VERIMAG Laboratory

CNRS – UJF – INPG

UMR 5104

Activity Report

2002 – 2005

VERIMAG
2, avenue de Vignate, F38610 Gières, France
http://www-verimag.imag.fr/

Contents

1 Introduction **11**
- 1.1 Overview . . . 11
- 1.2 The Embedded Systems Domain . . . 13
  - 1.2.1 Trends in Embedded Systems Engineering . . . 15
  - 1.2.2 Scientific Program: Fundamental Challenges . . . 17
- 1.3 How to Read this Report . . . 18

I Research Teams **19**

2 "Synchrone" Team **21**
- 2.1 History and Background . . . 21
- 2.2 The Lustre Toolbox . . . 21
- 2.3 Current Work Directions . . . 24
  - 2.3.1 Language Design . . . 24
  - 2.3.2 Multi-threaded and Distributed Implementations . . . 24
  - 2.3.3 Verification Techniques and Correct-by-Construction Development . . . 24
  - 2.3.4 Debugging, Testing, Simulation and Early Execution . . . 25
  - 2.3.5 Heterogeneous Components . . . 25
- 2.4 Distinguished Collaborations . . . 26
- 2.5 Perspectives for Future Work . . . 27
- 2.6 Team Members . . . 29
- 2.7 PhD Theses . . . 31

3 Distributed and Complex Systems Team **33**
- 3.1 History and Background . . . 33
- 3.2 IF: A Validation Environment for Real-Time Distributed Systems . . . 34
- 3.3 Component-based Design of Real-Time Embedded Systems . . . 34
  - 3.3.1 Fine-grain Control of QoS Requirements . . . 35
- 3.4 Program and System Validation . . . 35
  - 3.4.1 Programs with Dynamic Data Structures . . . 35
  - 3.4.2 Parameterized Networks Verification . . . 37
  - 3.4.3 Test Case Generation . . . 37
- 3.5 Security . . . 38

3.6 Distinguished Collaborations . . . 39
3.7 Perspectives for Future Work . . . 40
3.7.1 Component-based Design and the IF Environment . . . 40
3.7.2 Adaptive Systems . . . 40
3.7.3 Program Verification and Monitoring . . . 40
3.7.4 Security . . . 41
3.8 Team Members . . . 42
3.9 PhD Theses . . . 44

4 Timed and Hybrid Systems Team **47**
4.1 History and Background . . . 47
4.2 Current Work Directions . . . 47
4.2.1 Verification and Synthesis for Hybrid Systems . . . 48
4.2.2 Testing and Monitoring for Timed and Hybrid Systems . . . 48
4.2.3 Advanced Techniques for Embedded Compilation . . . 48
4.2.4 Fighting the State Explosion in Timing-related Problems . . . 49
4.2.5 Model-based Design and Implementation of Embedded Systems . . . 49
4.3 Distinguished Collaborations . . . 49
4.4 Perspectives for Future Work . . . 50
4.5 Team Members . . . 51
4.6 PhD Theses . . . 52

II Research Activities **55**

5 Heterogeneous Components **57**
5.1 A Framework for Component-based System Construction . . . 59
5.1.1 Introduction - Key issues . . . 59
5.1.1.1 Heterogeneity of Interaction . . . 60
5.1.1.2 Heterogeneity of Execution . . . 60
5.1.1.3 Heterogeneity of Abstraction . . . 60
5.1.2 Results . . . 61
5.1.2.1 Glue Operators and their Properties . . . 61
5.1.2.2 Correctness-by-Construction . . . 65
5.1.3 Perspectives for Future Work . . . 66
5.2 The IF Intermediate Format for Modeling Component Systems . . . 66
5.2.1 The IF Language and its Semantics . . . 67
5.2.2 Observers for the Expression of Dynamic Properties . . . 68
5.2.3 IF as a Semantic Framework for Timed Modeling Languages . . . 69
5.2.3.1 SDL for Real-time . . . 69
5.2.3.2 UML for Real-time Embedded Systems . . . 70
5.2.4 Case Studies . . . 72
5.2.5 Concluding Remarks . . . 75
5.3 Components for the Development of Embedded Systems . . . 75

5.3.1 A Language-based Approach . . . 76
5.3.2 The Development Environment . . . 77
5.3.3 First Results . . . 78
5.3.3.1 Synchronous Modeling of Asynchronous Systems . . . 78
5.3.3.2 Specification of Components . . . 78
5.3.3.3 Early Execution of Heterogeneous Systems . . . 78
5.3.4 Classes of Applications . . . 79
5.3.4.1 Lustre/SCADE Programs from Airbus . . . 79
5.3.4.2 Ad-hoc Sensor Networks . . . 79
5.3.4.3 Systems-on-Chip . . . 80

6 Language Design and Extensions 81
6.1 Evolutions of Lustre . . . 81
6.2 Mode-automata . . . 82
6.3 Contracts for Reactive Systems . . . 82
6.4 Description of Non-Deterministic Components . . . 83
6.4.1 Abstract Machine: Lucky . . . 84
6.4.2 User Friendly Language: Lutin . . . 84
6.5 AOP and Reactive Systems . . . 86
6.6 Projects and Contracts . . . 87
6.7 Perspectives for Future Work . . . 87

7 Validation Techniques 89
7.1 Verification . . . 89
7.1.1 The IF Toolbox for the Verification of Real-time Distributed Systems 89
7.1.1.1 An Open Platform for State Space Exploration . . . 90
7.1.1.2 Static Analysis for State Space Reduction . . . 91
7.1.1.3 A Front-end for Validation of UML . . . 93
7.1.2 Verification of Parameterized Networks . . . 95
7.1.2.1 Monadic Abstractions . . . 95
7.1.2.2 Transitive Closure of Regular Relations . . . 96
7.1.3 Verification of Software with Dynamic Memory . . . 98
7.1.3.1 Dynamic Data Structures . . . 99
7.1.3.2 Region-based Memory Management . . . 103
7.1.4 Verification of Numerical Programs by Abstract Interpretation . . . 104
7.1.4.1 Algorithmics of Convex Polyhedra. . . . 105
7.1.4.2 Exact Widening in Linear Relation Analysis. . . . 105
7.1.4.3 An Abstract Lattice for Addresses. . . . 106
7.1.5 Formal Construction and Verification of Lustre Programs . . . 106
7.1.5.1 Verification of Lustre Programs with Arrays . . . 106
7.1.5.2 Formal Construction of Lustre Programs . . . 107
7.1.5.3 Non-Deterministic Synchronous Observers . . . 108
7.2 Testing . . . 108
7.2.1 Automatic Testing of Reactive Numerical Programs . . . 109

7.2.1.1 The "Lurette" Tool . . . 110
7.2.1.2 Specification and Generation of Test Sequences . . . 111
7.2.1.3 Case Studies . . . 113
7.2.1.4 Application to Counter-example Search . . . 114
7.2.2 Conformance Testing . . . 114
7.2.2.1 Black-box Conformance Testing for Real-time Systems . . 114
7.2.2.2 Property-oriented Testing . . . 116
7.2.2.3 Property Testing . . . 117
7.2.2.4 Robustness Testing . . . 118
7.3 Debugging, Simulation and Early Execution . . . 118
7.3.1 Debugging Reactive Programs . . . 119
7.3.2 Simulation and Early Execution . . . 119

8 Implementation Techniques 123
8.1 Property-aware Compilation Technology . . . 124
8.1.1 Time-constrained Single-threaded Synchronous Programs . . . 125
8.1.1.1 Case Study: Radio Link of a GSM Terminal . . . 126
8.1.1.2 Case Study: ISDN Prototype Phone . . . 127
8.1.1.3 Case Study: Automated Vehicle Control Software . . . 127
8.1.2 Scheduled Native Code for Real-time Java Applications . . . 128
8.1.3 Multi-threading and Multi-processing . . . 129
8.1.3.1 Case Study: MPEG-4 Video Encoder . . . 130
8.1.4 Fine-grain Control of QoS Requirements . . . 131
8.2 Implementation of High-level Models . . . 136
8.2.1 Translation of Simulink/Stateflow to Lustre . . . 136
8.2.2 Implementation on TTA . . . 138
8.2.3 Implementation with Static-priority Preemptive Schedulers . . . 140
8.3 Perspectives for Future Work . . . 141

9 Security 143
9.1 Security Protocols . . . 143
9.1.1 Verification of Security Protocols within Dolev-Yao Model . . . 143
9.1.2 Beyond Secrecy and Authentication . . . 145
9.1.3 Timed Security Protocols . . . 145
9.1.4 Soundness of the Symbolic Model . . . 145
9.2 Certified Applications . . . 147
9.3 Mobile Code Specification and Verification . . . 150

10 Hybrid Systems 153
10.1 Introduction . . . 153
10.2 Verification and Synthesis of Continuous and Hybrid Systems . . . 154
10.2.1 Historical Survey . . . 154
10.2.2 Linear Systems and the d/dt Tool . . . 155
10.2.3 Extension to Non-Linear Systems . . . 156

10.2.4 Treating Parameters . . . 156
10.2.5 Differential Algebraic Equations . . . 157
10.2.6 Experience with d/dt . . . 157
10.2.7 Abstraction of Hybrid Systems . . . 158
10.2.8 Analysis of Polygonal Hybrid Systems . . . 158
10.2.9 Symbolic Reachability Computation . . . 159
10.2.10 Perspectives for Future Work . . . 160
10.3 A Unified Approach to Controller Synthesis . . . 160
10.3.1 The Game-Theoretic Approach to Design . . . 161
10.3.2 Scheduling under Uncertainty . . . 162
10.3.3 Search-based Verification and Control of Continuous Systems . . . 163
10.3.4 Synthesizing Low-Complexity Schedulers . . . 164
10.4 Decentralized Observation and Control . . . 166
10.5 Specification Formalisms for Hybrid Behaviors . . . 166
10.5.1 Timed Regular Expressions . . . 167
10.5.2 Temporal Logic for Continuous Signals . . . 167
10.5.3 Application: Property Monitors . . . 168
10.6 Fighting the Clock Explosion . . . 169
10.6.1 SAT Solvers for Difference Logic . . . 169
10.6.2 Abstraction Techniques for Timed Systems . . . 170
10.6.3 Getting Rid of Zones . . . 171
10.7 From Control Loops to Real-Time Programs . . . 172
10.8 Analog Verification . . . 172
10.9 Perspectives for Future Work . . . 172

11 Systems-on-Chip 175
11.1 Projects and Contracts . . . 175
11.2 Transaction-Level Modeling . . . 175
11.3 SystemC . . . 176
11.4 Formal Verification for SoCs . . . 177
11.5 Test and Runtime Verification for SoCs . . . 178
11.6 Expressing and Analyzing Non-Functional Properties of SoCs . . . 179
11.7 Perspectives for Future Work . . . 181

III Teaching and Training 185

12 Teaching and Administrative Tasks 187

IV Research Administration and Publications 191

13 Research Administration 193
13.1 Prizes and Awards . . . 193
13.2 Organization of Conferences and Workshops . . . 193

13.3 Evaluation of Research . . . 195
13.3.1 Steering Committees and Programme Committee Chairs . . . 195
13.3.2 Participation in Programme Committees . . . 196
13.4 Editorial Boards . . . 197
13.5 Invited Conferences, Tutorials and Summer Schools . . . 198

14 Publications **201**
14.1 Books and Edited Work . . . 201
14.2 PhD Theses . . . 202
14.3 Articles and Chapters of Books . . . 204
14.4 Contributions to Conferences and Workshops . . . 207
14.5 Research Reports and Internal Publications . . . 220
14.6 Miscellaneous . . . 222
14.7 Distributed Tools . . . 223

V Projects and Industrial Contracts **225**

15 Projects **227**
15.1 European Projects . . . 227
15.1.1 ADVANCE . . . 227
15.1.2 AGEDIS . . . 227
15.1.3 ARTIST and ARTIST2 . . . 228
15.1.4 ASSERT . . . 229
15.1.5 AMETIST . . . 230
15.1.6 CC . . . 231
15.1.7 INTERVAL . . . 231
15.1.8 NEVA . . . 232
15.1.9 Next TTA . . . 232
15.1.10 OMEGA . . . 234
15.1.11 PROSYD . . . 235
15.1.12 RISE . . . 235
15.1.13 SafeAir II . . . 236
15.1.14 SPEEDS . . . 237
15.1.15 TYPES . . . 238
15.1.16 VHS . . . 238
15.2 National Projects . . . 239
15.2.1 ARESA . . . 239
15.2.2 openTLM . . . 240
15.2.3 ALIDECS . . . 241
15.2.4 APRON . . . 241
15.2.5 AS “Verification of Quantitative Properties” . . . 241
15.2.6 AVERROES . . . 242
15.2.7 CORTOS . . . 242

15.2.8 CTRL-a . . . 242
15.2.9 DYNAMO . . . 243
15.2.10 EDEN . . . 243
15.2.11 EVA . . . 243
15.2.12 EXPRESSO . . . 243
15.2.13 MADEJA . . . 244
15.2.14 MASH . . . 244
15.2.15 OpenEmBeDD . . . 245
15.2.16 PERSIFORM . . . 245
15.2.17 POTESTAT . . . 246
15.2.18 PROUVE . . . 247
15.2.19 ROSSIGNOL . . . 247
15.2.20 SODA . . . 247
15.2.21 TAXYS . . . 248
15.3 Industrial Contracts . . . 248
15.3.1 Airbus 2002 . . . 248
15.3.2 Airbus 2004 . . . 248
15.3.3 ANACONDA . . . 248
15.3.4 ADHOC . . . 249
15.3.5 Esterel Technologies . . . 249
15.3.6 SocTLM . . . 250
15.3.7 FVTA . . . 250

VI Summary and Perspectives 251

16 Summary 253
16.1 Research . . . 253
16.1.1 Strengthen and Extend Existing Background and Competencies . . . 253
16.1.2 Opening to New Research Directions . . . 254
16.2 Transfer and Industrialization . . . 254
16.3 Visibility and Infrastructure . . . 255
16.3.1 The European Network of Excellence ARTIST . . . 255
16.3.2 The Center for the Technologies of Software (CTL) . . . 256
16.3.3 The joint laboratory with STMicroelectronics . . . 256
16.3.4 The “Pôle de Compétitivité” Minalogic . . . 256
16.3.5 ETAPS 2002 . . . 257
16.3.6 Awards . . . 257

17 Perspectives 259
17.1 General Perspectives . . . 259
17.2 Perspectives of the “Synchrone” team . . . 260
17.2.1 Research directions . . . 261
17.2.2 New projects . . . 262

17.3 Perspectives of the "Distributed and Complex Systems" team 263
17.3.1 Methods and tools for Component-based Adaptive Real-time Engineering 263
17.3.2 **Software Verification** 265
17.3.3 **Security** 265
17.3.4 Modeling and verification of autonomous systems. 266
17.3.5 Projects accepted for 2007-2011 267
17.4 Perspectives of the "Timed and Hybrid Systems" team 269
17.4.1 Hybrid Systems 269
17.4.2 Timed Systems 270
17.4.3 Monitoring 270
17.4.4 SAT modulo theories 271
17.4.5 Applications 271
17.4.6 Tools 273
17.5 Creation of a new team on the Implementation of Embedded Systems . . . 274
17.5.1 Multi-processor architectures 274
17.5.2 Dynamic memory management 275

A Organization and Resources **277**
A.1 Management and Administration 277
A.2 List of the Staff - January 2005 279
A.2.1 Teachers/Researchers 279
A.2.2 Engineers 279
A.2.3 Contract Employees 280
A.2.4 Administration 280
A.2.5 Temporary Teacher/Researchers (ATER) 280
A.2.6 PhD Students 281
A.3 Budget 282
A.3.1 Budget for 2001 282
A.3.2 Budget for 2002 285
A.3.3 Budget for 2003 288
A.3.4 Budget for 2004 291

B External References **297**
B.1 External References 297

Chapter 1

Introduction

1.1 Overview

VERIMAG was established in January 1993 as a joint research laboratory between VERILOG SA and the partners of the IMAG Institute (CNRS, INPG and UJF). VERIMAG is a public research laboratory since 1997. It is associated with CNRS, INPG and UJF. Currently, it has a total staff of 82 persons including 26 permanent researchers, 7 researchers under contract and 30 Ph.D. students.

VERIMAG carries out research in the area of embedded systems design. It aims to produce theoretical and practical tools for the cost-effective development of embedded systems of guaranteed quality. Quality includes dependability properties such as security, safety, availability and performance.

Between 1993 and 1997, VERIMAG has produced and transferred, in collaboration with Verilog SA, and subsequently with Telelogic, the Lustre language which is the basis for the Scade environment, as well as verification and testing results integrated into the Object-Geode environment. VERIMAG's results have given rise to transfer and to numerous contractual relations implying Verilog, Schneider Electric (nuclear plants), EADS for the development of safety critical systems in Airbus, Prover-Technology for a verification tool dedicated to Lustre, and Esterel-Technologies. Other industrial partners of VERIMAG are STMicroelectronics, Alcatel, CS-Transport, EDF, France Telecom, IBM, Intrasoft, ISD, Leti/CEA, Prover Technology, RATP, Silicomp, Trusted Logic.

VERIMAG has well-recognized competences in synchronous languages, validation and verification with focus on security and safety, modeling and temporal and hybrid systems analysis. It plays a significant role in real-time embedded systems. Our ambition is to contribute actively to the development of knowledge which constitutes the basis of informatics as a discipline for software and systems.

VERIMAG follows a development policy which relies on transfer to industry, to advance research. Particular attention is paid to the following aspects: 1) identification of innovative and realistic work directions and 2) ensuring financial support through industrial contracts.

These two points are very important. The first has permitted some of our techniques to be implemented not only in R&D projects, but also in real industrial applications. The

second point means that a significant share of our budget comes from contractual resources. Using technology transfer to allow relevant, high-quality research is at the heart of our approach. For its implementation, we work in the following complementary directions:

- Maintaining and reinforcing strategic industrial partnerships with companies such as STMicroelectronics, France Telecom, EADS and Airbus Industries;
- Setting up the structures required for transfer and industrial applications such as the "Critical Systems and Software Platform" (PSLC in French) and the "Centre for Software Technologies" (CTL in French - http://www.systemes-critiques.org/ctl/) structure for regional coordination of technology transfer.
- Participating in national, European and international R&D projects. VERIMAG is the scientific and technical leader of the ARTIST1 Accompanying Measure, and the subsequent ARTIST2 Network of Excellence proposal (http://www.artist-embedded.org/FP6/).

The laboratory is organized in teams carrying out work along three axes, applied to various domains and leading to specific development activities.

- Synchronous models and languages for the development of embedded systems
  `http://www-verimag.imag.fr/SYNCHRONE/`

  The Lustre language and a dedicated toolset (compilers, program validation tools) have been developed since 1984 and form the basis for the SCADE toolset, commercialized by Esterel Technologies. This tool has been adopted by major actors in critical embedded systems such as Airbus (flight control), Schneider Electric (nuclear plant control), CSEE-Transport (railway signalling). Current research topics include modeling, testing, verification and distributed code generation in relation with the Time-Triggered-Architecture technology. The most recent activities concern the use of the synchronous models and tools as a framework for component-based design and development of heterogeneous embedded systems, with applications to systems-on-a-chip, ad-hoc sensor networks and embedded control systems.

- Tools and methods for the validation of asynchronous and complex systems
  `http://www-verimag.imag.fr/~async/`

  During more than two decades, research on this topic has led to developments of tools, such as the Xesar, CADP and the IF toolset which have been connected to design languages such as Lotos, SDL and UML, and which have been partially transferred into the CASE tool ObjectGeode. Current research topics include semantics and real-time extensions of formalisms with heterogeneous execution and communication modes (SDL, UML) and modeling and formal verification of complex and timed component based systems. Rapidly expanding activities in this area are: development and study of rigorous semantic models and specification languages for security protocols; development of powerful and efficient analysis methods; and automatic generation of correctness proofs adapted to certification.

- Tools and methods for timed and hybrid systems
  `http://www.verimag.imag.fr/TEMPORISE/`

  Timed and hybrid models are extensions of discrete models to describe the interaction between a computer and the physical environment in which it operates. Since 1990, VERIMAG plays a leading role in this area, situated at the border between computer science and control theory. Current research directions in this domain are foundations of timing technology: investigation of new methods and algorithms to treat time-dependent phenomena, such as scheduling in real-time systems and in manufacturing, delay analysis in circuits and composition of timed systems as well as the connection between continuous modeling tools and code generators.

Over the past four years, VERIMAG's evolution has been characterized by the following:

- Development of research activities in new areas such as component-based modeling and implementation techniques for embedded systems. This development has been stimulated by industrial projects such as the IST Omega project on Real-Time UML and the IST Next TTA and IST RISE projects on the connection of SCADE with Simulink/Stateflow and its implementation on non-standard architectures. It was also boosted by strong collaboration with STMicroelectronics on topics such as "Transaction Level Modeling", property-aware compilation and QoS control for multimedia systems. Another new research area is Security, which started only 4 years ago, and has been consolidated in particular through the creation of new research positions.

- Evolution of the research activities within the teams. The activities of the "Synchrone" team have significantly shifted to cover new application domains, taking into account de-facto standard languages like Simulink and SystemC. The work on Lustre now covers topics such as high-level modeling and implementations for various distributed platforms. The activities on timed and hybrid systems have been diversified as their basic theoretical and technical results found applications in testing of real-time systems, circuit modeling, scheduling and implementation of real-time systems.

1.2 The Embedded Systems Domain

Embedded systems generally interact with their physical environment, which they can *control*. Moreover, they jointly integrate software and hardware. For these reasons, embedded systems are at the border between software, hardware, and control theory. These systems may be used in different types of applications, including transport (avionics, space, automotive, trains), electrical and electronic appliances (cameras, toys, television, washers, dryers, audio systems, cellular phones), power distribution, factory automation systems, etc..

The extensive use and integration of embedded systems in everyday products, marks a significant evolution in information science and technology. A main trend is the prolif-

eration of embedded systems, that should work in seamless interaction while respecting real-world constraints.

Embedded systems have a number of specific characteristics, which play a role in structuring the technical domain, and for determining the relevant areas of research and industrial development:

- Criticality - Embedded systems are often critical. The degree of criticality depends on the consequences of deviation from a nominal behavior that can impact safety, security, mission completion, business. As an example, safety criticality means that failures can damage human lives, or cause major catastrophes. Safety critical systems include flight control systems, railway signaling systems, industrial process management systems. As a second example, business criticality is where major financial stakes are associated with proper operation. Business critical systems include services deployed on an embedded infrastructure such as cell phones, power distribution, traffic management, distributed entertainment, etc..

- Reactivity - Embedded systems are deployed in the physical environment, and as such they have a continuous interaction with it. This is a central characteristic of embedded systems. Reactivity implies that embedded systems are subject to real-time constraints relating their execution speed to that of their environment.

- Autonomy - Embedded systems generally need to be autonomous, that is, to fulfill their functions without human intervention for extended periods of time. Autonomy is needed, especially where human reactions may be too slow or insufficiently predictable.

Embedded systems are of strategic importance in modern economies. They are used in mass-market products and services, where value is created by supplying either functionality or quality. Functionality is defined as the service rendered to the user. Quality for a given functionality characterizes extra-functional properties of the product or service, such as performance, or dependability. For instance, a cellular phone offers functionality for mobile communication, while quality is characterized by audio fidelity, battery life, etc.. Embedded technologies confer advantages to system and service developers, in generating added value and enhancing competitivity. The relative weight of software in the value of embedded systems is constantly increasing. Software allows new, complementary services, and differentiation, which brings competitive advantages.

Embedded technologies are the fastest growing sector in Information Technologies.

Europe currently has leading positions in sectors where embedded technologies are central to growth. These sectors currently include avionics, automotive, space, consumer electronics, smart cards, telecom devices, energy distribution, railway transport. It is anticipated that they will also include distributed services such as e-Health and e-Banking. Europe has a leading position in civil avionics where fly-by-wire technology provides an overwhelming competitive advantage in the cost of operating aircraft. Europe is also well-positioned in the space sector, specifically for launch vehicles and satellites. In the

automotive sector, European manufacturers and their suppliers enjoy a leading technological advantage for engine control, and emerging technologies such as brake by wire and drive by wire. Railway signaling in Europe relies on embedded systems, and allows faster, safer, and heavier traffic. Embedded technologies will be extensively used to make energy distribution more flexible, especially in view of the coming market liberalization. Embedded technologies are strategic for the European telecommunication sector, which is still well-positioned - in spite of the recent difficulties in deploying UMTS technology. Finally, Europe is also well-positioned for e-Services (e-Banking, e-Health, e-Training), based on the leading edge in smart cards and other related technologies.

1.2.1 Trends in Embedded Systems Engineering

In the last ten years, certain spectacular changes have been observed in systems engineering. The number of embedded system applications was relatively low, mainly used for control applications in aircraft, trains, production plants. These applications being intensively safety-critical, development costs and times were extremely high. The development costs were nonetheless a small portion of the overall system cost. The coming generations of embedded systems - used in mass-market products - will be characterized by the following factors:

- A great variety of component types. Components will be tailored to fit technical and economic requirements, with differing service levels. For instance, specific components must be developed for mobile phones, for the different automobile sectors, for consumer electronics, etc.. This is markedly different from the situation ten years, ago, where general-purpose processors were prevalent.

- Integration of heterogeneous components in a real-world environment. A main requirement for embedded systems is to be integrated within the larger embedded environment. The objective is to obtain smooth and harmonious cooperation with other components to provide global services.

- Market Constraints. Market constraints require careful positioning between cost/quality for a given functionality. For example, mobile communication functionality can be provided at different optima of cost and quality, depending on the market segment sought. This requires predictable engineering techniques allowing to estimate costs as a function of design choices (e.g., processor speed, hardware versus software implementation). Market constraints will deeply transform engineering of computerized systems.

These factors imply the need for system development technologies allowing to jointly consider functionality, quality, physical implementation, and market constraints:

- Functionality is the capacity to deliver the specific service.

- Quality covers a set of properties related to performance, and dependability. Performance properties concern dynamic aspects of the system behavior, such as: throughput, jitter, speed, efficiency, response time, latency, etc. Dependability properties include safety, security, availability, reparability, and all properties characterizing the system's capability to provide a service in the presence of faults, errors, overload, and any type of incident susceptible of disturbing nominal behavior.

  Amongst dependability properties, safety and security are the most important. Safety means resistance to faults or errors. This is an essential property particularly for transport and process control applications.

  Security means resistance to attacks and active hindrance. This is essential for networked applications, such as banking and other commercial applications.

- Physical implementation constraints deal with the use of resources and the system's deployment context, such as weight, physical size, resistance to vibration or radiation, etc.. For embedded systems it is crucial to make the best use of resources, either to minimize costs (e.g., memory), or to improve autonomy (e.g., minimizing power and/or energy)

- Market constraints cover aspects related to the product quality and time to market, for a given functionality and target consumer profile.

Currently, we lack methods and tools allowing to jointly take into account these different constraints. Achieving the capacity to build systems of guaranteed functionality and quality, at an acceptable cost, is a major technological and scientific challenge.

It is important to have a fairly clear idea of what is feasible under the current state of the art. We identify two main technological approaches and related know-how, with their associated application areas:

- Hard Real Time and Safety Critical applications are extensively used in controllers. Such applications must meet deadline requirements, that guarantee that the system will be fast enough with respect its physical environment. They are used in transport and process control. They are composed of strongly coupled/coordinated components, integrated into a dependable, highly constrained architecture.

- Distributed Soft Real-Time applications with a good level of reactivity and dependability. Such applications are used in telecommunications, networked applications, etc.. They are characterized by their quality of service, rather than by hard real-time requirements. Quality of service expresses dynamic properties other than strict deadlines, such as throughput, jitter, mean or statistical values for time-related parameters.

We currently have the capability to build relatively small hard real-time applications on the one hand, or large distributed applications on the other hand. In the coming years, it

will be important to extend the applicability of hard real-time and safety-critical technologies to other areas (e.g., automobile). This implies accompanying technical adaptations, and that these technologies are available at compatible costs.

In the long term, we believe that it will be necessary to achieve large-scale, dependable distributed real-time systems, to allow applications such as automated freeways, and next-generation air traffic control.

1.2.2 Scientific Program: Fundamental Challenges

VERIMAG's scientific program addresses issues at the heart of a fundamental research challenge.

The need to jointly consider functional and extra-functional constraints in the design of embedded systems leads to the concept of system-centric development. Here, the main focus is the end result: a system as the combination of hardware and software, in interaction with its physical environment.

System-centric approaches are necessary to determine trade-offs between cost and quality, taking into account specific features of three factors: hardware, software and the environment.

Taking into account the dynamic characteristics of the physical environment - when known - can drastically simplify the design and optimize the components chosen and the dynamic resources used. For example, knowledge of the physical environment for embedded signal processors in cellular phones (specified by international standards) allows optimal choice of hardware.

Current methods and tools do not allow system-centric approaches for designing embedded systems. In fact, these approaches raise difficult, fundamental research problems, which are the basis of an emerging theory that should bring together Informatics and Physics.

Informatics consider models of computation based on abstract notions of machines (e.g., automata, complexity and computability theory, algorithms, etc.), that do not take into account physical properties of computation (e.g., execution times, delays, latency, etc.). From this point of view, software is abstract and is essentially characterized by functional properties. There is no unified theory allowing to predict the behavior of an application software on a given execution platform. The latter is composed of physical elements whose characteristics determine the execution speed and other dynamic properties of the application.

Clearly, a unifying theory is needed for encompassing concepts of computation that take into account physical properties of the underlying platform. Its absence raises a major scientific challenge, and seriously limits the state of the art in systems engineering. For software there exist theory and models for the verification of functional properties, whereas these are not available for implementations. Extra-functional properties of implementations are validated essentially via testing. This makes system validation costly, and less amenable to analysis.

1.3 How to Read this Report

VERIMAG is structured into three teams, respectively entitled "Synchronous approach", "Distributed and Complex Systems", and "Timed and Hybrid Systems". These teams are briefly presented in Part I, with references to the next part: in order to highlight the cooperations between the teams, the presentation of the research activities of the last 4 years (Part II) is organized by scientific themes, which tend to be transversal with respect to the activities of the teams. Part II ends with Chapter 16, which subsumes the main achievements of the period, and gives some perspectives.

Then, Part III describes the teaching and training activities, and Part IV describes the animation activities of the laboratory and lists the publications of the period from mid 2001 to 2005: notice that these publications are referred by numbers — like [13] — in the text, while citations in letters — like [WB98] — refer to the general bibliography (Appendix B) which contains relevant references, from either outside the Laboratory or outside the period.

Part V lists the various projects, and the industrial relations. In Appendix, one can find some administrative and budget informations, together with the the general bibliography.

Part I

Research Teams

Chapter 2

"Synchrone" Team

2.1 History and Background

20 years ago, the "synchronous" team at the VERIMAG laboratory proposed the formally-defined synchronous language LUSTRE, for the development of critical control software. In 1993, LUSTRE was transferred successfully to industry. It is now the core of the industrial tool SCADE provided by Esterel-Technologies. SCADE is used by Airbus, Schneider Electric, and Eurocopter, among many other companies. More details on the story of LUSTRE may be found on our web page: http://www-verimag.imag.fr/SYNCHRONE/history.pdf

More recently, the team worked on a pure synchronous version of Statecharts called ARGOS [72], and studied several ways of mixing dataflow programs with imperative features [JLRM94, Jou94, 73, 36].

In parallel with this work on languages, the team has worked on general validation techniques such as model-checking (providing the tool LESAR [HLR92]), abstract interpretation (see the tool NBAC [JHR99, 65]), connection to theorem-provers (see [DC00]), test (see [RWNH98, 207]), simulation, and debugging (see [MG00, 206, 25]). These techniques have been implemented in tools connected to LUSTRE, and applied to industrial case-studies. LUSTRE and these tools can now be used as a validation tool-box for a wide variety of embedded systems.

The work during the past 4 years has extended further than the traditional topic on pure synchronous languages and systems. The present activities of the team are now organized into five long-term research topics, listed below, and which reflect these new orientations. The short-term projects are usually transverse to this structure.

We first give an overview of the LUSTRE toolbox, then we detail the five topics below.

2.2 The Lustre Toolbox

Figure 2.1 shows all the tools of the LUSTRE toolbox and the relationships between them.

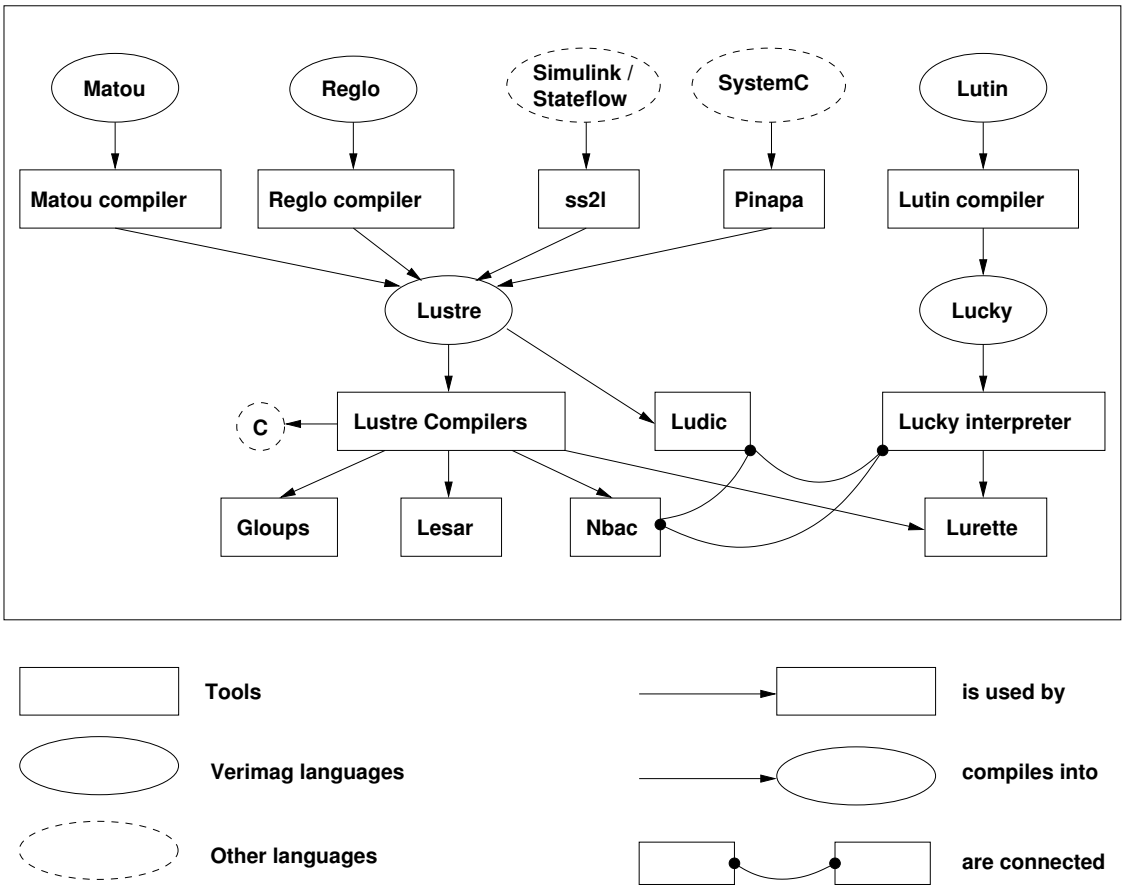


Figure 2.1: The Lustre toolbox

Compilers

- Lustre is a synchronous data-flow language for programming reactive systems. The compiler generates C code, using methods that optimize either the code size (single loop) or the code execution speed (automata based compilation).
- Reglo is a language based on regular expressions. It is used to express the dynamic behavior of a finite Boolean memory, or, equivalently, temporal properties over a finite set of Boolean variables. The associated compiler translates Reglo programs into either a set of language equations (similar to a sequential circuit net-list), or a Lustre observer (a program which accepts behaviors satisfying the underlying property).
- Matou is an implementation of Mode-Automata (see section 6.2) on top of Lustre.
- ss2l (see section 8.2.1) translates Simulink/Stateflow to Lustre.

- PINAPA (see section 11.3) is an open source SystemC front-end. It relies on GCC to parse the C++, and on the SystemC library to extract the architecture of the platform to analyze. It produces LUSTRE programs that can be analyzed by our validation tools (test or verification).

Test, simulation and early execution

- LURETTE is an automated testing tool of reactive programs. The environment of the system under test is described by LUSTRE, LUTIN or LUCKY programs.
- LUCKY is a language based on interpreted automata for the simulation of non-deterministic reactive code. The interpreter of this language is the core of the testing tool LURETTE. A stand-alone interpreter (simply called LUCKY) is also provided.
- LUTIN is also a language dedicated to the description of non-deterministic reactive systems. Compared to LUCKY, it aims at improving user-friendliness by providing functional abstraction and high level control structures. It compiles into LUCKY.

Verification

- LESAR is a symbolic model-checker, mainly based on the classical BDD-based approach. It also deals with simple static numerical constraints (linear algebra).
- NBAC is an abstract-interpretation based tool that deals with dynamic numerical properties. It implements an original approach for approximating the (potentially) infinite state space of a reactive program.
- GLOUPS is an automatic generator of PVS proof obligations for LUSTRE programs and safety properties.
- GOUPIL is a tool that slices LUSTRE programs according to their array structure, in order to generate proof obligations for various other verification tools. The proof obligations are rewritten in LUSTRE.

Debugging

- LUDIC is a debugger for LUSTRE programs. It has been connected to NBAC and LUCKY in order to provide an automatic "reach that state" capability [206].

Common libraries Most of these tools are relying on common libraries, in particular:

- GBDD is a Binary Decision Diagrams library, initially written for the LUSTRE compiler in the early 90s. It is also used by LESAR, NBAC, LUCKY/LURETTE. A stand-alone symbolic calculator is also provided (BDDC).
- POLKA is a convex polyhedron library. It is used by LESAR, LUCKY, and NBAC.

2.3 Current Work Directions

2.3.1 Language Design

Based on our experience in proposing languages that are both formally defined and adapted to industrial practices, part of our work is still devoted to language design: LUSTRE extensions, new synchronous languages mixing control and dataflow, languages for the description of non-deterministic environments, languages for the specification of components in the form of contracts, aspect-oriented programming for reactive systems, etc. These activities are detailed in section 6 and have been published in various papers: [232, 225, 224, 36, MR97, MR98, MRR00, MR00b, MR00a, 300, 226, 73, 243, 37, 207, 91, 92].

We also use LUSTRE as a target code for languages in other domains. In particular, the verification tool-box can now be used for various (not necessarily synchronous) formalisms provided that they can be given a semantics in LUSTRE. For example, gateways to LUSTRE exist for SIMULINK/STATEFLOW (see §8.2.1, the European projects Next TTA §15.1.9, and RISE §15.1.12, and the papers [158, 248]), and SystemC (see the project SocTLM §15.3.6 on the modeling and analysis of systems-on-a-chip, conducted with STMicroelectronics, and the papers [233, 234]).

2.3.2 Multi-threaded and Distributed Implementations

The problem of compiling synchronous languages efficiently for simple mono-processor architectures without operating systems was solved several years ago, and is implemented in several tools (the SCADE certified compiler, our academic compiler). The team now works on non-classical implementations, typically for on distributed architectures or multi-threaded programs running on a RTOS that uses preemptive scheduling. We study new implementation techniques mixing time-triggered and event-triggered execution schemes. We also study the problems due to the implementation of a discrete controller in a continuous world. These activities are conducted in the context of the projects Next TTA (see §15.1.9) and RISE (see §15.1.12). They are detailed in section 8.2. The related papers are: [158, 157, 248, 247, 80, 256].

2.3.3 Verification Techniques and Correct-by-Construction Development

The team develops and distributes the model-checker LESAR. The abstract-interpretation tool NBAC was originally developed by the team. We still work on general verification techniques, mixing model-checking and abstract interpretation, or connecting LUSTRE to PVS.

The work on abstract-interpretation techniques is described in section 7.1.4, and the related publications are [30, 203]. It takes place in the project APRON (§15.2.4).

The team also works on mixed methods:

- A development method based on a notion of *refinement* for reactive systems, à la B.

- A tool called GLOUPS, able to extract PVS proof-obligations from LUSTRE programs. GLOUPS can be used to prove properties that cannot be dealt with in model-checking or even abstract-interpretation tools. Typically, it can prove properties about non-linear dynamic behaviors.
- A tool called GOUPIL, able to extract proof-obligations (written in LUSTRE) from a LUSTRE program structured with arrays and local specifications in the form of contracts. The proof-obligations can then be sent to various verification tools (LESAR, NBAC, GLOUPS, ...)

These activities are described in sections 7.1.5.2 and 7.1.5.1. The related publications are [209, 32, 224].

2.3.4 Debugging, Testing, Simulation and Early Execution

Testing is complementary to formal verification. We are developing a complete solution for embedded systems, covering: the formal description of the non-deterministic environments, the automatic generation of realistic test sequences, the description of oracles, and the automatic execution of tests. Recently, this testing activity has been opened to include general simulation mechanisms for non-deterministic parallel processes and early-execution techniques.

These activities are described in section 7.2.1. They are now conducted in the context of several projects, with or without industrial partners: Alidecs (§15.2.3), ASSERT (§15.1.4), AdHoc (§15.3.4), SocTLM (§15.3.6).

The related publications are [37, 207, 77, 25, MG00, 206].

2.3.5 Heterogeneous Components for Embedded Systems and Model-Driven Development

Apart from the traditional use of LUSTRE and its verification toolbox in the domain of critical control software, the team has started to work on the modeling and analysis of complex systems in various domains. To tackle these new problems, we rely on several ideas and existing techniques:

- Synchronous languages may be used, in a wide variety of contexts, to *model* asynchronous phenomena
- LUSTRE and other synchronous languages are well suited for the description of synchronous hardware.
- LUSTRE and its variants can be used as an *executable* formal specification language.
- Most of the validation tools in the LUSTRE toolbox are not limited to LUSTRE as the input format.
- The work on synchronous languages, semantics and implementations, has led to several proposals for bridging the gap between pure centralized synchronous systems and completely asynchronous distributed systems: GALS, quasi-synchronous systems.

The kinds of heterogeneity we are interested in are:

- Hardware, application software and operating systems or middleware.

- Synchronous, asynchronous, GALS systems.
- Several degrees of granularity in the description, concerning both time and data.
- Determinism and non-determinism.
- Models of the environments cohabiting with partially described components or fully implemented ones.

Our approach for studying heterogeneous components is the following:

- Find application domains and industrial case-studies that constitute relevant instances of the general problem; we are currently working on three instances: Lustre/Scade programs from Airbus, Systems-on-a-chip (see 11), Ad-hoc sensor networks (see 15.3.4).
- Adopt a language-based approach, meaning that we propose high level languages for specifying and programming components, for describing the environments, for expressing properties to be checked on components or on their compositions (see the project ALIDECS 15.2.3).
- Start small and extend our models and tools progressively, including new models and implementation techniques as soon as their relationship with the existing core is well understood. For instance, starting from the first implementation techniques for Lustre, dedicated to centralized mono-processor systems, we have defined more general techniques addressing more complex architectures, without losing the benefits of the original simple techniques.

In the RISE project (see 15.1.12), we demonstrated our approach to model-driven development of embedded software and systems by developing a complete chain from models written in Simulink/Stateflow to embedded C code, via Lustre and TTA (Time-Triggered-Architecture).

These activities are further described in section 5.3.

2.4 Distinguished Collaborations

The "Synchrone" team maintains a long term cooperation with Esterel-Technologies, which develops and commercializes the Scade toolset — projects Next TTA (§15.1.9), RISE (§15.1.12), ASSERT (§15.1.4), contract Esterel-Technologies (§15.3.5 — , and with some of its customers: with Airbus, the cooperation is moving from the R&D department to people more involved in the production (flight commands, in particular); Hispano-Suiza (Snecma) was the coordinator of the SafeAir II project, and a β-tester of Lurette. Within the ASSERT project (§15.1.4), our tightest cooperation is with CS (on the connection of AADL, UML and Scade/Lustre), Prover-Technology (on modular verification), EADS (on case studies from aerospace), and of course with ESA (the coordinator).

Since 2002, we have established a very strong cooperation with the System-Platform-Group of ST Microelectronics, on the modeling and analysis of systems-on-a-chip at the transactional level (see chapter 11). Three PhD students are funded partially by STMicroelectronics, to work on the common project.

Recently, in 2004, we established a cooperation with France Telecom R&D and other academic teams on the modeling and analysis of ad-hoc sensor networks (the CIS team of

the TIMA laboratory and the DRAKKAR team of the LSR-IMAG laboratory in Grenoble, and INSA Lyon).

On the academic side, we cooperate with

- LIAFA (A. Bouajjani), IRCCYN (Ch. Mauras) and LSV (Ph. Schnoebelen, A. Finkel) in the Specific Action on "Verification of Quantitative Properties" (§15.2.5.
- LIP6 (M. Pouzet), CMA-ENSMP (F. Boussinot), INRIA Rhône-Alpes/PopArt (A. Girault), LAMI/EVRY Univ. (J.-M. Delhomme) in the project Alidecs ($15.2.3).
- the LIENS (P. Cousot), the "Ecole des Mines" (F. Irigoin) and IRISA (B. Jeannet) in the APRON project (§15.2.4).
- IRISA (A. Benveniste) and Berkeley (Alberto Sangiovanni-Vincentelli) in the ARTIST project.

The team has been active in the organization of the yearly SYNCHRON meeting, from 1994. This meeting gathers academic and industrial people interested in the synchronous technology, from all Europe, India, the USA. We also created the SLAP workshop (Synchronous Languages, Applications, and Programming) in 2002, in collaboration with S. Edwards (Columbia University) and M. Mendler (Bamberg University, Germany).

2.5 Perspectives for Future Work

For the next five years, the directions for future work are directly related to the recent reorganization into five main topics. All the research activities will continue to be conducted in strong cooperation with industrial partners, in the traditional application domain of critical control systems (Airbus, Schneider Electric, Audi, Esterel-Technologies, ...) or in new application domains where the synchronous technology may help (STMicroelectronics for systems-on-a-chip, France Telecom R&D for ad-hoc sensor networks).

We will continue to maintain the Lustre toolbox. Future work in language design is detailed in section 6.7. In the sequel we highlight three general research directions.

The Lustre/SCADE Environment as a Development *Method* for Various Architectures

At the beginning, Lustre was "only" a programming language, with a compiler for simple platforms. Then we developed validation tools (test, verification) and it became the Lustre/Scade toolbox.

The more recent projects have shown that Lustre/Scade is now becoming a candidate for a complete *development method*, encompassing: high level modeling in some engineering design languages that can be translated into Lustre (typically Simulink/Stateflow), formal verification on models (formal representations of software and operating system parts, formal models of the environments), and efficient implementation for specialized architectures (for instance TTA platforms).

Following the work done in the context of the Next TTA and RISE projects, we will continue to apply the Lustre/Scade methodology (high level modeling, formal verification, automatic implementation by generating application code and/or operating system

glue) to complex architectures.

Verification Methods and Tools

Concerning verification, the ultimate goal is to provide tools usable for each kind of usage considered for LUSTRE, from programming to modeling. We still need to work on the core of verification tools.

- For data-dominated programs, which appear in particular in numerical control, the interactive proof approach needs to be completed, as these kind of problems are obviously out of the scope of automatic verification.

- Control-dominated programs generally present some simple numerical aspects (counters), and are, in our opinion, the main application domain for linear relation analysis. Both the performances and the precision of the analysis need to be improved, which is an endless research.

- The use of LUSTRE as a modeling language reveals new verification problems, with the modeling of (partial) asynchrony, the modeling of new features like addresses. New abstract domains, and combined verification methods (enumerative/symbolic) must be developed for that.

Formal Models and Tools for the Modeling, Early Execution and Simulation of Heterogeneous Systems

This last topic is a longer term perspective. It aims at exploiting the expertise of the team in: high level languages, implementation techniques, various models of time and concurrency, validation methods. The idea is to study complex systems that do not necessarily belong to our traditional application domain (critical control software). This is the natural continuation of our work on the two new applications domains we started to study recently: systems-on-a-chip (since 2002) and ad hoc sensor networks (since 2004). This may be done by using the LUSTRE toolbox, but also other tools.

The main motivation for this research direction is the following fact: in the context of embedded systems development, heterogeneity is and will always be the law, and it concerns all stages of the development cycle. High level languages are different (not only because of their syntax, but also because of the computation models they rely on), the granularity of the components is different (for instance the time granularity of a hardware Register-Transfer-Level (RTL) design and that of a software part are quite different), the first specifications are also different in form and content (the development of control systems starts with SIMULINK models, while the development of software parts may start with UML modeling, and the development of hardware parts with VHDL or higher level models).

On the industry side, a tool like SystemC, and even the idea of modeling SoCs at the so-called *transaction level*, appeared because SoC design needs very fast and very early

simulations of the architecture together with the embedded software. SIMULINK has become a de-facto standard for almost the same reasons: the emphasis is on fast and early simulation of the designs. On the academic side, a lot of work has been spent on computation models, semantically well-founded models and languages, and formal validation tools.

It seems hopeless to find a unifying semantic model (and the appropriate language support) in which all the existing descriptions could be translated, manipulated, and used for fast simulation, formal validation and implementation purposes, the least reason being that new languages and tools appear each day.

We aim at providing methods and tools that help in bridging the gap between these industrial tools driven by simulation needs (but in which the need for formal validation appears as soon as the systems are complex enough), and the formal models and tools developed with semantic properties in mind. This very general goal can be divided into more dedicated objectives. Here is a non-exhaustive list:

- Open simulation environments, in which some components come from formally-defined languages, and others are taken as black-boxes. The architecture of the system and the various communication protocols are on the "formal" side. It means that, for instance, formal tools can be used to check synchronization properties. On the other hand, the detailed behavior of the components may have to be abstracted. A black-box component should have a formally defined interface, for understanding how it behaves in some environment, and also an API for the main simulator to activate it.

- Several notions of comparison between levels of abstraction and/or granularities. The first instance of this problem we will address is related to the RTL and TLM levels in SoC design (see section 11.7), but the problem is much more general.

- Extraction of models from engineering design languages. We already did that for SystemC and Simulink, and the same approach may be applied to other languages. Since the source language has no formal semantics, any translation into a formally defined language is, de-facto, a semantics. This semantics might differ from the official and informal semantics as given by the simulation tools interpreters. A translation into LUSTRE is therefore very interesting, because it is executable. The LUSTRE code can, at least, be simulated and its behavior compared to the behavior of the simulation tools.

2.6 Team Members

INPG/ENSIMAG stands for the computer science and applied mathematics department (ENSIMAG) of the National Polytechnical Institute of Grenoble (INPG). UJF stands for University Joseph Fourier. CNRS stands for the National Center for Scientific Research.

Permanent Members

K. Altisen, Assistant Professor, INPG/ENSIMAG
PhD students: D. Stauch
participates in : ALIDECS, CORTOS

P. Caspi, Director of Research, CNRS
PhD students: A. Curic, J. Mikac, C. Kossentini, C. Sofronis
projects and contracts: Next TTA, RISE
participates in : ALIDECS, ARTIST, ARTIST2

N. Halbwachs, Director of Research, CNRS
PhD students: D. Merchat, L. Danthony-Gonnord, M. Péron
projects and contracts: ASSERT, SafeAir II, APRON
participates in : Airbus-2002

E. Jahier, Engineer, CNRS
participates in : SafeAir II, Airbus 2004, ALIDECS

F. Lagnier, Assistant Professor, UJF

F. Maraninchi, Professor, INPG/ENSIMAG
Team leader
PhD students: L. Morel, M. Moy, C. Helmstetter, J. Cornet, L. Samper, D. Stauch
projects and contracts: ALIDECS, Airbus-2002, SocTLM, openTLM, adhoc
participates in : RISE, ARTIST, ARTIST2

C. Parent, Assistant Professor, UJF
PhD students: D. Merchat

X. Nicollin, Assistant Professor, INPG/ENSIMAG

A. Rasse, Assistant Professor, UJF

P. Raymond, Researcher, CNRS
PhD students: Y. Roux
projects and contracts: Airbus 2002, Airbus-2004, Esterel-Technologies
participates in : ALIDECS

P.-C. Scholl, Professor, UJF

Temporary Members

Y. Bouzouzou, Engineer, 2003-04
participates in : SafeAir II

D. Potop-Butucaru, Post-doc 2005
participates in : ASSERT

N. Scaife, Post-doc 2003-05
participates in: RISE

2.7 PhD Theses

Past PhD Theses

1. Fabien Gaucher (2000-2003) : *Ludic, a debugger for Lustre*, Florence Maraninchi
2. Yvan Roux (2000-2004) : *Description and execution of non-deterministic reactive systems*, Pascal Raymond
3. David Merchat (2001-2005) : *Algorithmics of polyhedra for Linear Relation Analysis*, Nicolas Halbwachs
4. Lionel Morel (2001-2005) : *Arrays and contracts for the development of regular embedded systems*, Florence Maraninchi
5. Matthieu Moy (2002-2005) : *Methods and Tools for the Verification of systems-on-a-chip at the transactional level*, Florence Maraninchi
6. Jan Mikáč (2002-2005) : *Refinement and proofs for Lustre programs*, Paul Caspi

Current PhD Theses

1. Jérôme Cornet (2004-2007) : *Expressing and validating non-functional properties of systems-on-a-chip*, Florence Maraninchi
2. Adrian Curic (2001-2005) : *Implementing Lustre Programs on Distributed Platforms with Real-Time Constraints*, Paul Caspi, Stavros Tripakis
3. Laure Danthony-Gonnord (2003-2006) : *Exact widening in Linear Relation Analysis*, Nicolas Halbwachs
4. Claude Helmstetter (2003-2006) : *Testing and runtime-verification techniques for systems-on-a-chip at the transactional and RTL levels*, Florence Maraninchi
5. Chiheb Kossentini (2003-2006) : *Robust implementation of distributed hybrid control laws. Application to Airbus flight commands*, Paul Caspi
6. Ludovic Samper (2004-2007) : *Modeling and analysis of ad-hoc sensor networks*, Florence Maraninchi, Laurent Mounier
7. Christos Sofronis (2003-2006) : *Embedded Code Generation from High Level Models*, Paul Caspi, Stavros Tripakis
8. David Stauch (2004-2007) : *Aspect-orientation and reactive systems*, Karine Altisen, Florence Maraninchi
9. Mathias Péron (2005-2008) : *Compositional Validation of Components for Embedded Systems*, Nicolas Halbwachs

Chapter 3

Distributed and Complex Systems Team

3.1 History and Background

The research of the DCS team aims at studying the foundations and at developing languages and tools that support the rigorous development of complex and distributed systems. Complex and distributed systems cover hard and soft real-time applications, security protocols and security applications as well as programs. The common feature of these systems is that their behavior is logically and combinatorially complex and that their design is error prone. Therefore, there is a need for a design discipline for this type of systems. This research is motivated by the conviction that system and program developers need adequate and abstract concepts and efficient tools to face the problems they encounter.

Initially, the research of the DCS team has focussed on the modeling and validation of communication protocols. The team has pioneered model-checking techniques and model-based test case generation. The first model-checker, CESAR, was build in 1982 by J.-P. Queille and J. Sifakis, and was able to handle CTL-like properties and systems described as Petri nets. Other model-checkers followed culminating in the IF-toolbox that handles real-time distributed systems whose specification may include C-code, dynamic process creation and priorities. The IF-toolbox includes static analyzes techniques for mitigating the state explosion problem and allows properties to be written as temporal properties as well as observers. It is currently widely used within and outside VERIMAG; we count approximately 20 downloads each month and 200 users. Maybe more importantly it used by our industrial partners such as France Telecom, EADS and IBM. The team has also built, together with IRISA, a model-based automatic test case generation that has been successfully transferred to industry and integrated in the SDL-based tool suite of Verilog.

While continuing the work on the modeling and the verification of distributed systems, the team has made substantial effort during the last four years to start new research topics: "Component-based design of real-time embedded systems", "Foundations and tools for computer security" and "Program verification". In the remaining of this section, we present our main results during the last four years (2001-2004).

3.2 IF: A Validation Environment for Real-Time Distributed Systems

The IF environment evolved from an **I**ntermediate **F**ormat, for integrating different tools, to a powerful toolbox that provides users with a number of different techniques for specifying and verifying *real-time distributed systems*. The toolbox supports different system-design notations, such as Real-time UML, SDL and the IF notation. It is also worth mentioning that it supports the validation of real-time systems that may include native code, e.g. C-code, dynamic process creation, structured states, different communication primitive and priorities [143]. To our knowledge, there is no other tool that is able to handle systems combining these primitives. The toolbox includes the following functionalities:

- Front-ends from different specification languages, such as UML and SDL, equipping these with rigorous semantics and validation support [139, 201, 197, 58].
- Explicit-state on-the-fly state space exploration techniques that are tuned towards process dynamic creation, real-time constraints, FIFO queues communication and priority rules [143, 140, 144, 59]. Properties can be written declaratively using temporal logic or operationally using observers.
- Static analysis and slicing techniques for mitigating the explosion problem of size of state space. These techniques are tuned to the particular feature of the models underlying the considered systems, such as timing constraints and FIFO queues [49, 19].
- Algorithms for test case generation, both timed or untimed [26, 181, 122, 182, 214, 215].

Since its first release in 1998, the toolbox has been acquired by about 200 groups and used in many projects (sometimes without the participation of Verimag).

Modeling and language design aspects related to the IF environment are discussed in Section 5.2 while the validation aspects are discussed in Section 7.1.1.

3.3 Component-based Design of Real-Time Embedded Systems

The aim of our work is to develop theory, methods and tools for component-based construction of real-time systems.

Theory aims aims at providing a general framework encompassing heterogeneous composition, that is composition of components with different execution and interaction models. A key idea is the study of a framework for composition clearly dissociating structure from behavior, due to specific properties of composition (glue) operators. This leads to a layered construction approach which we show to be essential for incremental construction.

Contrary to existing approaches e.g., process algebras, structure is considered to be a first class entity with specific laws and properties.

Methods focus on component-based construction techniques. We develop techniques for guaranteeing correctness by construction for generic properties such as deadlock-freedom, liveness, timelock-freedom. The techniques use a methodology which combines composability and compositionality results. Furthermore, we study problems related to the expressivity of the framework, by showing that the proposed operators are powerful enough to model scheduling and heterogeneous coordination. Our results are published in [42, 189, 188, 190, 57] and explained in more detail in Section 5.1.

This framework is currently partially implemented in the IF toolbox. This includes the language for behavior description, the use of connectors and of dynamic priorities, the principle of description layering. We are currently experimenting a new tool fully implementing the framework.

3.3.1 Fine-grain Control of QoS Requirements

Extra-functional properties are often crucial for embedded applications. In particular, Quality of Service (QoS for short) properties are for some applications, as video decoders, vital. For productivity reasons, it is desirable to obtain the needed embedded software by adapting some generic components. A difficult problem is, however, that QoS properties do not only depend on the considered software but also on the execution platform and on the mapping of this software on the platform. Thus, the generic component has to be adapted according to the execution platform and the QoS requirements. Thus, the problem we want to solve is the following : given an application software with hard deadlines constraints, construct a controlled software which uses the resources optimally. In order to meet hard real time deadlines, usual techniques are based on Worst Case Execution Time (WCET) analysis. Thus, the application is scheduled statically and there is no deadline miss. On the other hand, soft real time approaches, whose goal is to ensure an optimal utilization of the resources, are generally based on dynamic techniques and the control is made at task level. In [166], we propose a method bridging the gap between hard and soft real time. The considered method allows to generate from an application software a controlled application software that meets given QoS requirements. This method is explained in more detail in Section 8.1.4.

3.4 Program and System Validation

3.4.1 Programs with Dynamic Data Structures

The goal of our research is the development of verification techniques for programs that handle dynamic data structures, typically recursive data structures implemented using dynamic memory allocation and pointer data types. We are particularly interested in programs written in high level languages such as C# or Java, that offer means to dynamically handle Essentially, two approaches to the verification of programs with dynamic

data structures can be found in the literature. On one hand, there are techniques based on approximated fix-point computation using abstract interpretation techniques. Shape Analysis introduced in [SRW98] provides a powerful and flexible analysis method that allows to compute an abstraction of the heap graph of a given program using three-valued first-order logic [SRW02].

On the other hand, there are deductive verification methods based on Floyd-Hoare like program logics. As an important representative, we can mention here the work based on Separation Logic as assertion language [ORY99]. Deductive verification yields, in general, sound and complete methods but often, as in the case of programs with lists, they cannot be at the same time sound, complete and effective.

We are interested in the verification of three kinds of properties:

- Safe execution properties e.g., each pointer should be allocated prior to use (no segmentation faults), each allocated memory cell should be reachable starting with the program variables (no memory leaks).

- Topological properties, involving the characterization of the shapes that the memory configurations might have e.g., lists, trees, circularities, etc.

- Quantitative properties i.e., arithmetic reasoning about the sizes of data structures involving i.e., lengths of lists, heights of trees, etc.

Our approach is that of semi-algorithmic verification. In fact, the verification problems for such properties of dynamic programs are undecidable for the most [1]. However, many non-trivial algorithms (e.g., the Schorr-Waite graph markup algorithm) have been proven correct using manual proof methods. A verification tool based on a semi-algorithmic approach, that would perform the verification of classical toy examples, as well as of real-life case studies, would be a great help to the developers of complex and reliable software. In order to reach this goal, we address the following problems:

- A semantic model for programs with destructively updated data structures [145].

- Specification logics for reasoning about the topology of the dynamically memory [147, 146].

- Semi-algorithmic verification for programs handling linked-list data structures [148, 149], that is, decision procedures for the specification logics and techniques for automatically generating invariants.

A detailed description of our results can be found in Section 7.1.3.1.

[1] Notice that the existence of a semi-algorithm does not imply that the problem is r.e., since as usual we usual semi-algorithm in the sense of a partially correct algorithm whose termination is not guaranteed

3.4.2 Parameterized Networks Verification

A parameterized network is a family of systems of the form $S_1, \| \cdots, S_n$ where n is the parameter of the family. Many distributed algorithms are described as a parameterized network. Nancy Lynch's book, *Distributed Algorithms*, contains a rich variety of such networks. We are interested in the verification of such network and more specifically in their automatic verification. We follow two approaches both based on the fact the transition relations of many parameterized networks can be represented as a regular set. The first approach is based on monadic abstractions. The main idea here is that the transition relation of many parameterized networks can be described as monadic formulae with process indices are free variables. Then, exploiting the effective equivalence of monadic second-order logic and finite state automate, we can design algorithms for automatically computing safe abstract systems. Results related to this approach are published in [BBLS00, BLS00, 120, 46] and discussed in Section 7.1.2.

The second approach uses the fact that regular transition relations can be recognized by finite state transducers, and hence, on can design partial algorithms for computing the transitive relation of such transducers. Partial is here to be understood, in the sense of partial correctness in program verification, that is if the algorithms terminates then it computes the transitive closure. Our main achievement here is the use of forward and backward simulations to collapse infinitely many states in the infinite table corresponding the transitive closure [172, 53]. A detailed presentation of this approach is presented in Section 7.1.2.

3.4.3 Test Case Generation

Conformance testing consists in checking whether a given implementation, called implementation under test (IUT for short), behaves conformly to a given specification. In our case, the semantics of both the specification and the IUT is described as a labeled transition system and to behave with conformly means that the IUT does not exhibit an output which is not allowed by the specification. We are interesting in the problem of generating test cases automatically from the specification model. A test case is a sequence of inputs to the IUT and expected outputs from the IUT and a verdict stating whether the test has passed or not. Writing test cases is still an error prone and time consuming activity, hence, we are interested in techniques for automatically generating such test cases. Our approach is model-based and consists in applying algorithms akin to model-checking algorithms to explore the specification's model and generate test cases. As stated above, we have developed efficient algorithms for test case generation that have been implemented in commercially available tools.

However, model-based test case generation has some limitations as it requires a completed specification and is rather globally oriented in contrast to a property-oriented testing. Therefore, we aim at developing algorithms that to allow automatic test generation from a *partial specification* (not necessarily expressing the complete expected behavior of the system), and with respect to a particular property (test case execution should indicate whether the IUT satisfy or not this property) More precisely, the test generation technique

we propose [181] is based on the following ingredients:

- A (possibly partial) specification $\mathcal{S}$ is used as a "guideline" for the test case synthesis, and it is therefore supposed to be "close enough" to the actual behavior of the IUT. Note however that we do not require at this level any particular conformance relation between $\mathcal{S}$ and the IUT.
- A safety or bounded liveness property $\mathcal{P}$ is given through an observer $\mathcal{O}$bs recognizing sequences of $\neg\mathcal{P}$. This observer is a parameterized automaton on infinite words.
- Test cases are automatically generated by traversing the specification in order to find the "most promising" execution sequences able to show the non satisfiability of $\mathcal{P}$ by the IUT. These execution sequences are the sequences recognized by $\mathcal{O}$bs that are the "closest" to the ones provided by the specification.

In [241, 182], we extend and adapt property-based testing to *robustness testing*. Informally, robustness can be defined as the ability of a software to keep an "acceptable" behavior, expressed in terms of *robustness requirements*, in spite of *exceptional* or *unforeseen* execution conditions (such as the unavailability of system resources, communication failures, invalid or stressful inputs, etc.).

3.5 Security

Our research activities in computer security started in 2000 with the Ph.D. Thesis of Liana Bozga on the automatic verification of security protocols. Since then, we have been working on the semantic foundations and the verification of security protocols, on developing a certification methodology for security applications and more recently on mobile code verification.

Automatic Verification of Security Protocols Our main results are:

1. An effective, complete and sound weakest precondition calculus for bounded security protocols and a general logic that allows to specify secrecy, authentication and other properties [135].
2. An abstract interpretation-based method for the verification of unbounded security protocols [137, BLP06]. The method has been implemented in a tool called Hermes [138].
3. A decision procedure for bounded timed security protocols [136, 48].
4. A tool for the automatic generation of Coq proofs from Hermes that do not depend on the abstractions used in Hermes nor on its fix-point computation.
5. A decision procedure for bounded protocols and opacity properties [229, 227].
6. A partial decision procedure for infinite state systems and opacity [286].

Complexity-Theoretic Soundness of the Symbolic Semantics As most verification techniques for security protocols, our work is based on the so-called Dolev-Yao model (also called symbolic and formal model). In this model, data is represented by algebraic terms, freshness by fresh name creation and the adversary capabilities by a deductive system. In particular, cryptographic primitives are supposed perfect, that is, there is no way to learn any information from a cipher text except by deciphering it which requires having the secret key. The latter cannot be guessed by the adversary; he has to derive from its knowledge using the deductive system. Obviously, this is not true in practice. For, a secret key is simply a string of bits can very well be guesses by an adversary. Yes, the probability to efficiently guess the right key is exceptionally small in the length of the bit-string. In fact, in the so-called complexity-theoretic approach an adversary can be any randomized polynomial-time Turing machine and correctness of a cryptographic scheme or of a protocol is defined using complexity-theoretic notions. This approach is usually considered more realistic as the Dolev-Yao model. Thus, a fundamental question is the relationship between these semantic models. In [208, 285, 289, 288], we show that the Dolev-Yao model a safe abstraction of the complexity theoretic model for protocols that may combine asymmetric encryption, symmetric encryption, digital signature, hashing and Diffie-Helman exponentiation. More detail and related work are given in Section 9.1.4.

A Methodology for Certified Security The Common Criteria are a standard for the certification of security applications. It defines seven levels with increasing demand of use of formal methods. We are interested in the highest level, EAL7, that imposes the use of formal techniques. We defined a methodology with tool support for the development of certified applications aiming at the level EAL7, the most demanding level. Our methodology is presented in Section 9.2.

3.6 Distinguished Collaborations

The DCS team has strong collaborations with France Telecom on practically the whole range of the team's activities, component-based design, verification and testing and security. This collaboration is materialized in joint projects, national and European, as well as in research contracts between France Telecom and Verimag. In particular, we are working on transferring a version of the IF-tool box specialized to voice services to France Telecom. We also have a long-term cooperation with Trusted Logic. This cooperation started with the EVA project and the work on security protocols and is now extended to certification.

More recently, we started a cooperation with STMicroelectronics on two subjects: Fine-grain control of QoS requirements and program debugging. In particular, two PhD. Theses are co-supervised by members of the DCS team and researchers at STMicroelectronics.

Concerning academic cooperations, we closely collaborate with the SECSI [2] (Sćurité des systèmes d'information) team at LSV Ens Cachan, LIAFA, Paris 7 and the German

[2]SECSI:http://www.lsv.ens-cachan.fr/axes/secsi.php

research Institute OFFIS. We also started a new collaboration with the Institute for Software Integrated, Systems [3].

Each year, the team hosts long-term visitors. In 2005, Professor Kanchi Gopinath, from Indian Institute of Science Bangalore, has spent three months and Francis Klay, from France Telecom, one month.

3.7 Perspectives for Future Work

3.7.1 Component-based Design and the IF Environment

The implementation of the general framework based on a layered model of components and the explicit separation between aspects related to the execution model and aspects related to the interaction and communication between components will continue. At the same as implementing the general case, we aim at implementing domain specific instances. In particular, we are interested in avionic and automotive applications. The basic research related to the general framework has also to be pursued. Indeed, we still need to develop and study scalable techniques that guarantee that some generic properties such as mutual exclusion and deadlock freedom. However, we are also interested in non-functional properties, more specifically we are interested in performance evaluation. A problem that is encountered in practice is that different unrelated models are used for specifying the functional properties on one hand and the expected performances on the other hand. Clearly, this is not satisfactory, one should aim at developing a methodology that allows to extract an evaluation model from an enriched functional model. We are working on developing such a methodology that will be based on our general framework for components, its implementation in UML and the IF-tool box.

3.7.2 Adaptive Systems

We plan to build up on the existing theoretical framework on QoS control, to develop research on adaptive systems. The challenge is to develop design techniques for adaptive systems meeting both critical and best-effort requirements. The theoretical problems to be studied include influence on performance of parameters such as granularity of control and uncertainty, finite state quotients of controllers, composability of controllers. Application driven work will focus on efficient implementation techniques, as well as specialization to memory and power management.

3.7.3 Program Verification and Monitoring

As a long term goal, we plan to develop a tool for model-checking programs. In order to be efficient and applicable to realistic programs such a tool has to combine static analysis, model extraction techniques and model-checking. In fact, Verimag has the necessary knowledge to develop such a tool, that would be comparable to Microsoft's SLAM tool.

[3] ISIS: http://www.isis.vanderbilt.edu/people/default.asp

Indeed, we were the first to propose predicate abstraction as a verification method and we have worldwide recognized skills in building model-checking algorithms and tools. A major challenge in building this tool is to be able to analyze programs with dynamic data structures. Our work on the verification of these programs is recent, and started when Radu Iosif joint VERIMAG in September 2003. Yes, the obtained results are encouraging and have been published in prominent international conferences such as Fossacs and SAS.

We plan to continue this work with the following objectives:

- Continue our basic research on the verification of this class of programs with the focus on logic- and automata-based techniques and their combination with abstract interpretation. In particular, we plan to focus more specifically on embedded software.
- Implement and experiment the developed verification techniques.
- Integrate this work into a software model-checker.

Program Monitoring We plan to study runtime monitoring of real-time systems continuing our recent work [122] on this subject. Dealing with such systems requires, on the one hand, high-level specification languages which are able to express timing constraints, and on the other hand, techniques to synthesize monitors from such specifications. There are two efficiency issues involved: (1) the efficiency of the monitor synthesis procedure, and (2) the efficiency of the monitor execution procedure. The latter is especially important since the monitor is executed in real-time with the monitored system. The former is crucial as well, in order to be able to synthesize a monitor in the first place.

Most existing monitoring techniques are "centralized", in the sense that there is a unique monitoring agent receiving all the runtime events from the monitored system. Having a single monitoring point can often be infeasible. This is especially true when the monitored system is distributed (e.g., the automated braking system of a modern vehicle, distributed on the four wheels). We plan to investigate *decentralized monitoring* techniques, where the task of a central monitor is replaced by a number of local monitors, each observing directly one node of the distributed system, and together checking a global property at runtime. We will identify both fundamental limitations of decentralized monitoring and applications where it is practically feasible. We have shown that some versions of the problem are undecidable whereas others are not [260]: when no communication or some type of unbounded-delay communication is assumed, the problem becomes undecidable, whereas when bounded-delay communication is assumed the problem becomes efficiently decidable. We would like to relate these limitations with classical (im)possibility results in distributed systems, such as consensus, clock synchronization, etc. No-one appears to have examined the problem under this angle yet.

3.7.4 Security

Verification of Tamper-resistant Security Protocols Our current work on the semantics and verification of security protocols provides guarantees against an adversary

who has only *black-box* access to the secret information of honest parties. From the implementation point of view, this corresponds to the assumption that honest parties execute their code on some secure hardware in which the protocol and secret data are stored, denying the adversary any access to this data other by exchanging messages. This assumption is not always satisfied. Indeed, known attacks show that it is hard to preserve the security of Smart-cards because of "covert channels" based on differential power analysis, differential fault analysis etc.... More generally, embedded systems such as GSM mobile phones, PDAs and Smart-cards are vulnerable for these types of attacks as they can become in the adversary's possession. Thus a natural step in our research is to investigate semantic models and verification techniques for security protocols that take into account this type of attacks.

An other direction of research, we plan to investigate concerns the verification of open-ended protocols. These are protocols where the number of participants can change dynamically, that is, participants may join or leave a running session. Also here, we are interested in developing automatic verification techniques and in proving soundness of the hypothesis of these techniques with respect to cryptographic assumptions.

Certified Applications We intend to pursue our work on developing an efficient tool-supported methodology for certified applications. The next step is to integrate our methodology and tools to a commercially available tool and to study its applicability on a realistic case study. We are currently working with Trusted Logic, Axalto and CEA-CESTI (an evaluation organism) to setup a project. From a research perspective, we plan to investigate the extension of our methodology to cover more sophisticated security policies.

Mobile Code Specification and Verification Our work on the specification and verification of mobile code is recent, and began when Jean-Francois Monin joint VERIMAG in September 2003. Our aim is to one hand to develop verification techniques for mobile code that take into account functional as well as security aspects. An other line of work concerns the semantic foundations of mobility. We plan to continue our work on proving the coherence of proposed models for mobility such as the distributed join-calculus.

3.8 Team Members

Permanent Members

S. Bensalem Professor, University Joseph Fourier

M. Bozga Engineer, CNRS
Ph.D. students: Abdelkarim Kerbaa, Ramzi Ben Salah
projects and contracts: OMEGA
participates in : PERSIFORM, AMETIST, DYNAMO
Marius Bozga is the main developer of the IF environment.

C. Ene Assistant Prof, University Joseph Fourier

J.-C. Fernandez Professor, University Joseph Fourier
Ph.D. students: J. Combaz, H. Venturini, C. Pachon
projects and contracts: POTESTAT
participates in : MODESTE

S. Graf Researcher, CNRS
projects and contracts: OMEGA, PERSIFORM, ASSERT, ARTIST2

R. Iosif Researcher, CNRS
projects and contracts: DYNAMO

Y. Lakhnech Professor, University Joseph Fourier; DCS team leader
Ph.D. students: M. Amrani, L. Gottely, R. Janvier, L. Mazaré
projects and contracts: MODESTE, PROUVÉ, ROSSIGNOL, EDEN
participates in : POTESTAT, DYNAMO, ARTIST2

J.-F. Monin Professor, University Joseph Fourier
Ph.D. students: Haixing Hu
projects and contracts: AVERROES, TYPES
participates in : PERSIFORM

L. Mounier Assistant Professor, University Joseph Fourier
Ph.D. students: C. Pachon, L. Samper (partially)
projects and contracts: POTESTAT
participates in : MODESTE, PERSIFORM, adhoc

M. Périn Assistant Professor, University Joseph Fourier
Ph.D. students: R. Janvier
projects and contracts: EDEN
participates in : MODESTE, POTESTAT, PROUVÉ

J. Sifakis Director of Research, CNRS; Lab. director
Ph.D. students: Ch. Nakhli, J. Combaz, A. Basu
projects and contracts: ARTIST, ARTIST2 (European Network of Excellence)

A. Akhavan Engineer
participates in : EDEN

Temporary Members

1. Ileana Ober, Research Engineer 2002-2004
   projects and contracts: OMEGA

2. Iulian Ober, Research Engineer 2002-2005
   projects and contracts: OMEGA, PERSIFORM

3. Youssef Bouzouzou, Engineer, 2002-03
   participates in : AGEDIS

3.9 PhD Theses

Past PhD Theses

1. Karine Altisen (1998-2001) *Application de la synthèse de contrôleurs à l'ordonnancement de systèmes temps-réel*; Joseph Sifakis

2. Liana Bozga (2000-2004) *Automatic Verification of Cryptographic protocols*; Yassine Lakhnech

3. Manuel Aguilar-Cornejo (1998-2003) *Improving Validation of Asynchronous Systems: Static Analysis for FIFO Reduction*; Susanne Graf

4. Lucian Ghirvu (1997-2002) *Automatic Test Generation of Conformance Test Cases for Telecommunication Protocols*; Jean-Claude Fernandez and Alain Kerbrat

5. Aurore Annichini Collomb (1997-2001) *Verification of Extended Automata using Symbolic Methods*; Ahmed Bouajjani

6. Gregor Gössler (1998-2001) *Compositional Modeling of Real-Time Systems — Theory and Practice*, Joseph Sifakis

7. Kai Baukus (1999-2003) *Abstraction-based verification of parameterized networks*, University of Kiel; Yassine Lakhnech

8. Ralf Huuck (1999-2003) *Software verification for programmable logic controllers*, University of Kiel; Yassine Lakhnech

9. Karsten Stahl (1999-2003) *Exploring the limits of parameterized system verification*, University of Kiel; Yassine Lakhnech

Current PhD Theses

1. Ananda Basu (2005-2008) ; Joseph Sifakis

2. Hugo Venturini (2004-2008) *Debugging Optimized C Code*; Jean-Claude Fernandez (partially)

3. Laurent Mazare (2003-2006) *Proving Opacity Properties for Cryptographic Protocols*; Yassine Lakhnech

4. Ramzi Ben Salah (2003-2006) *Timing Analysis of Large Systems and Circuits*; Oded Maler and Marius Bozga

5. Romain Janvier (2002-2005) *Models and Verification for Cryptographic Protocols in e-Commerce*; Yassine Lakhnech and Michael Périn

6. Abdelkarim Kerbaa (2002-2005) *Conditional Scheduling with Timed Automata*; Oded Maler and Marius Bozga

7. Jacques Combaz (2002-2005) *Dealing with Real-Time within the Compilation Flow of Embedded Systems*; Jean-Claude Fernandez and Joseph Sifakis

8. Cyril Pachon (2001-2005) *Model-based Approach for Robustness Testing*; Jean-Claude Fernandez and Laurent Mounier

9. Moussa Amrani (2001-2005) *Analysis and Verification of Java Programs through Abstraction and Model-checking*; Yassine Lakhnech

10. Chacker Nakhli (2001-2005) *Real-Time Java for Critical Embedded Systems*; Joseph Sifakis

11. Laurent Gotely (2001-2005) *Verification of Security Properties for Java programs*; Yassine Lakhnech (partially)

12. Ludovic Samper (2004-2007) *Modeling and analysis of ad-hoc sensor networks* Laurent Mounier and Florence Maraninchi

Chapter 4

Timed and Hybrid Systems Team

4.1 History and Background

The timed and hybrid systems team, which is the youngest at VERIMAG, was formed in 1998 by Oded Maler who worked mostly on hybrid systems, together with Sergio Yovine who completed his PhD at VERIMAG on developing KRONOS, one of the first verification tools for timed automata. The group received a significant reinforcement, especially with respect to the mathematical background underlying hybrid systems, when Eugene Asarin joined the team in 1999 as a professor. Stavros Tripakis, who did his PhD at VERIMAG, also on verification tools for timed automata joined the team in 2000, followed by Thao Dang, who did her thesis at VERIMAG on the hybrid verification tool **d/dt**, who joined in 2001. Eugene Asarin left for Paris in 2003 and the most recent member is Christophe Rippert (2005) who works on component-based embedded operating-systems kernels.

As its name suggests, the group started its life focusing on timed and hybrid extensions of discrete models, where it was one of the pioneers in verification algorithms and tools for these models. The group organization is more in the spirit of American universities where each member is considered as an autonomous researcher free to choose his research directions. The group has gained a worldwide reputation, both in fundamental theoretical results, scientific administration and tool development. The strong international visibility of the group is manifested by numerous publications in top international journals and conferences, participation in and coordination of numerous European and national projects and close ties with leading US and European universities.

4.2 Current Work Directions

Below we give a short description of the research themes on which members of the group work. The list is far from being exhaustive and additional information can be found in the corresponding thematic chapters and sections, most notably Section 7.2.2 and Chapters 8 and 10.

4.2.1 Verification and Synthesis for Hybrid Systems

As elaborated in Chapter 10 the group is one of the pioneers in adapting verification techniques to deal with continuous and hybrid systems using methods that combine graph algorithms, numerical analysis and computational geometry to approximate tubes of trajectories. Ongoing work includes investigation of novel ways to perform continuous verification, such as new search techniques, new geometric objects and data structures as well as new approximation and abstraction techniques. We have recently finished the successful European project CC and based on the lessons gained in it, we plan to launch in the coming years a "second-generation" attack on this topic with the ambitious goal of scaling-up hybrid verification techniques to the point of treating problems of industrial significance. Application domains for this research theme range from the "traditional" control engineering via the new promising domain of analog circuits down to physical modeling in general. O. Maler and T. Dang are working actively on this theme while E. Asarin and S. Yovine contributed to it in the past. Additional activities related to continuous and hybrid systems are described in Sections 10.2 and 10.3.

4.2.2 Testing and Monitoring for Timed and Hybrid Systems

This research theme aims at studying the problems related to the generation and evaluation of test-cases for systems admitting real-time and continuous components. This topic evokes many interesting conceptual and technical questions including the role of time passage as an observable, the determinization of timed automata on-the-fly, the development of suitable formalisms for specifying properties of dense-time and real-valued signals as well as coverage and similitude metrics for continuous signals and systems. Applications include testing and lightweight verification of real-time software and analog circuits (the PROSYD project). S. Tripakis, T. Dang and O. Maler work on these topics. More details can be found in Sections 7.2.2 and 10.5.

4.2.3 Advanced Techniques for Embedded Compilation

We are developing an automated model-based design, analysis, and implementation framework for synthesizing correct-by-construction code. Special emphasis is put on multi-threaded software and multi-processor architectures. Our framework is grounded on well-established notions such as program analysis and transformation, refinement and scheduler synthesis. It consists in (1) a formal language which provides platform-independent constructs to specify the behavior of an application using an abstract execution model, and (2) a compilation chain for refining the application abstract model into its concrete implementation on a target platform. Our tool provides a pre-industrial, extensible, and customizable compilation chain for generating code without semantic break-downs. We are currently working on (1) applying our framework to other industrial applications, (2) strengthening the integration into STMicroelectronics' FlexCC2 compiler technology, and (3) generating code for other platforms such as SystemC for early simulation. S. Yovine

and Ch. Rippert are currently working on this theme; past contributors include the post-docs Ch. Kloukinas and Ph. Gerner. More details are given in Section 8

4.2.4 Fighting the State Explosion in Timing-related Problems

The expressive power of timed automata can be used in a variety of application domains, including timing analysis of software and hardware and the solution of complex dynamic scheduling problems. Albeit the decidability that guarantees solvability "in principle", and despite a lot of algorithmic effort invested in verification tools in VERIMAG and elsewhere, typical problems are often orders of magnitude larger than what the best current tools can handle. A large part of our work consists in struggling with this problem by various means. For scheduling problems we have worked a lot on heuristic search algorithms for finding strategies for problems of scheduling under uncertainty. For circuits we are developing new interesting abstraction techniques that hide internal clocks (this work is supported by the FVTA grant from Intel). As a more general solution we have developed a new and powerful SAT solver for difference logic, the logic that corresponds to timed systems, and had some promising results on scheduling and circuit analysis problems. These activities are part of the AMETIST project and are conducted by O. Maler with help with M. Bozga of the DCS team who is responsible for the IF toolset. Previous contributors to this effort include E. Asarin and, of course, S. Yovine and S. Tripakis who developed the first versions of Kronos, upon which IF is founded. These efforts are described in Section 10.6.

4.2.5 Model-based Design and Implementation of Embedded Systems

Model-based design is being established as an important paradigm for the development of contemporary embedded systems. This paradigm advocates using models all the way from design to implementation. Using models, rather than, say, building and testing prototypes, is important in order to cut development cost and time. However, using models alone is not enough. Being abstractions of reality, models often make "idealizing" assumptions, which break down during implementation. Thus, it is necessary to bridge, somehow, the gap between high-level models and low-level implementations. This research direction aims at doing precisely that. We are working with high-level models such as Simulink/Stateflow, the synchronous language Lustre, or even timed automata. We target a variety of execution platforms and programming styles, including mono- or multi-threaded programs, RTOSs with or without preemptive scheduling mono- or multi-processor platforms linked with time-triggered or other communication networks. This research is led by Stavros Tripakis together with Paul Caspi from the "Synchrone" team and is described in Section 8.

4.3 Distinguished Collaborations

The group has intensive relations with researchers from top US universities such as Berkeley, Penn, CMU and NYU where team members have spent long periods as post-docs

and visiting professors. Among the French collaborators we mention the LIAFA and LSV laboratories. Within the European projects we have collaborated with partners coming from control (ETH Zurich, CWI, Dortmund, Lund) as well as verification (Weizmann, Aalborg, Nijmegen, Twente and Trento) and system implementation (Vienna). As for industrial collaboration, the major current partner is ST Microelectronics (mainly within the ANACONDA project on compilation but also on analog verification within PROSYD). Other industrial collaborators or sponsors include Audi, TTTech and Esterel Technologies (within NEXT-TTA and RISE), IBM (PROSYD) and Intel (FVTA).

4.4 Perspectives for Future Work

As one can see, the group is involved in a large number of diverse topics, much beyond its size as measured by permanent researchers. In terms of research themes, two members of the team have been working recently on new research directions related to the namely the implementation of embedded software. The work of S. Yovine on property-aware compilation is done independently of the rest of the team, except for the newly recruited Ch. Rippert and the dedicated post-docs and contractual engineers. The work of S. Tripakis on model-based development of embedded control software is done in collaboration with P. Caspi from the "Synchrone" team.

These changes of orientation, together with the uncompensated departure of E. Asarin, reduced significantly the resources dedicated to hybrid systems research and to application-independent research on timed automata. Recruiting young researchers who are fluent in control and applied mathematics and who do not shy away from algorithmics and programming is crucial for maintaining our position in the hybrid arena. The importance of this position is emphasized now when Verimag approaches more "physical" disciplines.

We conclude this chapter with a wish list describing some of the (ambitious) goals we want to see accomplished four years from now.

1. For hybrid systems analysis we hope that our research effort will bring us to the point of being able to handle systems with dozens of continuous variables, a fact that will facilitate the acceptance of reachability analysis techniques in industry and in the physical sciences

2. The main challenge in property-aware compilation is fully automated code synthesis for complex applications (e.g., H.264 encoding) and architectures (e.g., Network-on-Chip), involving time, memory and energy constraints. We also envisage complete integration and transfer into an industrial design flow.

3. For the verification and synthesis based on timed automata models, we hope to see a performance breakthrough that will allow us to treat automata with hundreds of clocks and state variables. This will lead to proliferation of such techniques to scheduling and timing analysis

4. For model-based development of embedded control software, we will continue our efforts to develop a complete, end-to-end tool chain from high-level models all the

way down to executable code. Our main short-term goal is to complete our results by adding to the picture more execution platforms (mostly distributed, with communication infrastructure such as CAN or FlexRay), more application types (mostly multi-threaded, under a preemptive RTOS, both event-triggered and time-triggered) and more high-level models (e.g., timed or hybrid automata). The longer-term goal is to develop an implementation theory for mixed continuous-discrete systems that run on heterogeneous architectures.

5. We believe analog verification to be a potential driving force for hybrid systems research, as did digital circuits for formal verification. We hope to see the emergence of an active community around this topic. Transferring our monitoring tool for analog signals to industry with the help of the PSL label will be considered a success.

4.5 Team Members

Permanent Members

Eugene Asarin, Professor, UJF (1999-2003)
Contractual responsibilities: MASH
Contribution to projects: CC, AMETIST
Thao Dang, Researcher, CNRS
PhD supervision: T. Nahhal, A. Donze
Contractual responsibilities: CORTOS
Contribution to projects: CC, PROSYD
Oded Maler, Director of Research, CNRS
PhD supervision: R. Ben Salah, S. Cotton, A. Donze, A. Kerbaa, D. Nickovic
Contractual responsibilities: CC, AMETIST, FVTA, PROSYD
Christophe Rippert, Assistant Professor, INPG (since 2005)
Contribution to projects: ANACONDA, MADEJA, NEVA.
Stavros Tripakis, Researcher, CNRS
PhD supervision: N. Abed, A. Curic, M. Krichen, C. Sofronis
Contractual responsibilities: NEXT-TTA, CORTOS, RATP, Service Networks
also participates in : RISE
Sergio Yovine, CNRS researcher)
PhD supervision: I. Assayad, G. Salagnac
Contractual responsibilities: ANACONDA, DYNAMO, MADEJA, MASH, NEVA.

Temporary Members

Philippe Gerner, Post-doc (2004-2005)
Christos Kloukins, Post-doc (2002-2004)
Francois-Xavier Défaut, Engineer (2004-2005)
Olivier Quévreux, Engineer (2004-2005)

4.6 PhD Theses

Past Theses

1. Catalin Dima (2000-2001) *An Algebraic Theory of Real-time Formal Languages*; Eugene Asarin and Oded Maler

2. Yasmina Abdeddaim (1998-2002) *Scheduling with Timed Automata*; Oded Maler and Eugene Asarin

3. Moez Mahfoudh (2000-2002) *On Satisfiability Checking for Difference Logic*; Oded Maler and Eugene Asarin

4. Gerardo Schneider, (1998-2002) *Algorithmic Analysis of Hybrid Systems*; Sergio Yovine and Eugene Asarin

5. Marcelo Zanconi, (2001-2004) *Modeling and Analysis of Real-time Systems with Preemption, Uncertainty and Dependencies*; Sergio Yovine

Current PhD Theses

1. Adrian Curic (2001-2005) *Implementing Lustre Programs on Distributed Platforms with Timing Constraints*; Stavros Tripakis and Paul Caspi

2. Abdelkarim Kerbaa (2002-2005) *Conditional Scheduling with Timed Automata*; Oded Maler and Marius Bozga

3. Ismail Assayad (2003-2006) *Automated Scheduler Synthesis for Multiprocessor Embedded Applications*; Sergio Yovine

4. Ramzi Ben Salah (2003-2006) *Timing Analysis of Large Systems and Circuits*; Oded Maler and Marius Bozga

5. Christos Sofronis (2003-2006) *Embedded Code Generation from High-level Models*; Stavros Tripakis and Paul Caspi

6. Alexandre Donze, 2003-2006 *Optimal Control of Hybrid Systems*; Oded Maler and Thao Dang

7. Scott Cotton (2005-2006) *SAT Solving Modulo Theories*; Oded Maler

8. Moez Krichen (2004-2007) *Black-box Conformance Testing for Real-time Systems*; Stavros Tripakis

9. Tarik Nahhal (2004-2007) *Test Generation for Continuous and Hybrid Systems*; Thao Dang

10. Dejan Nickovic, (2004-2007) *Monitoring Real-time Temporal Properties*; Oded Maler

11. Guillaume Salagnac (2004-2007) *Dynamic Memory Allocation for Real-time Java*; Sergio Yovine

12. Nazha Abed (2005-2008) *Systematic Exploration of Large State Spaces*; Stavros Tripakis

Part II

Research Activities

Chapter 5

Component-based Modeling and Development of Heterogeneous Systems

A central idea in systems engineering is that complex systems are built by assembling components, or building blocks. Components are systems characterized by their interface, an abstraction that is adequate for composition and re-use. It is possible to get large components by "gluing" together simpler ones. "Gluing" can be considered as an operation on components.

Component-based frameworks and tools have been developed for general-purpose software, and some of them have very successful commercial implementations (.COM, .NET, EJB, etc.).

In the context of embedded systems, there is no such widely accepted framework. The strong constraints that have to be taken into account when developing an embedded system make it very difficult to use a general-purpose component framework. The ideal tool for component-based development and reuse of embedded systems will offer support for expressing and evaluating non functional properties (like timing, or power consumption), for formal verification of individual components and compositions of components, etc. There is still a lot to be done before such a framework can be used on industrial cases.

This chapter presents several complementary approaches for modeling the kind of heterogeneous components that are needed for the development of complex embedded systems. One approach is to develop new semantical models for components and their compositions. We present theoretical results, and the way they are supported in the IF toolbox. Another approach is to start from a well established and sufficiently general semantical model (namely, the synchronous setting), to apply it to practical problems in component design.

These approaches may be used for early execution, simulation and validation, but also as a basis for correctness-preserving implementations.

Heterogeneity is a key issue. A formal component-based framework for embedded systems should be usable for components that are heterogeneous in all the senses below:

- Heterogeneity of *interactions* between components, meaning that the interactions

that can be *atomic* or *non atomic*.

- Heterogeneity of *executions* of the global system, in which components may run synchronously, asynchronously, or by conforming to one of the intermediate models like GALS (Globally-Asynchronous-Locally-Synchronous) or quasi-synchronous systems.
- Heterogeneity of *abstraction levels*, meaning that one cannot expect a uniform level of abstraction in the component descriptions, if the components may be application software parts, hardware parts, physical environment models, etc. For instance, application software has a very abstract notion of time, while hardware architectures can be given precise timing descriptions.
- Heterogeneity of *development level*, meaning that it may be very interesting to validate a complex system before all the components have been completely designed. In this case, some of the components are deterministic executable programs or hardware parts, and others are given by non-deterministic specifications.
- Heterogeneity between the *computers and their environments*, because some components describe parts of a numerical computer system, and some other components model the behavior of the physical embedded system's environment. The latter may be discrete or continuous models.

Section 5.1 presents a general semantical framework of components and component composition. Heterogeneity of interactions, executions, and abstraction levels is dealt with by enforcing the separation between component implementation and component composition. This is done by considering three separate layers: behavior (which may be specified in any form); interaction (based on a generalization of n-ary rendez-vous); and execution (specified by means of dynamic priority rules). The strong point of this framework is the strict separation between the validation of the components correctness and the validation of the system correctness. Global safety properties reduced to deadlock freedom, and sufficient conditions are proven, allowing ensuring deadlock freedom of the system from (abstractions of) local behaviors and the interaction layer.

Section 5.2 presents the IF language on which is based the IF toolset presented in Section 7.1.1. The IF language has been introduced with a similar aim as the framework of the first section. It can be seen as an instance of this framework in the sense that the tool implementing its semantics clearly separates local behavior, interaction and execution. IF plays the role of a structured intermediate format between user-level modeling languages (such as SDL or UML) and the semantic models used by validation tools. It has been used for providing semantics to such formalisms and extensions of them, in particular real-time extensions. A strong point is the implementation, including abstractions at the syntactic level and a simulation engine with a set of validation tools ranging from random execution to exhaustive verification and test case generation (see section 7.1.1).

Section 5.3 presents an approach for the development of heterogeneous component-based embedded systems based on the following principles. The kind of heterogeneity we are interested in are the following: 1) we consider hardware, software or operating system parts of the computer system, either fully developed and thus executable, or available only as early non-deterministic specifications; 2) we also consider some non-deterministic models of the physical environment of the computer system. We adopt the synchronous semantical basis, and we define a notion of heterogeneous component in this setting. The

synchronous model is powerful enough to represent synchronous or asynchronous compositions, and all their variants (quasi-synchronous systems, Globally-Asynchronous-Locally-Synchronous systems, etc.) In order to get hints on the notion of component that is really needed, we study problems from three application domains: critical control avionics software, ad-hoc sensor networks, systems-on-a-chip. We aim at providing a development environment offering: 1) domain-specific high level languages for programming and specifying the computer system components, and for modeling the environments; 2) powerful static analysis methods; 3) implementation tools.

5.1 A Framework for Component-based System Construction

5.1.1 Introduction - Key issues

A central idea in systems engineering is that complex systems are built by assembling components (building blocks). Components are systems characterized by their interface, an abstraction that is adequate for composition and re-use. It is possible to get large components by "gluing" together simpler ones. "Gluing" can be considered as an operation on components.

The system construction problem can be formulated as follows. For a given property P, build a component c satisfying P, from a given set of atomic components C_a and a set GL of operators on these components. The component c to be constructed, can be considered as a term of the term algebra generated from C_a and GL. There exist only a few algebraic frameworks for formalizing system construction problems, such as boolean algebra for logical circuits, Hoare logic for programs, process algebras with modal logic e.g., HM logic. At the same time, most of the existing results about protocols and distributed algorithms define solutions to construction problems for component-based systems, meeting specific properties. For example, a token ring protocol guarantees mutual exclusion between interacting components. As a rule, construction problems can be formulated as highly intractable synthesis problems.

Existing frameworks for studying component-based construction fail to adequately treat at least one of the following important requirements:

- Encompass heterogeneous composition to ensure interoperability of components;
- Provide results guaranteeing correctness-by-construction for essential system properties such as deadlock-freedom, progress and liveness, in order to reduce a posteriori validation as much as possible.

We developed a framework meeting the above requirements for the following particular instance of the construction problem:

For a given a set of atomic deadlock-free components from C_a, find "glue" such that the resulting system is deadlock-free and meets a given safety property P.

Heterogeneity

System designers deal with a large variety of components, each having characteristics of different types. A central problem is "meaningful" composition of such components to ensure that they interoperate correctly. We need semantic frameworks encompassing heterogeneous composition. There exist three specific sources of heterogeneity: interaction, execution and abstraction.

5.1.1.1 Heterogeneity of Interaction

Interactions are combinations of actions performed by system components to achieve a desired global behavior. Interactions can be *atomic* or *non atomic*. For atomic interactions, the state change induced in the participating components cannot be altered through interference with other interactions. Synchronous languages and hardware description languages use atomic interactions. On the contrary, languages with buffered communication (SDL) or multi-threaded languages (Java, UML), generally have non atomic interactions.

Both types of interactions may involve *strong* synchronization or *weak* synchronization. Interactions involving strong synchronization, can occur only if all the participating components agree, for instance interaction via rendezvous involves strong synchronization (and is atomic). Interactions involving weak synchronization are asymmetric and require only the participation of an initiating action that may synchronize with other actions, for instance, "outputs" in synchronous languages. Finally, interactions can be static or dynamic.

5.1.1.2 Heterogeneity of Execution

Currently, there exists no formalism jointly encompassing both synchronous and asynchronous execution. Synchronous execution is typically used in hardware, synchronous languages and time triggered systems. It considers that a system execution is a sequence of steps. It assumes synchrony, meaning that the environment does not change during a step, or equivalently "that the system is infinitely faster than its environment". In each execution step, all the system components contribute by executing some "quantum" computation. The synchronous execution paradigm has a built-in strong assumption of fairness: in each step all components can move forward. Asynchronous execution does not use any notion of a global computation step. It has been adopted in most distributed system description languages such as SDL and UML and programming languages such as ADA and Java. The lack of built-in mechanisms for sharing computation between components can be compensated through scheduling. This paradigm is also common to all execution platforms supporting multiple threads and tasks.

5.1.1.3 Heterogeneity of Abstraction

System development involves the use of languages, models and physical implementations representing a system and its components at different abstraction levels. For heterogeneity,

a key abstraction is the one relating an application software to its implementation on a given platform.

Application software is *untimed* in the sense that it makes abstraction of physical time. The only references to physical time are time parameters of real-time statements such as timeouts and watchdogs. The expiration of watchdogs or timeouts is treated at semantic level as an external event.

An application software running on a given platform, is a *timed* system. The set of its state variables include not only the variables of the application software but also all the variables needed to characterize its dynamic behavior such as time, quantity of resources e.g., memory and power.

We need abstractions and theory relating application software to its implementations. An inherent difficulty in the definition of these abstractions is the lack of results relating functional to non functional properties.

5.1.2 Results

Our framework for heterogeneous system composition allows us to consider architectures and their elements as first class entities independent from the components' behavior. Such a decoupling between behavior and architecture is essential for correctness-by-construction theory depending mainly on architecture properties.

The framework introduces the notion of *glue* operators which transform sets of components into new components. We studied two independent classes of such operators.

- Interaction models which provide a general mechanism for modeling the interactions between a set of components.
- Priority orders which provide a general mechanism for restricting the behavior of a set of components by preserving deadlock-freedom.

For the two classes of operators, we studied results guaranteeing correctness-by-construction.

5.1.2.1 Glue Operators and their Properties

Notation: For a given vocabulary of actions A, we denote by B_i the *behavior* of transition systems on disjoint sets of actions $A_i \subset A$.

B_i is a set of transitions of the form (s, a, s') where $a \in A_i$ and $s, s' \in S_i$, the set of the states of B_i.

$GL = \{gl_i\}_i$ is a set of *glue operators* with a type function τ such that $\tau(gl_i)$ is a subset of A.

The following syntax defines the set of the components, by using an auxiliary function act which gives for a component c, the set of its actions $act(c)$:

- $c = gl\{B_1, \ldots, B_n\}$ is an atomic component, if $\tau(gl) = act(c)$, where $act(c) = A_1 \cup \ldots \cup A_n$;

- $c = gl\{c_1, \ldots, c_n\}$ is a component if $c_1, \ldots, c_n$ are components such that $\tau(gl) = act(c)$ where $act(c) = act(c_1) \cup \ldots \cup act(c_n)$, if the sets $act(c_1), \ldots,$ $act(c_n)$ are disjoint.

As in process algebras, components are characterized by their behavior and their structure. The behavior of a component is defined by considering that each glue operator transforms sets of transition systems into transition systems e.g., by using operational semantics.

To simplify the structure of components, we introduce on glue operators, a partial idempotent *composition operation* $\oplus$ such that $(GL, \oplus)$ is a commutative monoid. Furthermore, for $gl, gl' \in GL$, we take $\tau(gl \oplus gl') = \tau(gl) \cup \tau(gl')$ and we assume that the following property is satisfied:

$gl(U \cup\ gl'(U')) = gl \oplus gl'(U \cup\ U')$, where U, U' are sets of components.

This is a kind of "generalized" associativity property which allows separation between behavior and structure, by reducing any component into a term of the form $gl\{B_1, \ldots, B_n\}$. Such a separation is instrumental for considering structure as first class entity.

We present results related to the two classes of glue operators and their combination.

Interaction Models and their Properties Given an action vocabulary A, an *interaction* of A is any non empty subset of A. Interactions correspond to sets of compatible actions. Two interactions are incomparable if none of them is contained in the other. Interaction models are glue operators characterizing the interactions of a system of interacting components.

An *interaction model* im on an action vocabulary A, is a pair $im = (\Gamma, \Delta)$ where:

- Γ is a set of incomparable interactions of A called *connectors* of im;
- Δ is a set of incomparable interactions of A such that any interaction of Δ is contained in some interaction of Γ. Δ is the set of the minimal *complete interactions* of im.

The interactions defined by an interaction model im are contained in some connector of Γ. Furthermore, we require that either they contain some interaction of Δ or they are maximal. This mechanism allows to characterize both strong and weak synchronization. For strong synchronization, only the interactions corresponding to connectors are possible. For weak (asymmetric) synchronization, minimal possible interactions are defined by elements of Δ. In particular, this covers the case where an output can interact with a set of inputs. Possible interactions are all those containing the output; the singleton containing the output is an element of Δ.

Interaction models can be considered to be operators on transition systems. The term $im\{B_1, \ldots, B_n\}$ defines a transition system such that if (s_i, a_i, s'_i) is a transition of B_i , for $i = 1 \ldots n$, and $\{a_1, \ldots, a_n\} \in im$ then
$((s_1, \ldots, s_n), \{a_1, \ldots, a_n\}, (s'_1, \ldots, s'_n))$ is a transition of $im\{B_1, \ldots, B_n\}$.

Given interaction models $im_i = (\Gamma_i, \Delta_i)$ for $i = 1, 2$, the operation $\oplus$ is defined by $im_1 \oplus im_2 = im = (\Gamma, \Delta)$ such that $\Gamma = max(\Gamma_1 \cup \Gamma_2)$ and $\Delta = min(\Delta_1 \cup \Delta_2)$ where max and min, respectively, are the functions giving the set of the maximal and minimal, respectively, elements of their arguments.

Clearly, the operation is associative. The set of the connectors of the resulting interaction model contains all the connectors except those contained in some connector of the union of the connector sets. The dual operation is applied for minimal complete interactions.

In this framework, it is possible to define a generalization of *parallel composition*. Consider a component $im\{B_1, \ldots, B_n\}$, such that $im = (\Gamma, \Delta)$ and the B_i, are transition systems on disjoint vocabularies of actions A_i, where $A = \cup_{1 \leq i \leq n} A_i$. This component can be obtained as a composition of simpler components.

For $A' \subseteq A$, define the interaction model $im[A'] = (\Gamma', \Delta')$ where $\Gamma' = \{\gamma \in \Gamma | \gamma \subset A'\}$ and $\Delta' = \{\delta \in \Delta | \delta \subset A'\}$. This interaction model includes exactly all the interactions of im involving only actions from A'.

We show that $im\{B_1, \ldots, B_n\}$ can be obtained as the composition of the components $im[A_i]B_i$ for $i = 1 \ldots n$. We define for a set $\{A'_1, \ldots, A'_k\}$ of disjoint subsets of A, the interaction models $im[A'_1, \ldots, A'_k] = (\Gamma', \Delta')$ such that:

- Γ' consists of all the connectors γ obtained by assembling together one connector contained in each set A'_i. That is, $\gamma = \gamma_1 \cup \ldots \cup \gamma_k$ where $\gamma_i \in \Gamma'_i$, the set of the connectors of $im[A'_i]$.

- Δ' contains the subset of the elements of Γ' which are complete interactions of Δ.

We give below instances of the general results presented in [57, 189].

- $im[A_1, A_2]\{im[A_1]\{B_1\}, im[A_2]\{B_2\}\} = im[A_1, A_2] \oplus im[A_1] \oplus im[A_1]\{B_1, B_2\}$
  This is an expansion theorem for parallel composition between $im[A_1]\{B_1\}$ and $im[A_]\{B_2\}$. $im[A_1, A_2]$ is a parallel composition operator parameterized by the interaction model including all the interactions between A_1 and A_2

- $im[A_1 \cup A_2, A_3]\{im[A_1, A_2]\{im[A_1]\{B_1\}, im[A_2]\{B_2\}\}, im[A_3]\{B_3\}\} =$
  $im[A_1, A_2 \cup A_3]\{im[A_1]\{B_1\}, im[A_2, A_3]\{im[A_2]\{B_2\}, im[A_3]\{B_3\}\}\} =$
  $im[A_1 \cup A_2 \cup A_3]\{B_1, B_2, B_3\}$
  This is a kind of general associativity result for parallel composition parameterized by interaction models. It allows incremental construction of components. The two first terms correspond to two different ways of building the component $im[A_1 \cup A_2 \cup A_3]\{B_1, B_2, B_3\}$ by using binary parallel composition.

- $im[A_1 \cup A_2 \cup A_3] =$
  $im[A_1] \oplus im[A_2] \oplus im[A_3] \oplus im[A_1, A_2] \oplus im[A_2, A_3] \oplus im[A_1, A_3] \oplus im[A_1, A_2, A_3]$
  This is a canonical decomposition of the operator $im[A_1 \cup A_2 \cup A_3]$ into disjoint interaction models.

Priority Orders and their Properties Given a vocabulary of actions A, a priority order on A is a strict partial order $pr \subseteq A \times A$.

For a transition system B on a vocabulary A, $pr\{B\}$ represents the transition system which behaves as B with the difference that if from a state of B many different actions are enabled, only the ones which are maximal in pr can be executed in $pr\{B\}$.

Given two priority orders pr_1 and pr_2, we define a composition operator $\oplus$ such that $pr_1 \oplus pr_2$ is the least strict partial order such that $pr_1 \cup pr_2 \subseteq pr_1 \oplus pr_2$, if such a partial order exists.

We have extensively studied the use of priority orders as a means to restrict the behavior of a set of components, so as to meet a given property P. In [42] we show that priority orders provide a general framework for modeling and composing scheduling policies. In [190] we generalize these results in the following manner.

Given a system consisting of a set of interacting components and a global safety property P of this system, it is possible to define controllers (e.g., by fixpoint characterization) which interact with the system so that the controlled system satisfies P and is deadlock-free. We have characterized the behavior of the controlled system by a class of functions that restrict the enabling conditions of the interactions of the initial system. The main result is that priority orders characterize exactly the class of the restrictions induced by safe and deadlock-free controllers. That is, they are expressive enough to describe any solution to the system construction problem. A consequence of this result is that mutual exclusion properties can be modeled by using priorities, if they preserve deadlock-freedom. This leads to interesting techniques for correctness-by-construction.

Another class of results deals with composability of restrictions induced by priority orders. We have shown that in general, $pr_1\{pr_2\{B\}\} \neq pr_2\{pr_1\{B\}\}$ and $pr_1 \oplus pr_2\{B\} \subseteq pr_1\{pr_2\{B\}\}$. That is, the restriction by $pr_1 \oplus pr_2$ is stronger than the restriction by successive application of the two priority orders. We have shown that equality holds if and only if $pr_1 \cup pr_2 = pr_1 \oplus pr_2$.

As we consider priority orders to be glue operators, we use the axiom $pr_1 \oplus pr_2\{B\} = pr_1\{pr_2\{B\}\}$ by assuming that only flattened terms represent transition systems.

The Layered Component Model

We have developed a modeling methodology which considers that components consist of three distinct layers [57]. The bottom layer includes a set of transition systems modeling behavior. The intermediate layer is an interaction model and the upper layer is a priority order.

The principle of layered modeling has been applied in the IF toolset. Layered components can be represented by using composite glue operators of the form $< pr, im >$. We extend the composition operation $\oplus$:
$< pr_1, im_1 > \oplus < pr_2, im_2 > = < pr_1 \oplus pr_2, im_1 \oplus im_2 >$.

The combined use of interaction models and priorities confers numerous advantages. It can be shown through examples that only using interaction models to describe components' coordination may lead to cumbersome solutions. There are coordination problems

for which priority orders are more appropriate e.g., scheduling problems, while for other problems the use of interactions is instrumental for getting simple solutions e.g., data flow problems. Another advantage is the existence of results guaranteeing by construction generic properties (see next paragraph).

5.1.2.2 Correctness-by-Construction

A posteriori validation techniques based on flattening the system structure have well-known limitations. We need theory, methods and tools for establishing, by construction, overall system correctness from the properties of its components. The idea of building systems that are correct-by-construction is much more common than one would believe. All the proven algorithms, protocols and architectures provide recipes for building correct systems. Nevertheless, these are for very specific properties such as lossless message transmission, clock synchronization etc. and they involve more or less sophisticated construction principles. We need lightweight theory for establishing generic system properties such as deadlock-freedom or progress of integrated components.

Two types of rules are necessary for correctness-by-construction:

- *Compositionality rules* which allow inferring global system properties from properties of its components. That is, if for $i = 1, \ldots, n$ a set of components c_i satisfy properties P_i then it is possible to guarantee that the component $gl(c_1, \ldots, c_n)$ satisfies some property $\tilde{gl}(c_1, \ldots, c_n)$ where $\tilde{gl}$ is an operator on properties depending on gl.

- *Composability rules* which allow inferring that a component's properties are not affected when its structure is modified. That is, if components $gl\{c_1, \ldots, c_n\}$ and $gl'\{c_1, \ldots, c_n\}$ satisfy respectively properties P and P' then the component $gl \oplus gl'\{c_1, \ldots, c_n\}$ satisfies $P \wedge P'$. Composability means stability of component properties across integration. Property instability phenomena are currently poorly understood e.g., feature interaction in telecommunications, or non composability of scheduling algorithms.

In [188], we provide sufficient conditions for deadlock-freedom of a layered component of the form $< pr, im > \{B_1, \ldots, B_n\}$ from deadlock-free atomic components. We assume that for the behavior B_i of each atomic component, a deadlock-free invariant is given. That is, a set of states from which the component can be blocked only because of its environment. The conditions relate deadlock-free invariants of the atomic components to the enabling conditions of the interactions of $< pr, im >$. These are computed by analysis of the structure of the operator $< pr, im >$, represented as a bipartite dependency graph relating actions of atomic components to interactions of im.

We also provide composability results, sufficient conditions for individual deadlock-freedom and liveness of the atomic components which are integrated in $< pr, im > \{B_1, \ldots, B_n\}$. The atomic component $im\{B_i\}$ is individually deadlock-free (respectively live) if it is always possible (respectively always inevitable) to perform some action of A_i in $< pr, im > \{B_1, \ldots, B_n\}$. The conditions require that atomic components have a set of states from which they are both deadlock-free and it is possible (respectively inevitable) to enable interactions with other components.

5.1.3 Perspectives for Future Work

The developed framework and the underlying modeling methodology have been successfully applied in the IF toolset as well as in the Expresso project. We believe that the existing results and their application provide evidence that they adequately address many key issues in component-based systems modeling. We plan to continue this work in the following directions:

- Complete the formalization of the general framework for layered components by studying the laws of the corresponding term algebra. An interesting class of problems in this context, is expressivity of different sub-algebras obtained by using classes of glue operators. For instance, which construction problems can be solved by using only binary interactions and weak synchronization?

- Continue the study of correct-by-construction techniques especially to make their application more incremental and general. Their extension in order to deal with any safety property is possible, in principle. We need tractable techniques adapted to specific classes of properties e.g., mutual exclusion.

- Develop a general infrastructure supporting the framework and extending the functionality offered by the IF toolset. Marius Bozga has developed a prototype in C++ that is currently under evaluation. The prototype supports any kind of interaction model and dynamic priorities. The objective is to show that the framework can serve as a general "semantic middleware" that allows implementing system description languages.

5.2 The IF Intermediate Format for Modeling Component Systems

IF has been introduced as a structured intermediate format between user level modeling languages (such as SDL [IT03] or UML [OMG03b]) and the semantic models used by validation tools and as an exchange format between validation tools.

The concepts of IF have been chosen in order to allow a structural representation of models, that is including an explicit notion of parallel composition of time and of communication, so as to allow,

- the mapping of user level modeling languages without compromising validation by introducing unnecessary state explosion, in particular, by preserving concepts which can be exploited for making verification more efficient

- playing the role of a model exchange format between a large number of validation tools, meaning that it should contain the concepts used in these tools; indeed, the input formats of tools like KRONOS [Yov97], Lash [WB98] or TREX [102] are sublanguages of IF.

IF has been used successfully for providing semantics to such formalisms and extensions of them, in particular real-time extensions (see section 5.2.3 below) and for carrying out case studies in which systems have been modelled using user level modeling languages or directly IF itself and then validated using the IF tool-set (see section 5.2.4 below).

5.2.1 The IF Language and its Semantics

In [BFG+99] we introduced a first version of the IF notation for representing systems of components. We introduced the main concepts, where many of them have been taken from modeling formalisms based on communicating state machines, such as SDL: a system is represented by a set of processes (or components) running in parallel and interacting via asynchronous signals stored in message buffers, where processes are represented by timed automata extended with data and actions.

More recently, in [143, 274], we have introduced some additional features to increase the expressivity and the flexibility of the framework. The new features concern

- the possibility to dynamically create process instances
- richer language for the description of the behavior associated with a transition, including the possibility to include external code.
- dynamic priority rules on processes for explicitly reducing non determinism due to concurrency which allows the expression of general execution modes and/or scheduling policies.
- an explicit notion of *resource* for the definition of mutual exclusion constraints

The distinguishing features of IF are that it allows mixing a detailed structured behavioral description of a system with timing constraints and that it allows restricting the a priori completely asynchronous execution through two types of priorities: *urgency* for expressing priority between system and time progress and inter process priorities.

Time Furthermore, the behavior of a process may be subject to timing constraints expressed using a set of clocks which can be reset and whose values increase over time. The time model of IF is that of *timed automata with urgency* [BST98, BS00] where the execution of a transition is an *event* defining an *instant* of state change, whereas time is progressing in states. Urgency is expressed by means of an *urgency* attribute of transitions. This attribute can take the values *eager*, *lazy* or *delayable*. *Eager* transitions are executed at the point of time at which they become enabled - if they are not disabled by another transition. *Delayable* transitions cannot be disabled by time progress. *Lazy* transitions may be disabled by time progress.

Dynamic Priorities System models are often highly nondeterministic, due to the nondeterminism of the environment which is considered as open, to the concurrency between processes, an finally due to explicit non determinism. For validation of functional properties, leaving the second type of nondeterminism non resolved is important in order to

verify correctness independently of any particular execution order. Nevertheless, going towards an implementation means resolving a part of this non determinism and choosing an execution order satisfying time related and other nonfunctional constraints.

Such additional concurrency restrictions can be enforced by dynamic priorities defined by rules specifying that whenever for two process instances some condition (state predicate) holds, then one has less priority than the other. An example is

$$p1 \prec p2 \text{ if } p1.group = p2.group \text{ and } p2.counter < p1.counter$$

which for any process instances which are part of some "*group*", gives priority to those with the smallest values of the variable *counter* (e.g., the less frequently served). The notion of priority rules has been defined in [BGS00].

We have used such priority rules in order to define scheduling constraints in case studies and also to implement a special execution mode used in the Omega UML profile, requiring run-to-completion of a set of objects [DJPV05, 59].

In [274], a formal dynamic semantics of IF is given, describing a system as a transition relation with a dynamically evolving state: at any point of time, the state is defined as the tuple of the states of its living components: the states of a process are the possible evaluations of its attributes (control state, variables and message buffer content). The state of a signalroute is a list of signals "*in transit*". Transitions correspond to steps of one of a single component.

5.2.2 Observers for the Expression of Dynamic Properties

Observers [Pha94] are a well accepted concept for describing dynamic safety properties of systems in terms of acceptors which are defined in a similar way as a system component. Notice also that observer always "synchronize" with observed events even when rendezvous communication does not exist in the language (for example, the observers defined for synchronous languages in section 7.1.5.3 are in this respect closer to the the components they observe).

We have extended the notion of observer, in order to make them usable not only for the expression of properties but to serve also as a flexible means for guiding the state space exploration.

- In addition to processes, observers have special operators for monitoring the current states of the processes and of their queues. For monitoring the actions, each system step generates a set of *events* corresponding to signal output/delivery/ input, process creation/destruction, informal statements,... On termination of a system step, observers react to the generated events and the reached state. Meaning that observers are executed in parallel with an IF specification using a "weak synchronous composition".

- States are classified into accept, error and invalid states.

This allows us distinguishing different kinds of observers, from pure observers expressing properties, observers for guiding state space exploration (where invalid states are used to cut parts of the state space) to intrusive observers, e.g. for testing which may not only observe, but also change the system.

5.2.3 IF as a Semantic Framework for Timed Modeling Languages

An important motivation for the definition of the IF language was its use as a semantic underpinning for timing extensions of high level modeling languages. The implementation of these semantics in the from of compilers from the source language into IF (see also section 7.1.1) has provided us with an efficient means for experimenting language extensions used in several case studies .

IF has been used to contribute to SDL (see section 5.2.3.1) by providing the semantic underpinning and experimental platform for a proposal of a more flexible timing semantics and of timing extensions allowing capturing non functional properties and requirements. In a similar way, IF has been used as a source of inspiration for the definition of a real-time profile for UML and its semantic foundation (see section 5.2.3.2).

5.2.3.1 SDL for Real-time

SDL (Specification and Description Language) [IT03] is an ITU standard (International Telecommunications Union) initially intended for the specification of Telecom standards, but widely used for modeling of general concurrent and distributed applications. SDL models are used both for early validation and for code generation. SDL includes means for the operational expression of timing features (timers which can be armed and stopped and generate *timeout* events) but not for non functional timing features (such as communication delays and execution times). Notice that the standard semantics does not allow expressing any constraint on time progress, in order to guarantee property preserving implementations in absence of knowledge on the run-time platform, whereas most existing tools consider only minimal time progress (or *eager* transitions [BST98]) in order to make validation, based on state space exploration, feasible.

Our aim was the definition of a real-time extension of SDL separating clearly the parts of the model used for code generation from "annotations" used for validation. We extend the existing work — where the work closest to ours is [MTMC99] — by proposing

- a more flexible time progress semantics based on annotations with explicit urgencies [BGK+00, 282], thus allowing the modeling of systems consisting of components which are heterogeneous with respect to their time progress rule.

- annotations for the description of resource dependent time constraints, such as execution times associated with tasks, communication delays associated with communication paths and response time of the environment, as well as general timing constraints [139, 201, 197].

The time annotations are a generalization of those proposed in previous work as they allow time constraints depending on the dynamic behavior of the system and in addition, the absence of the implicit minimal time progress assumption, avoids the need for the introduction of explicit waiting which makes the obtained model unusable for code generation.

The formal semantics of our SDL extensions had also to be defined in terms of Abstract State Machines [Gur95] which are used for the definition of the formal semantics in the SDL standard [GGP03]. Abstract state machines have no native notion of time and most of the time extensions proposed so far do not allow zero time system progress [GH96]; this lead to a study on time in ASM [283, 199, 200].

5.2.3.2 UML for Real-time Embedded Systems

We have considered the Unified Modeling Language UML and the so called model driven engineering approach as a vehicle for smoothly integrating validation into the software development process for real-time and embedded systems. The strength of UML is to provide means for modeling different view points of a system building together a global model from which ultimately an implementation will be generated.

This vision of the development process rises several challenges:

In the context of the Omega project, we aimed at the definition of an UML profile appropriate for real-time embedded systems based on the work already done at OMG [OMG02], the definition of a clean semantic framework and a smooth integration of existing state-of-the-art validation tools that would be usable by designers. An overview on this project can be found in in section 15.1.10 and in [194, 281].

A Real-time Profile for UML We have chosen as a basis, the operational UML profile defined in [DJPV03] which is close to those of the commercial UML CASE tools for simulation and/or code generation in the context of real-time and embedded systems, such as Rhapsody, Telelogic TAU and Rose Real-Time, (that emerged from ROOM).

These tools, as well as the UML profile for Scheduling, Performance and Time (SPT) [OMG02] separate the timing issues from the functional and behavioral aspects of the system which makes timing analysis with a satisfactory level of accuracy very difficult. Our aim was to overcome this problem by giving up this too strict separation.

This proposal, which is defined in detail in [196, 58], is similar to the one made for SDL in the sense that it consists at least partially in lifting the expressive concepts defined for IF at UML level (such as the notions of *clocks*, *priorities*, *resources*, *observers*,...). This lifting of concepts was easier in the context of UML which has been defined as an extendible language. Nevertheless, we had to take into account some constraints:

- in order to provide a syntactic means for accessing semantic level state changes on which time constraints must be expressed, we introduce an expressive notion of *timed events* and access to different occurrences which turned out to be very useful. Events are associated with syntactic constructs, for example, with a state an *enter* and an *exit* event, with a signal transmission a *send*, a *receive* and a *consume* event.

Events are defined by classes stereotyped <<`event`>> with predefined parameters depending on their type (all events have an occurrence time, a send-signal event has a sender, a receiver, a signal type, signal type depending parameters) and possibly user defined attributes. Time constraints are then expressed by observers that observe occurrences of the above mentioned <<`events`>>.

- we had also to take into account the acceptance by users. For example, we have not lifted the concept of *urgency* directly to UML, as the experience from our previous work with SDL revealed that explicit urgencies on transitions are not well accepted and not well used by designers. Instead, we experimented some urgency patterns which associate urgencies to transitions depending on their type: in particular, in the context of real-time embedded systems studied in Omega, considering environment transitions as lazy and system transitions as eager (or delayable when they have time constraints) was the right solution.

  Concerning time constraints, the expressiveness of observers is required by users. In addition to constraint patterns for the expression of deadline and execution time constraints of actions,... for which SPT provides already several concepts, and which can be expressed easily as derived concepts of the general profile.

  We have not enough feedback on the use of priority rules in order to be able to confirm or infirm their usefulness at user level. We have used them in one case study for expressing a fixed priority schema.

Implementation in IF The semantics of the Several approaches to a formal definition of the semantics of the profile were proposed by different project partners. For the operational profile a global operational reference semantics has been defined by a mapping into symbolic transitions system [DJPV03]. For the real-time extensions, we give in [58] a semantics of duration expressions and constraints in terms of OCL so as to stick as close as possible to UML itself.

We defined a semantics of the operational Omega UML and the time extensions in the form of a mapping of the UML concepts into the IF language by respecting the operational untimed semantics defined in [DJPV03].

We have taken care to make this mapping as open as possible to alternative or additional semantic choices, especially for those choices where several alternatives have come out of the user requirements. For most concepts, such as objects, state machines, signals, object creation, non determinism, real-time extensions including observers, the mapping is straight forward as there exists a corresponding concept in IF. Associations are mapped into variables which may be shared amongst a set of processes. To each type of <<`event`>> corresponds an explicit transition in the IF specification allowing to map each UML level event observation into a corresponding one handled by some IF observer.

The main features which cannot be mapped directly are *inheritance*, *method calls* and the *run-to-completion* semantics of the set of objects in the same activity group. Operation calls are translated into two signal exchanges between the calling and the called object,

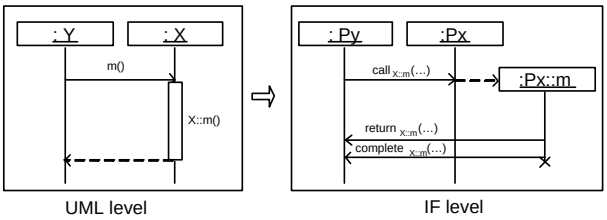


Figure 5.1: Handling primitive operation calls using dynamic creation.

and the operation body is represented by an IF process (see figure 5.1), thus allowing also recursive operation calls and easy introduction of asynchronous operation calls.

The run-to-completion strategy for the (dynamically evolving) set of objects building an *activity group* is obtained using a single dynamic priority rule, stating that the manager object associated with each group has always lower priority than all other objects, guaranteeing that a step — triggered by a new external request — is initiated only when the previous one has completely terminated.

This mapping is presented in more details in [236, 237, 144, 58]. The tool IFx which provides user interface of IF for users of this UML profile is presented in section 7.1.1.

The Omega UML profile and its time extensions have been successfully used for modeling the four case studies considered by the four industrial partners of the Omega project. For all of them, validation, in particular of timing properties, has been carried out using the IF toolbox to the greatest satisfaction of the users who continue to use the profile and the tool in other projects [281].

5.2.4 Case Studies

IF has been used for successfully modeling and validating systems in large variety of application domains, where the models have been made either directly using the IF language or using one of the high level languages (SDL or UML) for which translators exist:

***Embedded software*:** Examples are

- the Ariane 5 Flight Program [152, 291],
- the Medium Altitude Reconnaissance System MARS (see the OMEGA Project web page `http://www-omega.imag.fr/`),
- the K9 Rover Executive [89, 122];

These examples deal all with static architectures. The main problem is here the interaction between synchronous and asynchronous parts of the system and timing aspects.

***Telecommunication protocols*:** Examples are

- PGM, Pragmatic General Multicast Protocol (see the ADVANCE Project web page `http://liafa.jussieu.fr/~haberm/ADVANCE/`)

- association and channel management of MASCARA, Mobile Access Scheme based on Contention and Reservation for ATM [195];

In these examples the considered systems make important use of dynamic object creation (of users, channels...) and have a dynamic communication structure (mobile users in the first example), as well as complex data types. An important point here was to find the appropriate abstraction for making the verification problem finite.

***Planning and scheduling*:** The main example is providing optimal schedules for a chemical production plant of Axxom AG [275]. Here the architecture is static and the behavior acyclic. In this example we make use of the possibility to associate cost functions with transitions and have introduced some new exact heuristics for avoiding the exploration of too many useless solutions.

***Asynchronous circuits*:** Examples

- circuit validation [133, 131]
- and timing analysis [246]

In these examples of systems with static architecture, timing mixed with complex control are the main challenge.

We present in more details the Ariane 5 case study.

Ariane 5 Flight Software

The objective of the Ariane 5 Flight Software is to control the launcher mission from lift-off to payload release. This software operates in a completely automatic mode and has to handle both the external disturbances and different hardware failures that may occur during the flight. This case study presents the most relevant points required for embedded application and focuses on the real time critical behavior. More detailed descriptions of this case study can be found in [59, 291].

In an earlier experience with this case study using SDL as a modeling language and IF as a verification tool [152], we modelled only the asynchronous part of the system, responsible for controlling the flight phases and the exception handling. We succeeded in verifying the correctness of this part of the software using an almost deterministic environment (representing the synchronous part) obtained from a real flight, but we had no insurance that this environment represents an abstraction of the other parts of the software. Using Lustre [300], we had modelled and verified the functional correctness of the cyclic guidance and navigation algorithms under some assumptions on the asynchronous part. Nevertheless, using this combination of SDL and Lustre, we could not verify all the properties that we could consider with the common UML model, in particular mutual exclusive access to the bus and all the scheduling related issues.

Modeling with UML The Ariane-5 Flight Software is a composition of several synchronous and asynchronous modules with strong interactions. Synchronous, cyclically scheduled modules have a specific period and phase; they receive their inputs at the start

of their periods and shall produce their outputs before their next execution. Non cyclic, asynchronous modules are activated on external events, and are synchronized or not with the cyclic synchronous processes depending of their required deadline; which is in some cases very short.

Moreover, in order to verify the correctness of the real time software design, non-functional features such as the scheduling policy (task or thread definition, priorities between tasks) CPU consumption of each algorithmic function (associated to a task) and different timing constraints have to be expressed.

The design has been modelled in UML using Rational Rose and the Omega UML profile as a collection of objects communicating mostly through asynchronous signals, and whose behavior is described by state machines. Abstract operations are used to model the guidance, navigation and control tasks. For the modeling of timed behavior and timing properties, we used the OMEGA real-time features.

Validation Results using the IF toolbox The model obtained for the Ariane-5 Flight software is relatively large: 23 classes, each one with several operations and a state machine. In addition, the UML model has been annotated with 12 observers, expressing all the relevant untimed and timed requirements or assumptions. The translation into IF is completely automatic using uml2if and gives around 7000 lines of IF code.

To cope with the complexity of the model, in addition to static analysis and partial order reduction, some application specific abstractions and reductions have been needed. In order to verify the correct cooperation between the cyclic and the acyclic behavior, we have used three approaches: we verified the cooperation between the concrete specification of the cyclic behavior with an abstraction of the acyclic behavior and the other way round. This gave satisfactory results concerning the properties of the asynchronous part handling the global flight phases. In addition, we considered the cooperation between the concrete specifications of both parts by scaling down the durations of some phases of the mission.

This last method allowed to verify the main schedulability related property of the basic cycle ensuring that its execution finishes before its deadline (i.e. before being requesting to restart a new execution cycle) has been naturally expressed by a timed observer and validated for several scheduling policies and timing assumptions. Notice that an RMA approach simply concluded that the system is not schedulable.

On the reduced model, all properties have been validated through model checking. Intentionally added bugs have been discovered as well, and error scenarios provided for all of them.

The performances of the tool have been considered very satisfactory for this medium size system. In all cases, long verification runs meant presence of a bug (leading to an infinite state space), and after elimination of all bugs, verification times ranged from seconds to a few minutes depending on the properties. The user estimated that the analysis based on the use of an abstract version of either the cyclic or the acyclic part will also be possible for much larger applications, such as ATV.

5.2.5 Concluding Remarks

The IF language has proven to be well adapted as a modeling language for distributed and concurrent applications as well as a framework for providing a semantic underpinning for user level modeling languages, including timing features. The reasons are the expressiveness at the right level of abstraction of the concepts and the existence of the IF toolbox, providing tool support for model exploration and verification.

Model exploration is a powerful means for debugging, also for complex models with infinite state spaces for which no real alternatives exist. A general outcome from the Omega project — where several verification tools have been applied to the same case studies — is that the IF tool is the only one that always detected rapidly subtle modeling errors, where some of them where due to combinations of functional and timing aspects.

Concerning the UML real-time profile developed and its translation to IF, notice that the timing extensions proposed are almost independent of the choice of the operational profile and can be directly reused in other context. Also the mapping from the Omega operational profile to IF takes already into account a rich subset of UML and has been developed with flexibility in mind.

A related topic that we want to integrate in IF is *Automated Module Composition* in a context where components or modules are "black boxes" that are completely characterized by their *interfaces*, in terms of typed input and output ports. A *compatibility* relation between ports specifies the composition constraints: only compatible ports can be composed. Notice that the compatibility relation can be made as complicated as necessary, using static but also dynamic definitions of ports, like in interface theories [LX00, dAH01].

The problem considered is that of finding in library a set of existing components that can be used to implement some given interface by connecting them in the right way. When the libraries are big, this needs obviously to be automatised. This is, for instance, the situation in the automotive industry where controller models in Simulink/Stateflow (trademark of The Mathworks, Inc.) are built from model libraries containing thousands of components, from which hundreds must be chosen for a particular design [MC01].

In [259, 263]. we provide a solution for this problem of *module composition* [259, 263]. In the abstract setting, we are given a set of *available modules* (i.e., the library components, which can be parameterized) and a *target module*. We want to find automatically, if there exists, a *composition* of some of the available modules that implements the target module.

5.3 Components for the Development of Embedded Systems

For efficiency reasons, embedded software programs were originally written in low level language (assembly code) or general purpose sequential languages (C or Ada). Nonetheless, these languages were not adapted to the programming of these systems.

The first improvement step has been to develop domain-specific languages, like the

family of synchronous languages [Hal93, 47]. They are based on a precise mathematical model of time and concurrency, providing high-level constructs as well as efficient compilation into sequential code. Moreover, they are associated to verification tools that largely improve the safety of programs.

However, the programming of large size critical embedded systems raises several problems which are yet to be addressed. The increase in the size of embedded systems (e.g., over 5000 SCADE charts for the flight control of the Airbus A380), together with very strong constraints in time-to-market (typically 5 years for the design of this new generation of Airbus) requires the intensive *reuse* of software components.

There is an increasing need for an integrated development environment based on a formal semantic basis, allowing the specification, the design, the development, and the reuse of components. This environment should provide means to manage the whole life-cycle of a system, from early specification phases to efficient embedded code. It should provide composition and encapsulation mechanisms to be used both for the specification of programs and for the description of their environment. It should provide early validation tools for checking statically the absence of some run-time errors (e.g., type preservation, deadlock freedom, execution in bounded time and space). Early simulation of programs and specifications should be possible, hence the ability to execute *incomplete* programs is a key point.

5.3.1 A Language-based Approach

In the ALIDECS project (see section 15.2.3), we study the research problems related to the above mentioned needs. The use of an appropriate programming language being one of the key points contributing to safety in computer systems, we Cavour a *language-based* approach, for all the needs listed above. This means that we aim at defining (or adapting) high level languages for the components specifications and implementations, the description of the environments, etc. Moreover, we look at implementation problems: separate compilation, dynamic modifications of component systems, etc. Another important point is related to the expressive power of the specification languages, and to its "usability". In other words, since specifications may become as complex as programs, the user should be given a full structured language, with reusable blocks, in order to describe specifications easily.

From this language-based perspective, the notion of a component is necessarily *hierarchic*. That is, a given component can be composed with another component, such that the result is a new component.A component should always include its interface as well as its *specification*, and of course the composition operation should be *formally defined*. Notice that this point of view on components is not shared by all the existing approaches. For instance, in existing architecture description languages (ADLs), there is often a clear distinction between the entities that may be composed, and the global architecture.

5.3.2 The Development Environment

We do not study new semantical models for reactive software and components. Instead, we adopt the synchronous technology as the common semantical basis for all notions, from basic language constructs to high level component composition. This will avoid, in particular, to study implementations issues from scratch.

The main goal of the project is to study a integrated development environment based on this semantical model, and tailored for the specific constraints of embedded systems (ability to guarantee properties statically, existence of efficient code generators, etc.). These objectives may be viewed as covering the missing intersection between:

- General-purpose component-oriented design, which may not always be adapted to the specific constraints of reactive embedded systems. For instance, it may be quite difficult to guarantee bounded memory and reaction time statically.
- Several clean notions of modules introduced in general-purpose programming languages like ML, which may be two powerful. For instance, if we use a general higher-order functional language for the description of compositions, a lot of interesting static properties will become undecidable. We'll probably need to restrict the power of the language used for describing compositions.
- Engineering tools and methods like Simulink/Stateflow, dedicated to the development of reactive systems, that were originally designed as simulation tools for control engineers and therefore lack all the properties a structured programming language should have. These tools are interesting for two reasons: first, they are de facto standards of the application domains we are interested in; second, people have developed several notions of libraries for the easy reuse of "components", and these notions may give hints on the needs we have to cover.
- Formal frameworks like the so-called "reactive modules" in which the emphasis is put on formal compositional proofs, and which do not offer high-level programming languages, nor implementation techniques.
- Domain-specific languages for reactive systems, like the family of synchronous languages, which perfectly address the specific constraints of this domain, have compilers that produce good embedded code, but lack a notion of component.

We will consider a development environment in which several kinds of entities coexist: programs, specifications, models of physical environments, models of missing software or hardware parts. All these entities should be executable, and may be composed in synchronous, asynchronous or mixed ways.

Programs are deterministic and fully detailed executable objects, either hardware or software parts. *Specifications* are non-deterministic early specifications of components that will eventually become executable objects. *Physical models* are the necessarily non-deterministic abstractions of physical environments that will never be known in full details. *Missing parts* also need modeling, in the same form of early specifications.

5.3.3 First Results

The first key point is to understand how a synchronous framework may be used as a unifying one for heterogeneous components that may be synchronous or asynchronous, centralized or distributed. *Synchronous modeling of asynchronous systems* is the key. The next keypoints are the *specification of components* and *early execution*.

5.3.3.1 Synchronous Modeling of Asynchronous Systems

It is well-known [Mil83] that synchronous formalisms can be used for describing asynchronous system, as soon as they can involve non-determinism. Synchronous languages are intrinsically deterministic, but non-determinism can be introduced in synchronous programs through additional inputs. Moreover, one can arbitrarily restrict the non-determinism by expressing assertions about these additional inputs. As a consequence, one can model not only fully asynchronous systems — or asynchronous compositions of synchronous subsystems (GALS [BCG+97, BCG99, BS98]) — but also the full range of partially asynchronous systems that exist in real life [202]. Once modelled that way, systems can be simulated and verified using tools devoted to synchronous languages. This approach was used to validate the translation schemes to distributed or multi-task code [129] (see §8.2). This approach is also studied within the ASSERT Project (§15.1.4).

5.3.3.2 Specification of Components

We already have a lot of ways of specifying the functional behavior of non-deterministic synchronous reactive systems:

- Observers : written in any synchronous language, interpreted as an *acceptor* of input/output traces
- Reactive programs with undefined local variables : again, written in any synchronous language, interpreted as ordinary programs, where the value of undefined variables is considered as random
- Contracts : for instance a pair of two observers, specifying independently what is required on the inputs, and what is guaranteed on the outputs.
- Sets of parallel machines with weights (see §7.2.1.1) : convenient for the modeling of non-deterministic environments.

All these forms share a common semantical basis. We have to extend these forms to cope with asynchronous components (using the principles explained in the previous paragraph).

5.3.3.3 Early Execution of Heterogeneous Systems

As a first result of ALIDECS, we have connected our tool Lucky (explained in the context of testing methods in§7.2.1.2) to ReactiveML. Lucky takes as input a non-deterministic specification of a reactive system, and produces concrete scenarios corresponding to this specification. ReactiveML is a language developed by M. Pouzet and L. Mandel (Lip6, Paris), that offers a set of reactive constructs on top of ML. It's interpreted, and it allows

to modify programs dynamically. It is very adequate to serve as the "glue" code between components that are finished, and other components that are given as non-deterministic specifications in various forms.

The first use of this Lucky-RML connection is is the context of ad-hoc sensor networks (see below).

5.3.4 Classes of Applications

We need to study applications from various domains, in order to propose adequate languages, static analysis and implementation methods for embedded component-based software. The industrial cooperations of the Synchrone team offer interesting case studies in three very distinct applications domains: critical control avionics software, ad hoc sensor networks, systems-on-a-chip.

5.3.4.1 Lustre/SCADE Programs from Airbus

In his PhD, Lionel Morel has worked on a case-study from Airbus: the ELMU (Electrical Load Management Unit). The complete code contains a library of small software components that are intensively used, and that are specified informally in a way that resembles contracts. L. Morel proposed a rewriting of these informal specifications into contracts. We are currently doing re-engineering work on the specification of the ELMU itself, in order to obtain contracts.

5.3.4.2 Ad-hoc Sensor Networks

In the context of the project AdHoc (§15.3.4) VERIMAG works on the modeling and analysis of a sensor network. Our task is to provide a global model of such a system, which is compulsory for choosing efficient implementations. A crucial point is that there are a lot of interactions between the levels of abstraction that were traditionally separated in network design. For instance, the type of application we have in mind with a sensor network may influence the choice of the MAC layer. A typical MAC layer in this application domain is such that all nodes are sleeping most of the time (meaning the radio functions are not working, in order not to waste energy). Since it is costly for a radio system to switch from sleeping to active mode, this choice is convenient only if the application is such that the communication between nodes is sporadic.

The first result is a small simulator written in ReactiveML, and connected to Lucky. It allows to represent in a unified way all the hardware and software components of a sensor network:

- The basic nodes of the network (small pieces of hardware with radio capabilities, a very small memory, and limited computing power). A node is modeled according to the amount of energy it spends, depending on its mode (receiving, emitting, idle...). Transitions between modes are also qualified in terms of energy consumption.
- The MAC layer is programmed in ReactiveML
- A diffusion protocol is also programmed in ReactiveML

- A model of the application is included in the system. This is a very important point, since a network of sensors may have very different behaviors depending on the kind of sensor activations the application may provoke. In our first case-study, we consider a pollution detection application. We model a pollution cloud passing over a set of sensors spread on the ground. This is done in the language of Lucky, and the connection of Lucky to RML allows a global execution of the sensor network.

The implementation of ReactiveML is tailored towards the execution of thousands of processes, so we can simulate realistic sensor networks. Since ReactiveML has a formal semantics very close to that of more classical synchronous languages, the complete description will be reused to extract formal models of the whole system, for analysis purposes.

The ability to include components that model a physical environment will also be used to take radio perturbations into account.

The models of hardware components will be used as formal specifications for the actual implementation of the sensor nodes with asynchronous circuits.

The results are described in [SMMM06, SMM+06].

5.3.4.3 Systems-on-Chip

In the context of the project SocTLM (see §15.3.6 and chapter 11) we work on the description of Systems-on-Chip at the transactional level. This is a very interesting context in which to look at the modeling of heterogeneous components. In SystemC, the designers may describe: the hardware components, the software components, the operating system primitives, the communication protocols, etc.

By expressing the semantics of SystemC into communicating parallel machines (that can then be translated to Lustre in order to obain an executable model), we already made a step towards a formal model for harware/software components.

We are now working on the specification of non-functional properties of these components.

Chapter 6

Language Design and Extensions

The activities in language design and language extensions are related to the family of synchronous languages. Synchronous languages (Lustre, Esterel, Signal, ...) are high level languages dedicated to the programming of reactive systems. Since 1984, VERIMAG has been involved in the development of the synchronous language LUSTRE, which became the core language of the industrial environment SCADE, developed by Esterel-Technologies, and used, e.g., by Schneider-Electric, Airbus. For more details on the story of Lustre, see §2.1.

The "synchrone" team at VERIMAG has a long experience in proposing languages that are both formally defined and adapted to industrial practices. Part of the work is still devoted to language design: LUSTRE extensions, new synchronous languages mixing control and dataflow, languages for the description of non-deterministic environments (needed by testing tools, see 7.2.1.1), aspect-oriented programming for reactive systems, etc.

6.1 Evolutions of Lustre

The basic constructs of the language are very few, and well known from the very beginning. Recent work addresses problems due to the use of Lustre in very large projects. The extensions proposed were first inspired by industrial case-studies from Airbus. We also worked on multiparadigm languages, mixing dataflow with imperative features like explicit automata.

The recent work on LUSTRE is twofold: we introduce arrays, iterators and modules in the academic version (LUSTRE V6); we maintain a strong collaboration with Esterel Technologies for the definition of the next version of SCADE.

Arrays were first introduced in the language for helping the description of hardware systems [Rocheteau91]. Lustre-V4 includes operators like slice extraction, concatenation and a static recursion mechanism, all well adapted to the description of hardware systems. This set of operations is however not well-suited for the description of software systems: their compilation leads to imperative code with independent variables corresponding to array elements. In the new V6 version of Lustre, we introduce array iterators, inspired from classical functional operators like map and fold [232, 224, 32], that can always be

compiled into imperative code with arrays and loops.

This notion of array will be implemented in the next version of SCADE (we have a contract with Esterel Technologies for that, see §15.3.5).

In Lustre V6, we have introduced a concept of modules that we called *packages*. A Lustre package is basically a kind of module that can be parametrised by types, constants or nodes. It is something between ocaml Functors and BOO Classes, but tailored to the use of reactive systems in general, and Lustre in particular.

6.2 Mode-automata

The definition of mode-automata is the PhD thesis work of Yann Rémond [36]. It was motivated by the following: Lustre is well-suited for the development of regulation systems, which are first designed by control engineers, and can then be programmed as block-diagrams. Working with the systems Lustre is well-suited for, we observed that they are often specified informally using the notion of running modes. However, there seemed to exist no language in which the mode-structure of a complex system could be expressed directly. Following the approach of domain-specific languages, we proposed to extend Lustre with a new construct, called *mode-automaton*, devoted to the description of these running modes of regulation systems. We defined the language of mode-automata and its semantics, and implemented the prototype Matou. A mode-automaton is mainly a (possibly structured) automaton whose states are labeled by dataflow equations. The compilation process can take advantage of the imperative structure to produce code for the reactive kernel in such a way that the WCET is the maximum of the modes' individual WCETs. Without an explicit imperative structure, it is in general difficult to avoid a WCET which is the sum of all the modes' individual WCETs.

We discussed the mixed language with Airbus and Esterel Technologies, working on case-studies. J.-L. Colaço (from Esterel Technologies) and M. Pouzet (Lip6, Paris) worked on more basic constructs that can be added to a dataflow language in order to support imperative features, among which mode-automata. The next version of SCADE will offer this support.

Related papers are: [MR97, MR98, MRR00, MR00b, MR00a, 300, 226, 73]

6.3 Contracts for Reactive Systems

The definition of contracts is part of the work of Lionel Morel [32] in his PhD thesis. The first definition is detailed in [225, 224].

The "Design by contract" principle has been applied successfully in the context of object-oriented designs, since it was first proposed by B. Meyer for Eiffel. In OCL (the Object Constraint Language for UML) for instance, one can specify a contract for a method, in the form of a `pre-post` specification à la Hoare. This is very well adapted to the specification of transformational software.

However, a simple pre-post specification is not sufficient for *reactive* software. The ability to *model* the environment is crucial when developing embedded control systems,

because no control system is able to behave correctly in any environment. For instance, a heater controller always makes the hypothesis that, once it has turned the heater on (an output of the system), the temperature (an input) will start increasing, within a known range of slopes. The specifications of reactive embedded software have to take the intrinsic system/environment loop into account. The timing aspects make it even harder: we should be able to *assume* that the temperature input is increasing, when specifying what a reactive system *guarantees*. We need a notion of logical time defined by the sequence of reactions to the environment. It is *logical* because it needs not be directly related to a physical time scale.

What does *time* mean in contracts? The assumption and guarantee clauses may talk about the current execution of a method, and express in the guarantee clause that `x` is equal to `old x + 1`, but "old" means "just before this execution". However, we may have to talk about *previous* executions of the same method. For instance, consider a class C with methods $m1$ and $m2$, and a specification stating that method $m1$ should never be called before $m2$ has been called at least once. A usual way of expressing such contracts is to attach an explicit Boolean *state* Σ to each object of class C; an execution of $m2$ *guarantees* that Σ is set to true; executing $m1$ *assumes* Σ is true. In some sense, this means that the potential *memory* hidden in the assume or guarantee clauses has been compiled by hand into explicit variables. For reactive systems, we need this notion of time: first, we need to talk about the successive executions of the piece of code that constitutes *one* reaction of the system to its environment; second, the assumption on the environment (e.g., "*the temperature is increasing*" supposes successive uses of the temperature; and, finally, when executing several components "in parallel", a notion of logical time is needed to understand how their executions may interleave.

We propose to write a contract of a reactive component as a pair (assume clause, guarantee clause) in which both clauses are given by synchronous observers. The assume clause may talk about the inputs, the guarantee clause may talk about the inputs and the outputs, but should not constrain the inputs more than the assume clause does.

We propose a language support, together with the definition of contracts, insisting on the actual use of these local specifications in complex industrial applications. For these notions to be adopted by engineers, it it important to provide contracts that can be written in the same language as the programs. Using Lustre dedicated nodes for that also allows to separate clearly the memory needed for the program, and the memory needed for each of the contract clauses.

The first experiment with this contract language has been done on a case-study from Airbus, the Electrical Load Management Unit. The complete Scade specification of the ELMU includes a library of around 100 small reusable components, that are conveniently specified by contracts.

6.4 Description of Non-Deterministic Components

Testing needs a formal definition of the properties to be proven, as well as the formal description of the hypothesis made on the environment. The main challenge to automate

a testing or a simulation process is to be able to automate the generation of realistic input sequences to feed the program. In other words, we need an executable model of the program environment in which inputs are the program outputs, and outputs are the program inputs.

In the first Lurette prototype (see section 7.2.1.1), the behavior of the System Under Test (SUT) environment was described by Lustre observers specifying what realistic SUT inputs should be. In other words, the environment was modelled by a set of (linear) constraints over Boolean and numeric variables. The work of Lurette was to solve those constraints, and to draw a value among the solutions to produce one SUT input vector. But, from the point of view of language expressivity, Lustre observers happen to be too restrictive, in particular to express sequences of different testing scenarios, or to have some control over the probabilistic distribution of the drawn solution.

It was precisely to overcome those limitations that the new languages, Lutin and Lucky, were designed.

6.4.1 Abstract Machine: Lucky

Lucky was designed as a basic model for describing the behaviors of non-deterministic or stochastic reactive machines.

A Lucky program is an interpreted automaton whose transitions define the reactions of the machine. More precisely, each transition is associated with:
— a set of constraints (a Lustre-like formula) that defines the set of the possible outputs, and
— a weight that defines the relative probability for each transition to be taken, i.e., to be used to produce the output vector for the current step.

Lucky is the core of the testing tool Lurette (see an example in §7.2.1).

6.4.2 User Friendly Language: Lutin

Lucky was not designed as a high-level language, but rather as an abstract machine code. The idea is then to define high-level, user friendly languages that can be compiled into Lucky.

Such a language, called Lutin [243], has been defined. This language is similar to Lucky, except that it is based on control statements instead of explicit automata. The basic statements are sequence, loop, stochastic choice, concurrence. The language also provides classical features in user-friendly language, like functional abstraction, data-types, etc.

Here is a very simple example of Lucky program describing how the temperature varies *non deterministically* in a room according to the commands sent to an heater located in the room. The programs illustrates some of the features of the language:

- the program (`Temp`) is declared with its inputs (`on`, `off`) and its output (`t`),
- in a functional-like style, the user may define macros with parameters (generic constraint `between`), or without parameter (the simple alias `dt`),

- in a constraint, the user may refer to the previous value of a variable (like `pre t` in the definition of `dt`),
- the user mat introduce, via the *exist* statement, internal random variables2s; `running` denotes the internal state of the heater, it is treated as a hidden output,
- dynamic behaviors are built by combining atomic reactions with control statements; atomic reactions are constraints on the inputs, outputs and random variables,
- the control statement `assert` allows to distribute global constraints over a whole behavior; in the example, global assertions are that the slope always remain between -0.1 and 0.2, is positive when the heater is running and negative otherwise,
- the basic control statement is the sequence `fby`; in the example, the system starts with a constraint stating that the heater is off and the temperature is between 10 and 17; then the system enters its *working* behavior,
- the `loop` statement takes has parameter the average number of steps and the standard deviation; in the example, the system works properly for an average number of 10000 iterations,
- the stochastic choice (`|` together with `weight`) allows to express several possible behaviors; in the example, the system behaves (almost) correctly if:
  - it obeys a `on` command, which is probable (weight 100),
  - it obeys a `off` command, which is as probable (weight 100),
  - or it ignore any command, which is less probable (implicit weight 1),
- then the system breaks down and stays forever in a state where the heater remains off (unbounded `loop`).

```
node Temp(on, off: bool) returns (t: real) =
   (* ---- SOME DECLARATIONS --- *)
   (* Constraint macro *)
   let between(x, min, max: real) =
      (x > min) and (x < max)
   in
   (* Simple alias *)
   let dt = (t - pre t) in
   (* Random variable *)
   exist running: bool in
   (* ---- MAIN DYNAMIC BEHAVIOR --- *)
   (* Global invariants *)
   assert between(dt,-0.1,0.2)
      and (running => (dt > 0.0))
      and (not running => (dt < 0.0))
```

```
in
(* Initial constraint *)
(not running and between(t,10.0,17.0))
fby
(* Working behavior *)
loop~10000:1000 {
   (on and running) weight 100
   | (off and not running) weight 100
   | (running = pre running)
}
fby
(* Breakdown behavior *)
loop { not running }
```

Lutin compiles into Lucky. This compilation resembles classical algorithms that translate regular expressions into automata. This work is presented in the thesis of Yvan Roux [37].

6.5 Aspect-Oriented Programming (AOP) and Reactive Systems

Aspect oriented programming (AOP) has emerged recently. It aims at providing new facilities to implement or modify existing programs: implementing some new functionality or property in a program P may not be done by adding a new module to the existing structure of P but rather by modifying every module in P. This kind of functionality or property is then called an *aspect*. AOP provides a way to define aspects separately from the rest of the program and then to introduce or "weave" them automatically into the existing structure.

From our point of view, the AOP community is divided into two families:

(1) the first family is guided by the needs on tracing, profiling, debugging: aspects are used to describe functionalities that do not modify the behavior of the original program. Such an aspect should be allowed to access/observe every entity (variables, functions, ...) in the program, provided that it does not interfere with the execution of the program. The way it refers to the entities of the program may be syntactic, or even lexical, but its role is not intrusive.

(2) In the second family, aspects aim at *modifying* the behavior of a program. The way aspects are allowed to interfere with the program behavior, and with each other, has to be clearly defined and limited. The definition of an aspect needs to be more *semantic*.

We consider that AOP techniques and tools have demonstrated their interest in various domains. We suspect that some of the recurring problems people have encountered when programming complex reactive systems (and making the programs evolve) can be better understood and dealt with if we clearly consider them as aspects. In [91], we investigate

a construct that mixes imperative code with declarative constraints. The declarative part is *woven*, at compile time, in the imperative code, by controller synthesis techniques.

Since we are interested mainly in *critical* systems, we care about formal semantics, and therefore would like our notion of aspect to be more "semantical" than "syntactical". We will concentrate on the preservation of a behavioral equivalence.

To investigate these ideas further, we propose to define and implement a particular notion of aspects for a formally defined language, in the context of reactive systems. This means:

— *Choosing a base synchronous language for reactive systems.* We choose to work on a formalism whose structure is intermediate between the structure of high level languages (because it would be too specific) and the simple flat automata representing the reactive behaviors (because it would be degenerated). Our core language is made of Mealy machines composed in parallel, and communicating via the synchronous broadcast.
— *Proposing a language of aspects* and a weaving mechanism, in a somewhat "minimal" way. The weaving operation should preserve the usual behavioral equivalence used in the context of reactive systems.
— *Implementing* this language and this weaving mechanism in an existing compiler.
— *Applying it* to significant case studies.

Aspect-orientation and reactive systems is the PhD thesis work of D. Stauch, which started in October 2004. The first ideas where published in [92], and an extended version of this paper will appear in 2006 in a special issue of Science of Computer Programming [AMS06a]. Other papers are [AMS06b, SAM06].

6.6 Projects and Contracts

Activities related to language design are conducted in various projects, with or without industrial partners: in ALIDECS 15.2.3 we study languages for the description, implementation and validation of embedded components. In ASSERT 15.1.4 we study a language for expressing contracts of synchronous and asynchronous components. In CTRL-a 15.2.8 we studied aspect-oriented mechanisms. In the industrial contract 15.3.5 we have to validate the choices made by Esterel Technologies for the next version of SCADE, including arrays and iterators, a support for mode-automata, and other improvements.

6.7 Perspectives for Future Work

We will continue to make LUSTRE evolve to meet the user needs. This has to be done carefully, with a lot of discussions with both the users and the tool developers, in order not to lose the good properties of the language.

The main research direction in language design is to propose new high level languages for the specification of non-deterministic components, that are needed for two purposes: the modeling of physical environments, the specification of computer parts that are not fully implemented yet.

When these components are included in the system because they model the physical environment, there are two possibilities: either the physical phenomenon to be modelled is well known, and we can find simulation models in a quite operational form (for instance a program that starts by drawing a random number, and then executes a deterministic algorithm); or the phenomenon is not well known, and we cannot find simulation models. In the first case, we should be able to import an operational simulation model as a subprogram in our language. In the second case, it is often more convenient to propose a declarative style: the physical phenomenon is modeled by expressing temporal constraints on the chaos behavior. To meet all these needs, we have to define a mixed-style language in which operational simulation models (usually quite efficient in simulation) can be used together with constraint-based models.

Chapter 7

Validation Techniques

Since embedded systems are often among the most safety critical, the validation of these systems is a particularly important goal. In this chapter, we will present our recent activities in this area, while distinguishing between verification techniques (§7.1) — which aim at ensuring, automatically or interactively, that some properties are fulfilled by a system —, testing (§7.2) — which aims at discovering bugs —, and debugging (§7.3) — which aims at finding the source of a bug.

7.1 Verification

This section mainly concerns automatic verification. The IF toolbox (§7.1.1) is one of the main tool developed at Verimag; it is dedicated to the automatic verification of finite-state (or finitely abstracted) asynchronous real-time systems. The subsequent sections concern specific problems raised by some classes of infinite systems, like parameterized networks (§7.1.2), dynamic memory (§7.1.3), and numerical programs (§7.1.4). Finally, §7.1.5 addresses specific interactive methods for the correct construction and verification of Lustre programs.

7.1.1 The IF Toolbox for the Verification of Real-time Distributed Systems

The IF toolbox [143, 144, 140] is the validation environment built upon the IF notation presented in section 5.2.1. The toolbox is freely available at `http://www-verimag.imag.fr/~async/IF`.

The main validation technique implemented is explicit state-space exploration which is currently the most successful techniques used for the validation and test case generation of concurrent systems with complex control flow. Nevertheless, exploration is far from being trivial for dynamic systems that, in addition, use complex data, involve various communication mechanisms, mix several description languages, and moreover, depend on time constraints. The solution we propose is an open, modular and extensible exploration

platform designed to cope with the complexity and the heterogeneity of actual concurrent systems.

Overall, the toolbox is composed of three categories of tools (see Figure 7.2)

1. **behavioral tools** built on top of the exploration platform, for simulation and debugging, automatic verification of properties and automatic test generation. These tools implement state of the art reduction techniques such as *partial order reduction* and some form of *symbolic simulation*, and thus present a good level of scalability.

2. **static analysis tools** providing source-level optimizations that help reducing furthermore the state space of the models, and thus improve the chance of obtaining results from the behavioral tools. The implemented *data* and *control* flow analysis techniques, leading to exact abstractions of the initial model, are dead variable reduction, dead code elimination and *slicing*.

3. **front-ends and exporting tools** which provide an interface with higher-level languages (UML, SDL) and with other validation tools (Spin [Hol99], Agatha [LRG01], etc.).

7.1.1.1 An Open Platform for State Space Exploration

The exploration platform provides access to the operational semantics corresponding to IF descriptions. The platform offers primitives for representing and accessing execution states and labels as well as basic primitives for traversing the state space: an *init* function which gives the initial state, and a *successor* function which computes the set of enabled transitions and successor states from a given state. These are the key primitives for implementing any on-the-fly forward enumerative exploration or validation algorithm.

Figure 7.1 shows the structure of the exploration platform. The main features of the platform are simulation of the process execution, non-determinism resolution, management of time and representation of the state space.

The exploration platform can be seen as a tiny operating system where process instances are plugged-in and executed. Process instances are either application specific (coming from IF descriptions) or generic (such as time or channel handling processes).

In particular, simulation time is handled by a specialized process managing clock allocation/deallocation, computing time progress conditions and firing timed transitions. There are two implementations available, one for discrete time and one for dense time representation. For discrete time, clock values are explicitly represented by integers. Time progress is computed with respect to the next enabled deadline. For dense time, clock valuations are represented using variable-size Difference Bound Matrices (DBMs) as in tools dedicated to timed automata such as Kronos[Yov97] and Uppaal[LPY97].

The exploration platform is used as back-end of debugging tools (interactive or random simulation), model checking (including exhaustive model generation, on the fly μ-calculus evaluation, model checking with observers), test case generation, and optimization (shortest path computation).

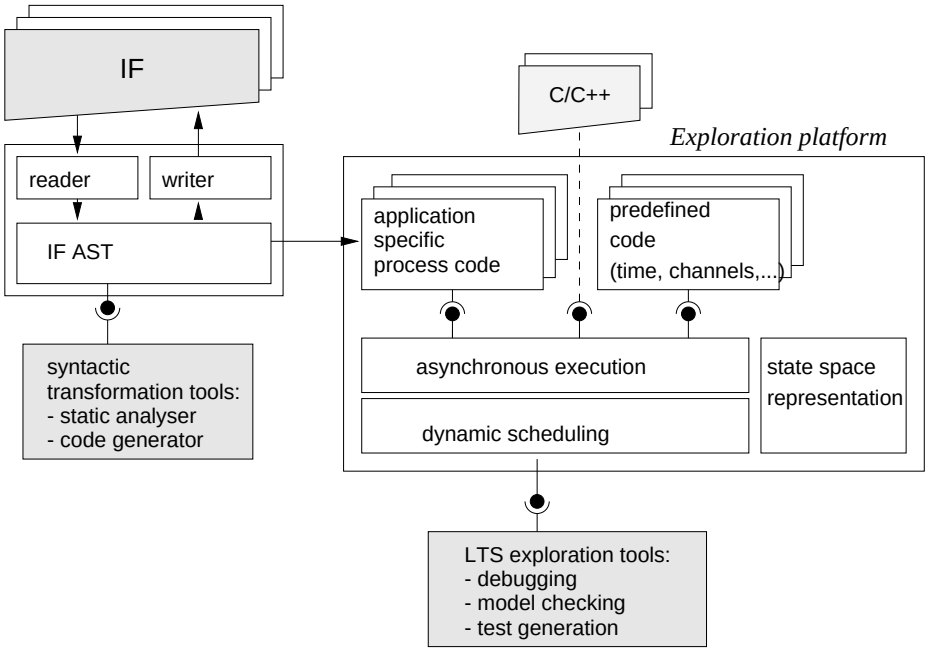


Figure 7.1: Functional view of the IF platform

This architecture provides features for validating heterogeneous systems. Exploration is not limited to IF descriptions: components with an adequate interface can be executed in parallel on the exploration platform. It is indeed possible to use native C/C++ code (either directly, or instrumented accordingly) of already implemented components.

Another advantage of the architecture is that it can be extended by adding new interaction primitives and exploration strategies. Presently, the exploration platform supports asynchronous (interleaved) execution and asynchronous point-to-point communication between processes and dynamic scheduling through dynamic priorities. Different execution modes, like synchronous or run-to-completion, or additional interaction mechanisms, such as broadcast or rendez-vous, are obtained by using adequate dynamic priorities as defined in [AGS00, 190].

Concerning the exploration strategies, reduction heuristics such as partial-order reduction or some form of symmetry reduction are already incorporated in the exploration platform. More specific heuristics may be added depending on the needs of particular application domains.

7.1.1.2 Static Analysis for State Space Reduction

It is recognized that static analysis techniques are useful for alleviating the state space explosion problem occurring in model-checking. Static analysis is used to detect facts which can be exploited during the model-checking phase by identifying a set of semantic level states into a single state by preserving a class of properties or a particular user defined property.

Static Analysis Derived from Code Optimization. This category of static analysis techniques has been particularly investigated in the context of conformance test generation for telecommunication protocols [BFG00, 49] during the PhD Thesis of Lucian Ghirvu [26].

For example, cone-of-influence reduction or slicing has been proposed in [Kur94]; it consists in a backwards program analysis detecting a superset of variables and instructions which may influence a given set of target variables or control points, called an observation, followed by the elimination from the initial system specification of all other variables and instructions. The reduced system strictly preserves any property expressed in terms of the set of observations.

Another successful technique considered is dead variable elimination. This technique consist in detecting and assigning a predefined value to variables whose values at some control point are not used in the future. This method turned out to be crucial for dealing successfully with real specifications: even such a simple analysis technique often drastically reduce the size of the state-space of the model, especially in the case where the system specifications are not written by hand but automatically generated from user level formalisms. Concrete results on some benchmarks are given in table 7.1

Example		reachable # states	live quotient # states	reduction factor
Token Ring Protocol	3 stations	20,953	1,903	11.0
	4 stations	550,153	17,887	30.7
	5 stations	14,408,801	163,775	87.9
File System Application	2 files	619,400	17,472	35.4
	3 files	n.a.	313,488	n.a.
Pragmatic General Multicast	2 messages	202,830	122,271	1.6
	3 messages	3,687,521	1,148,715	3.2
Bounded Retransmission Protocol	3 paquets	394,610	198,981	1.9
	4 paquets	678,192	315,003	2.1
Ariane 5 Flight Program		n.a.	2,201,760	n.a.

Table 7.1: State space reduction obtained by static analysis.

Abstraction of Communication Buffers An important source of state explosion is the communication between processes via asynchronous signals stored in buffers which are a priori unbounded. Moreover, these situations really occur, especially in open systems in which no assumption on the speed of the environment is made. Several techniques have been developed to overcome this problem, for example symbolic techniques using regular expressions to identify a possibly infinite set of equivalent buffer contents [ABJ98]. Nevertheless, this method is applicable only in the context of lossy buffers.

In the context of the PhD Thesis of Manuel Aguilar [19], we have used static analysis to reduce the size of the communication buffers by means of a particular partial order reduction which privileges for a particular set of buffers always a read-action over a write action.

Static analysis is used in a first step to refine the initial coarse grained flow-graph in which we determine *conflicts* which correspond to the existence of several alternative communication paths form a *conflict source* to a *conflict target* process. In a second step to determine a set B of buffers such that in the refined flow graph, in every cycle and in every path leading from a *conflict source* to a *conflict target*, there is at least one buffer in B. We have shown that minimizing the size of all buffers which are not in B by privileging always reading over writing preserves reachability of process states. We have implemented the minimization, corresponding to a particular partial order reduction using the notion of priorities between process instances introduced in IF (see section 5.2.1).

This reduction is particularly interesting in the case where it allows to reduce an infinite state space (due to a buffer with unbounded length) to a finite state space.

7.1.1.3 A Front-end for Validation of UML

The translation of UML models to IF and the validation techniques presented in section 5.2.3 are implemented in an extended version of the IF toolset, called IFx. The overall architecture is shown in figure 7.2. It allows a designer to simulate and verify UML models and observers developed in third-party UML editors[1] and stored in XMI format.

In a first phase, the tool takes as input a UML model and generates an IF specification and a set of IF observers by applying the translation scheme presented in section 5.2.3. During this phase, a first sanity check is performed on the model and results are provided in the form of compiler warnings and errors. They concern action syntax, timing annotation syntax, type errors, etc.

In a second phase, the tool drives the back-end IF simulation and verification tools, and translates the validation results back to the level of the original model. The idea is to make the back-end formalism and tools invisible to the designer, but also to enhance the functionality of the IF toolbox by providing more complex interactive simulation features like conditional breakpoints, scenario persistence, custom views for the system state, etc.

Using the IF tools as an underlying engine gives access to all state space reduction and analysis techniques existing in IF: *static analysis* and *partial order* optimizations for state-space reduction, *symbolic* model exploration, model minimization and comparison [143]. The use of reduction techniques improves tremendously the scalability of the tools, which is an essential feature in the context of UML where often large design models are manipulated.

The most complete publication on the extended toolbox can be found in [59]. The toolbox has been applied on four case studies in the context of the OMEGA project. Two of them are presented in some details in section 5.2.4 and the two other are described in [281] or on the Omega website `http://www-omega.imag.fr`

[1]The CASE tools that have been tested for compatibility are: Rational Rose Enterprise Edition 2002 / Unisys Rose XMI Add-in 1.3.6 and I-Logix Rhapsody Developer Edition, v4.1, v4.2 and v5.2

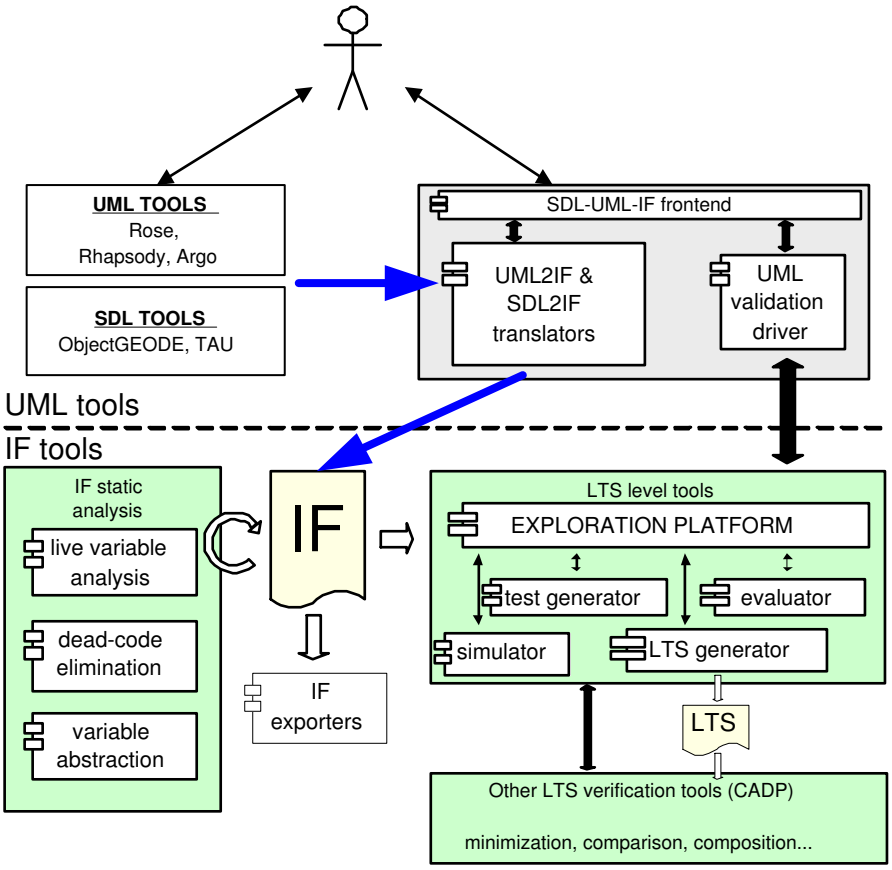


Figure 7.2: Architecture of the IFx validation toolbox.

7.1.2 Verification of Parameterized Networks

A parameterized network is a family of systems of the form $S_1, \| \cdots, S_n$ where n is the parameter of the family. Many distributed algorithms are described as a parameterized network. Nancy Lynch's book, *Distributed Algorithms*, contains a rich variety of such networks. We are interested in the verification of such network and more specifically in their automatic verification. It has been shown by Apt and Kozen (1986) that the verification problem for parameterized network, that is, the question whether $S_1, \| \cdots, S_n \models \varphi$, for all $n \in \omega$, is undecidable even in the case where each S_i is a finite state transition system. Thus, there is no hope to find a sound and complete algorithm for solving this question. Therefor, we are looking for sound verification methods based on abstraction techniques. We follow two approaches. The first one is based on monadic abstractions while the second is based on the computation of transitive closures of regular relations. Next, we explain these approaches and our results.

7.1.2.1 Monadic Abstractions

This approach consists in computing a finite-state system that is a safe abstraction of $S_1, \| \cdots, S_n$, for any $n \in \omega$. In other words, any behavior of an instance $S_1, \| \cdots, S_n$ of the family is represented by a behavior for the finite-state system. To do so, we have to address two problems:

1. How to find the abstraction function?
2. How to automatically compute the finite-state abstract system?

We have identified a class of parameterized networks whose transition relations can be expressed as WS1S formulae (weak monadic second-order logic of one-successor) and answered these questions for this class of networks. WS1S is a logic that allows to describe regular sets of finite words. It has been shown by Büchi in 1962 that there are effective semantic preserving transformations from WS1S to finite automata and back. This will be the main feature that will us to automatically compute the finite state abstraction. In summary our approach consists of the following steps:

1. Representing the infinite family of fair networks $S_1 \| \cdots \| S_n$ as a single fair transition system $\mathcal{S}$ whose variables range over finite sub-sets of ω and whose transitions are expressed in WS1S, the weak second-order logic of one-successor (e.g. [Tho90]). We call such systems fair WS1S transition systems.
2. Constructing an abstraction relation that maps the states of the WS1S transition system $\mathcal{S}$ to abstract states which are valuations of boolean variables. We present a generic abstraction relation for verifying universal properties.
3. Automatically constructing a finite abstract system $\mathcal{S}_A$ that is an abstraction of $\mathcal{S}$ which implies that every computation of $\mathcal{S}$ can be mapped to a computation of $\mathcal{S}_A$. Moreover, we construct an abstract formula ψ_A such that if $\mathcal{S}_A$ satisfies ψ_A then, using the preservation results of [LGS$^+$95, DGG97], we can deduce $S_1 \| \cdots \| S_n \models \forall p_1 \ldots, p_k \leq n : \psi(p_1 \ldots, p_k)$, for every $n \in \omega$.

4. Since the abstract system is finite, we can use model-checking to verify that it satisfies ψ_A. However, verifying progress properties using abstractions often fails because of infinite computations in the abstract system that do not correspond to fair infinite ones in the concrete one. To mitigate this problem, we augment the abstract system with *safe* fairness conditions, that is, conditions that only remove infinite computations that do not correspond to concrete ones. We present two techniques for synthesizing fairness conditions from the concrete system that can be safely added to the abstract one:

   (a) An algorithm that given a WS1S formula characterizing a ranking function computes pairs of sets of transitions expressing strong fairness conditions that are *guaranteed* to hold for the parameterized network, and hence, abstractions of them can be safely added at the abstract level.

   (b) A method that allows the generation of fairness conditions at the abstract level from the fairness conditions of the parameterized network. In particular, we discuss which kind of weak/strong fairness can be lifted from the concrete to the abstract level.

Our results are published in [BBLS00, BLS00, 120, 46].

7.1.2.2 Transitive Closure of Regular Relations

The starting point of this approach is the same as for monadic abstractions, namely that the configurations of a linear parameterized network can be seen as words over a finite alphabet. Hence, a set of configurations can be modelled by a regular set and a natural choice to represent the transition relation is *finite-state transducers*.

Taking finite-state automata and transducers to describe infinite sets of states and their operational evolution is, in general, not sufficient when doing state exploration. To capture all reachable states, one needs to characterize the effect of applying a transducer an arbitrary number of times, in other words, one needs to compute $\mathcal{T}^* \circ \mathcal{A}$, where $\mathcal{A}$ characterizes the initial states and $\mathcal{T}^*$ the transitive closure of $\mathcal{T}$. In this paper, we consider the slightly more general problem of calculating the transducer $\mathcal{T}^*$ instead of the automaton representing $\mathcal{T}^* \circ \mathcal{A}$. In general, $\mathcal{T}^*$ is not finite-state anymore (and neither $\mathcal{T}^* \circ \mathcal{A}$). Indeed, it is undecidable whether the iteration of a given transducer is regular.

The transitive closure of a finite-state transducer can be represented as an infinite-state transducer $\mathcal{T}^{<\omega} = \bigcup_{i \in \omega} \mathcal{T}^i$, the union of all finite compositions of $\mathcal{T}$. We developed a method that constructs a finite quotient of $\mathcal{T}^{<\omega}$ by identifying states that are equivalent in some way [172, 53]. We briefly explain this method below.

Our algorithm works by computing successively the approximants $\mathcal{T}^{\leq n} = \bigcup_{0 \leq i \leq n} \mathcal{T}^i$ for $n = 0, 1, 2, 3, \ldots$, while attempting to *accelerate* the arrival at a fixpoint by collapsing states. This quotienting is based on a novel behavioral equivalence defined in terms of *past* and *future bisimulations*. This equivalence has to be infinite to collapse the infinite-state transducer $\mathcal{T}^{<\omega}$ to a finite one. Therefore, one is faced with the problem of effectively computing and representing a suitable infinite equivalence. To solve this problem, we first

identify sufficient conditions on an approximant $\mathcal{T}^{\leq n}$ for its states (which are also states of $\mathcal{T}^{<\omega}$) to be equivalent as states of $\mathcal{T}^{<\omega}$. Then we show that the equivalence of two states of $\mathcal{T}^{\leq n}$ induces the equivalence of infinitely many states of $\mathcal{T}^{<\omega}$.

We illustrate the underlying intuition on a small example in which sets of unbounded natural numbers are represented as automata over the symbols 0 and *succ*. The transitions we consider are given by the function α, defined inductively by $\alpha(0) = even$ and $\alpha(succ(x)) = \neg(\alpha(x))$. It computes the parity *even* or *odd* of a number; $\neg$ is a function that toggles parities. Consider the transition relation $\rightarrow$ that corresponds to a single step in the evaluation of this recursive definition. Fig. 7.3(a) gives a transducer, $\mathcal{T}_\alpha$, that represents this transition relation. The slash (/) is used to separate the input symbol from

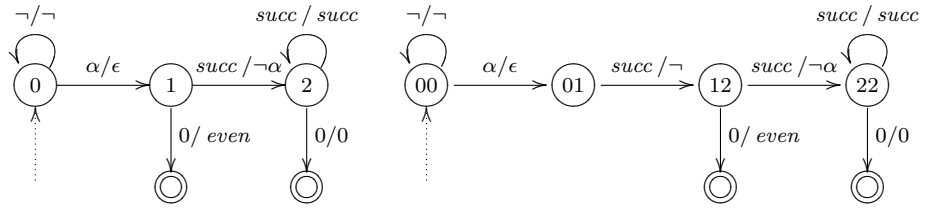


Figure 7.3: Left (a): The transducer $\mathcal{T}_\alpha$. Right (b): Its product $\mathcal{T}_\alpha^2$.

the output symbols; ϵ denotes the empty string. Note that by the self-loop on state 0, the transducer leaves any leading occurrences of the symbol $\neg$ unchanged, and similarly for the trailing occurrences of *succ* before the final 0.

To start approximating $\mathcal{T}_\alpha^{<\omega}$, consider the product transducer $\mathcal{T}_\alpha^2$ shown in Fig. 7.3(b): It moves the symbol α over one more occurrence of *succ*, while turning it into a $\neg$, as reflected by the edge from state 01 to 12. In every next product transducer $\mathcal{T}_\alpha^3, \mathcal{T}_\alpha^4, \ldots$, an additional such *succ* /¬-edge will appear. Clearly, the limit transducer $\mathcal{T}_\alpha^{<\omega}$, the union of all approximants, is going to have infinitely many states. On the other hand, the combined effect of the ever-growing sequence of *succ* /¬-edges would be captured by a simple loop if states 01 and 12 were identified. Collapsing $\mathcal{T}_\alpha^{<\omega}$ in this way, we can hope for a finite quotient. To do so, we need to address the following questions: First, how can we justify equating pairs of states like 01 and 12, which are obviously semantically different in that they realize different transductions, i.e., what is the equivalence notion on $\mathcal{T}_\alpha^{<\omega}$ employed for quotienting. Secondly, how to compute the quotient without prior calculation of the infinite $\mathcal{T}_\alpha^{<\omega}$?

As for the first point, we must ensure that identifying states in the quotient does not introduce transductions not already present in $\mathcal{T}_\alpha^{<\omega}$. Equating 01 with 12 in the above example, consider the run through the "collapsed" transducer that goes from 00 to 01 (or rather to the new state obtained by collapsing 01 and 12) and then continues from this state as if continuing from 12. Exploiting the equation 01 = 12, this run is introduced by the collapse. Even though the states 01 and 12 are semantically different, as observed above, identifying them does not change the *overall* semantics of $\mathcal{T}_\alpha^{<\omega}$, as there *exists* another state that "glues" together the past of 01 and the future of 12, namely state 1 of $\mathcal{T}_\alpha$. Another class of runs that are introduced by the collapse are those that go from 00

to 12 and then continue as if continuing from 01. But also in this case, there is a state in $\mathcal{T}_\alpha^{<\omega}$ that glues (this time) the past of 12 to the future of 01, although it has not been constructed when considering $\mathcal{T}_\alpha^{\leq 2}$. This state is 012 and would enter the scene as part of $\mathcal{T}_\alpha^3$, when constructing the next approximant. We formalize these ideas as follows: States q_1 and q_2 may be identified if there exists a past bisimulation P and a future bisimulation F such that the pair (q_1, q_2) is both in the composed relation $P; F$ and in $F; P$, thus ensuring the existence of both "gluing" states. Indeed, we will require that the bisimulations *swap*, i.e., $F; P{=}P; F$. So it will be enough to show that (q_1, q_2) is in either one of the composed relations. The situation is sketched in Fig. 7.4, containing $\mathcal{T}_\alpha$ and $\mathcal{T}_\alpha^2$ and where the state 012 of the not-yet-constructed $\mathcal{T}_\alpha^3$ is drawn by a dotted circle.

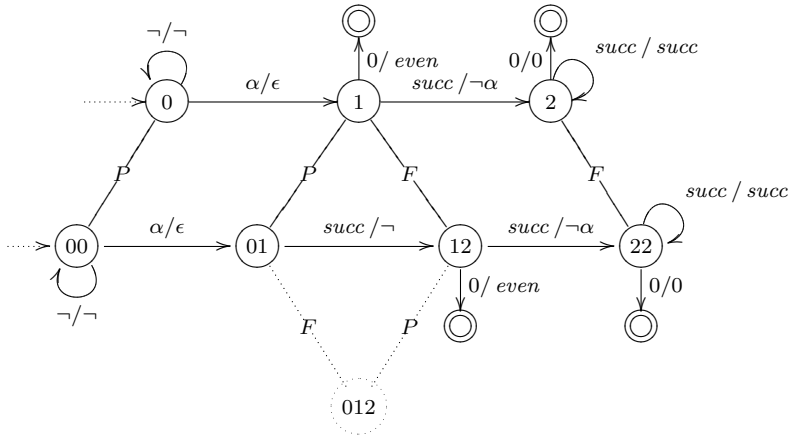


Figure 7.4: $\mathcal{T}_\alpha$ and $\mathcal{T}_\alpha^2$ with past and future bisimulations

The second question is how to detect equivalent states in some approximant $\mathcal{T}_\alpha^{\leq n}$, i.e., how do we know that there exists a state *somewhere* in $\mathcal{T}_\alpha^{<\omega}$ that is past-bisimilar to one and future-bisimilar to the other? To this end we exploit the structure of $\mathcal{T}_\alpha^{<\omega}$'s states, namely that they are sequences of states from $\mathcal{T}_\alpha$. It is easily seen that bisimulations are *congruences* under juxtaposition of such sequences. In the example above, we can conclude the existence of an appropriate state without actually having to construct $\mathcal{T}_\alpha^3$: By looking at $\mathcal{T}^{\leq 2}$ only, we see that 1 and 12 are future bisimilar, whence by congruence also 01 and 012. Similarly, past bisimilarity of 12 and 012 can be inferred from 1 and 01. Again, the inferred pairs $(01, 012) \in F$ and $(012, 12) \in P$ are indicated by dotted lines in Fig 7.4. Furthermore, exploiting the congruence property allows to extrapolate the quotienting relation found on a finite $\mathcal{T}_\alpha^{\leq n}$ to the whole $\mathcal{T}_\alpha^{<\omega}$, and thus to obtain a finite quotient of $\mathcal{T}_\alpha^{<\omega}$, without calculating the limit first.

7.1.3 Verification of Software with Dynamic Memory

The constant increase in complexity of programs, and, in principal, of mission critical embedded programs, leads to implementations that use recursive data structures for efficient bookkeeping and fast searching of data e.g. AVL trees, hash tables, etc. Such programs

need to run on platforms with limited amounts of memory, requiring *efficient memory management*.

In order to address this issue, implementors use low-level language primitives such as dynamic memory allocation, pointer referencing, etc. However these mechanisms are highly error-prone, perhaps the greatest danger of imperative programming. Examples of bad run-time behavior due to incorrect uses of pointers include: null pointer dereferencing (segmentation faults), memory leaks, etc. The first goal of this research is therefore ensuring the **absence of run-time errors**.

Explicit memory de-allocation may cause fragmentation, leading to an inefficient use of the available space. One way of easing the implementor's task is to provide a garbage collection mechanism, thus avoiding explicit de-allocation and fragmentation. However, embedded programs need to meet tight timing constraints, which could be delayed by an arbitrary scheduling of the garbage collector task. Understanding the structure of the program heap will therefore help optimizing the program in order to ensure that responses occur within acceptable delays. Hence, the second goal of our work is **program optimization**.

This activity is funded by projects MADEJA (§15.2.13) and DYNAMO (§15.2.9).

7.1.3.1 Dynamic Data Structures

Programs handling pointers usually perform selector updating assignments that generate extremely complicated graph structures. Understanding what the program does in this case is in general highly time-consuming. Even short programs, such as the in-place list reversal or the Schorr-Waite markup algorithm have non-trivial correctness proofs.

The use of dynamic data structures in programs is a source of undecidability even for the most basic problems, such as reachability of a control point. As a consequence, all existing approaches are based on partial decision procedures. The method we propose follows the general pattern used in verification of infinite-state systems, namely:

- define a set of initial states $Init$,
- define a set of erroneous states Bad,
- iteratively compute $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad)$,
- if $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad) \cap Init \neq \emptyset$ report an error.

In particular, we use *exact symbolic representations* of sets of states by means of logics describing the topology of the heap. Moreover, we focus on decidable logics, in order to be able to answer the question of the fourth point. However, due to the undecidable nature of the problem, the set $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad)$ is not definable in the same logic $\mathcal{L}$ used to define Bad, but in $\mathcal{L}$ augmented with higher-order constructs (e.g. transitive closure of binary relations). We deal with this problem by finding classes of programs for which $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad)$ is still definable in $\mathcal{L}$. Another possibility, considered as future work, consists in systematically over-approximating $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad)$ by a formula in $\mathcal{L}$.

Related Work The existing literature on the analysis of programs with pointers shows two different approaches.

On one hand, there are techniques stemming from *static analysis*, which are based on abstract interpretation [CC97]. Those methods yield sound but incomplete answers i.e. all real errors are reported, but not all reported errors are real. The most visible group that adopted this method has developed the *Shape Analysis* method [SRW98]. Their approach consists in abstracting a heap graph by a first-order formula interpreted over a three-valued domain [SRW02]. In principle the abstract domain is finite, which allows effective computation of fixed points in the abstract domain. The abstract domain can be refined by adding new predicates, however iterated refinement is not guaranteed to terminate.

Second, there are deductive methods based on precise symbolic representations of sets of states, using adequate logics. We distinguish here the work on *Separation Logic* [Rey02], [ORY99], a logic with a relatively simple formal proof system, yielding fairly readable correctness proofs. On the other hand, the logic is undecidable and verification is carried out manually. Another approach uses *Monadic Second Order Logic of k Successors* (SkS) to reason about recursive data structures (lists, trees) [KS94]. This method has a higher degree of automation, since SkS has an elegant decision method based on tree automata [Rab69], however the user still has to annotate the program with the loop invariants needed in order to prove a certain property.

Semantic Models We started by defining a simple imperative programming language consisting of three types of statements: creation of a new memory cell (`p := new`), selector updating (`p.car = q`) and pointer assignment (`p := q.cdr`). These statements are organized in control structures, where conditionals are pointer comparisons for equality (`p = q`). This elementary language (SPL for Simple Pointer Language) captures most of the functionality of high-level pointer handling languages.

In [145], we study three different semantics for these instructions. We start from a low level concrete semantics, denoted by $[\![\cdot]\!]_1$ in Fig. 7.5, where the state of a program is described by a stack and heap. We then show that this semantics does not distinguish between states that are equal up-to the identity of memory locations (addresses). In other words, renaming of locations induces an equivalence relation that is a bisimulation (Diagram (1) in Fig 7.5).

This semantics is, however, sensitive to garbage and dangling pointers from which we would like to abstract. Therefore, we present a storeless semantics $[\![\cdot]\!]_2$ where a heap is modeled as a structure that is a set of regular languages satisfying additional conditions. The idea behind this semantics is that each heap element can be represented by a Rabin-Scott automaton with a single accepting state.

The third semantics we introduce is an axiomatic semantics, i.e., a Hoare logic-like proof system. The main feature of this proof system is that it allows to prove properties of programs that are insensitive to garbage. Thus, our proof system is compatible with garbage collection, a problem difficult to deal with, as pointed out by Calcagno and O'Hearn in [CO01].

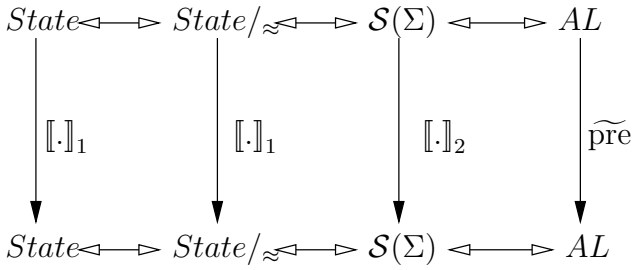


Figure 7.5:

Specification Logics The study of state representations naturally led to the definition of specification logics suitable for describing (possibly infinite) sets of graphs. As we show in [146], describing pointer aliasing suffices to characterize heap graphs up to isomorphism. Therefore, we take as primitive the alias relation between paths in the graph.

A *path* in a heap graph a path is a sequence of the form $u.\sigma_1.\sigma_2\ldots$, where u is a program variable and $\sigma_1, \sigma_2, \ldots$ are selector fields (e.g. `car`, `cdr`). The fact that two paths π_1 and π_2 are *aliased* in some node is denoted as $\pi_1 \Diamond \pi_2$. Boolean combinations of alias propositions describe finite portions of heap e.g. $u.cdr.cdr \Diamond u.cdr.cdr.cdr.cdr$ describes a cyclic list with a prefix of size 2 and period of size 1 or 2. This logic (named pAL for Propositional Alias Logic) can define a given graph up to isomorphism, and therefore finite sets of graphs can be defined. The main problem is that infinite sets of graphs can only be defined by characterizing all nodes up to a finite frontier and leaving everything else unspecified. For instance the formula $u.cdr.cdr \Diamond v.cdr \wedge \neg u.cdr \Diamond v \wedge \neg u.cdr \Diamond v.cdr$ defines the set of all heaps in which there are at least four nodes connected like in Fig. 7.6. The logic pAL is described in detail in [146].

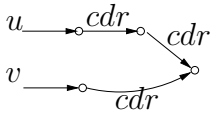


Figure 7.6:

To overcome the expressivity problems of pAL, we have designed an extension (named cAL for Counting Alias Logic) in which paths are specified by words with counters $(x_1, x_2, \ldots)$ of the form $u\sigma_1^{x_1}\sigma_2^{x_2}\ldots$. Counters are interpreted as natural numbers. Assigning values $v_1, v_2, \ldots$ to all counters, basically transforms a parametric path of the form above into a normal path in which σ_1 is repeated v_1 times, σ_2 v_2 times, etc. cAL is the logic of parametric alias propositions of the form $\pi_1(\vec{x}) \Diamond \pi_2(\vec{x})$ in which first order quantification over counters is used. For instance, the formula $\exists x\ u.cdr^x \Diamond v$ states that v is reachable from u by a sequence of cdr's. This logic can be used to define some simple recursive data structures e.g., non-circular singly-linked lists or doubly-linked lists. cAL is described in detail in [272].

In addition to describing heap topologies, the logic cAL is a tool for **quantitative reasoning**, providing a direct embedding of the graph domain into a numeric domain. Each sub-path of the form $\sigma\sigma\ldots\sigma$ is associated its length, which is the value of a counter, and each graph is associated a numeric vector. In addition to the alias propositions of cAL, we allow first-order arithmetic formulas, being able to reason about arithmetic relations between lengths of paths in the heap. For instance, in a list reversal program one would like to specify that the length of the input list equals the length of the output list. In order to prove this property, one needs the invariant that the sum of the lengths of all lists handled internally by the reversal program is a constant, equal to the size of the input list.

Semi-algorithmic Verification In order to use the pAL and cAL specification logics in an effective verification method, we have studied the decidability and the invariant generation issues. The latter is necessary to define the set of states leading to bad states i.e., $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad)$, whereas the former is used to effectively decide whether $\bigcup_{i=0}^{\infty} \mathrm{pre}^i(Bad) = \emptyset$.

Decidability Issues Logics interpreted over graphs are in general undecidable, take for instance first-order logic and monadic second-order logic of graphs. The problem of finding decidable logics over arbitrary graphs is the problem of sacrificing expressiveness for effectiveness. Another way of dealing with generally undecidable logics is to restrict the class of structures over which they are interpreted. In studying decidability of our specification logics for heap, we have explored both directions.

pAL This logic is decidable, the satisfiability problem being NP-complete. In [146] we developed two proof systems for pAL: a natural deduction system and a tableaux system. Both systems are sound and complete. Decidability is a consequence of the fact that any tableau procedure terminates eventually.

cAL This logic is undecidable in general, when interpreted over graph or dag structures [272]. Decidability of the purely existential subset is subject of ongoing work (in collaboration with Thibault Henin, intern). The logic is decidable when interpreted over trees and lists. The result on lists uses the decidability of a fragment of the arithmetic with addition and divisibility $\langle \mathbb{N}, +, |, 0, 1\rangle$[Rob49], which we have studied separately in [148]. The divisibility constraints occur due to the presence of cycles in lists. For instance, $\varphi(x, y) = u\Diamond u.cdr^x \wedge u\Diamond u.cdr^y \wedge \forall z < x \; \neg u\Diamond u.cdr^z$ is satisfied by all x and y such that x is a positive integer divisor of y.

Invariant Generation We have studied the invariant generation problem in the singly-linked list case, the case of trees being considered as future work. This work has been reported in [149]. The main idea is that the state space of a program with lists can be partitioned in a finite number of equivalence classes, each equivalence class being characterized by an open formula of cAL (parametric graph). Moreover, the transition relation

between equivalence classes can be given in terms of arithmetic constraints. We have thus reduced the problem of computing transitive closure of the transition relation in a program handling lists to the acceleration of integer counter automata.

In general, the input-output relation of a counter automaton cannot be expressed in a decidable logic e.g. Presburger arithmetic. However this is the case of *flat* automata with *affine* transition relations i.e., of the form $x' \leq y + c$, $c \in \mathbb{Z}$ [CJ98b]. As ongoing work, we identify interesting classes of programs with lists for which the transition relation can be encoded as a flat affine automaton, thus making verification possible without the use of information-losing abstraction. The idea of approximating the transition function of arbitrary counter automata using decidable theories, is considered as future work.

7.1.3.2 Region-based Memory Management

A particular mechanism that is quite problematic in real-time embedded contexts is automatic dynamic memory management. One problem is that execution and response times are extremely difficult to predict in presence of a garbage collector. There has been significant research work to come up with a solution to this issue, either by building garbage collectors with real-time performance, or by using a scoped-based programming paradigm. Another problem is that evaluating quantitative memory requirements becomes inherently hard. Indeed, finding a finite upper bound on memory consumption is undecidable. This is a major drawback as embedded systems have (in most cases) stringent memory constraints.

We are working on a method that attempts to tackle these two issues. The idea consists in using escape analysis to automatically synthesize a memory organization that associates a memory region [TT97] with computation units (typically methods or threads). The method also statically predicts the amount of memory to be dynamically allocated by the program.

Escape Analysis Escape analysis is a static analysis technique for object-oriented programs that aims at conservatively determining if an object escapes from or is captured by a method. Intuitively, an object escapes a method when its lifetime is longer than the method's lifetime, so it can not be collected when the method finishes its execution. An object is captured by the method when it can be safely collected at the end of its execution.

Several approaches to escape analysis for Java have been proposed, most of which aim at allocating objects on the stack, and removing unnecessary synchronizations. Blanchet's [Bla03] technique works on the bytecode, which brings in an additional complexity due to the stack-based model. Rinard's [WR99], Choi's [CGS+99], and Rugina's [CR04] methods use points-to analysis to determine if an object escapes a method through a path in the points-to graph. Gay [GS00] proposes a fast but very conservative escape analysis, based on solving a simple system of linear constraints obtained from a Static Single Assignment (SSA) form of the program.

In [245], we presented an algorithm for escape analysis inspired by, but more precise than, the one proposed by Gay. The primary purpose of our algorithm is to produce useful

information to allocate memory using a region-based memory manager. The algorithm combines intra-procedural variable-based and inter-procedural points-to analysis, without constructing the full points-to graph. This is a work in progress towards achieving an application-oriented trade-off between precision and scalability. We carried out experimental results of a first prototype on the JOlden benchmarks. Experiments showed our prototype is more precise than Gay's analysis and much faster than the analysis implemented in MIT's Flex compiler [2] and McGill's SOOT[3].

A prototype technique and tool for synthesizing regions from escape analysis results is described in [184, **?**]. Program instrumentation with region-based memory management is still work in progress. Anyway, preliminary experiments revealed a gain of 20% of total utilized memory, when using GC together with region- based manager, w.r.t. GC only, even if the actual region-allocated memory is about 5%. We are currently working on generating bytecode instrumented for the RTSJ API implemented by the JamaicaVM [4] from AICAS.

Estimating Dynamic Memory Use Dynamic memory estimation for object-oriented programs is intrinsically hard. Indeed, finding a finite upper bound on memory consumption is undecidable. Some approaches for analyzing memory consumption have been proposed. These techniques either produce linear approximations, or formulas which are difficult to evaluate, or rely on a type system and type annotations in the program.

In [**?**] we have developed a novel static analysis approach for over-approximating the amount of memory dynamically allocated by (Java-like) imperative object-oriented programs. Our technique is based on computing linear invariants that relate program variables to memory consumption. Roughly speaking, the amount of consumed memory is the number of integer points that satisfy the invariant. This number is given in a parametric form as a polynomial where unknowns are method input parameters. Our method does not require annotating the program in any form and does produce non-linear parametric upper-bounds on memory usage. The polynomials are to be evaluated on program (or method) inputs to obtain the actual bound.

We have developed a prototype tool that allowed us to experimentally evaluate the accuracy of the method on several Java benchmarks. The results were very encouraging, even producing exact results in many cases. We are currently improving the tool in order to thoroughly test the complete approach.

7.1.4 Verification of Numerical Programs by Abstract Interpretation

Concerning abstract interpretation, our research mainly concerns Linear Relation Analysis, for obvious historical reasons.

[2] www.flex-compiler.lcs.mit.edu
[3] www.sable.mcgill.ca/soot
[4] www.aicas.com/sites/jamaica.html

Linear relation analysis was introduced in Halbwachs'first thesis (1978), as an abstract interpretation finding invariant linear inequalities between numerical variables, at each control point of a program. It was reused, later, for the verification of synchronous programs and timed and hybrid systems. In particular, for its application to Lustre programs — which don't have any obvious control structure —, an original technique for building the suitable control structure to prove a given property, was defined in Bertrand Jeannet's thesis (2000) and implemented in the verification tool NBAC [65].

Our current work on these topics take place in the "APRON" project of the ACI "Sécurité Informatique" (see §15.2.4)

7.1.4.1 Algorithmics of Convex Polyhedra.

Convex polyhedra are often used to approximate sets of states of programs involving numerical variables. The manipulation of convex polyhedra relies on the so-called *double description*, consisting of viewing a polyhedron both as the set of solutions of a system of linear inequalities, and as the convex hull of a *system of generators*, i.e., a set of vertices and rays. The cost of these manipulations is highly dependent on the number of numerical variables, since the size of each representation can be exponential in the dimension of the space. In [290, 203], we proposed to reduce the dimension by detecting that a polyhedron is in fact a *Cartesian product* of several polyhedra of smaller dimensions, and we examined how to adapt the usual operations to work on such factored polyhedra. This approach provides significant improvements, but it highly dependent of the choice of variables: for instance, a hypercube can be highly factorised (as a product of intervals) in a suitable base, but a simple rotation can forbid any factoring. So, we extended the approach by allowing the best base to be chosen before performing the most expensive operations. This work constitutes David Merchat's Thesis [30], and was implemented as a layer above the "Parma Polyedra Library" [BRZH02].

7.1.4.2 Exact Widening in Linear Relation Analysis.

Improving the precision of the "widening" on polyhedra is also a hot topic [BHRZ03]. It was neglected for a while, because the precision can always be improved by delaying the application of the widening. Now, on one hand such a delaying is often expensive — since the simplification performed by the widening occurs later —, and on the other hand, there are many cases where the suitable delay is function of constants appearing in the analysed program: when these constants are large or symbolic, the delayed widening does not work. Instead of simply (and possibly endlessly!) looking for a better widening, we are trying to identify the cases where an "exact widening" can be applied: in some cases, techniques similar to "acceleration" [BW94, CJ98a, FS00] can be applied to compute the convex hull of the exact limit of possibly infinite iterations. The difference between widening and acceleration is the following: widening *always* allows an upper *approximation* of the limit of a sequence to be computed, by looking at several *terms* of the sequence; acceleration aims at computing, *in some cases*, the *exact* limit of a sequence, by looking at the *function*

used to compute a term from its predecessors. We are using the same idea, but without the same limitations, since we don't pretend to perform exact computations: on one hand, the convex hull approximation is still used, and on the other hand, the widening is applied when nothing else is possible. This research constitutes the ongoing thesis work of Laure Gonnord.

7.1.4.3 An Abstract Lattice for Addresses.

A last topic concerning abstract interpretation was suggested by our cooperation with ST-Microelectronics on the modeling and verification of "systems on chip". In these systems, an important goal is to verify the correctness of synchronisation algorithms, and in particular of the access rules to devices (memory areas, sensors, ...) known by their addresses. While these addresses are generally integers, they constitute a restricted data type, in the sense that many operations (e.g., sum, product) don't make sense on addresses. While usual interpretations on numbers (e.g., linear relation analysis) can be used to analyse these addresses, it is more efficient to design a specific lattice, restricted to meaningful operations and properties ((dis-)equalities, ordering, range) of addresses. This work is starting this year (Mathias Péron's ongoing Master thesis).

7.1.5 Formal Construction and Verification of Lustre Programs

Background

This section is devoted to verification techniques specifically devoted to synchronous programs, and particularly to Lustre programs. This topic has been studied for long at Verimag: in particular, the model-checker Lesar [RHR91] and the abstract interpreter NBac [JHR99] were developed and applied in the past, and are still our basic verification tools. In addition to automatic verification, the interactive proof and correct construction of Lustre programs was also studied [DC00].

Hereafter, we present the recent works on these topics, which concern the verification of programs containing the new array primitives (see §6.1), the continuation of research on correct construction, and some more general work about the expression power of synchronous observers used for specifying properties.

7.1.5.1 Verification of Lustre Programs with Arrays

In section 6.1, we describe a recent extension we made to Lustre: a set of constructs that allow to structure dataflow programs with *arrays*. Array manipulations are then described with *iterators* inspired from functional languages (map, fold, etc.).

This extension has been validated on several industrial case studies, and it appeared very convenient for the description of large regular systems. Code generation is also much better, since it is quite easy to produce imperative loops from functional-like iterators.

The last point to be studied is the verification of large programs with arrays. For the application domains we have in mind, the size of all arrays in a program are always known statically, so the program can be *expanded* in such a way that each array is transformed into

individual variables. Technically, this allows the use of all validation tools that are available for Lustre without arrays. But, of course, array expansion produces huge programs, and verification is often impossible.

Moreover, forgetting about the structure of programs is clearly not a good idea, since it may help a lot in decomposing the properties we want to verify on programs.

Part of the PHD work of Lionel Morel [224] is devoted to the definition and implementation of a prototype called GOUPIL that may "*slice*" programs and properties according to their array structure. In Lustre, properties are described in the language itself using *observer* components, so they are part of the program. GOUPIL is based on two ideas:

- First, if the program is structured with "map" iterators, it means that the elements of an array are treated independently of each other. In this case, slicing a program P manipulating arrays $T_1, T_2, \ldots$, and a property ϕ on these arrays gives a set of new independent proof obligations concerning *elements* of the arrays.
- Second, it the program is structured with "fold" iterators, it means that the elements of an array are not independent. Proving something on the result of such an iteration is as hard as proving general FOR iterations in imperative languages (since the size of arrays is known statically). We do not aim at providing a general technique for the validation of iterative programs (like invariant generation), since the use of arrays in dataflow programs is very often a way of structuring programs rather than data. It means that the complex iterative algorithms that may be involved are hidden in small well identified components for which other techniques can be used. Concerning the dataflow program itself, requiring that the property to be proved on the result of an iteration be indeed an invariant of this iteration is enough to cover most of the cases.

  From a fold iteration, we extract two proof obligations: one for the initialization, and one expressing that the property is indeed an invariant. Of course this reinforces the initial property.

All proof obligations are rewritten in Lustre, so that all the available tools (ranging from pure Boolean model-checking to assisted tools based on PVS) can be used to prove them. GOUPIL also allows to replace a component by its *contract* (see this point in section 6.3). GOUPIL is currently used in the context of the ASSERT project (15.1.4).

7.1.5.2 Formal Construction of Lustre Programs

Though Lustre/Scade has become a very popular system design environment, and though several formal validation tools have been devoted to it, some of them originating from VERIMAG, it still lacks a systematic proved-by-construction development methodology à la B, Z and VDM. The Flush project, where Flush stands for "Formal Lustre System Helper", is intended to study the possibility of filling this providing such a method.

The project is based on the Gloups tool [DC00] which provides Lustre with general purpose formal proof facilities by translation to PVS, the proof assistant developed at SRI, but it is not uniquely bound to that tool and is fully compatible with the use of other tools like model-checkers (Lesar, Prover) or abstract interpretation tools (Nbac). It consists of automatically generating refinement proof obligations in the form of Lustre "proof node"

that can then be discharged thanks to some proof tool. Elements of this methodology can be found in [209].

7.1.5.3 Non-Deterministic Synchronous Observers

The standard way for expressing (safety) properties in synchronous programming, is by using *synchronous observers*, i.e., programs behaving as a acceptors of traces: such an observer perceives the behavior of all the variables or signals relevant for the property, and computes a single (Boolean) output, which is true as long as the observed traces satisfy the property. A similar notion is used in IF (cf.§5.2.2), but in synchronous languages, the synchronous composition is the default one, so an observer is a standard program. Although defining sets of behaviors, the usual observers are deterministic programs (as all synchronous programs). However, it can be useful to consider non-deterministic observers, at least in two cases:

- to translate some specification languages, and in particular some interval temporal logics, which quantify over intervals;
- to express quantitative properties, by means of counters in the observers, while using only a finite number of counters.

Such non-deterministic observers can be used only for verification, since an implicit quantification on traces is performed by the verifier (the verification succeeds if *all* the possible traces satisfy the property), but not as oracles for testing nor for monitoring, since some traces can be non deterministically accepted or rejected by the observer. This is why it is interesting to investigate what is the limit of expressivity between deterministic and non-deterministic observers. In [187][278], we used these ideas to translate a safety fragment of P. Pandya's duration calculus QDDC [Pan01], into non deterministic Lustre observers with counters, and we identified a more restricted fragment which only needs deterministic acceptors. During the SAFEAIR-II project (see §15.1.13), we applied a similar approach to translate the specification language of the tool BOOST (an industrial model-checker developed at Infineon) into Lucky specifications of testing scenarios.

7.2 Testing

Testing and formal verification are complementary validation methods: their goals are rather different, since testing aims at finding bugs and does not, in general, provide any guarantee of absence of bugs; moreover, testing is often applicable to programs that are difficult to verify such as:

- programs with significant numerical aspects, which cannot be abstracted away;
- programs whose source code is hardly exploitable by verification tools — e.g., because the code is partly written in low level languages such as C, assembly languages etc;

- programs which are used as specifications, and are implemented by hand (e.g., distributed implementations from centralized programs).

A key issue in testing is the generation of the experiments to which the program or system under test (SUT) is confronted. Roughly speaking such an experiment is a sequence of stimuli sent to the SUT with a verdict (often called "oracle") that states whether the test passes or fails. The verdict depends on the SUT's reaction to the stimuli and the expected behavior. The work on testing at VERIMAG can be structured into two main activities:

1. Automatic testing of reactive numerical programs: The main achievement here is a method for testing whether a given reactive program P running in an environment E satisfies a property φ. The main difficulties here are the following:

   - The considered programs are infinite state systems; in particular, they often heavily manipulates numerical values.
   - The test scenario generation has to be performed while executing the program and its environment in closed feedback loop as the environment behavior may depend on the program outputs. In other words, the set of relevant input sequences to the program depends on the environment behavior which in turn depends on the program.

2. Conformance testing: The problem considered here is the following. Given a behavioral specification S and a SUT P check whether P correctly implements S. Roughly speaking, P correctly implements S means that any output of P is allowed by S, that is, P allows for potentially more inputs than S but never more outputs. The work done at VERIMAG concerns the development of algorithms and tools for the automatic generation of test cases (experiments) using model-checking techniques and the extension of this framework to timed specifications and systems as well as other types of test such as robustness testing.

7.2.1 Automatic Testing of Reactive Numerical Programs

Testing reactive programs, like Lustre programs, raises some specific problems. The behavior of such a program is a sequence of reaction to inputs coming from its environment. So, a "testcase" is a *sequence* of inputs. Now, in general, these sequences are not arbitrary: they must be *realistic* according to some known properties of the environment (e.g., "in some situation, the derivative of some numerical input should stay between some given bounds"). Moreover, it is often the case that the program *controls* its environment, meaning that the input at some instant may be constrained by the previous outputs of the program: this means that, since the considered input and output values generally belong to infinite domains (numerical variables), one has to execute the program during the test sequence generation.

An automatic testing tool, called Lurette, has been designed, which is able to generate arbitrary long sequences of realistic inputs — according to some specification of the environment — and to test the program against an oracle given as a synchronous observer.

This tool is described in §7.2.1.1. Moreover, it is used to search for actual counterexamples when approximate verification fails (§7.2.1.4), and for Lustre programs debugging (§7.3.1); it was also extended to permit early execution of partially specified components (see §7.3.2).

7.2.1.1 The "Lurette" Tool

The principle of the tool is to automate as far as possible black-box testing of reactive systems. This automation mainly concerns:

- the generation of relevant (or realistic or interesting) test scenarios according to a formal description provided by the user;
- the analysis of the test results, according to a formal description of the expected properties, also provided by the user.

Let us recall that, in the considered context, test sequences cannot be generated "off-line" and sent to the program: a reaction of the program is likely to influence the relevance of the forthcoming inputs. With this feedback aspect, testing reactive programs is similar to an execution within a simulated environment, as sketched in figure 7.7: the program under test and its environment run in closed feedback loop. An other component that must be provided, is the the *oracle*, which is just a synchronous observer deciding whether the test passes or not.

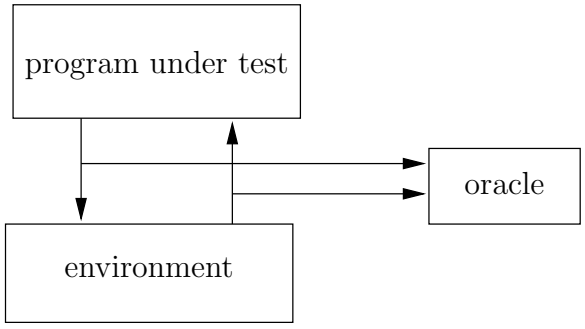


Figure 7.7: Lurette closed loop.

In the first version of Lurette — designed in the late nineties [RWNH98] — the environment was also described as an observer, written in Lustre. This approach showed some drawbacks:

- it is not always convenient for the user to think in terms of an *acceptor* of relevant test sequences. Thinking in terms of a *generator* is often more natural;
- the data-flow, declarative style of Lustre is not always suitable because the test process is often designed as a sequential, imperative scenario;

- using Lustre observers only allows to express that some sequences are possible and others are not, but not to express that some sequences are *more probable* (or realistic or interesting) than others.

To address those issues, we have defined new languages and tools for describing and generating test sequences.

7.2.1.2 Specification and Generation of Test Sequences

All the work described in this paragraph started in the PhD thesis of Yvan Roux [37].

For designing the new version of Lurette, we started from the observation that generating test sequences is just a special case of simulating a partially known reactive system. As shown in figure 7.7, the roles of the program and the environment are similar: they are two reactive systems running concurrently. The only difference is that the program is by nature executable (and, most of the time, deterministic), while the environment is only known by some properties (and is likely to be non-deterministic or even stochastic).

We have thus developed methods for describing and simulating non-deterministic or stochastic reactive systems. The first targeted application is indeed testing, but the approach is more general and can be used for other purposes, as mentioned before.

The core of the method is a language, called Lucky, for describing stochastic reactive systems as generators of constraints.

Principles. Basically, a Lucky program is an interpreted automaton where a transition represents a reaction of the system. Such transitions are labelled by:

- a constraint that must be satisfied by the current reaction,
- a weight representing the relative probability of the transition to be taken.

Concretely, a constraint is given as a Boolean expression over:

- the current values of the variables, which can be inputs, outputs, or local.
- the previous values of those variables (we note $\bullet x$ the previous value of x).

For simulation, we have to distinguish:

- controllable values (output and local) that must be generated at each step,
- uncontrollable values (inputs and previous values) which are either coming from the "outside" or the past.

Weights are basically numerical constants, but in some cases it may be convenient to use *dynamic weights*: for instance, an alive process can be described by using a looping transition labelled by a weight which decreases each time a loop is performed.

Lucky also provides a notion of transient states, which are useful for factorizing transitions and expressing conditional probabilities. They do not change the synchronous

semantics of the reactions: a complete reaction is a path of transient states between two stable states.

A test sequence is generated by finding, step by step, a feasible fair path within the automaton. Note that the inputs of the system (i.e., the output of the program under test) may influence the path search in two ways:

- constraints may depend on past values, so previous reactions of the program under test may forbid a transition;
- weights are numerical functions of past values, so previous reactions may influence the probability of a transition to be taken.

The tool associated to the Lucky language is a simulator (which is indeed the core of the tool Lurette). It uses a new constraint solver we have developed, and which handles both Boolean and linear numerical constraints.

Lucky was not designed to be a user-friendly language, but rather as the intermediate code of some abstract machine (the simulator). We have also worked on higher level languages that can be compiled into lucky (see more details in the section on language design 6.4)

Example. We illustrate the use of Lucky for describing the environment of a heater controller placed in a room with a window which is opened from time to time. The environment of this heater is the temperature of the room that comes from a sensor.

Figure 7.8 shows a Lucky that simulates the temperature in such a room. Input variables are a Boolean On, which is true if and only if the heater in on, and a rational T, which indicates the temperature outside the room.

The only output variable is the rational t, which indicates the temperature inside the room.

Local variables are the rational δ, which is used to compute the new temperature, and the integer j, which is used to count the number of steps the window remains open.

Previous variables are $\bullet t$ and $\bullet j$.

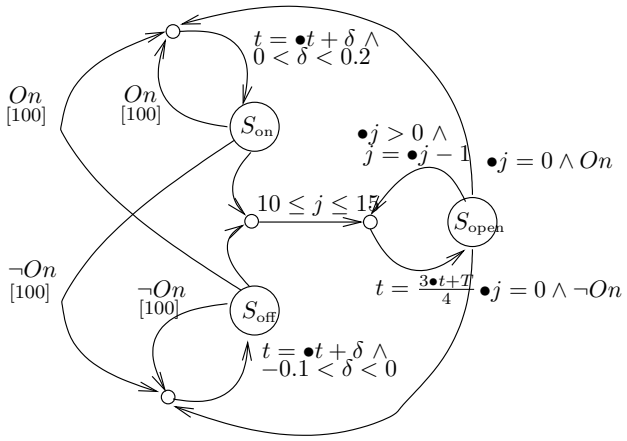
Stable states are denoted by s_{on}, s_{off}, and s_{open}. The other unnamed states are transient. The initial control state is s_{on}.

If the heater is initially on (resp. off), only 2 transitions are possible among the 3 output transitions of s_{on}, since the transition labelled by $\neg On$ (resp. On) is unsatisfiable. The first possible transition has weight 100, and the other one has weight 1.

- The first transition will therefore be drawn with a probability of 100/101. It leads to a transient state which has only one output transition leading back to s_{on} (resp. s_{off}); the elected formula is therefore $0 < \delta < 0.2 \wedge t = \bullet t + \delta$. It states that the local variable δ will be uniformly drawn between 0 and 0.2 (resp. -0.1 and 0), and that δ is then used to increase the temperature. This is intended to model that, when the heater is on (resp. off), the temperature slightly increases (resp. decreases).

- The second transition will be drawn with a probability of 1/101. After two transient states, the only stable node that is reachable is s_{open}, and the elected formula is therefore $true \wedge 10 \leq j \leq 15 \wedge t = \frac{3 \bullet t + T}{4}$. It states that the local integer variable j is drawn uniformly between 10 and 15, and defines how the new temperature is computed. This is intended to model that, whenever the window is open, the temperature become closer from the temperature outside.

- For the next step, from s_{open}, 3 transitions are possible, but they are labelled by formulas that can not be true at the same time (they form a partition): as long as j is greater than 0, the window will remain open; j is decremented at each step, and when it reaches 0, the control gets back to either s_{on} or s_{off}, depending on the variable On.



$V_i = \{On, T\}$, $V_o = \{t\}$, $V_l = \{\delta, j\}$, $V_p = \{\bullet t, \bullet j\}$, $Q_s = \{S_{on}, S_{off}, S_{open}\}$, $q_0 = S_{on}$.

Figure 7.8: Simulation of the temperature in a room with a heater and a window.

7.2.1.3 Case Studies

Case studies provided by Snecma/Hispano-Suiza and Renault have been analyzed by Lurette within the SAFEAIR II project (see §15.1.13). Actually, Hispano-Suiza and Renault used Lurette by their own, with our assistance. An important result is that Snecma/Hispano-Suiza found several bugs in an engine controller (M88). Some of those bugs were found in the validated version of the controller. This work is described in [207]. For instance, this system allows retaining the same deceleration for the same brake pedal angle, even when the brake heats. This application made extensive use of non-linear functions over floats, and is therefore out-of-reach of current state-of-the-art formal verification tools. Lurette was used successfully to generate several millions test sequences.

7.2.1.4 Application to Counter-example Search

Verification methods like model-checking work on a finite abstraction of the actual program. This abstraction is conservative, which means that if the proof succeeds on the abstract program, the actual program is certainly correct. However, when such conservative verification fails, it is generally undecidable whether an abstract trace corresponding to a counter-example has any concrete counterparts. For debugging purposes, one usually desires to go further than giving a yes/no answer, and generate such concrete counter-examples. We have studied a solution in which the testing techniques of Lurette are applied to searching for concrete counter-examples [77]: basically, the set of abstract erroneous traces is used to restrict the generation of concrete scenarios.

7.2.2 Conformance Testing

7.2.2.1 Black-box Conformance Testing for Real-time Systems

Extensive conformance testing theories exist for "untimed" models such as Mealy machines [LY96] or input/output labeled transition systems [BT01]. The objective of the line of work presented in this section is to extend and adapt these theories to "timed" models such as timed automata [AD94]. This work, started in 2002, is lead by Stavros Tripakis (CNRS researcher) and done jointly with Moez Krichen (PhD student).

The motivation behind this work has been both theoretical and practical. On the theory side, testing with timed automata is non-trivial, mostly because of issues of non-determinizability or observability of time. On the practical side, Mealy machines and IOLTSs are not well-suited for modeling specifications where real-time constraints are important. In Mealy machines, inputs and outputs are supposed to occur *synchronously* which makes the model appropriate for synchronous circuits but not so for asynchronous real-time systems.

The LTS framework abandons the hypothesis of synchronism between inputs and outputs and also introduces other desirable features for testing, such as conformance relations which are not equivalences. For example, the conformance relation ioco [Tre99b], used among others by the TGV tool [FJJV96], requires that the implementation accepts more inputs but produces no more outputs than the specification.[5] Still, time is not modeled in a satisfactory manner in these models, if at all. For example, in the ioco framework, time is not directly modeled. Instead, timeouts are abstractly captured with the notion of *quiescence*, which basically formalizes the observation "no observation". This poses problems upon execution of tests, where timeout values must be instantiated and the only way to do so is by using empirical methods.

Our contributions on the topic of testing real-time systems can be summarized as follows. First, we have proposed a framework which can fully handle specifications modeled as non-deterministic or partially-observable timed automata [214]. Non-determinism is essential, for instance, to express ranges of possible output delays or to abstract de-

[5]The implementation is supposed in fact to be input-complete, which avoids the need for more complex relations, such as alternating simulation.

tails. Partial-observability is also essential to model parts of the system the tester cannot observe, or when the specification is built in a compositional manner and internal communication between components is hidden. Also, it is undecidable to remove non-determinism from timed automata automatically [264], even when this is in principle possible (that is, when the automaton is determinizable), thus, having a framework which can handle such specifications is crucial for usability.

In this context, we have proposed a new conformance relation, called tioco (standing for "timed input-output conformance relation"). tioco is inspired from ioco, but requires urgency on the outputs to be quantified: the specification states not that an output will eventually come, but also what is the deadline. This is done simply by treating time delays as observable outputs. One of the benefits is that there is no need for quiescence, since timeouts are explicitly defined in the specification.

Our framework distinguishes between two types of testers, *analog* and *digital-clock* ones. The former are "ideal" in the sense that they can measure real-time precisely. The latter only have access to a digital clock, that is, a discrete counter, updated according to some physical process. Digital-clock tests are clearly more realistic to implement. However, analog-clock tests are also worth studying, first, because they are theoretically interesting, and second, because they can be useful in the case the system under test is a discrete-time system, but its granularity is not a-priori known.

Our framework includes algorithms for automatic test generation. Two types of test generation are considered: *off-line* test generation, where a suite of tests is generated and stocked, to be executed later; and *on-line* test generation, where the test is generated "on-the-fly", during execution. Analog-clock, on-line test generation is based on *on-the-fly subset construction* for timed automata, introduced in [262]. The technique consists in computing, for a given timed sequence ρ, the set of all possible states S_ρ that the timed automaton can be in after executing a run "labeled" by ρ. In the context of observation or testing, ρ corresponds to the observed sequence and S_ρ to the *uncertainty* or *state estimate* that the observer/tester has. On-the-fly subset construction permits to avoid determinizability issues for timed automata: even though we cannot compute S_ρ for every possible ρ (this would imply that timed automata are determinizable), we can compute S_ρ for a given ρ. Moreover, this computation can be done incrementally, as the observed sequence ρ is being generated.

Analog-clock tests can also be generated off-line. The obvious choice is to represent tests as timed automata themselves. However, again because of determinizability issues, the problem of synthesizing a sound and *strict* timed automaton tester[6] for a given specification is undecidable. However, when the *resources* of the tester (clocks and constants) are bounded, synthesis of such a tester is possible and can be done using symbolic techniques [215].

Digital-clock test generation is based on combining subset construction (on-the-fly or off-line) with appropriate modeling. In particular, the digital clock is modeled as a timed automaton emitting a special discrete event tick, observable to the tester. Then, the product of the specification and the digital-clock model is computed, which yields an

[6] strict in the sense that it rejects non-conforming implementations whenever this is possible

extended specification. The latter is observable by the tester only through an "untimed" interface, including the observable events of the original specification plus the new event tick. Thus, the problem of test generation becomes a problem of synthesis of an untimed deterministic controller (the tester) for a timed specification [214, 216].

When tests are generated off-line, a number of choices are possible for the tester, namely, to stop the test or continue, to wait for an input or issue an output, which output if many are possible, etc. These choices result in explosion of the number of tests, an issue not specific to timed testing. To limit explosion, we have proposed test generation methods based on *coverage criteria*, often used to guide test selection. Our framework currently supports criteria based on the coverage of states, locations, or edges of the specification model [215, 216].

We have built a prototype test-generation tool, called TTG, on top of the IF environment [143]. TTG takes as main input the specification automaton, written in IF language, and can generate digital-clock tests. TTG can be executed in *interactive* mode, where the user guides the test generation by resolving decision points. TTG can also be asked to generate a single test *randomly* or the *exhaustive* test suite, up to a user-defined depth. The depth of a test is the longest path from the initial state to a pass or fail state. The tests are output in IF language. Implementation of analog-clock test generation and test selection based on coverage criteria is underway.

In [214, 122, 216] we discuss in detail results obtained by using TTG on a number of case studies: generating tests for a simple light-intensity controller in [214], monitoring the real-time execution software of the K9 Mars rover of NASA in [122], generating tests for the bounded-retransmission protocol in [216]. The conclusions that can be drawn from this admittedly small set of experiments are as follows. First, the modeling framework is quite appropriate: for example, internal unobservable actions can be readily modeled, an indispensable feature for compositional specifications. Second, test-selection guided by coverage criteria permits to significantly reduce the number of generated tests. For example, in the case of the bounded-retransmission protocol, only a few tests can achieve 100% coverage with respect to all criteria, whereas exhaustive test suite number hundreds of tests even for small depths (see Tables 1 and 2 of [216]).

We currently pursue this line of work on multiple fronts, both theoretical and practical. On the theoretical side, we are studying other "classic" testing problems, such as state identification [Moo56], in the context of timed automata [217] or other generalizations of standard models [287]. We are also studying new coverage criteria, perhaps especially suitable when timing constraints are present. On the practical side, we are constantly improving TTG and searching for realistic case studies. Let us also note that this work is done in the context of national and international projects, including the French project ACI CORTOS and the European IST network of excellence ARTIST2.

7.2.2.2 Property-oriented Testing

Although the conformance testing framework is now well established, there is some limitations. First, it requires a rather *exhaustive* formal specification, since conformance is defined with respect to this specification, and any IUT exhibiting unexpected behaviors

(from this specification point of view) would be rejected. Second, the conformance relation is not very *flexible*: it is not always easy to understand what it does exactly preserve, and, more important, it is not possible to adapt it to the particular functionality one wants to test. We propose to extend this framework to the generation of *property oriented* test cases. The general idea is to allow automatic test generation from a *partial specification* (not necessarily expressing the complete expected behavior of the system), and with respect to a particular property (test case execution should indicate whether the IUT satisfy or not this property). This approach is outlined below. In the following, we distinguish between Property testing and Robustness testing.

7.2.2.3 Property Testing

The properties we consider are *linear* properties: each property defines a language (i.e., a set of sequences), and an IUT satisfies a given property if and only if all its execution sequences belong to its associated language. In this context it is a common practice to distinguish between *safety* properties, that can be checked by considering only finite execution sequences of the IUT, and *liveness* properties that need to consider also the infinite ones. Several characterizations of such properties have been proposed in the verification community. For example, automata on infinite words (recognizing ω-regular languages) is such a specification formalism. It can be shown in particular that any ω-regular language can be characterized by a Büchi automaton, or, equivalently, by a *deterministic* Rabin automaton. Since the use of a deterministic automaton is an important issue in the test generation technique we propose that the property to be checked is expressed by means of a deterministic Rabin automaton. Of course, testing the satisfiability of a liveness property is not possible: it would require an infinite execution time. However, automata on infinite words can be parameterized to specify so-called *bounded liveness* properties: the automaton recognizes a set of infinite execution sequences and some external parameters simply limit the "length" of the sequences to consider (this length being expressed for instance in number of interactions, or as an overall execution time). More precisely, the test generation technique we propose [181] is based on :

- A (possibly partial) specification $\mathcal{S}$ is used as a "guideline" for the test case synthesis, and it is therefore supposed to be "closed enough" to the actual behavior of the IUT. Note however that we do not require at this level any particular conformance relation between $\mathcal{S}$ and the IUT.

- A safety or bounded liveness property $\mathcal{P}$ is given through an observer $\mathcal{O}$bs recognizing sequences of $\neg\mathcal{P}$. This observer is a parameterized automaton on infinite words.

- Test cases are automatically generated by traversing the specification in order to find the "most promising" execution sequences able to show the non satisfiability of $\mathcal{P}$ by the IUT. These execution sequences are the sequences recognized by $\mathcal{O}$bs that are the "closest" to the ones provided by the specification.

7.2.2.4 Robustness Testing

Informally, robustness can be defined as the ability of a software to keep an "acceptable" behavior, expressed in terms of *robustness requirements*, in spite of *exceptional* or *unforeseen* execution conditions (such as the unavailability of system resources, communication failures, invalid or stressful inputs, etc.). Even if this kind of testing has been less studied than in the hardware community, several approaches have been proposed to automate software robustness testing. Most of them are based on *fault-injection*. We propose a robustness testing framework by modifying slightly the property testing framework [241, 182]:

- First, robustness is defined with respect to a *fault model* that may vary from an application to another. This element usually depends on the application architecture (e.g., unreliability of some communication links, lack of confidence in some external components, input channel feed by an untrusted user, etc.) and needs to be expressed at a rather abstract level. Note that some of these faults may be controllable by an external tester, whereas some others may not.

- Moreover, specifying the system behavior for *any* exceptional and/or invalid execution conditions expressed by a fault model is (by definition) hard to achieve. Therefore the specification should no longer be considered as "exhaustive" in this context (it may not always reflect the expected behavior of the implementation).

- As a consequence, test verdicts should no longer be based on a conformance relation between the implementation and this (approximated) specification, but directly with respect to the initial robustness requirements.

The solution we propose is based on the following elements: The initial *system specification* is expressed in a formalism those operational semantics can be defined in terms of Input-Outputs labelled transition systems and which explicits the system architecture (communication links attributes, component interface and internal structure). The *fault model* is expressed by *syntactic mutations* performed on this specification. The *robustness requirements* are expressed by a linear temporal logic formula describing the expected behavior of the system implementation in terms of tester interactions. Of course, checking whether a given implementation satisfies or not such a formula should remain decidable during a test (i.e., within a finite amount of time).

7.3 Debugging, Simulation and Early Execution

Debugging, simulation and early execution techniques are needed because formal verification does not always succeed, and because they may be applied very early in the development cycle. Some of them do not even require that the expected properties of systems be formally written, since they are based on mere human *observations* of systems' behaviors.

On the other hand, the work spent on verification tools has led to a complete set of core technologies that may be reused in the context of semantically well-founded debugging, simulation and early execution techniques.

7.3.1 Debugging Reactive Programs

The work on specific techniques for debugging reactive programs is the PhD of Fabien Gaucher [25]. It also appeared in [MG00].

For safety-critical reactive systems (in avionics, transportation, nuclear plants, etc.), a lot of effort has been put on the definition of safe languages, with a formal semantics. In the synchronous approach, several languages with distinct programming styles share a common simple view of the world that helps reasoning about programs formally. Formal verification and testing tools have been developed, based upon the formal semantics. But there are very few attempts at providing the user with a tool like a debugger. Simulators exist, but they are restricted to the observation of the global inputs and outputs of the reactive system, and to commands like step (on the base clock, for time-triggered implementations) and reset.

We proposed a debugging method for complex reactive systems written in the synchronous dataflow language Lustre. It is implemented in the tool Ludic, which combines: 1) the step-by-step execution with breakpoints on the *basic clock* of the reactive system; 2) visualization tools displaying timing diagrams, for all internals of programs. Ludic also offers sophisticated features: go backwards in time; jump to a state described by a safety property; automatically reduce the space in which bugs are to be searched for, using local *contracts* and defensive execution, etc.

In [206], we describe the "jump-to-state" functionality of Ludic. The user specifies a state to be reached with a safety property, itself expressed as a synchronous observer. This property is automatically translated into a proof obligation for the abstract-interpretation tool Nbac. The proof obligation is: prove that the state required is unreachable. If the tool succeeds, it means that the state is indeed unreachable, and a message is issued to the user. If the tool fails, it provides a diagnosis, which is a kind of symbolic representation of all the possible scenarios that could reach the state. However, these scenarios are abstract, and do not necessarily contain concrete executions. We use Lurette to try and find concrete executions (this is very similar to the work described in section 7.2.1.4 about the generation of counter-examples), and we play them in Ludic to reach the state. One of the key contributions of our proposal is the proper handling of numeric variables, because Nbac and Lurette can take numerical variables into account.

7.3.2 Simulation and Early Execution

Integrating formal methods in the design flow of complex systems means that, at very early stages of development, formal methods may provide automatic tools that help in writing and validating individual parts of the system, or partial compositions. A key point is to be able to simulate (or "execute") a complex system very early.

We distinguish between *simulation* and *early execution* because in the context of reactive systems, the development covers the specification of the computer part and the modeling of the physical environment. Technically, both can be done by writing non-deterministic reactive machines; but the modeling of the physical environment only provides a simulation of this environment (it is never perfect, it is an approximation of the physical world), while a non-deterministic specification of a component may indeed be considered to be *executed*. For instance, consider a component that takes five integer inputs and outputs the index of a maximal value. The specification may be non-deterministic, while any actual implementation will impose a static priority for searching for maximal values. The implementation is necessarily over-specified. It is quite convenient to *execute* the complete system by using the specification instead of one particular implementation. The question is then: how to execute a non-deterministic specification of a reactive system?

The answer is that it is a natural extension of the testing techniques described in section 7.2.1. During the development of a complex application, some modules are already programmed while others are still at the state of high level specifications. Moreover, complex applications are in general a mix of software and hardware components. Partially known and/or hardware components can be described as non-deterministic (or stochastic) systems.

In the testing framework, the program is executed as a black-box, and the environment is executed by constraint-solving techniques. In the early-execution framework, there are several modules that are finished, and also several modules that are given by their non-deterministic specification only. The modules that are finished can be executed as black-boxes. The modules that are only partially known are "executed" by constraint-solving techniques. Figure 7.9 sketches such a simulation: non-deterministic components (in particular the environment) are marked by a "?" sign.

Several languages or formalisms can be used for the non-deterministic descriptions of partially known components. We can use the languages Lucky/Lutin initially designed for the description of test scenarios. We can also use *contracts* (see §6.3), or classical Lustre observers.

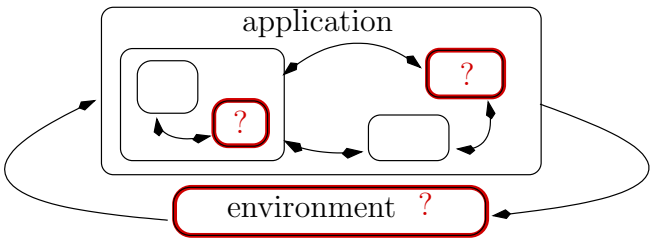


Figure 7.9: Early simulation of a complex application.

During the early execution of a set of components, global observers (i.e., components that only observe the others, and do not interfere with their behavior) can be added or removed on demand, depending on what has to be observed. Similarly, global inputs can temporarily be connected to some new components that generate them, in order to execute

the system without having to provide explicit values for these inputs at each step.

The problem of early simulation is studied in the ACI project ALIDECS (§15.2.3), and also in the European project ASSERT (§15.1.4), where we plan to extend the framework to synchronous/asynchronous components.

Chapter 8

Implementation Techniques

Design solutions for embedded applications are being strongly driven by software, which is becoming a dominant part of embedded systems. The relative weight of software in the value of embedded systems is constantly increasing. Software allows new, complementary services, and differentiation, which brings competitive advantages.

This trend in embedded systems design is leading to a significant grow in the workload of embedded processors as software gradually shifts towards an increasing computational complexity, requiring the execution of real-time tasks such as image, audio, and video compression and recognition. It has been observed that the increase of computational requirements is difficult to meet by hardware development alone [LTC+03].

Because more and more functions of embedded applications are implemented in software, multiprocessor architectures (e.g., TTA, Intel's network processors, Thales' PowerPC G4 Multiprocessor) are becoming an appealing hardware solution to provide high computational power. Consequently, embedded software is becoming multi-threaded, both because applications are composed of intrinsically concurrent tasks and to better exploit at software-level the physical parallelism provided by multiprocessor hardware. Hence, mastering multiprocessing and multi-threading is a key objective for embedded systems [HNO97, URS03].

Moreover, embedded systems are inherently real-time. Respecting timing constraints is a major challenge in embedded systems design. Systems must be built to provably meet end-to-end quantitative timing constraints (e.g., bounded response and communication delays), but also power and memory requirements (e.g., to optimize battery lifetime and memory access latency). These parameters are interdependent as, for instance, memory accesses are both time- and power-consuming, and energy savings can be achieved by slowing down the processor speed. In addition, other non-functional behavioral properties, such as deadlock freedom, must be verified as well.

Thus, building modern real-time embedded systems is becoming an extremely complex activity that requires mastering several interdependent and sometimes conflicting dimensions, both at software and hardware level. The way to overcome this complexity is to develop theoretically sound and automated implementation methods that cover all these dimensions. This approach will allow to bridge gaps in the design flow to ensure overall system correctness.

A rigorous design and implementation approach taking into account all these factors is missing today. Making it work requires understanding and mastering the dynamic behavior of systems, and the nature of relationships between non-functional constraints. Moreover, it requires a deep knowledge about the structure and the properties of the embedded application (including both software and hardware).

To address these issues the goal of this activity is to come up with techniques and tools for the design and implementation of predictable embedded systems that handle multi-threading, real-time constraints and multiprocessor architectures thoroughly the development cycle. This will bridge gaps in the design flow and help shorten time-to-market and limit costs, while guaranteeing overall system quality.

To achieve this goal, our approach is to develop model-based analysis and synthesis techniques to generate correct-by-construction code that complies with non-functional requirements. Special emphasis is put on integrating these techniques into industrial tools. This enables code generation for specific platforms (including software-to-processor mapping and scheduling), and platform-independent functional analysis, to be linked together in the same tool-chain without semantic gap. Such an integration considerably increases the overall quality of industrial systems designed with these tools, guaranteeing the correctness of the resulting solution.

In this context, we have been working on two main research directions:

- Development of property-aware compilation technology;
- Implementation of high-level models.

8.1 Property-aware Compilation Technology

In current engineering practices in industry, application requirements and design constraints are spread out and do not easily propagate through the development lifecycle. Moreover, the increasing complexity of applications tends to enlarge the abstraction gap between the application description and the hardware. Consequently, guaranteeing conformity with non-functional requirements is costly and error-prone.

To overcome this problem, we think compilation tools and their associated technologies (analysis, optimization, ...) must play the central role of mapping platform independent software into target execution platforms (operating system and hardware), while ensuring at compile time that non-functional requirements provided by system's engineers will be met at runtime.

Integrating in the same toolset a compiler and model-based formal analysis and synthesis modules handling non-functional constraints and heterogeneous architectures is an innovative way to provide correct by construction code. This allows the tasks of code generation (including software-to-processor mapping and scheduling) for a specific hardware implementation and hardware-independent functional analysis to be linked in the same tool.

The bases of this approach have been presented in [79]. The idea consists in using model-extraction techniques that allow relating the semantic model of the programming

language with its realization on the hardware platform, thus automatically creating the appropriate vertical bindings between non-functional parameters.

We have first successfully applied this approach in the TAXYS tool for single-threaded programs. In the EXPRESSO project we extended the technique for multi-threaded applications running on mono-processor architectures. We are currently working on developing a framework, called JAHUEL, for programming and compilation for embedded multi-processor platforms in the context of the VERIMAG-STMicroelectronics joint project ANACONDA. Together with STMicroelectronics we have also been working on developing a domain-specific property-aware implementation technique for ensuring QoS requirements in multimedia applications.

8.1.1 Time-constrained Single-threaded Synchronous Programs

The goal of the TAXYS project 15.2.21 was to provide a framework for developing real-time embedded sequential code and verifying its correct behavior with respect to quantitative timing requirements. To achieve so, we have developed the TAXYS tool [163, 130] (Fig. 8.1) which connects France Telecom's Esterel compiler SAXO-RT with VERIMAG's model-checker KRONOS.

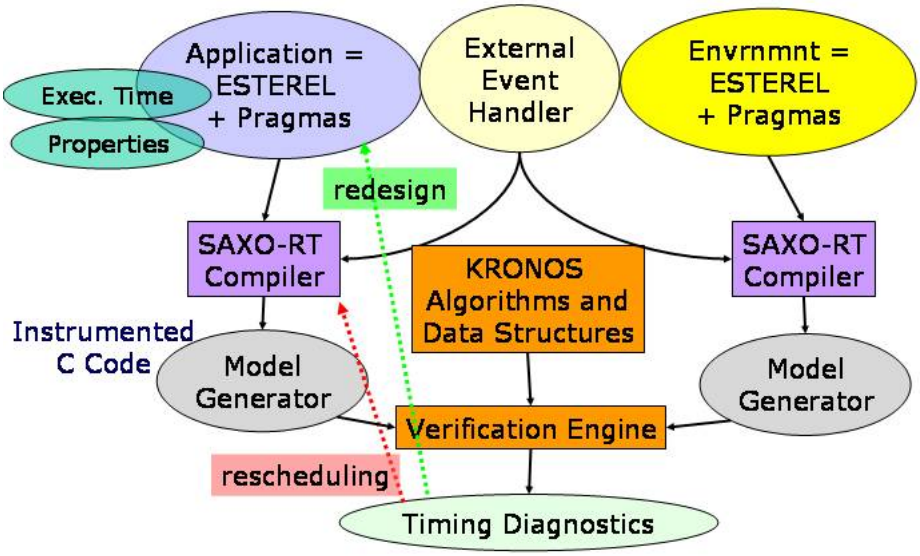


Figure 8.1: TAXYS.

In TAXYS, an embedded real-time program is decomposed into a control part, written in Esterel and a data-manipulation part written in C. The program is compiled with the Esterel compiler SAXO-RT that generates very efficient sequential C code targeted to

mono-processor hardware architectures. The order of execution of the specified parallel modules is resolved at compile time. Such an implementation is (functionally) correct modulo the so-called *synchrony* assumption, which states that the program always reacts faster than its environment. However, it may not satisfy the real-time requirements which depend on the execution speed of the target machine and the timing constraints imposed by the environment. In practice, validating the synchrony assumption amounts to show that the environment does not take too much lead over the application. This requires the use of a "realistic" synchrony assumption strongly depending on the application and its interactions with the environment. To interface the real-time system with its environment, we need to use an external event-handler which keeps track of the history of the environment and triggers computation steps of the application.

To check whether the program meets the timing requirements, the basic underlying idea of TAXYS consists in producing a formal model (actually, a timed automaton), that captures the real-time behavior of the whole application composed of the program, its execution constraints, and the external environment. The Esterel source code is annotated with timing constraints of two kinds: (a) time intervals associated with the execution times of the external C functions, and (b) the requirements to be satisfied. TAXYS produces timed model of the self-sequenced executable code of the application. This model is composed with a timed model of the external environment. The global model is analyzed to validate timing requirements. Whenever a property is not satisfied, TAXYS generates a source-level error trace. In many cases, the violation of a timing constraint is indeed caused by the wrong execution order of some computations triggered by independent events within a reaction (or synchronous step). Such problems can be remedied by automatically re-arranging the sequential code at compile time. To do so, the information derived from the error trace can be used as a compiler directive to make SAXO-RT generate a correctly scheduled sequential implementation of the Esterel program.

TAXYS has been successfully applied on real industrial telecommunication systems, such as a GSM radio link from Alcatel and a phone prototype from France Telecom, and PATH's automated-vehicle control software [257].

8.1.1.1 Case Study: Radio Link of a GSM Terminal

This case study is reported in [79]. It consists in the programming and verification of the radio link of a GSM terminal developed by Alcatel. We describe here a small part of the application which is composed of two modules. When the event `Prepar` arrives, the first module takes 50ms to prepare the radio front end of the mobile terminal in order to receive data. Then, when the event `Receipt` arrives, it goes through a demodulation phase, that takes between 80ms and 100ms, followed by a decoding phase, that finishes in 20ms. The second module is triggered by the event `Freq` and calculates the frequencies on which the data are going to be received and completes in 40ms. The computations are subject to the time constraints annotated in the code. The code of the full application developed by Alcatel consists of 815 lines of Esterel and 48000 lines of C. The application was validated for 62 test environments provided by Alcatel. 4 scenarios were found to lead to deadline violations caused by a wrong scheduling of calls. These errors were corrected by slightly

Name	Buff. size	Symb. States	Verif. Time	Diagnostic
$ISDN_1$	5	2 200	1.27 s	incorrect
$ISDN_2$	6	10 849	5 s	OK
$ISDN_3$	5	15 894	6.29 s	incorrect
$ISDN_4$	6	633 472	10 mn 47 s	OK
$ISDN_5$	5	22 695	13.6 s	incorrect
$ISDN_6$	6	$> 10^7$	?	aborted

Table 8.1: Experimental results for the prototype phone.

changing the Esterel code.

8.1.1.2 Case Study: ISDN Prototype Phone

This case study is reported in [163]. It deals with a prototype phone carrying simultaneously voice and data produced by a graphic tablet, implemented on a 32 MIPS Digital Signal Processor. The prototype has an audio input channel sampled at 8kHz that is connected to the microphone, an RS232 input channel carrying data from the graphic tablet, and an input channel sampled at 8 kHz to retrieve audio and graphic data sent by the network (TNR). Processing audio data consumes 3900 CPU cycles over the 4000 CPU cycles available every 125μs. Graphic data are compressed by a vectorization algorithm which consumes sporadically between 15000 and 20000 CPU cycles. The program consists of 258 lines of Esterel and 3000 lines of C code. We have used TAXYS to analyze the relationship between the size of the input buffers and the arrival rate of graphic data. We have analyzed the behavior of the system with three different environment models. For each one, we have experimented with buffers of sizes 5 and 6. In $\mathbf{ISDN}_1$ and $\mathbf{ISDN}_2$, the environment model is composed of two strictly periodic and independent tasks (the first carrying audio data at 8kHz and the second the graphic tablet data at 100Hz). In $\mathbf{ISDN}_3$ and $\mathbf{ISDN}_4$, the second task is aperiodic and emits bursts at rates varying non-deterministically between 25Hz and 100Hz. In $\mathbf{ISDN}_5$ and $\mathbf{ISDN}_6$, there is an additional periodic task that models switching between several audio modes. In all cases, the 8kHz periodic event is specified to be separator. The results presented in Table 8.1 show that the buffer needs to be of size at least 6 for the implementation to be correct. The current prototype was unable to handle the complexity of $\mathbf{ISDN}_6$.

8.1.1.3 Case Study: Automated Vehicle Control Software

This case study is reported in [257]. It involves the software developed by the PATH Advanced Vehicle Control and Safety Systems (AVCSS) project at the University of California at Berkeley [Var93]. This software is responsible for controlling a set of cars moving autonomously in a *platoon* formation (one car behind the other, with a small distance, e.g., 4-6 meters, between them), on the highway and at high speed (e.g., 65 miles/hour). The software consists of a set of processes running concurrently on a PC, reading data

from various sensors (e.g., radar, speedometer, accelerometer, magnetometer), writing to actuators (throttle, brake and steering), and using radio to communicate data to other vehicles [261]. The control part of the system has been re-programmed in Esterel. We have verified with TAXYS that the implementation is correct and that all periodic tasks complete their execution before the next period. TAXYS explored the entire reachable set (2022 symbolic states) in a less than a second.

8.1.2 Scheduled Native Code for Real-time Java Applications

Current trends in industry are leading towards the use of Java as a programming language for implementing embedded and real-time applications. From the software engineering perspective, the Java environment is indeed a very attractive development framework. Object-oriented programming provides encapsulation of abstractions into objects that communicate through clearly defined interfaces. Dynamic loading eases the maintenance and improvement of complex applications with evolving requirements and functionality. Besides, Java provides built-in support for multi-threading. However, the semantics of Java do not guarantee a predictable run-time behavior, which is an essential issue in embedded real-time software. To overcome this problem, work has been done to extend the language and the platform to accommodate to the requirements of real-time systems by focusing on current practices. Among such work, we should mention the Real-Time Specification for Java (RTSJ), that provides support for real-time programming (timers, clocks, handlers, priorities, ...

Still, these extensions leave some important issues unspecified, like the scheduling algorithm to be used, allowing an implementation to resolve them at will. In order to obtain more precise semantics, domain-specific profiles have also been defined, such as the Ravenscar-Java high-integrity profile for safety-critical systems. This profile settles an execution model based on a two/three-phase program execution, comprising an initialization phase and a mission phase (possibly followed by a termination phase), and multi-threading semantics relying on fixed priority preemptive scheduling and priority ceiling inheritance. Though designed to ease analysis and programming, this profile still has some drawbacks. For instance, it does not directly support threads which synchronize and communicate using the wait and notify methods of Java. Besides, the underlying schedulability analysis is pessimistic by nature and not well adapted for systems with heterogeneous tasks and constraints. The other important issue is performance.

Though there are efforts to produce efficient implementations of Java Virtual Machines, the slowdown due to the VM remains an argument against adopting Java for real-time applications. Real-time systems can afford neither the overhead nor the non-determinism of using a Just-In-Time (JIT) compiler. An alternative approach consists in using an Ahead-of-Time (AOT) compiler to generate executable code for a run-time system and to provide a native implementation of the real-time primitives. A major advantage of AOT compilation is that it allows performing sophisticated analysis techniques to produce highly optimized code.

In the EXPRESSO project 15.2.12 we developed an approach that takes into account the demands of both precise semantics and performance. Our work is based on

the two/three-phase execution model and API of the EXPRESSO High-Integrity Profile, which itself inherits concepts from the Ravenscar-Java profile and the RTSJ API. However, the semantics proposed by the profile do not fit the needs of applications that would demand alternative scheduling and synchronization paradigms, and handling quality of service requirements. To accommodate to such demands, our approach focuses primarily on the application.

We first automatically extract a formal model of the behavior of the application program as an extended automaton. Then, we synthesize an application-dependent scheduler which is safe (i.e., it is deadlock free and meets all timing constraints) and QoS extendible (i.e., it can be extended to handle QoS requirements, such as reducing response jitter, power consumption or context switches). This synthesized scheduler is meant to be used with an appropriate native run-time support we have developed, which itself uses the underlying R-T OS's primitives. Our scheduler also needs an instrumented version of the original Java code (also produced by our model extracting tool), so as to be able to follow the changes of thread states (i.e., the instrumentation implements an abstract program counter).

The test-bed we have developed [212, 211, 213] has been integrated together with the EXPRESSO High-Integrity API (from Aonix), the TurboJ [177] Java-to-native compiler (from Silicomp), and several real-time embedded operating systems such as eCOS and Thales' FAST-OS. This framework provides a complete analysis and compilation chain for embedded real-time systems based on Java.

8.1.3 Multi-threading and Multi-processing

The ANACONDA 15.3.3 project is a Verimag-STMicroelectronics joint effort to develop a theoretically sound and automated model-based design, analysis, and implementation tool-chain for generating correct-by-construction code for embedded multi-processor architectures.

We identify and contrast two classes of parallelism in embedded applications. The first one is the parallelism inherent in the specification of an application, typically corresponding to logically independent activities, e.g., computations performed by an MPEG-4 encoder on distant macro-blocks, which can be done concurrently [116]. We call this type **application** parallelism. The second one is the concrete or **execution** parallelism exhibited by an execution of a program on a particular architecture. It may either exploit hardware capabilities (e.g., pipelining, dedicated devices, multiprocessors, ...) to implement application parallelism, or restrict the latter because of architectural constraints (e.g., single processor, bus contention, ...). In any case, the programmer must handle, and therefore needs appropriate mechanisms for, both types of parallelism during the development cycle of an embedded application.

We have defined a language, and its associated semantics, such that, it (a) gives the programmer simple and architecture-independent means to describe application parallelism, and (b) provides a coherent framework for handling both types of parallelism all along the compilation chain. In other words, the language is designed to serve both as a tool for the programmer to express the parallel structure of the program, and as a representation to

be directly manipulated by the compiler to perform program analysis and transformations to exploit platform parallelism.

In contrast to synchronous languages, the language proposed in our work provides mechanisms for expressing parallelism in a very general way, in the sense that we avoid as much as possible committing to a particular execution model. This approach has the main advantage of factoring out analysis and transformations which do not depend on precise execution models, pushing down code generation for specific execution targets as the last step in the compilation flow.

The language and the compilation chain (called JAHUEL) are described in detail in [115]. We have successfully applied it in an MPEG-4 encoder/decoder provided by STMicroelectronics [116]. The current compilation prototype is implemented using an XML intermediate representation format, and Java for the algorithmic transformations. JAHUEL is connected with STMicroelectronics FlexCC2 compilation-suite which takes as input annotated C programs and generates the intermediate representation. Currently, Jahuel generates POSIX-compliant C code, which is then compiled for a specific target architecture, and SystemC code for simulation. Ongoing theoretical work concerns further structuring the compilation chain so as to welcome the addition of more quantitative architecture description features in the language.

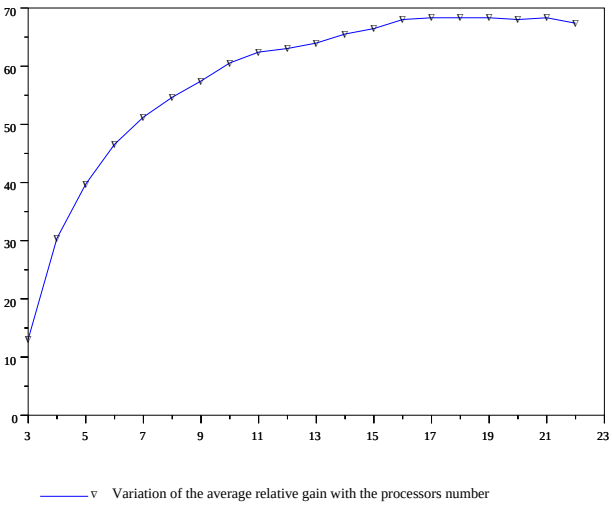


Figure 8.2: Average relative gain in compression time.

8.1.3.1 Case Study: MPEG-4 Video Encoder

The MPEG committee has defined widely used standards for digital video encoding providing high quality images and high compression rates. However, real-time MPEG encoding also demands high computational power, usually far beyond traditional sequential computers can provide. Fortunately, the algorithms compliant to the MPEG standard can be parallelized.

In [116], we proposed (1) a parallel model for MPEG-4 video encoding algorithm, which expresses the maximal parallelism originally hidden in the standard by the given block diagram specification, and (2) a parallel implementation based on the exploitation of fine grain parallelism at the macro-block level expressed in our model. The starting point was an MPEG-4 encoder provided by STMicroelectronics in the context of the ANACONDA project 15.3.3.

We first developed a detailed description of the MPEG-4 encoder's phases and the macro-block-level computations' dependencies. The parallel model is described in the formal language of the Jahuel tool 8.1.3. From this parallel model, we generated an efficient non-executable parallel specification for SMP multi-processors. To do so, we proposed a mapping and scheduling heuristic based on results from automatic loop parallelization techniques. Our heuristic extracts parallel loops without iterations' dependencies from the model and then schedules these loops on the SMP multi-processors using a greedy dynamic load balancing strategy. An implementation is (semi-)automatically produced by Jahuel into an MPI-based, message-passing parallel program.

Theoretical speedup analysis of our mapping and scheduling algorithm on a CREW-PRAM architecture model, and experimental performance results, showed the usefulness of our approach for SMP machines. To evaluate the parallel encoder performances we measured the relative gain in compression time, compared with the sequential encoder, for a benchmark of one hundred frames with resolution size of 640×480 (Fig. 8.2. Even if the speedup is sub-linear since some encoding phases cannot be parallelized, the results show that the addition of new processing units to the architecture has a positive impact in encoding time: the parallel compression time is smaller than the sequential time by nearly 15% for two processors and reaches nearly 70% for sixteen processors.

Besides, our approach has the advantage to be flexible since it can be applied to generate an MPEG-4 parallel implementation even if some features, like the rate control mechanism, are changed, or new ones are added to the proposed MPEG-4 model. We also outlined a method to select encoding algorithms and parameters especially for the motion estimation phase, in a two step performance quality and bit-rate control approach, in order to obtain real-time parallel MPEG-4 implementations on SMP multi-processors. These applications are useful in the context of real-time on-line encoding under QoS requirements, like video conferencing.

In [117] we have generated a parallel implementation of the MPEG-4 algorithm on an experimental custom multiprocessor board currently being designed at STMicroelectronics. The implementation relies on a scheduler obtained using a novel assume/guarantee-based scheduler-synthesis algorithm that we have developed.

8.1.4 Fine-grain Control of QoS Requirements

Cost-effective development of embedded software should be based on the extensive reuse of generic software components. It should also be supported by tools and methodologies for guaranteeing given functional and extra-functional properties. In particular for multimedia embedded software the fast evolution of market needs, user requirements and platforms requires reliable adaptation of standard features at minimal costs. Currently, adaptation

of available application software to target platforms and needs is too costly. To meet given QoS requirements a significant amount of experimentation is needed on virtual or real prototypes involving fine tuning of parameters of the components of the application software. After tuning, the behavior of application software can be modified only by changing user-defined input parameters. Thus, adaptability is coarse grain as it can be achieved only by modifying global parameters. Furthermore, some delay is necessary for adaptation due to limited controllability of the application software over the underlying execution system.

Starting from a real application, a videophone of STMicroelectronics, which is a real-time video coder/decoder, we study a method for fine grain QoS control meeting both soft and hard real-time requirements. For these applications uncertainty about execution times, makes necessary the use of control (dynamic scheduling) techniques. The problem we want to solve is the following : given an application software with hard deadlines constraints, construct a controlled software which uses the resources optimally. In order to meet hard real time deadlines, usual techniques are based on Worst Case Execution Time (WCET) analysis. Thus, the application is scheduled statically and there is no deadline miss. On the other hand, soft real time approaches, whose goal is to ensure an optimal utilization of the resources, are generally based on dynamic techniques and the control is made at task level. We propose a method bridging the gap between hard and soft real time. The considered method allows to generate from an application software a controlled application software that meets given QoS requirements, as follows.

- The initial application software performs cyclically input/output transformations of data streams. It is described by a precedence graph modeling dependency between actions (C-functions) and from which all the possible execution sequences can be extracted. Its execution during a cycle can be controlled by choosing *quality levels* which are parameters of the actions. We assume that the execution times of actions are increasing with quality.

- We consider single threaded implementations of the application software on a platform for which it is possible by using timing analysis and profiling techniques, to compute estimates of worst case execution times and average execution times of actions for the different levels of quality. Action execution is assumed to be atomic. A compiler is used to generate from the initial application software, for given QoS requirements and execution times, the controlled software. QoS requirements are deadlines on the termination of actions since the beginning of a cycle.

The controlled software can be considered as the composition of the initial application software with a *controller*.

- The controller monitors the progress of the computation in a cycle and chooses the next action to run and its quality level. It uses the model of the application software as well as knowledge for each action of its deadline, average and worst case execution times.

- At some state of a cycle, the controller chooses quality levels guided by two kinds of constraints. Safety constraints ensuring that no deadline is missed during the cycle. Optimality constraints ensuring near-optimal time budget utilization, that is, the available time for completing an action is used as much as possible to achieve the best quality (without violating safety).

Our method significantly differs from existing ones for QoS control and adaptive scheduling. The main difference is fine grain control of the execution. Existing control techniques act at task level e.g. at the beginning of a cycle, and their reactivity is slow. They do not require any deep knowledge of the data-flow structure of the application software. Our method consists in controlling execution during a cycle; the controlled software is produced by compilation (automatic code instrumentation).

Another important difference is that fine granularity allows combination of optimality and safety of the produced schedules. Most control techniques focus on optimality criteria and are adequate only for soft real-time. The integration of safety criteria is useful in applications where quality should remain above some minimal level or hard deadlines must be respected.

Our approach is as follows. The model of the application software is parameterized by an actual execution time. The model we consider is defined by precedence graph G, a finite set of action Q and three functions : the average time function C^{av}, which associates with any action and any quality level an average execution time, the worst case time function C^{wc}, which associates with any action and any quality level the worst case execution time, and a function D which associates with an action its absolute deadline. Moreover, the actual execution time, which has the same profile than e.g. worst-case execution time, is the parameter of the model. We design a controller which has two parts : a quality manager and a scheduler. The execution of the controlled software can be described as follows:

- A state of the controlled software is characterized by the history of actions already executed, a partial quality assignment function associating a quality with each action of the history and the overall time corresponding to the actual execution time. Given a state, the controller determines the next action with its quality level.

- The scheduler computes one optimized schedule for each quality level q. These optimized schedules give possible strategies.

- The quality manager selects among the strategies which satisfy the quality management policy, the one with corresponds to the highest quality level.

The method relies on a solid theoretical base. We proved that the control policy is safe under some reasonable assumptions about the controlled system — actual execution times are less than worst case execution times. We also proved optimal budget utilization. Furthermore, we studied specific conditions guaranteeing smoothness in terms of variations of quality levels chosen by the controller.

We applied this method to an MPEG 4 encoder provided by STMicroelectronics and written in C (more than 7000 loc). The encoder treats frames cyclically. Each frame

is split into N macro-blocks of 256 pixels. We developed a prototype tool that allows the generation of controlled application software when the order between the deadlines is independent of the quality (see Figure 8.3). The inputs of the tool are

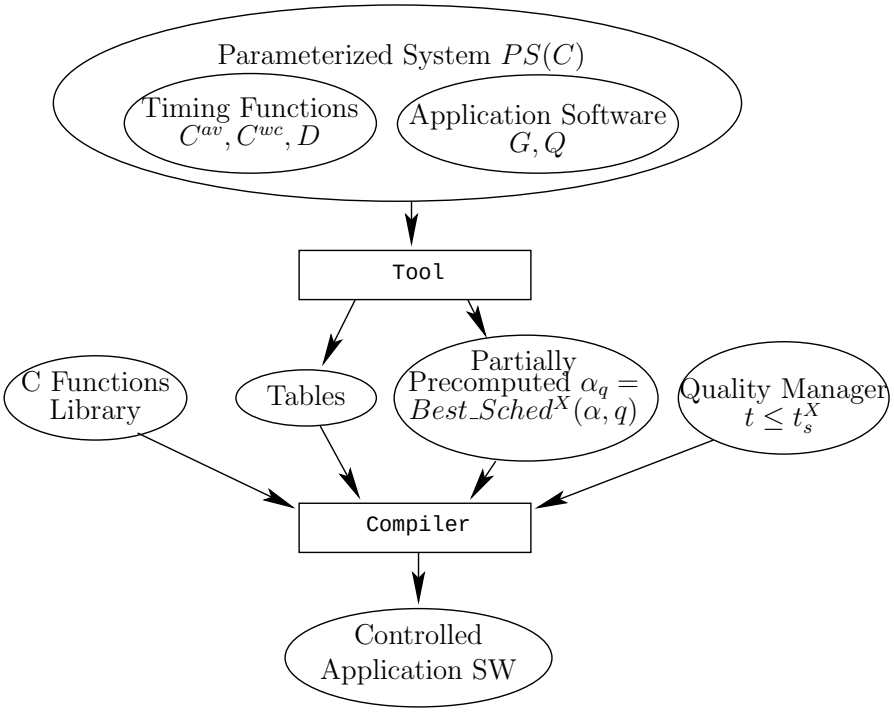


Figure 8.3: Tool overview

• the precedence graph G corresponding to the treatment of a macro-block and its iteration parameter N,
• tables describing the functions C^{av} and C^{wc} for the actions of G,
• the order relation between the deadlines.
From these inputs the tool computes
• C code corresponding to an optimized schedule α_q.
• tables containing pre-computed values used by the controller for the computation of quality management policy $t \leq t_s^X$.

The overhead due to the instrumentation of the application software in the size of the compiled code is of the order of 2% for the considered benchmarks. During execution, the corresponding overhead in memory allocation is not more than 1%. Finally, the overhead in runtime is estimated less than 1.5% of the overall execution time.

Experimental results given in Figure 8.4 show that for constant quality level, load fluctuation leads to a high amount of frame skips even if an additional input buffer is used. On the contrary, controlled quality completely avoids frame skips and overloads lead to smooth reduction of the video quality.

Figure 8.5 shows how optimized schedules increase the time budget utilization of the controlled application. We have also improved the smoothness of the chosen quality levels by using an adequate quality management policy called mixed policy (see Figure 8.6).

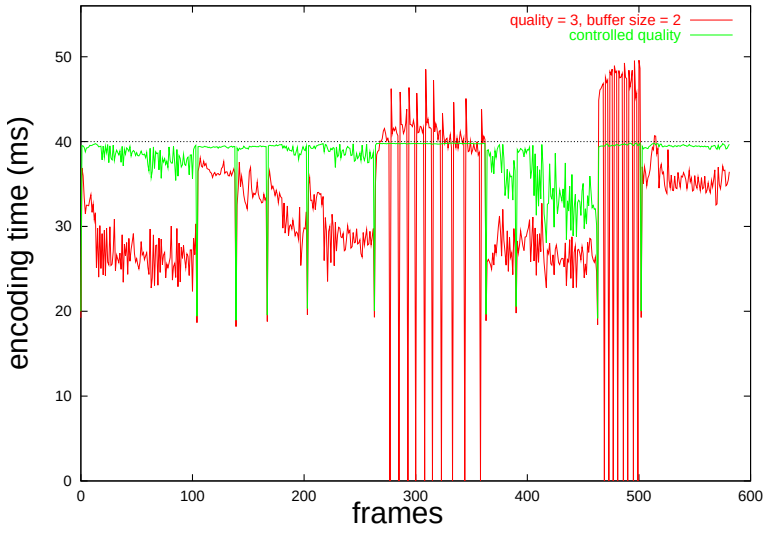


Figure 8.4: Load fluctuation for constant quality level

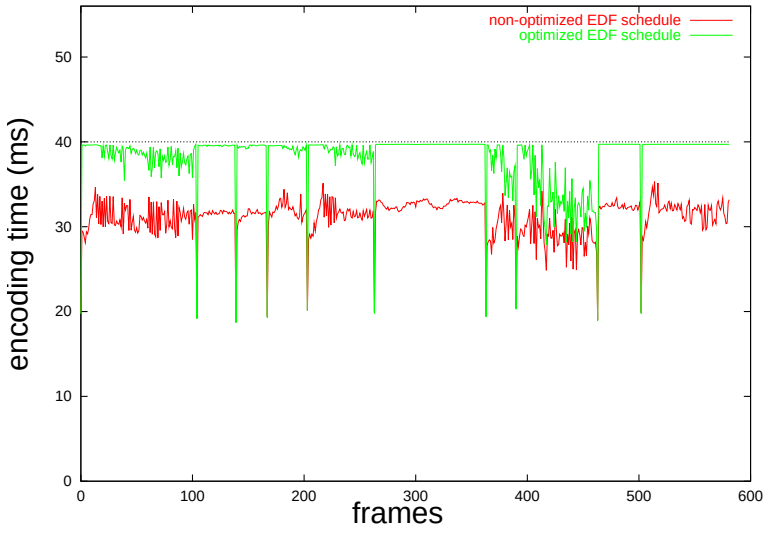


Figure 8.5: Time budget utilization for optimized schedules

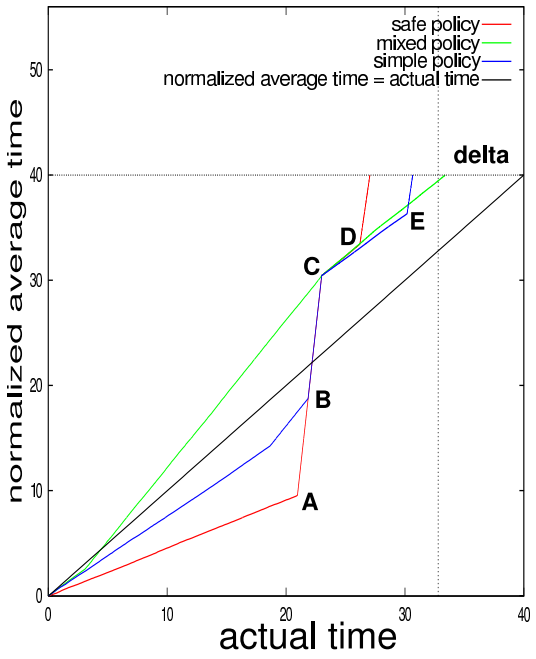


Figure 8.6: Quality management

8.2 Implementation of High-level Models

SCADE, the industrial toolsuite based on Lustre, is well-established in the avionics industry for the development of critical embedded control software. For instance, it has been employed for the design of the major part of onboard software of the Airbus A340-600 and A380. On the other hand, other industrial sectors, and in particular the automotive industry, largely employ the Matlab/Simulink/Stateflow (TM Mathworks) tools. Given this context, we have developed methods and tools to translate existing Simulink/Stateflow designs into Lustre. This permits to treat legacy or new Simulink/Stateflow models with Lustre-related tools, in particular for code generation or verification.

Besides, the automotive and avionics industries are developing control applications to be deployed on distributed architectures, in particular Time-Triggered Architecture (TTA), typically running an OSEKtime operating system, with non-preemptive scheduling and TTP-based communication (see Next TTA 15.1.9. Given this context, there is a motivation to develop a Lustre-centric end-to-end tool-chain from high-level design in Simulink/Stateflow all the way down to a running distributed implementation on multi-processor architectures.

8.2.1 Translation of Simulink/Stateflow to Lustre

The translation of Simulink to Lustre is natural, since both are dataflow-like models. Therefore, the structure of the original Simulink model can be preserved in the translated Lustre program. Still, there are differences between Simulink and Lustre. Lustre has been conceived primarily as a programming language, in particular for critical applications, and

has several features desirable in such applications, such as formal semantics, strong typing and modularity. Simulink, on the other hand, started purely as a simulation environment and lacks many desirable features of programming languages. Simulink has a multitude of semantics (depending on user-configurable options), informally and sometimes only partially documented. Although commercial code generators exist for Simulink (e.g., "Real-Time Workshop Embedded Coder" from Mathworks, "TargetLink" from dSpace) these present major restrictions. For example, TargetLink does not generate code for blocks of the "Discrete" library of Simulink, but only for blocks of the dSpace-provided Simulink library, and currently handles only mono-periodic systems. Another issue not addressed by these tools is the preservation of semantics. Indeed, the relation between the behavior of the generated code and the behavior of the simulated model is unclear. Often, speed and memory optimization are given more attention than semantic consistency.

Our translation is restricted to the discrete-time part of Simulink. This is justified by the fact that in an embedded control application only the controller must be implemented and, to be implementable on a computer, this controller must ultimately be designed in discrete time. Our translation aims at generating a Lustre program which preserves the simulated behavior of the original Simulink model. However, this behavior depends on the simulation options used, as said above. Thus, in our translation method, we have made explicit the options under which the resulting Lustre program is supposed to respect the semantics of the original model.

The translation consists of three separate steps. The first two steps involve type inference and clock inference on the Simulink model. These steps are independent of each other thus can be performed at any order. In fact, they are also independent of Lustre thus can also be used when translating Simulink to another target language. The first two steps produce important typing and timing information which is then used by the third step, namely, the translation itself.

The type system of Simulink can be formalized as an extension of Milner's polymorphic type system for ML which handles parametric polymorphism with type classes, as used for instance in Haskell to handle ad-hoc polymorphism. The situation is simpler for Simulink, which has only three classes, corresponding to "any type", "numerical type" and "boolean". A unification algorithm can be used to infer Simulink types. These types are then mapped to Lustre types.

Clock inference is more sophisticated. We had to develop a new type system where types correspond to Simulink's *sample times*, pairs of the form (π, θ), where π is a period and θ an initial phase. We infer such types from the Simulink model using a special type of unification algorithm, which incorporates properties of sample times. In particular, it incorporates the so-called "greatest-common divisor (GCD) rule" which roughly states that the sample time of the output of a basic Simulink block is the GCD of the sample times of the inputs.

The third step consists in translating hierarchically blocks of Simulink into nodes of Lustre, starting from the basic blocks and moving higher to composite blocks.

The translation of Stateflow is more challenging. On one hand, Stateflow is based on hierarchical state machines, à la Statecharts, a model quite different from the dataflow style of Lustre. On the other hand, Stateflow raises a number of serious semantic problems:

possibility of non-termination of a simulation cycle due to event-stack overflow or junction loops, side effects of junction branches which are not taken, returning to an undefined state after a recursive call of the simulation algorithm (the so-called "early logic return" problem), dependence of behavior on graphical layout, to mention just a few.

Our first task, therefore, has been to identify a "clean" subset of Stateflow which is amenable to translation. For this, we have proposed a number of restrictions which can be checked by static analysis, and ensure absence of the above problems.

We then turned attention to the translation method. The latter is inspired from the standard way of encoding automata into flow diagrams: a signal s encodes the current state of the automaton and equations such as

```
s = if (guard_to_next_state_s1) then s1 else pre(s);
```

encode the transition function. However, this simple scheme is not directly applicable to Stateflow, because of the complexity of its transitions. Indeed, each transition in Stateflow is generally an entire network of *junctions*, each containing many fields (event, condition, condition action, transition action). Some of these fields (the actions) are themselves entire programs written in an imperative language with assignments, etc. The Stateflow interpretation algorithm also adds complexity to the above picture. Indeed, the algorithm is recursively called and run-to-completion, every time a new event is broadcast in the model. All the above make translation extremely complicated, and we had to invent various techniques to deal with such complications. The reader is invited to consult [248] and the corresponding technical report for details.

The translation of Simulink/Stateflow to Lustre has been implemented in a prototype tool, written in Java and Objective Caml (see section 14.7). The tool has been used in various real-life case studies, mostly provided by our industrial partner Audi in the context of IST projects "NEXT TTA" and "RISE".

This work is described in more detail in the papers [158, 248, 80].

8.2.2 Implementation on TTA

In the scope of IST projects Next TTA and RISE, we use Lustre as a high-level programming language for applications which are to be executed on the Time-Triggered Architecture (TTA). So far, development for TTA has relied on low-level descriptions and tools: basically, the designer expresses a set of tasks and messages exchanged between the tasks along the bus. Then, scheduling tools are used to map the tasks and messages to processor and bus slots, using a TDMA scheduling scheme. No global view of the application exists within this scheme.

Our goal is to develop an end-to-end tool-chain from high-level design in Lustre all the way down to running implementations on distributed execution platforms such as TTA. In fact, high-level design can be done not only in Lustre, but also commercial and highly-used tools such as Simulink/Stateflow (TM Mathworks). Such designs can then be translated automatically to Lustre using the techniques we have developed [158, 248, 80], described in Section 8.2.1.

In the course of the above projects the goal of mapping Lustre to TTA automatically has been partially achieved, for the case of time-triggered applications (note that it is difficult to support urgent event-triggered applications in TTA, as in the worst case an event must wait for the next available bus slot in order to be propagated). We now have a tool-chain which can automatically generate distributed TTA code for a multi-periodic Lustre program, provided the user specifies the allocation of "tasks" (that is, parts of the Lustre program) to processors. The processors are assumed to run the OSEKtime operating system, use non-preemptive scheduling, and communicate via the TTP protocol.

In particular:

- We have proposed a set of extensions to the Lustre language, in order to facilitate its implementation on a distributed, real-time architecture such as TTA. These extensions can be seen as "pragmas" and do not change the logical semantics of the program. The extensions include primitives to express assumptions on periodicity of inputs and on worst-case execution times, real-time requirements such as deadlines of the form "output must be produced at most x time units after input", as well as primitives to indicate which part of the program should be allocated to which processor.

- We have proposed methods to partition a Lustre program into tasks and schedule these tasks on TTA, respecting the TTP constraints on bus communication. We have also proposed methods to refine the partition based on feedback from the scheduler for cases where a schedule is infeasible. Finally, we have proposed methods to automatically generated "glue code" (interface code) so that the final running system preserves the logical semantics of the original Lustre program.

The above have been implemented in prototype tools, to be integrated in the Lustre tool-suite. The tools have been used, along with existing tools such as Lustre to C code generators, in a number of case studies provided by Audi, our main industrial partner in the above projects. One of the case studies concerned the implementation of a steer-by-wire controller (camera based) for a prototype Audi vehicle. We used the Simulink-to-Lustre translator and a Lustre-to-C code generator to automatically generate code from Audi's Simulink design. This code was then integrated and run on the prototype vehicle. A demonstration of the system was performed for the reviewers and project partners during the final review of the Next TTA project, on January 22, 2004, at the Audi testbed in Ingolstadt, Germany.

Audi was also particularly interested in verification techniques, at different levels: verification of design, verification of schedules, verification of the distributed run-time system. We have provided methods to address these verification problems. For design verification, we used our Simulink-to-Lustre translator and the Lustre model-checker Lesar. For schedule and run-time system verification, new techniques and prototype tools have been developed.

Parts of this work are described in the paper [157], as well as in the PhD thesis report of Adrian Curic (under preparation).

8.2.3 Implementation with Static-priority Preemptive Schedulers

The above work on implementation of top of TTA is based on an underlying OS that uses time-driven, non-preemptive scheduling to execute the tasks. It is often the case that a preemptive scheduling policy is used instead. Indeed, most real-time operating systems implement a static-priority preemptive scheduler. Each task is assigned a priority: the priority is static in the sense that it does not change during execution of the task. At any point in time, the task with the highest priority among the ready task is executed. When a task becomes ready, the scheduler checks whether this task has higher priority than the task already running, and if so, preempts the running task and executes the new one. When a task becomes idle, the next highest-priority ready task is chosen.

An impressive amount of work exists on schedulability analysis for static-priority preemptive scheduling. However, most of this work focuses solely on schedulability analysis (determining whether a set of tasks can be scheduled without missing their deadlines) and ignores the issue of semantic preservation. The latter is a fundamental issue for achieving a guaranteed model-based implementation chain for embedded software. The goal is to find a method which, (1) on one hand ensures schedulability and (2) on the other hand guarantees that the semantics of the executed system will be equivalent to the semantics of the high-level model.

We propose such a method, where the high-level model is a network of event-triggered tasks communicating with each other. The semantics of this model are ideal, that is zero-time: a task executes instantaneously when it is triggered; it reads its inputs and computes its outputs at zero-time. In such a semantics, each task uses the last value produced by a task it receives inputs from. This is the semantics we wish to preserve.

The assumption is that the tasks are executed using the above static-priority scheduling scheme and the priorities of the tasks are known. Also, for each task, there is a known minimal inter-arrival time of the event triggering the task (this is used for schedulability). Standard schedulability analysis methods are applied to ensure property (1). A novel method is invented for guaranteeing (2). This method is based on a double-buffer scheme for inter-task communication. The fundamental idea is to dissociate the treatment of the triggering of the task with the execution of the task. In our scheme, the operating system handles the triggering of the task as soon as it occurs, even when the task is not chosen to execute. Handling the triggering of the task means appropriately updating the inter-task communication buffers. A double buffer is necessary in order to hold the two values, current and previous, of data communicated to the task. The previous value is necessary when we have a high-priority task A receiving data from a low-priority task B: since A can preempt B, it is impossible to guarantee the ideal semantics unless A uses, not the current, but the previous value produced by B.

This work is described in more detail in the paper [247].

8.3 Perspectives for Future Work

Our ongoing activities focus on consolidating our achievements, as well as studying how to extend and improve them in several ways. We will continue our efforts into developing complete, end-to-end tool chains from models all the way down to executable code. We envisage considering heterogeneous execution platforms (e.g., networks-on-chip, or distributed with communication infrastructure, such as CAN or FlexRay), and more application domains.

In the short-term, we should mention the following goals:

- Applying the Jahuel framework in other industrial applications, in particular, other video encoding standards such as H.264, and strengthening its integration into an industrial development framework.
- Finding appropriate ways to express QoS properties to be used throughout the compilation chain to automatically synthesize the controlled application. Another crucial issue, that needs further study, is efficiency of the implementation to avoid overhead and probe effects due to instrumentation.
- Generalizing our results on preemptive implementations of Lustre to other scheduling policies (e.g., dynamic-priority based) as well as for optimizing the intercommunication scheme (e.g., when the tasks are multi-periodic) [256].

Particular emphasis is put on fostering the cooperation with our industrial partners.

Chapter 9

Security

We aim at developing a theoretic and formal grounding for computer security. From a practical point of view, the goal of our research is the development of efficient and sound tools for the analysis of security properties and security applications.

9.1 Security Protocols

Security protocols are designed to allow secure communication and authentication using insecure channels. They rely upon cryptographic primitives aiming at ensuring the integrity of data, authentication or anonymity of the participants, confidentiality of transactions, and so on. The design of these protocols is often erroneous, even when assuming that cryptography is perfect, i.e. when encrypted messages cannot be deciphered without the inverse key. Today, we still lack semantically clean and widely accepted languages for describing security protocols. The situation concerning their properties is worse. Indeed, confidentiality and authentication are more or less well understood (it is actually not completely true of the latter) but anonymity, opactness, fairness, denial of service etc.... are far from being well studied. The analysis of cryptographic protocols is complex also because the set of configurations is huge or infinite: it is necessary to take into account any number of sessions, any size for the messages, the interleaving of sessions, the algebraic properties of cryptographic primitives or data structures. Interesting surveys can be found in [Mea03, Mea04].

Our main contributions in this area are the following:

9.1.1 Verification of Security Protocols within Dolev-Yao Model

We developed a general verification method for security protocols that can handle unbounded sessions, unbounded message size and unbounded fresh nonce creation [137, BLP06]. The distinguishing feature of our verification method is that, given a protocol and a message, it computes a precondition that guarantees that the message remains secret. This is done by a combination of static abstractions and approximated fixpoint computation using a symbolic representation of sets of terms. This verification method is implemented in the tool Hermes [138]. Hermes can be tested online at

Yahalom	OK	13.81
Needham-Schroeder Public Key	Attack	0.04
Needham-Schroeder Public Key (with key server)	Attack	0.90
Needham-Schroeder-Lowe	OK	0.03
Otway-Rees	OK	0.01
Kao-Chow_1	OK	0.79
Kao-Chow_2	OK	16.89
Neumann-Stubblebine	OK	0.04
Needham-Schroeder Symmetric Key	Don't know	0.08
TMN	Attack	2.78
TMN-Lowe	Attack	3.04
Denning-Sacco	Don't know	0.04
ISO Symmetric Key One-Pass Unilateral Authentication	OK	0.01
ISO Symmetric Key Two-Pass Unilateral Authentication	OK	0.01
ISO Symmetric Key Two-Pass Mutual Authentication	OK	0.01
Andrew Secure RPC	OK	0.02

Table 9.1: Running time of Hermes

`http://www-verimag.imag.fr/~Liana.Bozga/hermes/hermesgui.php`. Table 9.1 gives an idea how our tool Hermes performs on classic authentication protocols.

In [135], we define a sound and complete inference system for bounded cryptographic protocols. Bounded protocols means that a bounded number of sessions is considered but the message size is still unbounded. It is not difficult to encode a counter machine as an unbounded cryptographic protocols (CP, for short). Hence, it is not possible to have an effective and complete (and sound) inference system. We show that such a system exists for bounded protocols. This provides an alternative proof of the decidability of secrecy for bounded CPs and covers more properties than in existing work. We introduce a logic, called SPL for Security Properties Logic, for describing security properties and develop a calculus for computing the weakest condition that has to be satisfied by the initial configurations of the protocol in order to guarantee that a property described by an SPL formula is satisfied. We prove soundness and completeness for the introduced calculus. Then, we study the decidability of SPL and show that although the satisfiability (existence of a model) of SPL formulae is, in general, undecidable, it is decidable for its existential fragment, i.e., the satisfiability of formulae of the form $\exists X.\varphi$, where φ is quantifier-free can be decided effectively. Now, it turns out that interesting security properties are expressible in the universal fragment of the logic and that the weakest precondition of a universal formula is expressible as a universal formula. Hence, given a protocol π and a property φ, using the calculus one can compute a formula $wp(\pi, \varphi)$ such that there is an attack starting for an initial state satisfying ψ iff $\neg wp(\pi, \varphi) \wedge \psi$ is satisfiable. Thus, if ψ is given in the existential fragment, which is the interesting situation, one can effectively check whether $\neg wp(\pi, \varphi) \wedge \psi$ is satisfiable.

9.1.2 Beyond Secrecy and Authentication

We also studied anonymity properties and developed new results. In [229] it is proved that opacity (the inability of intruder to decide for an execution of a security protocol, that a property is satisfied or not) is decidable in the case of a passive intruder. This work is further extended in [227], where the hypothesis of encryption with only atomic keys is dropped, and in [228] where it is proved that a restricted version of opacity is still decidable for active intruders. In [286], opacity is extended to labelled transition systems and generalized in order to better represent concepts from the work on information flow. In particular, in this work, links between opacity and the information flow concepts of anonymity and non-interference such as non-inference are established. Also ways of verifying opacity when working with Petri nets are studied.

9.1.3 Timed Security Protocols

Most work done in verification of security protocols uses a formal model, called the Dolev-Yao model. Important hypotheses of Dolev-Yao model are: 1) the encryption algorithms are perfect (the intruder must know the inverse key to obtain the plaintext from a ciphertext) and 2) nonces (random numbers used to ensure the freshness of a message) are ideally generated (they do not collude, and can not be "guessed"). In [136, 48] the hypothesis of "perfect nonces" is weakened and we extend the method of [135] to provide a sound and complete method for the verification of security protocols that use timestamps. To do so, we augment security protocols with clocks and messages with timestamps. The protocol transition are then guarded by constraints on the clock and timestamp values. Such constraints may be used to model computation time as well as timeouts and delays.

9.1.4 Soundness of the Symbolic Model

Historically, verification of cryptographic protocols has been separated in two distinct branches. *Formal verification* of cryptographic protocols, originates from the work of Dolev and Yao [DY83]. The essential part of this approach is the perfect cryptography assumption that can be roughly summarized as follows: messages are represented as algebraic terms, fresh nonce creation is perfect, that is, nonces range over an infinite domain and freshness is absolute, the same holds for key creation. Moreover, there is no way to guess a nonce or a key and no information can be extracted from an encrypted message unless the inverse of the key used to encrypt the message is known. In this approach there is a single attacker that is modeled as an infinite process without bounds on its computational resources.

In the *complexity-theoretic approach*, cryptographic primitives operate on strings of bits and their security is defined in terms of high complexity and weak probability of success [GM84, BKR94] of any attacker. Protocols as well as attackers are randomized polynomial-time Turing machines. This computational approach is recognized as more realistic than the formal approach, however, its complexity makes it very difficult to design automatic verification tools.

Therefore, results of the type:

> If protocol Π uses the cryptographic schemes $S_1, \cdots, S_N$, if each schema S_i is correct with respect to the security notion C_i and if some additional syntactic conditions are satisfied by Π then the formal model is a safe abstraction of the computational, that is, correctness of the protocol established in the formal model implies its correctness in the computational one.

are of extreme importance for gaining confidence that a cryptographic protocol is secure.

In the last years, attempts have been made to bridge the gap between the formal and complexity-theoretic approaches. The ultimate objective is to be able to prove security in the formal model, then to prove properties on the encryption scheme and with that to deduce security of the protocol in the complexity-theoretic model. The first work to mention here is maybe Abadi and Rogaway's paper [AR00]. They proved that a notion of message *indistinguishability* in the formal model is valid in the computational model when making some assumptions on the encryption scheme. This means that if two messages are not distinguishable in the formal model, then their computational equivalents cannot be separated by a Turing machine in a reasonable (polynomial) time. This paper deals, however, with passive attackers that do not intervene during protocol execution. Active attackers are considered in [War03, MW04, BPW03, Lau04, CW04, 284]. Essentially, these papers prove that if the encryption scheme verifies a certain property (e.g. IND-CCA), then security in the formal model implies security in the computational model. However, each of this papers lacks generality. For instance the results in [War03, MW04] does not deal with secret key transmission. Cortier and Warinschi prove in [CW04] safety of the formal model for protocols that use asymmetric encryption and signature under the assumption that secret and signature keys are not sent. A similar result is proved by Backes, Pflizmann and Waidner [BPW03]. In the latter, the formal model is not exactly the Dolev-Yao model. P. Laud [Lau04] proves safety of the formal model for symmetric encryption. In particular, he deals with encryption cycles.

None of these papers deals with hashing and protocols that combine asymmetric and symmetric encryption, signature and hashing.

In [208], we prove that the Dolev-Yao model is a sound abstraction of the complexity-theoretic model for protocols that combine asymmetric encryption and digital signature, provided that the underlying encryption scheme is Indistinguishable under Chosen Cyphertext Attack (IND-CCA) and that digital signature is correct with respect to Selective Forgery Against Adaptive Chosen-Message Attack. Informally, the first condition states that an attacker can do only slightly better than flipping a coin in order to distinguish between the encryptions of two messages he chooses. Moreover, this is the case even when he has access to encryption and decryption oracles. Here an attacker has to be understood as a polynomial time randomized Turing machine. So the idea is that if we call the advantage of an attacker A the difference between the probability of his success in distinguishing the two messages and the probability of getting the answer correctly by simply flipping a coin is negligible in the formal sense. Informally speaking, the second condition, which is related to digital signature, states that an attacker cannot exhibit a message and its signature. Our work extends previous ones in several ways: 1) generalizes to multi-party protocols, 2) allows symmetric/asymmetric encryption as well as digital

signature, 3) encoding of secret keys is allowed, 4) applies to secrecy properties as well. In [285], this result is extended to symmetric and asymmetric encryption, signature and hashing. This result is extended to include Diffie-Hellman exponentiation in [288].

The soundness of the symbolic model when probabilistic opacity is considered is proved in [289]. A predicate is opaque for a given system, if an adversary will never be able to establish truth or falsehood of the predicate for any observed computation. In the Dolev-Yao model, even if an adversary is 99% sure of the truth of the predicate, it remains opaque as the adversary cannot conclude for sure. In this paper, a computational version of opacity is introduced in the case of passive adversaries called cryptographic opacity. The main result is a composition theorem: if a system is secure in an abstract formalism and the cryptographic primitives used to implement it are secure, then this system is secure in a computational formalism. Security of the abstract system is the usual opacity and security of the cryptographic primitives is IND-CPA security. To illustrate the results, two applications are given: a short and elegant proof of the classical Abadi-Rogaway result and the first computational proof of Chaum's visual electronic voting scheme.

This work is conducted in tight connection with the following projects:

- EVA: Explication et Vérification Automatique de protocoles cryptographiques
- ACI Rossignol: Verification of Cryptographic Protocols.
- RNTL PROUVE: Verification of Cryptographic Protocols.

9.2 Certified Applications

The International Common Criteria, CC for short, is an international standard for certified security applications. It defines requirements on the development as well as evaluation of the product under certification. In the United States, the NIST [1] together with the NSA [2] are undertaking effort to define a national program for the evaluation of information technology products for conformance to the International Common Criteria.

There are seven *Evaluation Assurance Levels* from EAL1 to EAL7 with increasing demand of formal specifications and proofs. At the highest, level formal proofs of refinement steps are required from the developer. We are interested in what is called the ADV-class in the CC that defines requirements for the stepwise refinement of the application from the security policy specification. The CC distinguish several abstraction levels of the application:

- the *Security Policy Model* (SPM) defines the executions that fulfill security rules;
- the *Functional Specification level* (FSP);
- the *High Level Design* (HLD) describes the modules, their interface and their links;

[1] The National Institute of Standards and Technology
[2] National Security Agency

- the *Low Level Design* (LLD) defines the implementation of the modules and their functionalities;
- the *Implementation level* (IMP) refers to the executable code of the applications.

The certification process aims at establishing that the implementation respects the security policy. The CC impose to prove a chain of *Refinement Conformance Relation* (RCR) relating one level to its predecessor: [3]

$$\text{IMP} \underset{IMP \to LLD}{\sqsubseteq} \text{LLD} \underset{LLD \to HLD}{\sqsubseteq} \text{HLD} \underset{HLD \to FSP}{\sqsubseteq} \text{FSP} \underset{FSP \to SPM}{\sqsubseteq} \text{SPM}$$

It turns out that the CC leave open the nature of the refinement relations, the specification languages and the verification techniques that should be used. These issues are particularly relevant for the highest evaluation level, the EAL7, as it requires formal proofs and as there are hardly any product certified at this level.

In 2002, we started a project, the RNTL project EDEN, with the industrials partners TRUSTED-LOGIC, CESTI, AXALTO and LETI to answer these questions and develop a certification methodology for level EAL7. Our goal is to provide tools and a methodology to support the *certification of smart card applications* at the highest level of formalization. The vendor TRUSTED-LOGIC – specialized in security applications – develops an environment based on UML that helps filling the documents requested for the certification review. This tool has been used for certification until the level EAL4 but does not provide any formal verification. VERIMAG is in charge of the tools for formal verification and the methodology.

Methodology

Our main contribution is the definition of a development methodology with tool support that aims at the highest level of certification. In our methodology, refinement relations correspond to trace inclusions with the following difference: two related levels are not necessarily expressed on the same alphabet. Indeed, a security policy (SPM) maybe expressed in terms of high level events such as authentication, transaction, setting of a variable, reading of a memory area, ... and focusses solely on the security aspects while the functional specification also specifies the functional properties of the application. It turns our that our definition of refinement relations is compatible with the notion of assurance mapping of the TCSEC.

We then introduced a notion of mapping relating two levels A and B. A mapping $\mathcal{M}$ is a set of tuples $(\varphi, \omega, \omega', \psi)$ where φ, ψ are predicates on the states of $A \times B$, and ω, resp. ω' are sequences of events on the alphabet of A (Σ_A), resp. B (Σ_B). A tuple $(\varphi, \omega, \omega', \psi)$ relates two chunks of traces, $s_a \xrightarrow{\omega} s'_a$ and $s_b \xrightarrow{\omega'} s'_b$, if $\varphi(s_a, s_b)$ and $\psi(s'_a, s'_b)$ hold. Now, a trace of A and a trace B are in the $\mathcal{M}$-induced refinement relation, if they can be split into related trace chunks. In other words, for a trace t_A of A we associate a set $\alpha_{\mathcal{M}}(t_A)$ of related traces on the alphabet Σ_B. We extend this pointwisely to the set of traces of

[3] The RCR proof between IMP and LLD is not considered in the EDEN project.

A. Let us denote by $Image_{A\to B}(Trace(A))$ this set. Then, we formalize the RCR relation $\underset{A\to B}{\sqsubseteq}$ as a trace inclusion through the mapping

$$A \underset{A\to B}{\sqsubseteq} B \stackrel{def}{=} Image_{A\to B}(Traces(A)) \subseteq Traces(B)$$

Note that a mapping definition where all φ predicates equals $false$ leads to a constant function $Image_{A\to B}$ equals to $\emptyset$. So, for the definition to make sense each trace of A must have a non-empty image on the domain of B traces. We use this criterion to reject meaningless mapping.

Supporting Tools In our methodology, the description of the application at all levels of abstraction as well as the refinement relations are specified in a Java-like syntax. VERIMAG's contribution consists in plug-ins for the ECLIPSE environment that translate the specifications into IF communicating state machines. These are then used to check the refinement between successive specification levels as follows. The verification of the refinement $A \underset{A\to B}{\sqsubseteq} B$ between two levels A and B is done in four steps:

1. each level is translated into IF processes and observers and each basic mapping $(\varphi, \omega, \omega', \psi)$ is translated into an IF observer.

2. The IF generator produces the model of level B.

3. The model of $Image_{A\to B}(A)$ is obtained by running the mapping observers in parallel with the processes of level A and the process Σ_B^*. The observers filter the traces of Σ_B^* which map onto traces of A.

4. Finally, the plug-in uses the CADP toolbox associated to IF in order to check the inclusion of the models $Image_{A\to B}(A)$ into those of B.

The levels FSP,HLD and LLD correspond to programs written in a restricted subset of Java. An interesting part of the methodology concerns the specification of the security policy model: the set of authorized traces is defined in two steps: 1) the user specifies all the commands of the smart card that matter for the security policy. For each command, he describes the condition of success, failure, error and the property that must hold in case of interruption of the command (*e.g.,* getting the card out of the device). From the commands we generate an IFprocess which executes all the commands in any order with any possible arguments (using an abstraction on arguments). Its model is the most liberal policy. 2) The IFprocess is instrumented to emit events of interest for the security policy. 3) By the mean of temporal properties the user restricts this model to his wishes. These properties are translated into IFobservers which reject the traces not satisfying the constraints.

Following the TCSEC, the security policy model defines the authorized behaviors. Several computation models have been experienced to specified a formal SPM. They differ on the style of specification algebraic versus operational and on the observable events (see [FP] for a survey). The Mealy or Moore state machines provides an operational way to

define the authorized traces ; the set of observable events depends on the security policy. Algebraic framework (like Parnas or Hoare traces) defines the set of legal traces as the traces satisfying inductive properties and logical constraints on the order of events. Parnas traces focuses on functions call events while Hoare traces are designed for studying the communication of processes.

Our framework is a combination of these approaches: the most liberal policy is described using a state machine, the observable events are defined by the output function of a state machine ; and the legal traces is the subset of liberal traces that satisfy additional temporal properties.

9.3 Mobile Code Specification and Verification

Many embedded systems are not standalone systems : they communicate with other embedded systems or with base stations. This is clearly the case in aircrafts, smartcards, and in mobile phones, for instance. Things get even more complicated by the fact that the environment of mobile systems changes during execution in various ways, in particular their connection topology : an entity may be disconnected, then become connected to other one or several other entities, which can themselves subject to similar changes. Note that, upper bound of communication delays can become arbitralily high, making failures indistinguishable from long delays and thus undectable.

Dually, in some applications, code is successively executed on several locations during the same run, e.g. in order to provide a load-balancing mechanism, or in order to collect pieces of information in a distributed environment.

Both kind of mobility—hardware mobility and software mobility—can happen together, and they be abstractly considered as two instances of the same phenomenon, that is, mobile computing. However, it is not clear that classical approaches, say based on fixed labelled transition systems, provide the appropriate abstraction levels for designing, modeling and reasonning about such systems. In particular, the concept of location, either physical or virtual, becomes a key notion. Several approaches have been proposed in the scientific community in order to deal with mobile computing. A number of implementations are based on scripting languages (e.g. telescript), object-oriented languages (Java) and agents. We concentrate on process calculi, which are can represent all underlying mechanisms and are better for analysing concepts and reasoning on programs. Three of the most prominents are the pi-calculus of Milner and Parrow (as well as a number of variants, such as the asynchronous π-calculus or the π_l-calculus), the join-calculus of Gonthier, Levy, Maranget and Fournet ; and the ambient-calculus of Cardelli and Gordon.

The π-calculus is their common ancestor. It is essentially a new version of CCS where processes exchange (channel) *names*. The join-calculus can be considered as a specialized version of the π-calculus where interactions are restricted in such a way that commputational decisions can always be taken *locally* (there is no need for distributed consensus). The ambient-calculus provides high-level features for expressing moves in a dynamic tree of locations and reasonning on security barreers. All these calculi have the same expressive power (each of them can be encoded in the others).

A convenient paradigm for explaning the behavior of systems composed of mobile processes is the chemical abstract machine (CHAM) of Berry and Boudol. Onb the sematical side, this boils down to distinguishing two kind of transitions: reductions, which produce new components (ie. processes or messages) from old ones, and structural rules, which just rearrange components (mathematically, this amounts to expressing that certain operators for composing processes are associative and commutative).

Ongoing Work at Verimag

The research described above deals mainly with *describing* general mobile systems and *reasoning* about them using algebraic laws and bisimulation techniques. We expect to extend such results in (at least) two directions, in the context of VERIMAG:

- studying the case of embedded systems;
- relying on well-known automated verification techniques.

The research trend is quite recent at VERIMAG (it started in 2004) and at the moment, only the second direction is worked out. A first idea is to take advantage of the support of the IF-2.0 environment. H. Venturini, with the help of M. Bozga end JF. Monin, studied a systematic translation of the ambient-calculus into IF-2.0. This translation has been applied on the Result Delivery Protocol proposed at the Universidade de São Paulo, which is used in a distributed information system for mobile agents devoted to collect traffic data through a fleet of vehicles. Only a fragment of the ambient-calculus was translated and needed for this application, and the simulation environment for IF-2.0 was successfully experimented. These results are described in [295]. The next step is to specify and verify relevant properties.

A complementary line of research that we follow consists of securing the reasoning on mobile systems using a proof assistant. We already achieved a formalization of the semantics of the distributed join-calculus in a version of type theory called the calculus of inductive constructions, wich is supported by the Coq proof assistant. This formalization was presented at IWFMS'04 [301].

Chapter 10

Hybrid Systems

10.1 Introduction

The research on hybrid systems at VERIMAG has as a major objective to export some ideas and insights originating from computer science toward other domains of applied science and engineering that do not enjoy these insights and for which the computer is often viewed merely as a tool, rather than a source of inspiration for a new kind of mathematical models. This research direction is motivated by the conviction that concepts developed within various areas of computer science and engineering have a large potential contribution to the emergence of a modern *system design discipline* for complex and heterogeneous systems, embedded or not. The concepts that we have in mind include the distinction between syntax and semantics (in logics, programming languages and system description formalisms), relationships between models at different levels of abstraction, verification and controller synthesis using graph-theoretic techniques, algorithms and complexity, hierarchical and modular modeling and so forth. To be able to communicate these ideas to diverse communities, one must invest in understanding their languages, models (systems over continuous state variables and metric time), and performance criteria which, more often than not, are quantitative rather than qualitative ("non functional" in contemporary parlance).

The hybrid systems group at VERIMAG is one of the founders and world-wide leaders in hybrid systems, as witnessed by membership in the steering and program committee of the major hybrid systems conference [13], the foundation of new workshops on timed systems [270] and on verification of analog circuits [277], the coordination of and participating in several top Europeans projects on the topic (VHS, CC, AMETIST), writing some introductory articles [70, 71, 223] and, of course, some important contributions to the domain in concepts, theory and tools. All these achievements are due to readiness to invest in understanding topics that cannot be solved by a small modification of one's own favorite concepts and tools, an investment that does not always pay in terms of immediate recognition by a close community (which is a major reason why so few computer scientists work on the topic).

10.2 Verification and Synthesis of Continuous and Hybrid Systems

The major innovative contribution of computer science to hybrid systems is the definition of the *hybrid automaton* model and the adaptation of verification techniques to such automata, where the hard part is, of course, the treatment of the continuous dynamics. After a short historical survey we explain the contribution of VERIMAG to this area during the reporting period.

10.2.1 Historical Survey

The hybrid automaton model was proposed in the early nineties in order extend verification toward systems that include both discrete and continuous components. The first papers on the topic advocated a deductive approach for verification but soon, influenced by decidability results for timed automata, the emphasis moved to algorithmic verification which means essentially to compute all the reachable states of a system, starting from an initial set and subject to an admissible set of inputs (disturbances). The early days of the domain focused on the so-called "linear" hybrid automata, in which the continuous variables have a *constant derivative* in every discrete state (mode). Like in timed automata, the set of time successors of a polyhedral set of states can be computed using linear algebra without resorting to the solution of real differential equations.[1]

However it turned out soon that except for some very special cases, the reachability problem for such automata is undecidable, a fact that, together with the expressive limitations on the continuous dynamics, led to a decay in the interest in this domain, except for some interesting theoretical results. From a tool development point of view, the outcome of this period include Kronos, by Sifakis and Yovine for timed automata and HyTech by Henzinger et al for "linear" hybrid automata. These tools and related papers have also established the standard form of reachability computation for timed and hybrid systems by alternating continuous dynamics and discrete transitions. Starting from a set P of states the procedure works as follows:

1. Compute the set of time successors of P according to continuous dynamics of the current mode

2. Intersect the result with the staying conditions of the mode to obtain all states reachable from P via transition-free trajectories

3. For each transition, intersect the set of time successors with the transition guard to obtain all "source" states of the transition

4. Compute the "target" set of the transition by applying the reset map to the set of source states

[1] This is part of the attractiveness of such automata for computer scientists.

5. Restart the computation from the obtained set according to the continuous dynamics of the target mode.

It is important to mention that interest in hybrid systems was (and still is) very high in the control community, relatively much more than within computer science. This is partly due to the fact that many of the analytical methods used in control design are tailored for purely continuous systems and hence are restricted to idealized subsystems (for example, ignoring saturation or assuming linearity) and do not extend easily to the hybrid case. However, much of the early work from the academic control community was either on the fringe ("intelligent control") or more orthodox mathematical work devoid of significant computational content. Toward the end of the 90s, together with the reorganization of the hybrid systems community around the HSCC workshop, the focus moved toward reachability computation for "real" continuous systems, those that their continuous dynamics are defined by non-trivial differential equations. Of course, given all the undecidability results for simpler models, there is no hope for *exact* computation which, when you come to think of it, is a too strong requirement for physical continuous systems. In fact, the process of accompanying ideas from finite-state systems to systems with numeric variables involves an ongoing lowering of expectations, starting from exact computations which are guaranteed to terminate (timed automata), via exact computations that may not terminate ("linear" hybrid automata) to approximate computations (arbitrary differential equations). As we shall see later, the process does not stop there.

In the next section we describe briefly the techniques underlying **d/dt**, the major contribution of VERIMAG to reachability computation for hybrid systems. Although this is not intended to be a scholarly survey, we mention some preceding relevant work such as that of Kurshan and McMillan on using reachability to construct finite-state abstractions of circuits from transistor models, related work by Greenstreet, various works in numerical analysis using *interval arithmetics* and the work of Kurzhanski et al on *ellipsoids*. Other tools that were developed in parallel with (and independently of) **d/dt** are CheckMate by Chutinan and Krogh at CMU, which used similar ideas, and Hysdel by Bemporad and Morari at ETH, which employs hybrid optimization to solve verification and synthesis problems for bounded horizon. Later tool development contributions to the domain include the work of Mitchell and Tomlin at Stanford on using PDE solvers for reachability computations and the Verishift tool of Botchkarev and Tripakis (then a post-doc at Berkeley) which uses ellipsoids.

10.2.2 Linear Systems and the d/dt Tool

The tool **d/dt** [106, 107, 108] has been developed within the thesis of Thao Dang, 1996-2000 under the supervision of E. Asarin and O. Maler. It treats systems that are piecewise-linear or, more generally, piecewise-affine, that is, hybrid automata where the continuous dynamics in each mode q is defined by an equation of the form $\dot{x} = A_q x + B_q u$ where u is an external disturbance bounded inside some convex set. The basic problem is to compute an over-approximation of the time successors of a convex polyhedron P under the continuous dynamics. For constant derivative systems this can be achieved for all times by linear

algebra, but in the general case we need to imitate numerical integration by fixing a time step r and computing a sequence of sets $\{P_i\}$ such that each P_i contains states reachable from P in the time interval $[(i-1)r, ir]$. Note that for discrete-time systems, P_i consists of points reachable from P *exactly* at time ir. This computation is done by numerical integration from the vertices of P, followed by a convex hull and additional modification to guarantee over-approximation. The approximation error does not accumulate and can be controlled by making r smaller. The sequence $\{P_i\}$ is stored either as a union of convex polyhedra or over-approximated by a non-convex orthogonal polyhedron for which we have developed a canonical representation. It is typically the intersection with transition guards that makes the shape of the reachable set more complex and requires additional vertices.

In addition to verification, **d/dt** also offers a procedure for synthesizing switching controllers for safety specifications and handles under-specified disturbances by performing an optimization procedure at every step for finding the input u that pushes each face of the polyhedron in "outermost" way. The tool is available publicly and its reachability algorithm has been integrated as the computational engine behind the Charon modeling and verification tool [96] developed at the university of Pennsylvania. Due to the need to manipulate polyhedra whose dimensionality is that of the state space, the tool is currently limited to system of a modest dimension (around 5). In the sequel we describe extensions to **d/dt**, mostly due to the work of Thao Dang and collaborators, as well as some other investigations in hybrid reachability.

10.2.3 Extension to Non-Linear Systems

The extension of **d/dt** to deal with non-linear systems [105], was one of the major developments during this period. This work was to a large extent a result of a successful collaboration between VERIMAG (E. Asarin., S. Yovine, T. Dang) with the applied mathematics MOSAIC team (J. Della Dora) within the IMAG Mash project [179, 180], culminating in the thesis of Antoine Girard [27] . Underlying this extension is a new method of "hybridization", that is, transforming any reasonable non-linear system $\dot{x} = f(x)$ into a *piecewise-affine* system that approximates its behavior. This transformation is achieved by first triangulating the continuous state space into simplices, and then for each simplex, computing by interpolation the linear operator A that agrees with f on the vertices of that simplex. The maximal error between f and A is estimated and represented in the piecewise-affine dynamics as an additive disturbance. This way the system is modeled as a hybrid automaton where each state corresponds to a simplex and the transitions correspond to boundary crossings. Other good properties of the transformation include its continuity along the boundaries and its good error convergence. These results, which demonstrate also how hybrid system ideas can sometime breathe life into old disciplines like numerical analysis, have been integrated into **d/dt** and tested on several examples.

10.2.4 Treating Parameters

In the analysis of a continuous system $\dot{x} = f(x, u)$ we make a distinction between state variables x and input variables u. The former need to be stored at each step because

they determine the future evolution of the system. Hence each of them increases the dimensionality of the stored polyhedra. Inputs, on the other hand, are not part of the state space and need not be stored because at every step they can "choose" a value taken from a bounded set, independently of their previous value. Parameters allow to define systems of the form $\dot{x} = f(x, u, p)$ where p is chosen at time zero and does not change during the lifetime of a trajectory. Hence parameters are like state variables in the sense that they need to be stored, but different from them in the sense that they do not undergo any dynamics. A specialized procedure that treats parameters properly and hence reduces the dimensionality of matrix computation has been developed and integrated.

10.2.5 Differential Algebraic Equations

The dynamics of electrical circuits, a new application domain for hybrid systems, is typically described using *differential algebraic equations* whose numerical integration is a crucial part in circuit simulation. When those equations are of index one, their numerical solution can be performed as a combination of standard integration and projection on the constraint surface. This technique has been implemented and tested on small analog circuits [173].

10.2.6 Experience with d/dt

Most of the few dozens of **d/dt** users are academic (we have some downloads from industry but without feedback). These users come from all over the world including France (INRIA, IRISA, Supelec), USA (Notre Dame, Vanderbilt, Penn, SRI, Honeywell), Germany (Oldenburg, Aachen, Karlsruhe, Hannover), as well as Netherland, Italy, Russia, China and Brazil. We currently have no resources to maintain an interactive user group, but we are aware that some use it for educational purposes. At VERIMAG we have applied **d/dt** to various case studies originating from diverse domains, including:

Automotive: Two case studies were treated by the predicate abstraction tool build on top of **d/dt** within the DARPA-funded MoBies project. The first was an automotive coordination system, with 4 continuous variable and a dozen of modes which was proved to be collision free. The second was an electronic throttle control system, with 7 continuous variables and 8 modes. Within the CC project **d/dt** has been used to synthesize an idle-speed controller on a model with 6 real variables and 3 modes [54].

Computational biology: The non-linear extension of **d/dt** was used to analyze a model of the genetic regulatory network of Vibrio fisheri with 3 variables, non-linear dynamics and several modes [109].

Analog circuits: A model of a biquad low-pass filter with 3 continuous variables, and dynamics defined by differential algebraic equations with some non-linearity has been verified in [173]. On the other hand, to illustrate the current limitations, we have not succeeded so far in verifying a sigma-delta A-to-D converter with 3 state variables and two modes due to the frequent mode switching that may happen at every moment, and had to resort to bounded horizon techniques.

10.2.7 Abstraction of Hybrid Systems

Abstraction techniques fight state explosion by replacing a complex system S by a simpler system S' which over-approximates S in the sense that every behavior exhibited by S can be exhibited by S'. We have focused on two commonly-used approaches to abstraction, one is the reduction to a finite-state system by *partitioning* the state-space ("predicate abstraction"), and the other is by *projection*.

Predicate abstraction works by partitioning the continuous state space into regions, each of which is viewed as a discrete state of S'. Then the transition relation of S' consists of all pairs (R, R') of regions such that there exists a direct trajectory (a trajectory that does not visit any intermediate region) between some $x \in R$ to some $x' \in R'$. The computation of this relation using reachability techniques is much simpler than standard reachability because the complexity of the sets does not accumulate. In fact, one can avoid explicit reachability computation altogether by analyzing the vector field along the boundary between R and R' and establishing the existence of a boundary point in which it has a positive normal component.

However the artificial "transitivity" in the transition relation of S' may create many false counter examples, that is, abstract behaviors that do not correspond to any concrete trajectory. When such a counter example is encountered, one can refine the partition hoping to create eventually an abstraction fine enough, either for proving the property or for finding a counter example that can indeed be concretized. This methodology, popular for verification of software and infinite-state systems in general, has a lot of heuristic ingredients such as the choice of the initial partition (based sometimes on the properties to be proved) and its refinement procedure. In a series of works [97, 98, 99, 43, 44] the applicability of counter-examples based abstraction refinement for hybrid systems has been explored.

The other natural abstraction technique is to reduce the state-space of the system by projecting away certain variables. A variable thus projected moves from the category of state variables to that of under-specified inputs, resulting in a more non-deterministic system. In other words, we over-approximate a system of (nonlinear) differential equations by a hybrid system with differential *inclusions* in lower dimension. The technique developed in [109] is based on this principle, augmented with ideas coming from qualitative physics and with a method for error control. While this abstraction approach can be used for dimension reduction, its effective application requires the ability to deal with nonlinear differential inclusions. We developed a reachability analysis method for uncertain bilinear systems which, in combination with the abstraction by projection, allows us to treat multi-affine systems. An experimental implementation of this technique enabled us to study a bacteria model in bio-regulator networks, but more experimentation is needed to assess its scope and limits.

10.2.8 Analysis of Polygonal Hybrid Systems

In parallel with approximate computations, the limits of exact reachability have been explored. Polygonal hybrid systems are a subclass of planar hybrid automata which can

be represented by piecewise-constant differential inclusions. We have studied qualitative properties of such systems and established several decidability and undecidability results within the thesis of Gerardo Schneider (supervised by S. Yovine and E. Asarin).

In [112], we have developed an algorithm for solving the reachability problem. Our procedure is not based on the computation of the reach-set but rather on the computation of the limit of individual trajectories. A key idea is the use of one-dimensional affine Poincaré maps for which we can easily compute the fixpoints. We have shown that between any two points linked by an arbitrary trajectory there always exists a trajectory without self-crossings. Thus, solving the reachability problem requires considering only those. We proved that, indeed, there are only finitely many qualitative types of those trajectories. For each type, we constructed a decision procedure based on the analysis of the limits of extreme trajectories.

In [113] we have studied phase portraits. We analyzed the qualitative behavior of sets of trajectories having the same cyclic pattern. Using the classification of cyclic behaviors of [112], we have given a classification of cyclic behaviors, we have shown how to compute the *viability* kernel of a cycle, that is, the set of points which can keep rotating in the cycle forever. We have shown that this kernel is a non-convex polygon and give a non-iterative algorithm for computing the coordinates of its vertices and edges. Furthermore, have defined and computed the *controllability* kernel, a cyclic polygonal stripe within which a trajectory can reach any point from any point, analog of the notion of limit cycle. Indeed, we have proven that the distance between any infinite trajectory performing forever the same cyclic pattern and the controllability kernel always converges to zero. These results are the analog of Poincaré-Bendixson for polygonal differential inclusions. A tool for qualitative analysis of such systems and based on these results has been developed by Gordon Pace [111].

In [114] we have revisited the decidability of the reachability problem for hybrid systems of a low dimension. We have characterized several classes of two-dimensional hybrid systems for which the problem is undecidable, and found other classes for which reachability is as hard as reachability for piece-wise affine maps, which is a very well known open problem.

10.2.9 Symbolic Reachability Computation

In [68] we have presented the first known families of linear differential equations with a decidable reachability problem. This is achieved by posing the reachability computation as a quantifier elimination problem in the decidable theory of the reals. We have characterized families of linear control systems of the form $\dot{x} = Ax + Bu$, where u belongs to a set U of possible inputs, for which the set of reachable states is definable in the theory of real numbers. Indeed, we have identified fragments of the real field extended with exponential and trigonometric functions that admit quantifier elimination by applying an appropriate change of variables. A quantifier-free representation of the reachable sets can then be obtained by quantifier elimination using tools such as REDLOG and QEPCAD. This result is the basis of the REQUIEM tool later developed at the University of Pennsylvania.

10.2.10 Perspectives for Future Work

Computing trajectory tubes for continuous and hybrid systems subject to disturbances and parameter variations is a novel contribution to control and to the analysis of dynamical systems. It can replace analytic methods when those fail, and complement simulation for small critical subsystems. These techniques suffer, however, from severe *state explosion* that limits their applicability. Like discrete formal verification, they are not easy to explain to engineers who are accustomed to reasoning about *individual trajectories*. Our current thinking is that progress should be made along two complementary directions, algorithmic improvements and further relaxations of completeness criteria.

The first direction is to improve the performance and usability aspects of **d/dt** and change its nature from a proof-of-concept to a more mature tool. A lot can be done in improving the "main loop" of reachability analysis by using alternative geometric objects and data structures (we are currently exploring the applicability of *zonotopes*, a special class of polyhedra, whose usefulness for reachability computation has been recently demonstrated by Antoine Girard), employing variable-step time discretization which can be coarser in uninteresting parts of the state space, using more sophisticated forms of abstraction or by exploiting analytic information extracted from the system dynamics.

Such an effort should aim at increasing the capabilities of the tool by one order of magnitude, say, few dozens of modes and few dozens of state variables. Such an improvement together with a more flexible user interface will increase the user base of **d/dt** and provide the necessary feedback for future developments and eventual industrial transfer. To achieve this ambitious goal an *additional researcher* and an *engineer* are badly needed.

The second direction consists in seeking alternatives to the "classical" approach for analyzing hybrid (and timed) systems by exhaustive computation of reachable sets of states, and develop methods based on adaptive search in the space of (simulated) trajectories. Such methods can be more intuitive for the engineer and will offer different tradeoffs between computational complexity and coverage, much in the spirit of testing for discrete system. These ideas will be discussed in the next Section.

10.3 A Unified Approach to Controller Synthesis

Working with discrete and continuous systems, being exposed to control, verification, scheduling and other domains, one cannot but observe that many problems treated under different names within different disciplines, have more resemblance if we look at them through an appropriate abstraction that filters their domain-specific details. Among these problems and techniques we mention the algorithmic approach to discrete systems verification by forward or backward fixpoint computation, the derived reachability algorithms for continuous and hybrid systems, bounded model checking (using satisfiability solvers to verify correctness for a bounded horizon), computational techniques for optimal control such as dynamic programming and model-predictive control, simulation, search methods in AI and Markov decision processes. Much of our effort during the reporting period was concerned with building a general unifying game-theoretic scheme, for which various system design and validation problems are concrete instances. We have also invested a significant

effort in attacking several concrete instances of this scheme, most notably scheduling under uncertainty and hybrid optimal control. We have also investigated controller synthesis for distributed systems and some practical aspects related to

10.3.1 The Game-Theoretic Approach to Design

The paper [223] lays down a *unified* and *domain-independent* model for control in the presence of adversaries. The model uses the metaphor of a two-player game between the controller to be synthesized and the environment it is supposed to control. Control synthesis is viewed as finding an optimal (or satisfactory) strategy for the controller, where optimality is parameterized by two factors:

1. The cost function associated with *individual* trajectories

2. The way the costs of all possible adversary-induced trajectories under a given strategy are combined to compute the overall value of the strategy (worst case, average case).

Verification, and open-loop control ("ballistic" control, planning) are obtained as special cases where either one of the players is suppressed, i.e. is assumed to be deterministic with no choice. On this model we identify three generic approaches for solving controller synthesis problems:

- For *bounded timed horizon* the problem can be posed as a constrained optimization problem as is done in model-predictive control and bounded model-checking (in the latter, it is often the case that *feasibility* of the constraints is emphasized rather than cost optimality).

- The other class of methods, roughly characterized as *dynamic programming* (DP), is based on iterative computation of a value function (cost-to-go), which determines the optimal cost and action at every state of the system. For discrete systems such techniques are used in backward verification and synthesis for automata and for Markov decision processes. In continuous systems the value function is often computed as a solutions of some partial differential equations due to Hamilton, Jacobi, Bellman and Isaacs (HJBI).

- The third approach is to perform a forward search in the space of strategies and trajectories, an approach used by game-playing programs. In verification, when the search is not exhaustive over all adversarial behaviors, this activity can be viewed as testing. For control this approach is not mainstream but it seems to be popular in some related domains such as reinforcement learning and robot motion planning. The advantage of this approach compared to DP is that the value function needs to be computed only for the reachable part of the state space, a fact that may moderate the curse of dimensionality.

The separation between the domain-independent abstract scheme and the specific computational aspects of each domain, may facilitate the development of a "universal" controller synthesis tool based on this model. The specificity of each domain will be manifested by the type of computational engine used, for example a SAT solver for discrete systems and an LP solver when the dynamics is linear. The study of solvers for hybrid constrained optimization problems, the basic computational tool for all verification and synthesis problems, is becoming an important research issue by itself and our contribution to the domain is described in Section 10.6.1. We started exploring the applicability of the game-theoretic model and its variants to several domains as described in the sequel.

10.3.2 Scheduling under Uncertainty

One important instance of the game-theoretic scheme is the problem of scheduling under uncertainty. Here the controller is a scheduler which decides at certain points in time whether to allocate resources to enabled tasks, and the adversary is used to model various types of uncontrolled uncertainty such as in task arrivals, durations, outcomes or faults. Thanks to the state-space modeling framework for scheduling using timed automata, developed within the the thesis of Yasmina Abdeddaïm (1999-2002) [18], we can pose such problems very naturally as finding a strategy for a timed game automaton and solve them rather efficiently (recall that all the problems we deal with are NP-hard at best). All this work had been done within the European projects VHS (Verification of Hybrid Systems) and Ametist (Advanced Methods for Timed Systems).

In [41] the problem of scheduling under temporal uncertainty, where task durations are bounded within an interval, has been addressed. Since in this problem worst-case optimality is somewhat trivialized because the worst-case optimum can be obtained by a *static* scheduler that assumes a maximal duration for each task, a more flexible criterion called *d-future optimality* has been defined, and strategies optimal with respect to this criterion have been computed using dynamic programming on timed automata. On randomly-generated instances the average performance of schedules thus obtained was within 1.5% from the clairvoyant solution while static schedules deviated from it by more than 12.5%. This result can serve as a basis for a new approach for handling "soft" real-time systems.

Note that this problem is a special case of the more general controller synthesis problem for timed automata, for which an algorithm which uses zones but works on-the-fly in a forward manner, has been developed and implemented in [93].

In [151] we tackled the problem of optimal scheduling under discrete uncertainty. Tasks are related by precedence constraints as in standard scheduling problems as well as by *conditional dependencies*, where the outcome of one task may determine whether another task should be executed. This is a natural model for scheduling programs with *if-then-else* branches on parallel machines, and can be very useful for high-level synthesis. The whole situation is modeled as a timed game automaton where the adversary chooses the task outcomes, that is, values for certain Boolean variable that may appear in the activation condition of further tasks. The problem is then reduced to a shortest path problem in discrete weighted *game graphs* and solutions are found using a variant of depth-first min-max search. We are currently comparing various heuristics for more efficient exploration

of this game graph. For the moment we can easily find optimal solutions for problems with 20 tasks and few conditions, and sub-optimal (but good) solutions for problems with hundreds of tasks and up to 10 conditions. This work is done within the thesis of Abdelkarim Kerbaa, supervised by O. Maler and M. Bozga.

Note that in this work, due to the discreteness of the adversary, we can easily apply forward search algorithms, while for the previously-mentioned problem of duration uncertainty, the adversary is dense and the solution is found by the much more costly dynamic programming procedure. We are currently exploring a discretization approach for the duration uncertainty problem by assuming only finitely many possible durations for each task. Although using this approach the system may reach states for which a strategy has not been computed, we believe that using approximate strategies, a significant increase in performance can be achieved. Not covering all adversary choices while computing strategies is also an interesting option for more general continuous and hybrid systems as discussed in the next section.

The behavior of real-time systems with preemptive schedulers can be modelled by *stopwatch automata*. Nevertheless, the expressive power of stopwatch automata discouraged for a long time their use for verification purposes. Indeed , the reachability problem (even for a single stopwatch) has been proven to be undecidable. There are, however, some decidable sub-classes such as the so-called *integration graphs*, *suspension automata* and *timed automata with tasks*. In [40] we have studied systems with preemptive scheduling (with static, e.g. RMA, or dynamic, e.g. EDF, priorities), uncertainty (lower and upper bounded execution times), and precedence dependencies. The behavior of these systems cannot be straightforwardly translated into a decidable extension of timed automata. We showed that the reach-set can be represented by formulas involving difference constraints on clocks, and time-invariant equalities capturing the values of stopped clocks. This result implies decidability and leads to an efficient implementation. Moreover, it gives a precise symbolic characterization of the state space for the considered class of systems.

In [219] we developed an abstract model of the problem of control under limited computational resources, where the controller had to keep the performance measures of a multi-plant system bounded by mode switching, where each mode represents a different mix of attention and resources, and hence a different derivative for the performance vector. We show how controllability for safety can be reduced to some reachability problem on "linear" hybrid automata whose decidability status is still open. Other geometric investigations in scheduling were reported in [185].

10.3.3 Search-based Verification and Control of Continuous Systems

For a system $\dot{x} = f(x, u)$, when $u \in U$ is interpreted as an external disturbance, exhaustive verification requires computation with subsets of $\mathbb{R}^n$ with all the associated difficulties. An alternative approach would be to use discrete time (which is equivalent to restricting the input space to piecewise-constant signals) and discretize U into a finite set $\bar{U}$. By doing so we reduce the input space into the familiar and tree-like object $\bar{U}^*$ that we all love. If the discretization is sufficiently dense, correctness with respect to $\bar{U}^*$ is a good enough

approximation for exhaustive correctness. Our first work in this direction was [210] and can be viewed as an alternative to "classical" reachability where instead of storing all points reachable within a time interval in a single geometric object, neighborhoods of sample trajectories are stored and merged. This approach allows one to export many graph-searching ideas from discrete to continuous systems.

When u is interpreted as a control input, this approach becomes even more interesting because in synthesis we are sometimes happy to find *one* control sequences which is good enough and do not always care about being exhaustive or optimal. We started exploring this idea in the control context in [269] and demonstrated its applicability on a toy problem of guiding a missile inside a tunnel. Is is clear, however, that search alone, no matter how intelligent, will cause an explosion which, in this approach, is more related to the number of input sequence ($|\bar{U}|^k$ where k is the decision horizon) rather than to the dimensionality of the state space. Further progress requires some exploitation of knowledge of the system dynamics and combination with more traditional techniques for optimal control.

As a test case for search-based controller synthesis we looked at the ABB case study of the CC project. This is a small-scale model of a power distribution network where the goal is to restabilize the system after a failure at time zero. The controller has 3 different types of actions (tap ratio, compensating capacitance and load shedding) to influence the system, each with its cost and effectiveness. A naive application of this idea would involve searching over all sequences ranging over all the 60 combination of the control variables values, however the monotone effect of each of them on the dynamics, together with the ordering of their costs, allowed us to use a segregated search that tries the cheap controls first. The resulting algorithm, reported in [293], is a variant of model-predictive control where the optimization for the bounded horizon is performed as a search over a finite set. The algorithm was tested on various scenarios in which the impedance of the system jumps from 0.25 p.u. to a higher value. When this value was 0.60 or smaller we could quickly find stabilizing sequences. The quality of the solutions found by segregated search was as good as exhaustive search for the decision horizons for which the latter could be applied.

The current effort in this direction, conducted within the thesis of Alexandre Donzé (supervised by O. Maler and T. Dang), is to develop more general-purpose optimal control algorithms that are based on search, without sacrificing rigorous analytic properties. Recently a new variant of the temporal-difference algorithm has been developed [280] and performed well on a toy inverted pendulum problem. The next problems that we intend to tackle are those of finding optimal paths for mobile robots amidst obstacles, and stabilizing a double pendulum.

10.3.4 Synthesizing Low-Complexity Schedulers

In order to synthesize a scheduler, the basic idea consists in constructing the set of reachable states and, thus, identify the deadlocks. These are the states where the application threads are deadlocked, or the states where some thread has missed its deadline or period. Having obtained the deadlocked states, we do a backwards traversal of the whole state space starting from the deadlocked states, and eliminating transitions leading to them, until there are no more deadlocks. Even though the basic idea is simple, it is evident that

in practice it suffers from the state explosion problem. Therefore, it is imperative that we use techniques to minimize the size of the state space.

In [212] we have developed a method for systems composed of cooperating threads scheduled on a preemptive real-time operating system running on a mono-processor hardware architecture. Our method consists of synthesizing schedulers for successively more detailed models, adding thus complexity to a model only when we have already calculated how we can constrain the more abstract one. The scheduler synthesis is performed in five major steps:

- Compositional Synthesis: first, we decompose the system and synthesize constraints independently for each of the components.We then apply the synthesis algorithm again on the parallel composition of the already constrained models.

- Abstraction of Time: second, we consider the issue of time. We examine the untimed model of the system. Searching for deadlocks in the untimed model allows us to examine a much smaller search space. In order to make the problem more tractable, we then reduce the timed model modulo the branching bisimulation equivalence, which eliminates unobservable actions (in our case time-passing transitions) but only when doing so preserves the branching structure of processes.

- Execution Model: third, we analyze the behavior of the system for two different execution models, namely preemptive and non-preemptive. We first consider that the application threads cannot be preempted while they are computing. The non-preemptive execution model hypothesis reduces the state space, since it removes all the cases where the execution of a thread is suspended so as to handle an interrupt. Once we can indeed safely schedule the system under the hypothesis that threads are never preempted, then we can use the constraints obtained during this step to reduce even further the state space that we have to construct and analyze when we do allow threads to be preempted.

- Observability of Clocks: the constraints we produce during the synthesis use the state of the system to decide what are the safe choices at each point during the execution and, therefore, also make reference to the values of the local clocks of the threads. However, these clocks do not really exist in the application but are introduced as a way to model computations. As using timers may substantially increase the execution time of the scheduler, we investigate the possibility of synthesizing a clock-free one. Not observing clock values, actually defines an equivalence relation that further reduces the state-space.

- Policies: once we have synthesized a scheduler, which is indeed non-deterministic, we can compose it with policies to reduce non-determinism (or even to make the scheduler deterministic).

This methodology has been successfully used in [211] for generating schedulers for real-time Java applications.

In [213] we have shown how one can adapt data-mining techniques to decrease the size of a synthesized scheduler to allow optimizing its implementation.

10.4 Decentralized Observation and Control

This research action aims at studying problems of observation and control in a distributed setting: instead of a single observer or controller, there are many such agents which monitor or control a plant at the same time. Each of them has partial information about the plant, and this is what makes the problem difficult. The problem is worth studying, however, since it embodies many interesting applications, including the long-pursued goal of protocol synthesis [242].

The two basic properties of most centralized observation and control settings[2], namely, (1) that existence of an observer-controller implies existence of a finite-state observer-controller, and (2) that checking existence and synthesizing an observer-controller is undecidable, break down in many decentralized settings. The precise features of decentralization which result in loss of such properties as decidability are still only partially understood.

Our work has contributed in the understanding of the computability limits in decentralized observation and control. In [258, 84] we have shown that undecidability occurs even in very basic decentralized observation and control problem on regular star-languages. This work also revealed how a decentralized observation problem where decisions are made off-line at a central decision point (whereas observations are gathered on-line by the decentralized observers) can be reduced to a decentralized control problem with no direct communication among controllers: the reduction is possible because communication can be modeled indirectly using an appropriate plant. In [260, 83] we have studied the problems where communication among observers-controllers is built in the setting: we have shown that bounded-delay communication makes the problem decidable, although unbounded-delay communication does not help in this direction. Finally, in [266] we have established links of the above decentralized observation problem to the theories of traces and rational relations. These links have permitted to identify a number of special cases where the problem is decidable.

10.5 Specification Formalisms for Hybrid Behaviors

The added value of "formal methods" to software and hardware engineering is not restricted to verification procedures. A large part of its contribution to system design is in putting *properties* at the center stage of the validation process. Properties are syntactic objects that specify in a rigorous manner which traces of I/O behaviors the system may exhibit while interacting with its external environment. Properties are typically expressed in some "declarative" formalism such as predicate logic, temporal logic or regular expressions over the observable alphabet.

The validation of a system with respect to a given property is based on transforming the property into a *property monitor* (observer, tester), a mechanism that checks whether a given behavior (sequence of I/O events) satisfies the property. This monitor can be viewed either as an automaton accepting exactly the set of satisfying behaviors or as a

[2]Including partial-observation settings

procedure working recursively both on the length of the sequence and on the syntactic structure of the property.[3]

The theoretical aspects of such transformation are related to the theories of formal languages and logic that deal with the expressive power of various formalisms for specifying discrete behaviors. One of the remarkable of this classic theory is that regular (finite-state) languages admit a variety of logical, algebraic and operational description formalisms which are all equivalent. We want to extend such results for timed and hybrid systems, partly for the purpose of exporting the language-theoretic aspects of formal methods to the continuous domain, and partly due to interest in fundamental theoretical problems related to timed and hybrid formal languages. Much of this work is carried out within the European project PROSYD (Property-based System Design) and some of its aspects are related to the work on testing timed automata mentioned in Section 7.2.2.

10.5.1 Timed Regular Expressions

The result in [45] is one of the strongest theoretical results concerning timed formal languages. It starts by defining decent semantic domains for timed behaviors (the monoid of *time-event sequences* and the monoid of discrete-valued *signals*) and then goes on to define timed regular expressions as means for specifying sets of such timed behavior. This class of expressions extends untimed expression by adding a time restriction operator, and by using conjunction and renaming (both are provably necessary to match the power of Alur-Dill timed automata). The translation from these expressions to timed automata is simple and the hard part is the proof of the other direction from automata to expressions, which involves the solution of a new type of language equations. This Kleene theorem for timed automata is part of our effort to elevate the theory of timed languages and automata to the level of the classical untimed theory. In another recent work in this direction [221] we define a notion of recognizability for timed languages and show that it coincides with deterministic timed automata. Minimization and determinization of timed automata have been studied also in [264] and other algebraic aspects of the theory of timed languages and expressions were explored in [24, 55, 178, 110].

10.5.2 Temporal Logic for Continuous Signals

The PROSYD project is concerned with the *property specification Language* (PSL), accepted as a standard by the semi-conductor industry. This is essentially Pnueli's temporal logic augmented with syntactic sugar, regular expressions and additional features which are needed in order to move from a clean theory to industrial-strength tools. Our role in this project is to develop an extension of PSL to describe properties of analog and mixed signals (AMS). This work is related to a new promising application domain, verification of analog circuits, to be discussed in Section 10.8, which exemplifies the inter-cultural gap between computer science and more "physical" domains.

[3]It is worth mentioning that some researchers and tool vendors advocate the direct use of monitoring procedure without using an external specification formalism.

As a starting point [220] we took a bounded version of the real-time logic MITL, and augmented it with static predicates on the continuous domain that transform analog signals into Boolean ones. Using the obtained logic STL (Signal Temporal Logic) one can express "sequential" properties that specify the order of occurrence of certain events (threshold crossings of continuous signals) as well as the temporal distance between them. These properties are quite different from the "properties" currently and implicitly employed in analog design which are more of a macroscopic nature (frequency, signal-to-noise ratio, energy). So far we have demonstrated how control-oriented properties such as stabilization and tracking can be expressed in STL and are currently working on properties related to flash memories, as communicated to us by the STM partners in PROSYD.

The development of future versions of STL depends on an ongoing interaction with the potential users of the technology including partners in PROSYD and other collaborators including the group at CMU with which we have recently collaborated on defining and checking oscillation properties [183]. Among the extension to STL that we consider we mention frequency-domain properties as well as property that allow continuous values at different times to be related directly and not only through their Boolean abstractions, for example something in the spirit of $t' - t < a$ implies $\xi[t] - \xi[t'] < b$. Other "properties" to be considered are those that output numerical values that reflect some quantitative characteristics of the signal. Other properties such as following a reference signal, require further investigations in new metrics for signals. This work on STL and its monitoring is part of the thesis of Dejan Nickovic supervised by O. Maler.

10.5.3 Application: Property Monitors

Large systems with some hundreds of thousands of transistors will have to wait for quite some time until exhaustive hybrid verification techniques will be able to handle them. The semi-formal alternative is to use monitors in conjunction with numerical simulators so that the monitors observes simulation traces, detects property violations and liberates the user from the need to construct monitors by hand or to inspect long waveforms or large files manually. We have built an offline monitoring procedure for STL [220] which takes a sampled-time simulation trace, and marks the truth values of sub-formulae backwards until the truth value of the formula itself is determined for time zero. The monitor has been applied to simulation traces generated by a Simulink model of a CC project case study, the water-level controller by EDF.

The disadvantage of offline monitoring is that it can start only after the simulation terminates, while it is often the case that a property is already satisfied or violated by a short prefix of a long trace, a fact that an online monitor can detect. This can be very meaningful when simulation is costly as is the case with certain analog circuits: it may take few weeks to simulate one behavior of an RF circuit. Another advantage of online monitors is that they can be used to monitor *real* systems during execution and not only simulated systems. Online monitoring requires, however, deterministic (or determinizable) monitor, which is a problem because MITL can express properties that are not acceptable by a deterministic timed automaton. We mention several possible remedies to this problem:

1. Use the observation of Tripakis that on-the-fly subset construction with respect to a *given* trace is possible, an observation used for testing timed automata [214].

2. Use specification formalisms that admit deterministic timed automata, for example the *past* fragment of MITL [298].

3. Develop (as we did) a piecewise-backward procedure that cuts a signal ξ into finitely many pieces $\xi_1, \ldots \xi_n$ and monitors them, one after the other, in a backward fashion.

10.6 Fighting the Clock Explosion

Timed automata can be viewed either as a special class of hybrid automata with a very simple dynamics in each mode (all clocks variables have the same derivative) and a restricted reset map, or as an extension of automata where the dynamics takes into account not only the qualitative input history but also the time elapsed since certain events have occurred. This requires as many clock variable as there are events whose occurrence times need to be simultaneously memorized.

The basic verification questions for timed automata are decidable and are solved by symbolic reachability algorithms like the one sketched in Section 10.2, specialized to the dynamics of clocks. The decidability comes from the fact that there are finitely many "timed polyhedra" encountered during the reachability computation, but from a complexity point of view this does not help much. The analysis of a system consisting of n components each with m states and one clock may generate as much as $m^n k^n n!$ zones, where k is the largest constant appearing in transition guards. The representation size of each zone can be n^2 which does not make life easier.

Although a lot of effort has been invested during the last decade in finding more efficient ways to represent and manipulate the reachable states of timed automata, to the best of our knowledge, this problem has not been solved and very few systems with more than a dozen of clocks has ever been verified using timed automata, which is a pity given their extremely useful expressive power. By analogy, the current status of timed verification is like that of discrete verification before the BDD revolution. One of the major problems seems to be the lack of an effective symbolic representation for both the discrete and continuous parts. Since we are interested in large-scale application like circuit timing analysis and scheduling, we have invested quite a lot in various attempts to solve the problem. In the past we have tried BDD representation (binary-encoded discrete clocks) and a canonical representation of non-convex timed polyhedra, while more recently we have focused on the methods described below.

10.6.1 SAT Solvers for Difference Logic

Bounded model-checking (BMC) for discrete systems profited from the impressive progress in the performance of propositional SAT solvers. Systems that made reachability computation explode, can now be treated in a satisfactory manner using such solvers. In a series of works (in collaboration with P. Niebert and E. Asarin) we have tried to apply

these ideas to timed automata. We have first identified *difference logic*, propositional logic augmented with difference constraints of the form $x - y < d$, as the appropriate logic for expressing the transition relation of timed automata as well as other timing-related problems such as scheduling.

Our first experience with the art (or more appropriately, the black magic) of SAT solvers was through the MX-Solver developed within the thesis of Moez Mahfoudh [235], during which we have studied different approaches for enhancing SAT to treat enriched logics. Our approach was based on a dynamic interaction between the Boolean and numerical parts. Each time a difference constraint was implied by the current assignment (i.e. became a unit clauses) it was put in a large difference-bounds matrix (DBM) which was checked for consistency using algorithms for negative cycle detection. Note that the DBMs used for SAT are several orders of magnitude larger that those used in TA verification. The performance of this solver on problems coming from BMC for timed automata was still much inferior than that of standard TA verification tools. It was much superior, however, to several other solvers developed around the same time (the topic was in the air) on problems dominated by difference constraints such as job-shop scheduling.

The second round of work on the topic started with the masters thesis of Scott Cotton. The first version of his DLSAT solver [169] employed conflict analysis and learning as well as a variety of techniques that improved the performance results significantly. The second (and still unreported) version of DLSAT has already obtained some impressive results. To begin with, for purely-propositional formulae it beats a world-class solver such as zChaff on some instances. Secondly, for some notorious job-shop scheduling problems it found the optimum (!) while most solvers would explode much before that. Finally the results on bounded model-checking for TA are much more encouraging than the previous results (for example is is able to check a formula obtained from a 25-unfolding of a 64-gate circuit (2^{64} states and 65 clocks) but more work still needs to be done concerning more efficient translations of BMC to SAT. It might be the case that like other asynchronous systems, TA are not amenable for efficient BMC. Like in untimed systems, the use of the solver is not restricted to BMC and, it can serve as an alternative computation engine inside (unbounded) reachability algorithms.

10.6.2 Abstraction Techniques for Timed Systems

Another attack on the clock explosion problem is conducted within the project *Combining Formal Verification and Timing Analysis* sponsored by Intel. The goal is to use the expressive power of timed automata to verify circuits at the level of abstraction of Boolean functions plus bi-bounded delays. Although we have managed in the past to verify some non-trivial asynchronous circuits in this level of abstraction [150], the size of most circuits is orders of magnitude larger than what TA tools can handle. To tackle this problem we resort to divide-and-conquer methods that cut circuits into small enough pieces, analyze each piece separately to construct a small abstract model of its I/O behavior for further use.

In [246] we have applied this idea to combinational circuits whose inputs change only at time zero and hence their automata are acyclic and every run reaches a stable state

within a finite amount of time. We add an additional clock which is never reset to zero and hence it measures absolute time. We analyze the circuit using reachability analysis (which eliminates qualitative behaviors which are impossible due to timing constraints) and then, after having used the clocks we throw them away by projecting guards on the absolute clock, hide transitions in internal gates and minimize the automaton. This way we obtain an over-approximation of the circuit which is faithful to its qualitative behavior and to the absolute occurrence times of events but is more liberal with respect to their temporal distances. Using this compositional approach we could analyze circuits with almost 100 gates, a great step for timed automata, but admittedly, still a far cry from the size of circuits analyzed by traditional methods.

Targeting more sophisticated asynchronous circuits, we have extended this technique to deal with open reactive systems whose inputs may change *any* time. This is a necessary step toward "hierarchical" compositional reasoning. Here there is no absolute clock to project on because we need to memorize input events that happen at different times. Our solution is to use dynamic clocks generated at every input event. Since we deal with well-formed circuits without auto-generated oscillations, every input event is propagated within finite time to the output, its occurrence time can be forgotten and its clock can be reused, a fact that guarantees a finite number of clocks. Applying the abstraction technique, this time projecting on the input clocks, we obtain an abstract model that relates the occurrence times of input and output events. This work, performed within the thesis of Ramzi Ben Salah (supervised by O. Maler and M. Bozga), involves a lot of implementation effort and is currently in its final phase [302]. It could serve as a method for creating timed abstraction of components in general, not necessarily those made of hardware.

10.6.3 Getting Rid of Zones

As mentioned in Section 10.2.10, we believe that further progress in hybrid verification will require some backtracking from the orthodox approach to reachability analysis which is too "breadth-first" and less single-trajectory oriented. Our work on scheduling illustrates how this can be done. The modeling of acyclic optimal scheduling problems with timed automata is straightforward, as well as its solvability by standard reachability analysis. However trying to solve it using zone-based reachability we have quickly reached the limits. Upon closer inspection we realized that the dense temporal uncertainty is, in fact, a result of a choice of a scheduler which may start a task at any time since it is enabled. It turned out that the optimum is achievable by a special class of "non-lazy" strategies that either grant the resource upon arrival to a state where the task in question is enabled, or wait until something else happens. This observation allowed us to analyze a finite number of runs and store states as clock vectors rather than polyhedra. Using this technique we could solve scheduling problems much larger than possible by standard TA tools [86, 41]. Moreover, a similar observation let us solve preemptive scheduling problems [88] without the trouble associated with the different type of zones that correspond to stopwatch automata. Although the optimality of non-lazy solutions does not extend to all types of scheduling problems, we believe that restricting the exploration to a discrete

subset of the scheduler actions is a promising idea and its applicability to the actions of the adversary is worth being investigated.

10.7 From Control Loops to Real-Time Programs

Hybrid systems research, being situated is one of the intersections of computation and control, helps also in understanding topics related to the computer realization of control loops. Issues such as sampling, numerical integration, code generation, scheduling, robustness or power consumption are better understood when equipped with the conceptual tools of both disciplines. In this sense, the hybrid research at VERIMAG has been privileged to enjoy the presence of Paul Caspi as a never-ending source of practical and theoretical knowledge. His collaboration with Stavros Tripakis on transforming control loops written in Simulink/Stateflow into scheduled Lustre programs, described elsewhere in Chapter 8 is a prime example of this synergy. It is part of the theses of Adrian Curic and Christos Sofronis, with a contribution of Norman Scaife. On the pedagogical side, the introductory survey [51] attempts to classify and explain the various implementation strategies for control-oriented embedded systems.

10.8 Analog Verification

Digital circuits have been one of the driving forces behind discrete verification, due to their mass production, which justifies large investments in exhaustive validation prior to irreversible fabrication, as well as their complexity which demanded performant verification techniques. There are some reasons to believe that analog and mixed-signals circuits will play a similar role in the development and acceptance of validation techniques based on hybrid models. It is commonly-accepted that the analog parts of systems-on-a-chip risk becoming a bottleneck in the design flow due to the lack of system-level EDA tools in general and validation tools in particular. We are actively involved in preliminary exploration in this domain which includes the verification of some small circuits [173, 183], the generation of monitors for signal temporal logic [220], the organization of a workshop on analog verification [277] and interaction with academic collaborator and industrial partners of the PROSYD project. Hopefully these activities will bear fruits in the future.

10.9 Perspectives for Future Work

Inter-disciplinary research is a double-edged sword. On one hand it can serve as a fertile ground for new ideas, sometimes much more interesting that those encountered in "standard" disciplines which very often degenerate into a boring repetition. On the other hand, inter-disciplinarity can be used to hide mediocre and even bad work that could not be accepted within any established serious community. It could also be an opportunity to re-cycle old results and find imaginary nails for one's own hammer. We hope that the

work described in this chapter does not fall completely within the latter category. We believe to have exported some useful computer science ideas to hybrid systems research and learned few things in the process. A large part of the research described here represents a conscious choice concerning the niche in the theory-to-application spectrum that we want explore, which is closer to the theoretical side than other approaches to embedded systems. This choice is based on the belief that when developing new models for new phenomena it is important to focus on a few new aspects as possible rather than go directly and try to solve practical problems with all their dirty details, which often take more than 80% of the effort. We believe that the possibility to opt for long-term goals and not produce complete solutions to industrial problems (or pretend to produce such) is a privilege of the researcher, compared to the engineer, and when not abused, it can eventually contribute to technological progress, sooner or later.

We close this chapter with a final remark on *timed systems*. While hybrid systems have their natural niche in the intersection of control and computer science, and are easy to explain to the relative layman, this is unfortunately not the case for timed systems whose understanding requires some sensitivity to subtle issues concerning levels of abstraction. The application of timed automata are spread all over the place (scheduling at various scales, timing analysis of hardware and software, time management and planning) and their treatment is often domain-specific and community-specific. Timed automata may give all these phenomena a state-space based model as do automata for the discrete case and differential equations for the continuous case. However, much of the theoretical work on timed automata only pays a lip service to applications, and then goes on to prove theorems, some of which are interesting. We believe that this unhealthy situation will change when the scalability barrier will be broken, and we keep on directing our efforts toward this goal.

Chapter 11

Transaction-Level Modeling and Analysis of Systems-on-Chip

"Systems-on-Chip", or SoCs, are heterogeneous systems made with hardware and software components that have to be developed together for a specific goal. A SoC can be made of several processors (like ARM cores), several memories, some busses, etc. The whole system is implemented on a single chip. In the consumer-electronics domain for instance, very short development cycles imply that the embedded software be developed and tested before the architecture of the SoC is completely defined, and long before the chip is indeed built. This has led to a kind of revolution in the way chips are designed.

11.1 Projects and Contracts

This research is being carried out in strong cooperation with our industrial partners, in particular, in the context of the joint Verimag/STMicroelectronics research effort, spanning several specific projects such as SocTLM 15.3.6, ANACONDA 15.3.3 and NEVA 15.1.8.

In the context of the Minalogic Competitivity Pole, we defined with STMicrolectronics and others partners a projet named openTLM, which will be funded in 2006.

11.2 Transaction-Level Modeling

The Register Transfer Level (RTL) used to be the entry point of the design flow, but the simulation environments for such models do not scale up well. Developing and debugging embedded software for these low level models before getting the physical chip from the factory is no longer possible with a reasonable cost. A new abstraction level, the *Transaction Level Model (TLM)*, is emerging. It uses a component-based approach, in which hardware blocks are modules communicating with so-called *transactions*. This level of abstraction is quite far from the cycle-accurate RTL. The TLM models are used for early development of the embedded software, because the high level of abstraction allows a fast simulation (around 1000 times faster than pure RTL models, and still much faster than cosimulation).

New models corresponding to this abstraction level must thus be developed, in languages such as C or SystemC 2.0.

As TLM models appear first in the design flow, they become *de facto* reference models for systems-on-a-chip. This raises several problems: 1) What does it mean to validate properties at the TLM level? (including the problem that SystemC or similar languages do not have a formal semantics; by validation we mean either simulation or formal verification). 2) Since automatic synthesis from TLM to RTL does not exist (and will not exist soon), how can we *compare* a TLM reference design and a RTL design that is supposed to implement it, and is partly written by hand? 3) How can we express and validate non-functional properties of SoCs at the TLM level?

These questions are key points for the wide adoption of TLM models in the industry, and are being addressed by a joint project between VERIMAG and the SPG team of STMicroelectronics Central R&D. The project currently involves three PHD students, a master student, a Professor from VERIMAG, and an project leader from ST.

11.3 SystemC

SystemC is becoming a de-facto standard for the description of complex systems-on-a-chip, in particular in the consumer-electronics domain where the constraints are: reusability, heterogeneity, very short development cycles. It enables system-level descriptions of SoCs: the same language is used for the description of the architecture, the software parts, and the description of the hardware parts. SystemC comes with a simulation environment enabling fast simulation and early execution of the software parts on a model of the hardware architecture.

A number of other approaches have been proposed for the description of heterogeneous hardware/software systems with an emphasis on formal analysis. See, for instance, Metropolis [BLP+02]. In this type of approach, the definition of the description language is part of the game. The language can be defined formally, and tailored to allow easy connections to validation tools. On the contrary, SystemC has not been defined with formal analysis in mind. it is primarily a simulation and coordination language, aiming at accepting all kinds of hardware or software descriptions in a single simulation.

SystemC is deliberately based on open standards like C and C++, for two reasons: first, it guarantees a fast learning-curve for the engineers of the domain; second, it guarantees that the models of systems developed in SystemC can be exploited even if the tool that was used to build them is no longer available. SystemC is normalized by the Open SystemC Consortium Initiative, involving the major actors of the domain. An IEEE standardization is ongoing. SystemC is currently used by major silicon companies like Intel, STMicroelectronics, Philips, Texas Instruments, ...

Developing analysis tools for SystemC is therefore interesting. SoC case-studies written directly in SystemC can be obtained from industry, and used to validate an analysis approach. The use of any academic formal language as the core of an analysis tool would imply that any case study is first *translated* into this language, before analysis can be performed. Then the validity of the results obtained, with respect to the original system,

can be questioned.

Dealing with SystemC: Parsing

In his PHD thesis, M. Moy develops a tool called Pinapa. Pinapa is the compulsory tool for anybody who wants to work on realistic SoCs designs: it is able to extract both architecture and behavior information from a piece of SystemC code, with very few limitations. It can be used as a front-end for various analysis tools, ranging from "superlint" to model-checking. It is open source and available from `http://greensocs.sourceforge.net/pinapa/`. There exists no equivalent tool for SystemC up to now.

Pinapa is also advertised by GreenSocs (`http://www.greensocs.com`). Quoting their website: "*GreenSocs undertakes and co-ordinates engineering projects to develop SystemC infrastructure, with and on behalf of the user community. The aim is to extend the use and applicability of SystemC by promoting collaboratively developed standards.*".

Pinapa is described in [234].

Dealing with SystemC: Semantics

The next compulsory step is to define a formal semantics of SystemC. In [233], we describe the translation of SystemC user code, and the modeling of the SystemC scheduler, into HPIOMs (for "heterogeneous parallel input/output machines"). This translation is performed on the abstract tree built by Pinapa. Translating SystemC into HPIOMs is a way of giving a formal semantics to SystemC. The faithfulness of the translation relies on the executability of HPIOMs. The HPIOMs obtained may be tested against the official SystemC execution engine (we currently use the Lustre back-end to execute HPIOMs — It allows either user-friendly debugging or efficient compilation).

The resulting tool is called Lussy. It is an open tool, allowing other tools (SAT solvers, model-checkers) to be experimented on HPIOMs obtained from SystemC.

11.4 Formal Verification for SoCs

SystemC designs being *circuit* designs, we could think of using one of the verification tools (model-checkers, SAT-solvers, etc.) developed for hardware verification, for instance SMV [McM93]. However, these tools are tailored for the RTL, exhibiting a clear notion of logical time, while we need to deal with heterogeneous designs. Heterogeneity comes from several places: determinism and non-determinism, synchronous and asynchronous systems, hardware and software components. Moreover, these tools cannot take general SystemC as input.

As far as we know, all the work on verification techniques and tools for SystemC designs are limited to the subset of SystemC that allows to write RTL designs. It cannot be used for real TLM designs (See [DG] for instance). [SF02] treats the SpecC language (similar to SystemC). A date of execution is associated with each instruction of the program, which

can then be considered as a dependency graph, on which synchronization properties can be proved. The approach is very limited: since only one date can be associated with an instruction, this does not allow to consider general loops.

Now, since SystemC is mainly a C++ library, one could think that we have to face the same problems as those addressed by general-purpose software model-checking tools. This is not the case. Verifying SystemC designs is, on the one hand *simpler*, because we do not have to deal with general dynamic data structures and general algorithmics; on the other hand *harder*, because we have to take parallelism into account, and to know about the scheduler specification. General software model-checking techniques concentrate on dynamic data structures and general algorithms. They provide sophisticated tools like invariant extraction, loop unrolling, etc., but are not directly usable to exploit the particularities of the SystemC constructs provided as a C++ library. Using these tools for SystemC would need to include the non-deterministic scheduler specification in the tool. Moreover, they usually do not take parallelism into account. For instance, CBMC [CK03, CKL04] can apply bounded model-checking techniques on pure C models, but does not deal with parallelism, or with infinite loops. SLAM [BR00] uses clever abstractions and refinement techniques, but also focuses on sequential programs. VeriSoft [God97] can handle parallel processes written in any language. They are executed as black boxes, communicating via calls to operating system primitives. These calls are intercepted to build a model of their parallel behavior. We cannot exploit such a black-box approach, because we need to extract the transaction-level specific constructs of SystemC, and aim at treating addresses in a specific way (see below).

The closest related work is to be found in Java model-checking, since they take a scheduler specification into account. The first version of the Java Path Finder model-checker [Hav99] used an approach similar to ours, translating Java into the intermediate representation Promela, and using the model checker SPIN to prove the properties. Version 2 [HPV00a] checks the byte-code directly, using a dedicated JVM with backtracking capabilities, and lots of other model-checking techniques. However, the techniques dedicated to Java are not directly applicable, neither to SystemC and its scheduler, nor to the modeling of synchronous and asynchronous mechanisms.

Our approach is to connect HPIOM to verification tools like Lesar [RHR91] developed in the "Synchrone" team, SMV [McM93] or the Prover plugin of SCADE. This is also part of M. Moy's PHD.

The journal paper [MMM06] summarizes the verification chain.

11.5 Test and Runtime Verification for SoCs

Since formal verification does not always succeed (because of the intrinsic size of the problems, or because of undecidable properties), we also investigate testing and runtime-verification methods. This is the subject of C. Helmstetter's PhD.

For the moment, the testing process on SystemC designs is the following: first, the system-under-test is connected to simple components that make it *closed*. In general, it consists in adding master components that generate some traffic on the buses. The test

patterns are entirely described in the code of these manually written additional components. Second, communications with a memory are added. Then the system is run, and at the end the state of the memory is inspected. The main problem is the number of tests, and an adequate notion of coverage.
For testing SoCs designs written in SystemC, we concentrate on the following points:

- Isolating the part of the SoC design that is *under test*, and identifying the external *ports* and the protocols that define what kind of components could be connected there. This is a kind of interface definition.
- Generating automatically the most general components to be connected to the external *ports* of the system-under-test (SUT).
- Writing SystemC *observers*, i.e., components that may observe the behaviour of the SUT without modifying it. This is a way of writing test oracles in the same language.

As far as runtime-verification methods are concerned, one of the first problems we studied is related to the scheduler of the SystemC runtime environment. A SystemC program has *modules*, *signals* to model the low-level communication and synchronization of the system, and a notion of *simulation time*. SystemC modules and programs are made of several processes that run *"in parallel"*, according to a scheduling policy that is part of the SystemC library specifications. The SystemC norm does not impose all details of the implementations. In particular, the scheduler definition leaves room for several implementations.

The problem is the following: there exist SystemC designs for which two distinct schedulers would produce observably distinct behaviors. We call these designs "scheduler-dependent". A scheduler-dependent design has several drawbacks: first, it is in general impossible to augment it with an "observer" process, i.e. a process that observes the behavior of the original design without modifying it; second, some program modifications may have quite unpredictable effects, depending on the scheduler used.

We propose a runtime-verification method for determining whether a SystemC design is scheduler-dependent. It can be viewed as a particular case of a dynamic partial-order reduction technique (see [God05]). The idea is to execute the SystemC design D with a particular scheduler implementation S, both instrumented in such a way that special events called *synchronization actions* can be observed. The execution forks when the tool is able to detect that a different choice in S at a given point in time would have generated observably different behaviors. This principle relies on a notion of causal ordering between synchronization actions.

11.6 Expressing and Analyzing Non-Functional Properties of SoCs

One of the key points related to the definition of Transaction-Level-Modeling is the ability to obtain early results on the quality of the architecture. In particular, it should be possible to evaluate the timing properties of a design before the SoC is actually built.

This topic is being studied by J. Cornet in his PhD, started in October 2004. The first step is to define what it means to specify *non-functional timing properties* of SoCs, at

the TLM level. Then we'll have to find a way of expressing these properties in SystemC components (either by reusing some of the constructs, or by providing a dedicated library, or even by proposing annotations). The next step will then be to provide an implementation of SystemC that simulates systems according to these timing properties. Finally, the expression of timing properties at the TLM level will be a keypoint for the comparison between the TLM and RTL levels.

This work has to be conducted in strong cooperation with STMicroelectronics. Their TLM team develops a methodology for separating functional and non functional timing aspects in TLM designs. The idea is to have, first, a model as abstract as possible with respect to the timing aspects. Then, new models can be developed, adding timing aspects. For instance, a complex behaviour can be specified to take "around" n units of time. For the moment the unit of time is related to the (unique) clock of the final synchronous design. The timed model is expressed using the `wait` primitive of SystemC, which enables it to be simulated by the standard SystemC engine. This allows to observe the influence of these timing elements on the global behavior, and to measure performances.

However, since the knowledge we can get on the average execution time of the system components is never exact, adding this timing information in a design should not imply that strong synchronizations appear, that were not visible in the untimed version (in other words, adding this type of timing information should not transform the asynchronous initial design into a perfectly synchronous one, where counting time is a valid way of synchronizing components). To overcome this problem, ST proposed to execute a `wait (T)` as a `wait (T')` where `T'` is a random number in a small interval around `T`.

The first result we have is a proposal for a formalization of this approach in the context of so-called *fuzzy-time* systems (a variant of fuzzy sets). Consider two asynchronous components C1, C2 on which we have some fuzzy information; for instance, it can be formalized by delays attached to states. The parallel execution of C1 and C2 is somewhere between the full set F of asynchronous executions (ignoring the delays), and a restricted set R of executions in which the delays attached to states are considered as exact and play a synchronization role. F is too large, because it ignores the timing completely, and will consider executions in which very long tasks terminate before very short ones. Conversely, R is too small, because it rules out executions as if the timing information were strict enough to decide that two tasks that have the same length do terminate at the same time. The set of executions we want to consider is that of *timing-realistic executions*: a very long task will terminate after a very short one, but for two tasks that have comparable execution times, we will consider both termination orders.

We proposed a fuzzy parallel composition, and showed that the set of executions is the same as the set of executions that can appear with the random trick in SystemC. But we also showed that the proposed trick is not enough to guarantee that all timing-realistic sequences have a sufficient probability to be actually observed during simulation.

11.7 Perspectives for Future Work

Component-Based Verification for SystemC

In his PhD, M. Moy [33] provides the building blocks for verification tools taking SystemC code as input. We are currently applying the Lussy toolchain to the TLM model (provided by STMicroelectronics) of the EASY (ARM Limited, 1999) platform. The EASY platform is a small-size system-on-a-chip. In Lussy, all the steps of the translation are performed correctly. Up to now, the generated Smv and Lustre codes are too large to be provable as a whole, which makes EASY an interesting case-study for component-based verification methods.

Exploiting the intrinsic component-structure of systems-on-a-chip described at the TLM level can be done mainly in two ways: either we perform agressive automatic abstractions on the models of the components, before we try to prove something on the system as a whole; or we ask the designers to provide the appropriate abstractions for their components. The two methods can be used together in a complete verification environment.

Automatic abstractions can be based on the structure of the source code, since Pinapa provides all the information. It may need to extend HPIOM to support annotations coming from the structure of the program, and usable by dedicated verification techniques. For example, HPIOM automata for the individual processes in SystemC use transitions as "microsteps" to represent the computation between two synchronization points, and also to represent these synchronizations steps. However, since the scheduler is non-preemptive, the parts of the control flow between explicit synchronization points are atomic, with respect to the parallel composition of the processes. It would be easy to use Pinapa in order to identify the parts of the automaton associated to a process that constitute atomic reactions of this process. Of course, since processes are decribed with a general programming language, an atomic reaction can be any algorithm, with loops and conditional structures. This means that a sub-automaton of any shape has to be abstracted by *one* transition. The difficulty is to label this transition with a kind of *summary* of the computations that happen inside the component, and that would be sufficient for the global view and analysis of the system. We will consider abstract interpretation techniques for determining these "summaries".

If we consider the other approach, in which we ask the designer to provide local specifications, a promising approach is the systematic use of *contracts* for some of the components. We already mentioned the case of processor components: a processor is an interpreter of the binary code, and it has to deal with complex data which is the C code to be executed! C should be abstracted (the values exchanged being replaced by unknown values encoded by inputs) but we need some assumptions about the behavior of the processor concerning the way it synchronizes with other components. This can be described by a contract. We are starting a collaboration with Edmund Clarke at Carnegie Mellon University, to apply their component-based verification methods an tools to SystemC and HPIOM.

Comparing the TLM and RTL Levels of Abstraction

One of the key problems induced by TLM modeling, given that there are no automatic synthesis techniques from TLM to RTL, is the comparison between the TLM and the RTL abstraction levels. The difficulty is that these levels of abstraction do not have the same time granularity, nor the same data granularity. Moreover, a so-called "transaction" between components at the TLM level corresponds to several steps of the base clock at the RTL level, but two successive transactions may have interleaved corresponding steps at the RTL level.

We need to study a formal relationship between levels. Then we will have to develop tools for the automatic validation of TLM-RTL conformance. We already have developed a tool that takes SystemC TLM designs and produces a format that can be exploited by validation tools. This tool formalizes the semantics of SystemC. For the RTL level, most developers use Verilog or VHDL, for which there exist analysis tools. The synchronous semantics of the register-transfer-level is quite clear.

Formal verification of a comparison relationship seems out of reach, but testing techniques, coupled with automatic test scenario generation, are promising.

Guidelines for TLM Design in SystemC

Based on the experiments we made on the formal verification of SystemC designs, we will proposed *guidelines* for TLM modeling in SystemC. The idea is that, among the various ways of describing the same system, some are more tractable by automatic validation tools than others. For instance, as soon as the addresses that are used for expressing how components are connected to each other are *dynamic*, it becomes extremely hard for an automatic tool to know about the structure of the design. A guideline can encourage designs in which addresses are clearly static when they can be (given by explicit constants, for instance).

The tool Pinapa can be used to develop a "superlint" tool, in which the guidelines can be enforced.

The TLM Lab

Figure 11.1 gives an overview of the set of tools we are working on with STMicroelectronics in the context of the joint laboratory.

The main blocks are:

- The connection to verification tools via Lussy (the SystemC front-end plus the formalization of the semantics in terms of communicating interpreted automata).
- The runtime-verification platform, including the testing tools, that exploit both the executions of SystemC platforms by the SystemC engine, and the information provided by the front-end Pinapa.
- The timing analysis module, that takes annotated TLM models and performs several kinds of timing analysis.

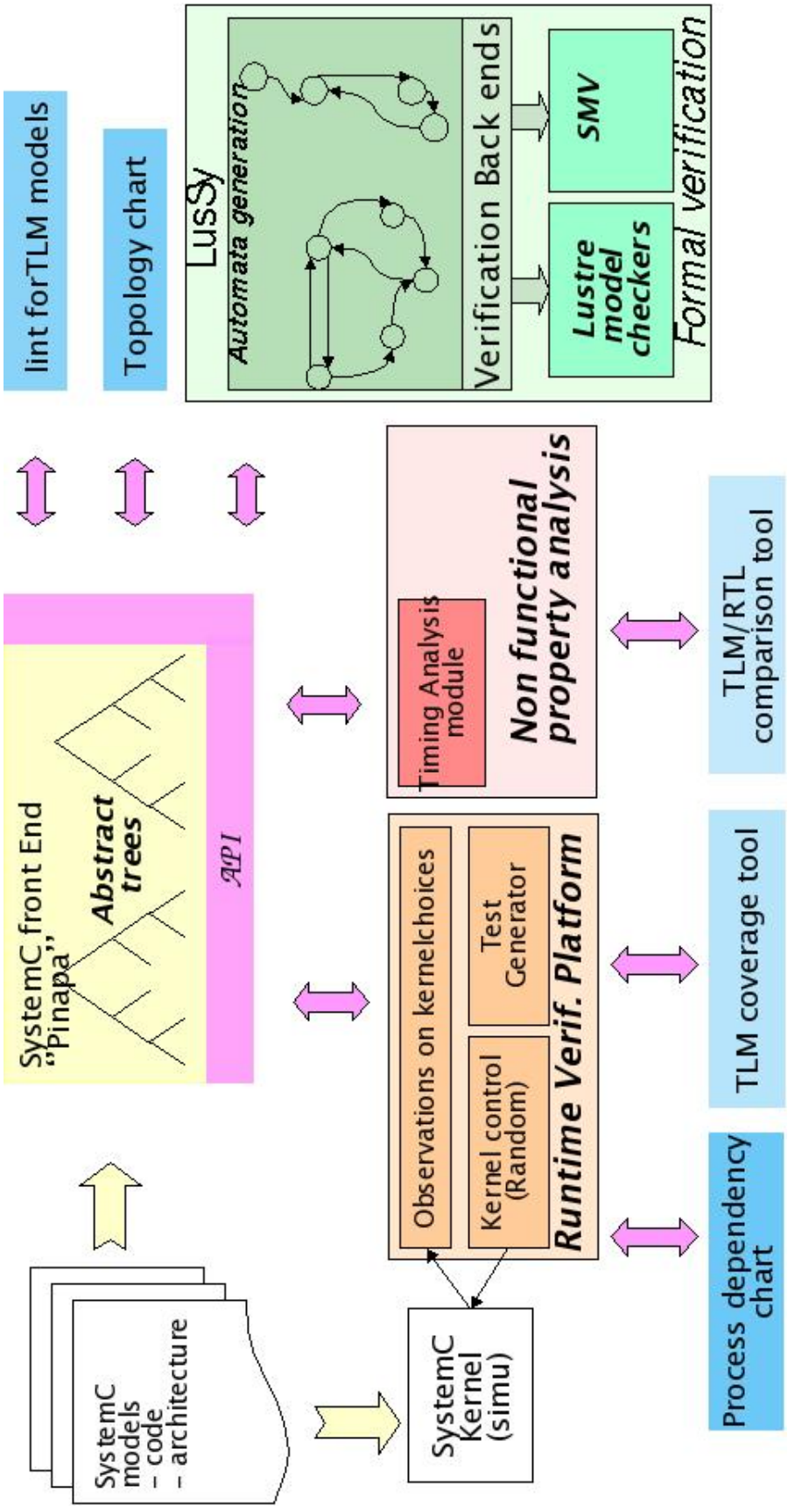


Figure 11.1: The TLM lab

Automated Generation of SystemC/TLM Code

In the context of the ANACONDA project 15.3.3 we are developing a technique to automatically generate SystemC/TLM code from the formal language of the Jahuel tool. The main motivation for this is the early prototyping, verification and testing of embedded applications on simulated hardware platforms. The automated generation of both executable and simulation code from the same formal model ensures simulation results are trustworthy.

This activity puts primary emphasis on multi-threaded processors and applications, and multiprocessor boards. An important issue here is simulation performance, which is likely to be compromised for such systems. We are investigating how specificities of communication and synchronization primitives and patterns can improve simulation time.

Part III

Teaching and Training

Chapter 12

Teaching and Administrative Tasks

VERIMAG's staff includes full time researchers as well as professors and associated professors (Maître de Conference). Permanent stuff of the latter two categories have teaching duties corresponding to 192 class hours. They participate in the activities of the two scientific teaching institutes in Grenoble: the Joseph Fourier University (UJF) and the National Polytechnical Institute of Grenoble (INPG). The concerned departments at UJF are: the department of Computer Science and Applied Mathematics (UFRIMA) and Polytech'Grenoble; while at INPG the Computer Science and Applied Mathematics (ENSIMAG) and the Telecom departments are concerned.

The implication of VERIMAG's stuff in the organization, the planning and the definition of the teaching curricula is relatively high. The list of co-ordination tasks filled by VERIMAG members includes the following:

- Licence Sciences et Technologie, niveaux L1 et L2, enseignements d'informatique : Fabienne Lagnier, Pierre-Claude Scholl (co-responsibles)
- Master Compétence Complémentaire en Informatique, M2 : Catherine Parent
- Master Mathématiques,Informatique majeure Informatique : Yassine Lakhnech
- Master Mathématiques,Informatique : Yassine Lakhnech (co-responsible)
- Spécialité "Systèmes et Logiciels" du master Mathématiques,Informatique : Nicolas Halbwachs.
- Spécialité "Informatique" de l'Ecole doctorale "Mathématiques, Siences et Technologies de l'Information, Informatique" commune à l'UJF et l'INPG: Nicolas Halbwachs.
- École d'informatique : Jean-Claude Fernandez (jusqu'à 2004).
- ENSIMAG, deuxième année : Florence Maraninchi, co-responsible
- ENSIMAG, stages d'été des première et deuxième années : Xavier Nicollin
- Département Télécommunications de l'INPG, tronc commun des première et deuxième années : Xavier Nicollin
- Département Télécommunications de l'INPG, Option "Applications Communicantes Embarquées" : Florence Maraninchi, co-responsible

Besides the above listed tasks, VERIMAG's members are actively participating in hiring and administrative committees :

- UJF Hiring Committee for Computer Science : Jean-Claude Fernandez, Fabienne Lagnier, Yassine Lakhnech, Pascal Raymond, Pierre-Claude Scholl

- INPG Hiring Committee for Computer Science and Applied Mathematics : Karine Altisen, Florence Maraninchi, Xavier Nicollin
- INPG Committee for HDR (Habilitations à Diriger les Recherches) : Florence Maraninchi
- Conseil du Département Scientifique Universitaire de l'UJF : Catherine Parent, Pierre-Claude Scholl
- Conseil de l'UFR IMA : Yassine Lakhnech, Laurent Mounier, Pierre-Claude Scholl
- Conseil d'Administration de l'INPG : Xavier Nicollin
- Conseil de l'ENSIMAG : Florence Maraninchi
- Conseil d'orientation du département Télécommunications de l'INPG : Florence Maraninchi
- Commission consultative d'enseignement du département Télécommunications de l'INPG : Xavier Nicollin
- Conseil Interuniversitaire de Coopération Documentaire (INPG, UJF) : Xavier Nicollin
- Equipe de direction de l'ENSIMAG : Florence Maraninchi
- bureau de la direction de l'UFR IMA : Fabienne Lagnier
- Gestion des services d'enseignement des filières informatiques de l'UFR IMA : Pierre-Claude Scholl, responsable

International Activities

ARTIST Education Group

In the context of the FP5 European Project ARTIST, Verimag has been involved in the activities related to the definition of curricula for Education and Training in advanced real-time systems. P. Caspi, Y. Lakhnech and F. Maraninchi worked on the first year deliverable with a lot of other European teaching institutions. The deliverable may be obtained at the following address:
`http://www.artist-embedded.org/Education/Education.pdf`

A short version (in french, see [52]) has been published in 2003 in the journal of SPECIF (a French association of teachers and researchers in computer science) and also as an interview in "Le monde Informatique" (September 2004).

A rewritten version is submitted to the Special Issue on Education of the ACM Transactions in Embedded Computing Systems.

An EMSOFT'05 workshop will be devoted to education in the field of Embedded Software and Systems.

Preparation of an International Master Programme in Embedded Systems

UJF and INPG are willing to propose together an international master programme on embedded software and systems. Y. Lakhnech (for UJF) and F. Maraninchi (for INPG)

are working on the definition of this programme, both from a scientific point of view, and from an administrative point of view.

The idea is to build on the experience we got in the ARTIST Education Group, in order to propose a programme in which we could welcome students from other European Universities, and also teachers from these universities.

INPG - Hanoi University of Technology

INPG and the Hanoi University of Technology (HUT) in Vietnam have been collaborating for 20 years in the context of exchange programs for teachers and students. Recently, INPG has taken part in the PFIEV (literaly: "Programme for Excellence Engineers in Vietnam"), helping HUT to create engineer programmes with a strong link to industry. The programme is now validated by the French committee that controls all engineer diplomas in France (the CTI). Even more recently, INPG started a cooperation programme in the context of the HUT International Training Programme (ITP). The idea is to build 5 years curricula in computer science and communication technologies, and to allow young Vietnamese students to enter the 3-year engineer programmes at INPG. F. Maraninchi is in charge of the ITP programme on the INPG side.
See `http://www.hut.edu.vn/itp/en/en_inpg.htm`.

Part IV

Research Administration and Publications

Chapter 13

Research Administration

13.1 Prizes and Awards

- Joseph Sifakis received the "Médaille d'argent" of the CNRS in 2002.

- Paul Caspi and Nicolas Halbwachs received the Prize "Michel Monpetit" from the French Academy of Science in 2004.

13.2 Organization of Conferences and Workshops

- MARTES 2005: Workshop on Modeling and Analysis of Real Time Embedded Systems, October 2005, in association with MODELS 2005 (see `http://www.martes.com`)

- WESE 2005: Workshop on Embedded Systems Education, September 2005, New Jersey, US, a satellite event of EMSOFT

- COSMICAH 2005: First International Workshop on the Verification of COncurrent Systems with Dynamic Allocated Heaps, July 2005, Lisbon, Portugal, a satellite event of ICALP 2005

- Workshop on the Link between Formal and Computational Models for Cryptographic Protocols, June 2005 (see `http://www.loria.fr/~cortier/workshop.html`).

- FAC 2005: Workshop on Formal Verification of Analog Circuits, April 2005, Edinburgh, U.K., in association with the European Joint Conferences on Theory and Practice of Software, ETAPS 2005

- SLAP 2005: Synchronous Languages, Applications, and Programming, April 2005, Edinburgh, U.K., in association with the European Joint Conferences on Theory and Practice of Software, ETAPS 2005

- Artist Workshop: Embedded Systems Design: An Emerging Unified Discipline, March 2005, Munchen, Germany, in association with the Conference on Design Automation and Test in Europe, DATE 2005

- Workshop and Review of the IST project Control and Computation March 2005, Zurich (CH)

- Workshop and Review of the IST project OMEGA, February 2005, Grenoble

- FMCO 2004: Third International Symposium on Formal Methods for Components and Objects, November 2004, Leiden, the Netherlands, (see `http://fmco.liacs.nl/fmco04.html`)

- SVERTS 2004, Workshop on Specification and Validation of UML models for Real Time and Embedded Systems, October 2004, Lisbon, Portugal, in association with the Conference on UML, UML 2004 (see `http://www-verimag.imag.fr/EVENTS/2004/SVERTS`)

- FORMATS and FTRTFT 2004: Joint Conference on Formal Modeling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant Systems, September 2004, Grenoble, France

- SLAP 2004: Synchronous Languages, Applications, and Programming, March 2004, Barcelona, Spain, in association with European Joint Conferences on Theory and Practice of Software, ETAPS 2004

- FMCO 2003: Second International Symposium on Formal Methods for Components and Objects, Leiden, the Netherlands, November 2003 (see `http://fmco.liacs.nl/fmco03.html`)

- SVERTS 2003, Workshop on Specification and Validation of UML models for Real Time and Embedded Systems, October 2003, San Francisco, in association with UML 2003 (see `http://www-verimag.imag.fr/EVENTS/2003/SVERTS/`)

- SIVOES-MDA 2003: Model Driven Architecture in the Specification, Implementation and Validation of Object-oriented Embedded Systems, October 2003, San Francisco, US, in association with UML 2003 (see `http://www-verimag.imag.fr/EVENTS/2003/SIVOES/`)

- QoSCBSE 2003: Workshop on QoS in CBSE, in association with the Conference on Reliable Software Technologies Ada-Europe, June 2003, Toulouse, France

- Spring School "Jeunes Chercheurs en Programmation", May 2003

- SLAP 2003: Synchronous Languages, Applications, and Programming, in conjunction with Euromicro Real-Time Conference, July 2003, Porto, Portugal

- HSCC 2003: Hybrid Systems: Computation and Control Prague, Czech Republic, April 2003, in conjunction with ETAPS 2003
- ISVQP 2003: Workshop on Infinite Systems and Verification of Quantitative Properties, March 2003, Grenoble
- FMCO 2002: First International Symposium on Formal Methods for Components and Objects, November 2003, Leiden, the Netherlands (see `http://fmco.liacs.nl/fmco02.html`)
- EMSOFT 2002: Organization of the Second Workshop on Embedded Software, October 2002, 2002, Grenoble, France (see `http://www-emsoft02.imag.fr/`)
- SIVOES-MONA 2002: Component-Based Software Engineering and Modeling Non-Functional Aspects Workshop, October 2002, Dresden, Germany, in association with UML 2002 (see `http://www-verimag.imag.fr/EVENTS/2002/SIVOES-MONA/`)
- ETAPS 2002: Organization of the *European Joint Conferences on Theory and Practice of Software*, April 2002, Grenoble, France (see `http://www-etaps.imag.fr/`). In the context of ETAPS 2002, we have organized the following workshops:
  - SLAP 2002: Workshop on Synchronous Languages, Applications, and Programming
  - TPTS 2002: Workshop on Theory and Practice of Timed Systems
- Organization of the *Journées PSLC* Grenoble, November 2001, in the context of the local *Plateforme Systèmes et Logiciels Critiques* of the *Centre des Technologies du Logiciel* (see `http://www.systemes-critiques.org/journees2001.php`)
- Co-organization of Workshops on Synchronous Programming, 1998-2001
- Workshop on Formal Methods for the Design of Safety Critical Systems: Synchronous Approaches, February 2001, Bangalore, India
- Workshop on Real-Time Tools, RT-TOOLS 2001, in association with CONCUR 2001, August 2001, Denmark, (see `http://www.docs.uu.se/~paupet/rttools-2001/`)

13.3 Evaluation of Research

13.3.1 Steering Committees and Programme Committee Chairs

- Steering Committee of the Conference on Computer Aided Verification, CAV, (Joseph Sifakis)
- Steering Committee of the Workshop on Embedded Software, EMSOFT, (Joseph Sifakis)

- Steering Committee of the European Joint Conferences on Theory and Practice of Software, ETAPS, (Susanne Graf 1999-2003, Nicolas Halbwachs 2004-2005)
- Steering Committee of the Conference on Hybrid Systems: Computation and Control, HSCC (Oded Maler, 1998 and 2002)
- Steering Committee of SPIN, the Workshop on Model-checking of Software (Susanne Graf 2003-2005)
- PC Co-chair of the Confererence Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2005 (Nicolas Halbwachs)
- PC Chairs of SPIN 2004 (Susanne Graf and Laurent Mounier)
- PC Chairs of the Conference FORMATS-FTRTFT 2004 (Yassine Lakhnech and Sergio Yovine)

13.3.2 Participation in Programme Committees

- Conference on Computer Aided Design, CAV 2001 to 2005
- Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2001 to 2005
- Workshop on Embedded Software, EMSOFT 2001 to 2005
- Workshop on Run-Time Verification, RV 2001 to 2005
- Conference Design Automation and Test in Europe, DATE 2003 to 2005
- Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2003 to 2005
- Conference on Software Engineering and Formal Methods, SEFM 2004 and 2005
- SDL Forum 2001, 2003 and 2005
- SPIN, the Workshop on Model-checking, 2002 to 2005
- Colloquium on Theoretical Aspects of Computing, ICTAC 2004 and 2005
- Symposium on Automated Technology for Verification and Analysis, ATVA 2005
- Workshop on Verification, Model Checking and Abstract Interpretation, VMCAI 2004 and 2005
- Workshop on Foundations of Interface Technologies, FIT 2005
- Journée sur l'Ingénierie Dirigée par les Modèles, IDM 2005
- Euromicro Conference on Real-Time Systems, ECRTS 2004 and 2005.
- Conference on Application of Concurrency to System Design, ACSD 2001, 2003 and 2005
- IEEE Symposium on Object-oriented Real-time distributed Computing, ISORC 2004 and 2005.

- Conference on Formal Techniques in Real-Time and Fault Tolerant Systems 2002, 2004
- Conference on Algebraic Methodology And Software Technology AMAST 2002 and 2004
- Workshop on Model-Based Testing, MBT 2004
- Workshop on SDL and MSC, SAM, 2002 and 2004
- Formal Methods in Computer-Aided Design, FMCAD 2002 and 2004
- Workshop on Principles of Software Engineering, PRISE 2004.
- IFIP WCC 2004 Workshop on Architecture Description Languages ADL 2004.
- Workshop on State-oriented vs Event-oriented thinking in Requirements Analysis, Formal Specification and Software Engineering, ST.EVE 2003
- Workshop on Designing Correct Circuits, DCC 2002 and 2003
- Workshop on Software Model Checking, SMC 2001 and 2003
- HSCC 1998 a 2002.
- Conference on Design, Automation and Test in Europe, DATE 2001
- Conference on Logic in Computer Science, LICS 2002
- Tools in Formals Methods, FM-Tools
- Workshop on Real-Time Systems, RTS 2001 and 2002
- European Symposium on Programming, ESOP 2001 and 2002
- Workshop on Semantic Foundations of Engineering Design Languages, SFEDL 2002
- Workshop on Real-Time Tools, RT-TOOLS 2001 and 2002
- Conference on Formal Aspects of Software Engineering, FASE, 2001
- Conference on Correct Hardware Design and Verification Methods, CHARME 2001
- Workshop on ASM, ASM 2001
- Workshop on Foundation of Software Science and Theoretical Computer Science, FSTTCS 2001

13.4 Editorial Boards

- Formal Methods in System Design (Joseph Sifakis)
- Software Tools for Technology Transfer (Joseph Sifakis)
- European Journal of Control (Oded Maler)
- European Association of Software Systems and Technologies, EASST, (Susanne Graf)

13.5 Invited Conferences, Tutorials and Summer Schools

- ARTIST2 Summer School on Component & Modeling, Testing & Verification, and Static Analysis of Embedded Systems, Sept 29 - Oct 2, 2005
  - *Fault-Diagnosis for Real-time Systems*
  - *Verification of UML models*
  - *Components & Modeling with Priorities in IF*

- Invited conference *Verification of Infinite State Systems with Applications to Security*, NATO Advanced Research Workshop, VISSAS 2005.

- Half day tutorial on the *IF language and toolbox* at the SDL Forum 2005 and at SPIN 2004

- Half day tutorial on *Abstract Interpretation* at the school MOVEP 2004, Bruxelles, December 2004. s

- 2 day course at the International Doctoral School "Chambéry - Torino" in Theoretical Computer Science and in Semantic Web, Aussois, France, June 2004

- Half day course on *The IF toolbox*, International School on Formal Methods for the Design of Real time Systems, Bertinoro, Italy, September 2004

- Invited conference *Composition of Real-time Components*, at International School on Model Driven architectures for Embedded Systems, Brest, France, September 2004.

- Keynote talk *Modeling Real-time Systems*, at RTSS 2004, Lisbon, Portugal, December 2004

- Invited conference *Embedded Systems, Challenges and Work directions*, at the 8th International Conference on Principles of Distributed Systems, Grenoble, France, December 2004

- One-week tutorial on *Static analysis: application to dynamic memory management in embedded systems.* Winter school on Computer Science, Universidad de Buenos Aires, Argentina, July 2004

- One day lectures on *Synchronous programming* at the Summer Schools "Jeunes Chercheurs en Programmation", Cargèse 2001, Rennes 2002 and Aussois 2003

- 2 days lecture at the European Summer School on Embedded Systems (ESSES 2003) Västerås, Sweden.

- Invited conference *Component-based construction of deadlock-free systems*, at FSTTCS 2003, Mumbai, December 2003.

- 1 day course at the Summer School TYPES 2002 on Theory and Practice of Formal Proofs, Giens, September 2002
- Invited conference *Scheduler modeling based on the controller synthesis paradigm*, at FTRTFT 2002, Oldenburg, Germany, September 2002
- Invited conference *Composition for Component-Based Modeling*, at FMCO 2002, Leiden, the Netherlands, November 2002
- Invited conference *Decentralized Fault Diagnosis for Discrete Event Systems*, at Conference on Decision and Control
- Tutorial at the Workshop on Networked Autonomous and Semi-autonomous Vehicles, CDC 2001, Orlando, USA, December 2001.
- Invited conference *Modeling real-time systems – challenges and work directions*, at EMSOFT 2001, Tahoe City, 2001
- Invited presentation *Modeling, Timing Analysis and Code Generation of PATH's Automated Vehicle Control Application Software*, at CDC 2001 Workshop on Networked Autonomous and Semi-autonomous Vehicles

Chapter 14

Publications

14.1 Books and Edited Work

[1] J. S. Alberto L. Sangiovanni-Vincentelli (ed.), *Embedded Software, Second International Conference, EMSOFT*, Grenoble, France, Springer, 2002.

[2] F. de Boer, M. Bonsangue, S. Graf, and W.-P. de Roever (ed.), *1st Symposium on Formal Methods for Components and Objects, revised lectures*, *LNCS Tutorials*, *2852*, 2003.

[3] F. de Boer, M. Bonsangue, S. Graf, and W.-P. de Roever (ed.), *2nd Symposium on Formal Methods for Components and Objects, revised lectures*, *LNCS Tutorials*, *3188*, 2004.

[4] F. de Boer, M. Bonsangue, S. Graf, and W.-P. de Roever (ed.), *3rd Symposium on Formal Methods for Components and Objects, revised lectures*, *LNCS Tutorials*, *3657*, 2005.

[5] W.-P. de Roever, F. de Boer, U. Hanneman, Y. Lakhnech, M. Poel, J. Zwiers, *Concurrency Verification: Introduction to Compositonal and Noncompositional Methods*, Cambridge University Press, 2001.

[6] A. Girault, F. Maraninchi, and M. Pouzet (ed.), *SLAP, Synchronous Languages, Applications, and Programming*, *ENTCS, Electronic Notes in Theoretical Computer Science*, 2004.

[7] A. Girault, F. Maraninchi, and E. Rutten (ed.), *SLAP, Synchronous Languages, Applications, and Programming*, *ENTCS, Electronic Notes in Theoretical Computer Science*, *65-5*, 2002, 785–938p.

[8] S. Graf and L. Mounier (ed.), *11th International SPIN Workshop on Model Checking of Software, 2004*, *Lecture Notes in Computer Science*, *2989*, 2004.

[9] S. Graf (ed.), *Tools and Algorithms for the Construction and Analysis of Systems: An STTT special section*, *STTT, Software Tools for Technology Transfer*, *vol 4, n 2*, 2003.

[10] S. Gérard, S. Graf, O. Haugen, I. Ober, and B. Selic (ed.), *1st workshop on Modelling and Analysis of Real-Time Embedded Systems (MARTES 2005)*, *Online proceedings available at* `http://www.martes.org/`, 2004.

[11] N. Halbwachs and L. Zuck (ed.), *Tools and Algorithms for the construction and Analysis of systems, TACAS'05*, *LNCS*, *3440*, 2005.

[12] Y. Lakhnech and S. Yovine (ed.), *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings*, *Lecture Notes in Computer Science*, *3253*, Springer, 2004.

[13] O. Maler and A. Pnueli (ed.), *Hybrid Systems: Computation and control, HSCC'03*, *LNCS*, *2326*, 2003.

[14] O. Maler (ed.), *Verification of Hybrid Systems*, Hermes, 2001. A special issue of the European Journal of Control.

[15] F. Maraninchi, A. Girault, and E. Rutten (ed.), *SLAP, Synchronous Languages, Applications, and Programming*, *ENTCS, Electronic Notes in Theoretical Computer Science*, *88*, 2003, 1–121p.

[16] F. Maraninchi, M. Pouzet, and V. Roy (ed.), *SLAP, Synchronous Languages, Applications, and Programming*, *ENTCS, Electronic Notes in Theoretical Computer Science*, 2005.

[17] J. Sifakis and B. Bouyssounouse (ed.), *Embedded Systems Design: The ARTIST Roadmap for Research and Development*, *LNCS*, *3436*, 2005.

14.2 PhD Theses

[18] Y. Abdeddaïm, *Scheduling with Timed Automata*, Phd thesis, INPG Grenoble, 2002.

[19] M. Aguilar, *Contribution à la validation de systèmes de processus communiquant par files d'attente : analyse statique pour la réduction de files*, Phd thesis, Institut Polytechnique de Grenoble, 2003.

[20] K. Altisen, *Application de la synthèse de contrôleurs à l'ordonnancement de systèmes temps-réel*, Phd thesis, Institut National Polytechnique de Grenoble, 2001, In French.

[21] L. Bozga, *Methods and Tools for Verification of Security Protocols*, Phd thesis, Université Joseph Fourier - VERIMAG, 2004.

[22] A. A. Collomb, *Vérification d'automates étendus : algorithmes d'analyse symbolique et mise en œuvre*, Phd thesis, Université Joseph Fourier, Grenoble, 2001.

[23] A. Curic, *Implementing Lustre programs on distributed platforms with real-time constraints*, Phd thesis, Université Joseph Fourier, Grenoble, 2005.

[24] C. Dima, *Théorie algébrique des langages formels temps réel*, Phd thesis, Université Joseph Fourier, Grenoble, 2001.

[25] F. Gaucher, *Etude du débogage de systèmes réactifs et application au langage synchrone Lustre*, Phd thesis, Institut National Polytechnique de Grenoble, 2003.

[26] L. Ghirvu, *Automatic Test Generation of Conformace Test Cases for Telecommunication Protocols*, Phd thesis, Université Joseph Fourier, 2002.

[27] A. Girard, *Analyse Algorithmique des Systèmes Hybrides*, Phd thesis, Institut National Polytechnique de Grenoble, 2004.

[28] G. Gössler, *Compositional Modelling of Real-Time Systems — Theory and Practice*, Phd thesis, Université Joseph Fourier, Grenoble, France, 2001.

[29] M. Mahfoudh, *On Satisfaiblity Checking for Difference Logic*, Phd thesis, UJF Grenoble, 2003.

[30] D. Merchat, *Réduction du nombre de variables en analyse de relations linéaires*, Phd thesis, Université Joseph Fourier, 2005.

[31] J. Mikáč, *Raffinements et preuves de systèmes Lustre*, Phd thesis, Institut National Polytechnique de Grenoble, 2005.

[32] L. Morel, *Exploitation des structures régulières et des spécifications locales pour le développement des systèmes réactifs de grande taille*, Phd thesis, Institut National Polytechnique de Grenoble, 2005.

[33] M. Moy, *Methods and Tools for the Verification of systems-on-a-chip at the transactional level*, Phd thesis, Institut National Polytechnique de Grenoble, 2005.

[34] C. Nakhli, *Approche Fondee sur les Modeles pour Java Temps-Reel*, Phd thesis, Ecole Normale Supérieure de Cachan - VERIMAG, 2005.

[35] C. Pachon, *Une approche basée sur les modèles pour le test de robustesse*, Phd thesis, Université Joseph Fourier - VERIMAG, 2005.

[36] Y. Rémond, *Un support langage pour les modes de fonctionnement des systèmes temps-réel : extension de Lustre par des automates de modes*, Phd thesis, Université Joseph Fourier, Grenoble, 2001.

[37] Y. Roux, *Description et simulation de sytèmes réactifs non-déterministes*, Phd thesis, Institut National Polytechnique de Grenoble, 2004.

[38] R. Salem-Habermehl, *Répartition de programmes synchrones temps-réel*, Phd thesis, Université Joseph Fourier, 2001.

[39] G. Schneider, *Algorithmic Analysis of Hybrid Systems*, Phd thesis, Université Joseph Fourier, 2002.

[40] M. Zanconi, *Modélisation et analyse de systèmes temps-réel avec préemption, incertitude et dépendences*, Phd thesis, Université Joseph Fourier, 2004.

14.3 Articles and Chapters of Books

[41] Y. Abdeddaïm, E. Asarin, O. Maler, "Scheduling with Timed Automata", *Theoretical Computer Science*, 2005.

[42] K. Altisen, G. Gössler, J. Sifakis, "Scheduler Modeling Based on the Controller Synthesis Paradigm.", *Journal of Real-Time Systems, special issue on Control Approaches to Real-Time Computing 23*, 1-2, 2002, p. 55–84.

[43] R. Alur, T. Dang, F. Ivancic, "Reachability Analysis of Hybrid Systems via Predicate Abstraction", *ACM transactions on embedded computing systems (TECS)*, 2004.

[44] R. Alur, T. Dang, F. Ivancic, "Counter-example Guided Predicate Abstraction of Hybrid Systems", *Theoretical Computer Science*, 2005, to appear.

[45] E. Asarin, P. Caspi, O. Maler, "Timed Regualr Expressions", *Journal of the ACM 49*, 2002, p. 172–205.

[46] K. Baukus, Y. Lakhnech, K. Stahl, "Verification of Parameterized Protocols", *Journal of Universal Computer Science 7*, 2, 2001.

[47] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, R. de Simone, "The synchronous languages 12 years later", *Proceedings of the IEEE 91*, 1, 2003.

[48] L. Bozga, C. Ene, Y. Lakhnech, "A symbolic decision procedure for cryptographic protocols with time stamps", *Journal of Logic and Algebraic Programming 65*, 2005, p. 1–35.

[49] M. Bozga, J.-C. Fernandez, L. Ghirvu, "Using Static Analysis to Improve Automatic Test Case Generation", *Software Tools for Technology Transfer 4*, 2, 2003.

[50] G. C. Buttazzo, A. Efrati, J. N. Hooker, C. L. Pape, J. Sifakis, "Abstracts of Invited Talks.", *Electr. Notes Theor. Comput. Sci. 65*, 6, 2002.

[51] P. Caspi, O. Maler, "From Control Loops to Real-Time Programs", *in : Handbook of Networked and Embedded Control Systems*, CRC Press, 2005.

[52] P. Caspi, F. Maraninchi, "La place de l'nformatique dans l'enseignement des logiciels et systèmes embarqués", *Bulletin de SPECIF*, 2003.

[53] D. Dams, Y. Lakhnech, M. Steffen, "Iterating Tarnsducers", *The Journal of Logic and Algebraic Programming*, 2002.

[54] T. Dang, "Application of Reachability Analysis to Idle Speed Control Synthesis", *International Journal of Software Engineering & Knowledge Engineering IJSEKE Special issue of selected papers from the International Embedded and Hybrid Systems Conference IEHSC'05*, 2005, p. to appear.

[55] C. Dima, "Real-Time Automata", *Journal of Automata, Languages and Combinatorics 6*, 2001, p. 3–23.

[56] J.-C. Fernandez, M. Bozga, L. Ghirvu, "State space reduction based on live variables analysis.", *Sci. Comput. Program. 47*, 2-3, 2003, p. 203–220.

[57] G. Gössler, J. Sifakis, "Composition for component-based modeling.", *Sci. Comput. Program. 55*, 1-3, 2005, p. 161–183.

[58] S. Graf, I. Ober, I. Ober, "Timed annotations in UML", *STTT, Software Tools for Technology Transfer*, 2005, DOI: 10.1007/s10009-005-0205-x.

[59] S. Graf, I. Ober, I. Ober, "Validating Timed UML models by simulation and verification", *STTT, Software Tools for Technology Transfer*, 2005, DOI: 10.1007/s10009-005-0205-x.

[60] T. A. E. Group, "(P. Caspi, A. Sangiovanni-Vincentelli, L. Almeida, A. Benveniste, B. Bouyssounouse, G. Buttazzo, I. Crnkovic, W. Damm, J. Engblom, G. Folher, M. Garcia-Valls, H. Kopetz, Y. Lakhnech, F. Laroussinie, L. Lavagno, G. Lipari, F. Maraninchi, P. Peti, J. de la Puente, N. Scaife, J. Sifakis, R. de Simone, M. Torngren, P. Veríssimo, A. J. Wellings, R. Wilhelm, T. Willemse, W. Yi) Guidelines for a graduate curriculum on embedded software and systems", *ACM Transactions on Embedded Computing Systems (TECS) 4*, 2005, p. 587 – 611.

[61] R. Huuck, B. Lukoschus, Y. Lakhnech., "Verifying Untimed and Timed Aspects of the Experimental Batch Plant", *European Journal of Control*, 2001.

[62] R. Iosif, M. Dwyer, J. Hatcliff, "Translating Java for Multiple Model Checkers: the Bandera Back End", *Formal Methods in System Design (FSMD) 26*, 2, 2005.

[63] R. Iosif, R. Sisto, "Temporal Logic Properties of Java Objects", *Journal of Systems and Software*, 2003.

[64] R. Iosif, "Symmetry Reductions for Model Checking of Concurrent Dynamic Software.", *Software Tools for Technology Transfer (STTT) 6*, 4, 2004, p. 302–319.

[65] B. Jeannet, "Dynamic Partitioning in Linear Relation Analysis. Application to the Verification of Synchronous Programs", *Formal Method in System Design 23*, 1, 2003, p. 5–37.

[66] B. Jonsson, E. Brinksma, G. Coulson, S. G. et I. Crnkovic, S. Gerard, H. Hermanns, J.-M. Jezequel, A. Ravn, P. Schnoebelen, F. Terrier, A. Votintseva, "Roadmap: Component based design and Integration platforms", *in : Embedded Systems Design: The ARTIST Roadmap for Research and Development*, *LNCS*, *3436*, 2005.

[67] J. S. Karine Altisen, Gregor Goessler, "Scheduler modeling based on the controller synthesis paradigm", *Journal of Real-Time Systems special issue on Control Approaches to Real-Time Computing*, 23, 2002, p. 55–84.

[68] G. Lafferriere, G. Pappas, S. Yovine, "Symbolic reachability computation of families of linear vector fields", *Journal of Symbolic Computation 32*, 3, 2001, p. 231–253.

[69] D. Lesens, N. Halbwachs, P. Raymond, "Automatic Verification of Parameterized Networks of Processes", *Theoretical Computer Science 256*, 2001, p. 113–144.

[70] O. Maler, "Guest Editorial: Verification of Hybrid Systems", *European Journal of Control 7*, 2001, p. 357–365.

[71] O. Maler, "Control from Computer Science", *Annual Reviews in Control*, 2002.

[72] F. Maraninchi, Y. Rémond, "Argos: an Automaton-Based Synchronous Language", *Computer Languages*, 27, 2001, p. 61–92.

[73] F. Maraninchi, Y. Rémond, "Mode-Automata: a new Domain-Specific Construct for the Development of Safe Critical Systems", *Science of Computer Programming*, 3, 2003, p. 219–254.

[74] M. Moy, "Functional Verification: Formal Verification", *in : Transaction-Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems*, Kluwer Academic Publisher, 2005.

[75] P. Niebert, S. Yovine, "Computing Efficient Operation Schemes for Chemical Plants in Multi-batch Mode", *European Journal of Control 7*, 4, 2001, p. 440–453.

[76] I. Ober, S. Graf, I. Ober, D. Lesens, "Un profil UML et un outil pour la modélisation et la validation de systèmes temps-réel", *Numéro spécial du journal Génie Logiciel consacré à la Journée NEPTUNE 05 : Ingénierie des Modèles - vérification de modèles 73*, 2005, p. 33–38.

[77] G. Pace, N. Halbwachs, P. Raymond, "Counter-example generation in symbolic abstract model-checking", *Software Tools for Technology Transfer 5*, 2—3, 2004.

[78] M. Périn, "Cohérence de spécifications multi-vues", *in : Technique et Science informatiques*, *30 n° 7*, Hermès, 2001, p. 875–900.

[79] J. Sifakis, S. Tripakis, S. Yovine, "Building models of real-time systems from application software", *Proceedings of the IEEE 91*, 1, 2003, p. 100–111.

[80] S. Tripakis, C. Sofronis, P. Caspi, A. Curic, "Translating Discrete-Time Simulink to Lustre", *ACM Transactions in Embedded Computing Systems*, 2005.

[81] S. Tripakis, S. Yovine, A. Bouajjani, "Checking Timed Buchi Automata Emptiness Efficiently", *Formal Methods in System Design 26*, 3, 2005, p. 267–292.

[82] S. Tripakis, S. Yovine, "Analysis of Timed Systems using Time-abstracting Bisimulations", *Formal Methods in System Design 18*, 1, 2001, p. 25–68.

[83] S. Tripakis, "Decentralized control of discrete event systems with bounded or unbounded delay communication", *IEEE Transactions on Automatic Control 49*, 9, 2004.

[84] S. Tripakis, "Undecidable problems of decentralized observation and control on regular languages", *Information Processing Letters 90*, 1, 2004, p. 21–28.

14.4 Contributions to Conferences and Workshops

[85] Y. Abdeddaïm, E. Asarin, O. Maler, "On Optimal Scheduling under Uncertainty", *in: Proc. TACAS*, H. Garavel, J. Hatcliff (ed.), *LNCS*, *2619*, Springer, 2003.

[86] Y. Abdeddaïm, A. Kerbaa, O. Maler, "Task Graph Scheduling using Timed Automata", *in: FMPPTA*, 2003.

[87] Y. Abdeddaïm, O. Maler, "Job-Shop Scheduling using Timed Automata", *in: Proc. CAV'01, LNCS 2102*, G. Berry, H. Comon, A. Finkel (ed.), Springer, p. 478–492, 2001.

[88] Y. Abdeddaïm, O. Maler, "Preemptive Job-Shop Scheduling using Stopwatch Automata", *in: Proc. TACAS'02*, J.-P. Katoen, P. Stevens (ed.), *LNCS*, 2280, Springer, p. 113–126, 2002.

[89] A. Akhavan, S. Bensalem, M. Bozga, E. Orfanidou, "Experiment on Verification of a Planetary Rover Controller", *in: Proceedings of the 4th Intl. Workshop on Planning and Scheduling for Space, IWPSS'04, Darmstadt, Germany*, 2004.

[90] K. Altisen, P. Bouyer, T. Cachat, F. Cassez, G. Gardey, "Introduction au contrôle des systèmes temps-réel", *in: Modélisation des Systèmes Réactifs (MSR'05), session invitée de l'ACI CORTOS*, Autrans, France, 2005. in French.

[91] K. Altisen, A. Clodic, F. Maraninchi, E. Rutten, "Using Controller Synthesis to Build Property-Enforcing Layers", *in: European Symposium on Programming (ESOP)*, 2003.

[92] K. Altisen, F. Maraninchi, D. Stauch, "Exploring Aspects in the Context of Reactive Systems", *in: Workshop on the Foundations of Aspect-Oriented Languages (FOAL), affiliated with AOSD (International Conference on Aspect-Oriented Software Development)*, Lancaster, GB, 2004.

[93] K. Altisen, S. Tripakis, "Tools for Controller Synthesis of Timed Systems", *in: RT-TOOLS*, 2002.

[94] K. Altisen, S. Tripakis, "Implementation of timed automata: an issue of semantics or modeling?", *in: Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, *LNCS*, *3829*, Springer, 2005.

[95] K. Altisen, S. Tripakis, "Implementation of timed automata: an issue of semantics or modeling?", *in: Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, *LNCS*, *3829*, Springer, 2005.

[96] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, O. Sokolsky, "Hierarchical Hybrid Modeling of Embedded Systems", *in: Embedded Software, LNCS 2211*, T. Henzinger, C. Kirsch (ed.), Springer, p. 14–31, 2001.

[97] R. Alur, T. Dang, F. Ivancic, "Reachability Analysis of Hybrid Systems via Predicate Abstraction", *in: Hybrid Systems: Computation and Control*, M. Greenstreet, C. Tomlin (ed.), Springer, 2002. to appear.

[98] R. Alur, T. Dang, F. Ivancic, "Counter-example Guided Predicate Abstraction of Hybrid Systems", *in: Tools and Algorithms for the Construction and Analysis of System*, Springer-Verlag, 2003.

[99] R. Alur, T. Dang, F. Ivancic, "Progress on Reachability Analysis of Hybrid Systems using Predicate Abstraction", *in: Hybrid Systems: Computation and Control*, O. Maler, A. Pnueli (ed.), *LNCS 2623*, Springer-Verlag, 2003.

[100] P. Amblard, F. Lagnier, M. Levy, "Using formal tools to study complex circuits behaviour", *in: Euromicro Symposium on Digital System Design (Architectures, Methods and Tools) (DSD'02)*, p. 180–186, Dortmund, Germany, 2002.

[101] P. Amblard, F. Lagnier, M. Levy, "Finite state machines: composition, verification, minimization: a case study", *in: 10th International Conference on Mixed Design (MIXDES03)*, Lodz, Poland, 2003.

[102] A. Annichini, A. Bouajjani, M. Sighireanu, "TReX : A Tool of Reachability Analysis of compleX systems", *in: 13th International Conference on Computer Aided Verification, CAV'01*, *LNCS*, *2102*, Springer-Verlag, p. 368–372, 2001.

[103] E. Asarin, S. Bansal, T. Dang, B. Espiau, O. Maler, "On Hybrid Control of Under-actuated Mechanical Systems", *in: Hybrid Systems: Computation and Control*, A. Sangiovani-Vincentelli, M. di Bendetto (ed.), *LNCS*, 2034, Springer, p. 77–88, 2001.

[104] E. Asarin, A. Bouajjani, "Perturbed Turing Machines and Hybrid Systems", *in: Proc. 16th IEEE Symp. on Logic in Computer Science (LICS'01)*, IEEE, p. 269–278, 2001.

[105] E. Asarin, T. Dang, A. Girard, "Reachability Analysis of Nonlinear Systems Using Conservative Approximation", *in: Hybrid Systems: Computation and Control*, O. Maler, A. Pnueli (ed.), *LNCS 2623*, Springer-Verlag, p. 20–35, 2003.

[106] E. Asarin, T. Dang, O. Maler, "d/dt: A Tool for Reachability Analysis of Continuous and Hybrid Systems", *in: Proc. IFAC Symposium on Non-linear Control*, IFAC, 2001.

[107] E. Asarin, T. Dang, O. Maler, "d/dt: A Verification Tool for Hybrid Systems", *in: Proc. of the 40th IEEE CDC, Orlando, Florida*, 2001.

[108] E. Asarin, T. Dang, O. Maler, "The d/dt Tool for Verification of Hybrid Systems", *in: Proc. CAV'02*, *LNCS*, 2404, Springer, p. 365–370, 2002.

[109] E. Asarin, T. Dang, "Abstraction by Projection", *in: Hybrid Systems: Computation and Control*, R. Alur, G. J. Pappas (ed.), *LNCS 2993*, Springer-Verlag, p. 32–47, 2004.

[110] E. Asarin, C. Dima, "Balanced timed regular expressions", *in: MTCS'2002*, *ENTCS*, *68*, Brno, Czech Republic, 2002. issue 5.

[111] E. Asarin, G. Pace, G. Schneider, S. Yovine, "SPeeDI – a Verification Tool for Polygonal Hybrid Systems", *in: Proceedings of "Computer Aided Verification, CAV'02"*, *Lecture Notes in Computer Science*, Springer, Denmark, 2002.

[112] E. Asarin, G. Schneider, S. Yovine, "On the Decidability of the Reachability Problem for Planar Differential Inclusions", *in: Hybrid Systems: Computation and Control*, A. Sangiovani-Vincentelli, M. di Bendetto (ed.), *LNCS*, 2034, Springer, p. 89–104, 2001.

[113] E. Asarin, G. Schneider, S. Yovine, "Towards Computing Phase Portraits of Polygonal Differential Inclusions", *in: Hybrid Systems: Computation and Control*, M. Greenstreet, C. Tomlin (ed.), Springer, 2002. to appear.

[114] E. Asarin, G. Schneider, "Widening the Boundary Between Decidable and Undecidable Hybrid Systems", *in: CONCUR'2002*, *LNCS*, 2421, Springer, p. 193–208, Brno, Czech Republic, 2002.

[115] I. Assayad, V. Bertin, F. Defaut, P. Gerner, O. Quévreux, S. Yovine, "Jahuel: A formal framework for software synthesis", *in: Proceedings of ICFEM 2005 Seventh International Conference on Formal Engineering Methods*, LNCS, Springer, Manchester, UK, 2005.

[116] I. Assayad, P. Gerner, S. Yovine, V. Bertin, "Modelling, Analysis and Implementation of an On-line Video Encoder", *in: Proceedings of "The First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'2005)"*, IEEE Computer Society, Besancon, France, 2005.

[117] I. Assayad, S. Yovine, "Compositional Constraints Generation for Concurrent Real-time Loops with Interdependent Iterations", *in: Proceedings of I2CS'05: the international conference on innovative internet community systems*, LNCS, Springer, Paris, France, 2005.

[118] R. Attias, D. Lee, A. Puri, R. Sengupta, S. Tripakis, P. Varaiya, "A Wireless Token Ring Protocol for Intelligent Transportation Systems", *in: IEEE Conference on Intelligent Transportation Systems*, 2001.

[119] R. Attias, D. Lee, A. Puri, S. Tripakis, "A Token-Ring Medium-Access Control Protocol with Quality of Service Guarantees for Wireless Ad-hoc Networks", *in: MobiCom*, 2001.

[120] K. Baukus, Y. Lakhnech, S. Bensalem, K. Stahl, "Networks of Processes with Parameterized State Space", *in: Satellite Workshops, ICALP 2001 (Crete, Greece)*, ENTCS, p. 388–402, 2001.

[121] C. Belta, J. Schug, T. Dang, V. Kumar, G. Pappas, H. Rubin, P. Dunlap, "Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium Vibrio fischeri", *in: Proc. of the 40th IEEE CDC, Orlando, Florida*, 2001.

[122] S. Bensalem, M. Bozga, M. Krichen, S. Tripakis, "Testing conformance of real-time software by automatic generation of observers", *in: Workshop on Runtime Verification (RV'04)*, *ENTCS*, Elsevier, 2004.

[123] S. Bensalem, M. Bozga, M. Krichen, S. Tripakis, "Testing conformance of real-time applications: Case of Planetary Rover Controller", *in: International Workshop on Verification and Validation of Model- Based Planning and Scheduling Systems(VVPS'05 - included in the conference ICAPS'05)*, Montery, California, 2005.

[124] S. Bensalem, S. Graf, Y. Lakhnech, "Abstraction as the Key for Invariant Verification", *in: Int. Symposium on Verification celebrating Zohar Manna's 64th Birthday*, LNCS 2772, Springer Verlag, 2003.

[125] S. Bensalem, K. Havelund, "Scalable Deadlock Analysis of Multi-threaded Programs", *in: IBM Verification Conference*, S. Ur (ed.), *Lecture Notes in Computer Science*, *3875*, Springer, Haifa, Israel, November 13 - 16 2005.

[126] S. Bensalem, D. Peled, H. Qu, S. Tripakis, "Automatic Generation of Path Conditions for Concurrent Timed Systems", *in: 1st International Symposium on Leveraging Applications of Formal Methods (ISOLA'04)*, Paphos, Cyprus, 30th October - 2st November 2004.

[127] S. Bensalem, D. Peled, H. Qu, S. Tripakis, "Generating Path Conditions for Timed Systems.", *in: Integrated Formal Methods, 5th International Conference, IFM 2005*, J. Romijn, G. Smith, J. van de Pol (ed.), *Lecture Notes in Computer Science*, *3771*, Springer, p. 5–19, Eindhoven, The Netherlands, November 29 - December 2 2005.

[128] A. Benveniste, B. Caillaud, L. Carloni, P. Caspi, A. Sangiovanni-Vincentelli, "Heterogeneous Reactive Systems Modeling: capturing causality and the correctness of loosely time-triggered protocols", *in: Emsoft04*, G. Buttazzo (ed.), ACM, 2004.

[129] A. Benveniste, P. Caspi, P. Le Guernic, H. Marchand, J.-P. Talpin, S. Tripakis, "A Protocol for Loosely Time-Triggered Architectures", *in: EMSOFT'02*, A. Sangiovanni-Vincentelli, J. Sifakis (ed.), LNCS 2491, Springer Verlag, p. 252–265, Grenoble, 2002.

[130] V. Bertin, E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venier, D. Weil, S. Yovine, "Taxys = Esterel + Kronos. A tool for verifying real-time properties of embedded systems", *in: Proc. IEEE CDC'01*, IEEE Control Systems Society, Orlando, 2001.

[131] D. Borrione, M. Boubekeur, L. Mounier, M. Renaudin, A. Sirianni, "Validation of asynchronous circuit specifications using IF/CADP", *in: IFIP Intl. Conference on VLSI, Darmstadt, Germany*, 2003.

[132] A. Bouajjani, A. Collomb-Annichini, Y. Lakhnech, M. Sighireanu, "Analysing Fair Parametric Extended Automata", *in: The 8th International Static Analysis Symposium, SAS'01*, *LNCS*, *??*, Springer-Verlag, p. 335–355, 2001.

[133] M. Boubekeur, D. Borrione, L. Mounier, M. Renaudin, A. Sirianni, "Modelling CHP descriptions in Labelled Transition Systems for an efficient formal validations of asynchronous circuit specifications", *in: Forum on Specification and Design Language (FDL'03), Frankfurt, Germany*, 2003.

[134] L. Bozga, C. Ene, R. Janvier, Y. Lakhnech, L. Mazare, M. Périn, "Automatic Verification of Security Properties Based on Abstractions", *in: Proc. NATO Workshop on Verification of Infinite-state Systems with Applications to Security (VISSAS 2005)*, 2005.

[135] L. Bozga, C. Ene, Y. Lakhnech, "On the existence of an effective and complete inference system for cryptographic protocols", *in: FOSSACS'04*, *LNCS*, 2987, 2004.

[136] L. Bozga, C. Ene, Y. Lakhnech, "A symbolic decision procedure for bounded security protocols with time stamps (Extended abstract)", *in: CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*, P. Gardner, N. Yoshida (ed.), *LNCS*, 3170, 2004.

[137] L. Bozga, Y. Lakhnech, M. Périn, "Abstract Interpretation for Secrecy using Patterns", *in: TACAS'03*, *LNCS*, 2619, 2003.

[138] L. Bozga, Y. Lakhnech, M. Périn, "HERMES: An Automatic Tool for Verification of Secrecy in Security Protocols", *in: 15th International Conference on Computer Aided Verification (CAV)*, *LNCS*, *2725*, 2003.

[139] M. Bozga, S. Graf, L. Mounier, I. Ober, J. Roux, D. Vincent, "Timed Extensions for SDL", *in: Proceedings of SDL FORUM'01 (Copenhagen, Denmark)*, *LNCS*, *2078*, Springer, p. 223–240, 2001.

[140] M. Bozga, S. Graf, L. Mounier, I. Ober, "IF Tutorial", *in: 9th SPIN'04 Workshop on Model-Checking of Software, Barcelona, Spain*, *Lecture Notes in Computer Science*, *2989*, 2004.

[141] M. Bozga, S. Graf, L. Mounier, "Automated validation of distributed software using the IF environment", *in: IEEE International Symposium on Network Computing and Applications (NCA-2001), Cambridge, MA, USA, October 8-10, 2001*, IEEE, 2001.

[142] M. Bozga, S. Graf, L. Mounier, "Automated validation of distributed software using the IF environment", *in: Workshop on Software Model-checking, associated with CAV 2001, Paris*, S. D. Stoller, W. Visser (ed.), *Electronic Notes in Theoretical Computer Science*, *55*, Elsevier Science Publishers, 2001.

[143] M. Bozga, S. Graf, L. Mounier, "IF-2.0: A Validation Environment for Component-Based Real-Time Systems", *in: Proceedings of Conference on Computer Aided Verification, CAV'02, Copenhagen*, *LNCS*, 2404, Springer Verlag, 2002.

[144] M. Bozga, S. Graf, I. Ober, I. Ober, J. Sifakis, "The IF toolset", *in: 4th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Real Time, SFM-04:RT, Bologna, Sept. 2004*, M. Bernanrdo, F. Corradini (ed.), *LNCS Tutorial*, *3185*, Springer, 2004.

[145] M. Bozga, R. Iosif, Y. Lakhnech, "Storeless Semantics and Alias Logic", *in: Proc. ACM SIGPLAN 2003 Workshop on Partial Evaluation and Semantics Based Program Manipulation*, ACM Press, p. 55 – 65, 2003.

[146] M. Bozga, R. Iosif, Y. Lakhnech, "On Logics of Aliasing", *in: Proc. 11th Intl. Static Analysis Symposium (SAS04)*, p. 344 – 360, 2004.

[147] M. Bozga, R. Iosif, "On Model Checking Generic Topologies", *in: Proc. LICS-ICALP Workshop on Logics for Resources, Processes and Programs (LRPP'04)*, p. 73–87, 2004.

[148] M. Bozga, R. Iosif, "On decidability with the arithmetic with addition and divisibility", *in: Proc. Foundations of Software Science and Computation Structures*, 2005.

[149] M. Bozga, R. Iosif, "Quantitative Verification of Programs with Lists", *in: Proc. NATO Workshop on Verification of Infinite-state Systems with Applications to Security (VISSAS 2005)*, 2005.

[150] M. Bozga, H. Jianmin, O. Maler, S. Yovine, "Verification of Asynchronous Circuits using Timed Automata", *in: Proc. TPTS 2002*, *ENTCS*, 65, 2002.

[151] M. Bozga, A. Kerbaa, O. Maler, "Scheduling Acyclic Branching Programs on Parallel Machines", *in: Proc. RTSS'04*, IEEE Press, 2004.

[152] M. Bozga, D. Lesens, L. Mounier, "Model-Checking Ariane-5 Flight Program", *in: Proceedings of FMICS'01 (Paris, France)*, INRIA, p. 211–227, 2001.

[153] V. Braberman, A. Ferrari, D. Garbervetsky, P. Listingart, S. Yovine, "JScoper: Eclipse support for Research on Scoping and Instrumentation for Real Time Java Applications", *in: ETX 2005: eclipse Technology eXchange (eTX) Workshop at OOPSLA*, San Diego, USA, 2005.

[154] V. Braberman, D. Garbervetsky, S. Yovine, "Synthesizing parametric specifications of dynamic memory utilization in object oriented programs", *in: FTfJP 2005: Formal Techiques for Java-like Programs*, To appear in Journal of Object Technology, Glasgow, Scotland, 2005.

[155] J. W. Bryans, M. Koutny, L. Mazare, P. Y. Ryan, "Opacity Generalised to Transition Systems", *in: The third international Workshop on Formal Aspects in Security and Trust (FAST'05), Newcastle, UK*, 2005.

[156] P. Caspi, A. Benveniste, "Toward an Approximation Theory for Computerised Control", *in: EMSOFT'02*, A. Sangiovanni-Vincentelli, J. Sifakis (ed.), LNCS 2491, Springer Verlag, Grenoble, 2002.

[157] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, P. Niebert, "From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications", *in: Languages, Compilers, and Tools for Embedded Systems (LCTES'03)*, ACM, 2003.

[158] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, "Translating Discrete-Time Simulink to Lustre", *in: EMSOFT'03*, Philadelphia, USA, 2003.

[159] P. Caspi, C. Mazuet, N. Reynaud-Parigot, "About the design of distributed control systems: the quasi-synchronous approach", *in: 20th International Conference Safecomp 2001*, U. Voges (ed.), *Lecture Notes in Computer Science*, *2187*, 2001.

[160] P. Caspi, P. Raymond, "From control system design to embedded code: the synchronous data-flow approach", *in: 40th IEEE Conference on Decision and Control (CDC'01)*, Orlando, Florida, 2001.

[161] P. Caspi, "Embedded control: from asynchrony to synchrony and back", *in: First International Wokshop on Embedded Software*, T. Henzinger, C. Kirsch (ed.), *Lecture Notes in Computer Science*, *2211*, 2001.

[162] P. Caspi, "Automatique continue, automatique discrete, informatique industrielle : le triangle des Bermudes", *in: MSR03*, Hermès, Metz, France, 2003.

[163] E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venier, D. Weil, S. Yovine, "TAXYS: a tool for the developpment and verification real-time embedded systems", *in: Proc. CAV'01, LNCS 2102*, G. Berry, H. Comon, A. Finkel (ed.), Springer, 2001.

[164] A. Collomb-Annichini, M. Sighireanu, "Parameterized Reachability Analysis of the IEEE 1394 Root Contention Protocol using TReX", *in: Proceedings of RT-TOOLS'01*, 2001.

[165] J. Combaz, J. Fernandez, J. Sifakis, T. Lepley, "QoS Control for Optimality and Safety", *in: Proceedings of EMSOFT'05, ACM Conference on Embedded Software Jersey City, USA*, 2005.

[166] J. Combaz, J.-C. Fernandez, J. Sifakis, T. Lepley, "Fine grain QoS control for multimedia application software", *in: Proceedings of DATE'05, Design, Automation and Test In Europe*, 2005.

[167] J.-F. Condotta, G. Ligozat, S. Tripakis, "Ultimately periodic qualitative constraint networks for spatial and temporal reasoning", *in: 17th IEEE Intl. Conf. Tools with Artificial Intelligence (ICTAI'05)*, 2005.

[168] M. A. Cornejo, H. Garavel, R. Mateescu, N. de Palma, "Specification and Verification of a Dynamic Reconfiguration Protocol for Agent-Based Applications", *in: 3rd IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems DAIS'2001 (Krakow, Poland)*, 2001.

[169] S. Cotton, E. Asarin, O. Maler, P. Niebert, "Some Progress in Satisfiability Checking for Difference Logic", *in: Proc. FORMATS/FTRTFT'04*, Springer, 2004.

[170] I. Crnkovic, J. Axelsson, S. Graf, M. Larsson, R. C. van Ommering, K. C. Wallnau, "COTS Component-Based Embedded Systems - A Dream or Reality?", *in: ICCBSS 2005, Bilbao, 2005*, *LNCS Volume 3412*, 2005.

[171] M. D'Amorim, C. Artho, H. Barringer, S. Bensalem, A. Goldberg, K. Havelund, K. Sen, "Runtime Verification with Java PathExplorer (JPAX)", *in: invited paper in Joint CAV/ISSTA Special Event on Specification, Verification, and Testing of Concurrent Software*, 2004.

[172] D. Dams, Y. Lakhnech, M. Steffen, "Iterating Transducers", *in: Computer Aided Verification'01*, 2001.

[173] T. Dang, A. Donze, O. Maler, "Verification of analog and mixed-signal circuits using hybrid systems techniques", *in: FMCAD'04*, Springer, 2004.

[174] T. Dang, G. Pappas, "Approximate Reachability Decomposition of Singular Perturbed Linear Systems", *in: Hybrid Systems: Computation and Control*, 2002. Submitted??

[175] V. Darmaillacq, J.-C. Fernandez, R. Groz, L. Mounier, J.-L. Richier, "Éléments de modélisation pour le test de politiques de sécurité", *in: Colloque sur les RIsques et la Sécurité d'Internet et des Systèmes, CRiSIS*, Bourges, France, 2005.

[176] S. Dasgupta, D. Potop-Butucaru, B. Caillaud, A. Yakovlev, "Moving from weakly endochronous systems to delay-insensitive circuits", *in: FMGALS 2005 (ENTCS)*, Verona (Italy), 2005.

[177] V. C. de Verdiere, S. Cros, C. Fabre, R. Guider, S. Yovine, "Speedup Prediction for Selective Compilation of Embedded Java Programs", *in: Proceedings of "Workshop on Embedded Software, EMSOFT'02"*, *LNCS*, *2491*, Springer, Grenoble, 2002.

[178] C. Dima, "Computing Reachability Relations in Timed Automata", *in: LICS*, 2002.

[179] J. D. Dora, A. Maignan, M. Mirica-Ruse, S. Yovine, "Hybrid computation", *in: ISSAC'01*, 2001.

[180] J. D. Dora, S. Yovine, "A methodology for analyzing the dynamics of hybrid systems", *in: European Control Conference, ECC'01*, 2001.

[181] J. Fernandez, L. Mounier, C. Pachon, "Property Oriented Test Case Generation", *in: Proceedings of FATES'03 (Satellite workshop of ASE'03), Montreal, Canada*, 2003.

[182] J. Fernandez, L. Mounier, C. Pachon, "A model-based approach for robustness testing", *in: Proceedings of TestCom'05, Montreal, Canada*, *LNCS*, *3502*, 2005.

[183] G. Frehse, B. Krogh, R. Rutenbar, O. Maler, "Time Domain Verification of Oscillator Circuit Properties", *in: Workshop on Formal Verification of Analog Circuits, FAC'05*, T. Dang, O. Maler (ed.), p. 1–13, 2005.

[184] D. Garbervetsky, C. Nakhli, S. Yovine, H. Zorgati, "Program Instrumentation and Run-Time Analysis of Scoped Memory in Java", *in: Proceedings of "Runtime Verification, RV'04"*, *Electronic Notes in Theoretical Computer Science*, Elsevier Science, Barcelona, Spain, 2004.

[185] P. Gerner, T. Dang, "Computing Schedules for Multithreaded Real-Time Programs Using Geometry", *in: FORMATS-FTRTFT'04*, Y. Lakhnech, S. Yovine (ed.), *LNCS 3253*, Springer-Verlag, p. 325–342, 2004.

[186] A. Girault, X. Nicollin, "Clock-Driven Automatic Distribution of Lustre Programs", *in: 3rd ACM International Conference on Embedded Software (EMSOFT'03)*, Philadelphia, USA, 2003.

[187] L. Gonnord, N. Halbwachs, P. Raymond, "From Discrete Duration Calculus to Symbolic Automata", *in: 3rd International Workshop on Synchronous Languages, Applications, and Programs, SLAP'04*, Barcelona, Spain, 2004.

[188] G. Gössler, J. Sifakis, "Component-Based Construction of Deadlock-Free Systems: Extended Abstract", *in: Proceedings of FSTTCS'03*, *LNCS*, *2914*, p. 420–433, 2003.

[189] G. Gössler, J. Sifakis, "Composition for Component-Based Modeling", *in: 1st Symposium on Formal Methods for Components and Objects, revised lectures*, *LNCS Tutorials*, *2852*, 2003.

[190] G. Gössler, J. Sifakis, "Priority Systems", *in: proceedings of FMCO'03*, *LNCS 3188*, p. 314–329, 2004.

[191] G. Gössler, "Prometheus — A Compositional Modeling Tool for Real-Time Systems", *in: Proc. Workshop RT-TOOLS 2001*, P. Pettersson, S. Yovine (ed.), Technical report 2001-014, Uppsala University, Department of Information Technology, 2001.

[192] S. Graf, S. Gérard, O. Haugen, I. Ober, B. Selic, "MARTES - Specification and Validation of Real-time and Embedded Systems, workshop overview", *in: MoDELS 2005 International Workshops Doctoral Symposium, Educators Symposium Montenegro Bay, Jamaica, October 2005 Revised Selected Papers*, *LNCS 3844*, 2005.

[193] S. Graf, O. Haugen, I. Ober, B. Selic, "SVERTS - Specification and Validation of Real-time and Embedded Systems, workshop overview", *in: Workshops associated with UML 2005*, *LNCS 3297*, 2004.

[194] S. Graf, J. Hooman, "Correct Development of Embedded Systems", *in: European Workshop on Software Architecture: Languages, Styles, Models, Tools, and Applications (EWSA 2004), co-located with ICSE 2004, St Andrews, Scotland*, *LNCS 3047*, 2004.

[195] S. Graf, G. Jia, "Verification Experiments on the Mascara Protocol", *in: Proceedings of SPIN workshop 2001, Toronto*, *LNCS*, 2001.

[196] S. Graf, I. Ober, I. Ober, "Timed annotations in UML", *in: Workshop SVERTS on Specification and Validation of UML models for Real Time and Embedded Systems, a satellite event of UML 2003, San Francisco, October 2003*, 2003.

[197] S. Graf, I. Ober, "A Real-time profile for UML and how to adapt it to SDL", *in: SDL Forum 2003, July 1-4, Stuttgart*, *LNCS*, 2708, 2003.

[198] S. Graf, I. Ober, "How useful is the UML real-time profile SPT without Semantics?", *in: SIVOES 2004, associated with RTAS 2004, Toronto Canada*, 2004. position paper.

[199] S. Graf, A. Prinz, "A Framework for Time in FDTs", *in: FORTE 2004, participants proceedings*, 2004.

[200] S. Graf, A. Prinz, "Time in Abstract State Machines", *in: ASM 2005, Paris*, *to appear in LNCS*, 2005.

[201] S. Graf, "Expression of time and duration constraints in SDL", *in: 3rd SAM Workshop on SDL and MSC, University of Wales Aberystwyth*, *LNCS*, 2599, Springer Verlag, 2002.

[202] N. Halbwachs, S. Baghdadi, "Synchronous modeling of asynchronous systems", *in: EMSOFT'02*, A. Sangiovanni-Vincentelli, J. Sifakis (ed.), LNCS 2491, Springer Verlag, Grenoble, 2002.

[203] N. Halbwachs, D. Merchat, C. Parent-Vigouroux, "Cartesian factoring of polyhedra in linear relation analysis", *in: Static Analysis Symposium, SAS'03*, LNCS 2694, Springer Verlag, San Diego, 2003.

[204] N. Halbwachs, "A synchronous language at work: the story of Lustre", *in: Third ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE'2005*, Verona, Italy, 2005.

[205] R. Iosif, "Symmetry Reduction Criteria for Software Model Checking", *in: Proc. 9th SPIN Workshop on Software Model Checking*, *Lecture Notes in Computer Science*, Springer Verlag, p. 22 – 41, 2002.

[206] E. Jahier, B. Jeannet, F. Gaucher, F. Maraninchi, "Automatic State Reaching for Debugging Reactive Programs", *in: AADEBUG'2003 – Fifth International Workshop on Automated Debugging*, Ghent, 2003.

[207] E. Jahier, P. Raymond, P. Baufreton, "Case Studies with Lurette V2", *in: First International Symposium on Leveraging Applications of Formal Method, ISoLa 2004*, Paphos, Cyprus, 2004.

[208] R. Janvier, Y. Lakhnech, L. Mazaré, "Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries", *in: Proceedings of the 14th European Symposium on Programming (ESOP'05)*, *Lecture Notes in Computer Science*, *3444*, Springer, Edinburgh, U.K., 2005.

[209] J.Mikáč, P. Caspi, "Temporal refinement in Lustre", *in: Slap'05*, *Electronic Notes in Theoretical Computer Science*, Elsevier, 2005. to appear.

[210] J. Kapinski, O. Maler, O. Stursberg, B. Krogh, "On Systematic Simulation of Open Continuous Systems", *in: Hybrid Systems: Computation and Control*, O. Maler, A. Pnueli (ed.), Springer, 2003.

[211] C. Kloukinas, C. Nakhli, S. Yovine, "A Methodology and Tool Support for Generating Scheduled Native Code for Real-Time Java Applications", *in: EMSOFT 2003*, *LNCS*, *2855*, 2003.

[212] C. Kloukinas, S. Yovine, "Synthesis of Safe, QoS Extendible, Application Specific Schedulers for Heterogeneous Real-Time Systems", *in: Proceedings of the 15th Euromicro Conference on Real-Time Systems (ECRTS'03)*, *ISBN 0-7695-1936-9*, 2003.

[213] C. Kloukinas, "Data-Mining Synthesised Schedulers for Hard Real-Time Systems", *in: Proceedings of the 19th IEEE Conference on Automated Software Engineering (ASE 2004)*, IEEE Computer Society Press, Linz, Austria, 2004.

[214] M. Krichen, S. Tripakis, "Black-box conformance testing for real-time systems", *in: 11th International SPIN Workshop on Model Checking of Software (SPIN'04)*, *LNCS*, *2989*, Springer, 2004.

[215] M. Krichen, S. Tripakis, "Real-time Testing with Timed Automata Testers and Coverage Criteria", *in: Joint conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS-FTRTFT'04)*, *LNCS*, *3253*, Springer, 2004.

[216] M. Krichen, S. Tripakis, "An Expressive and Implementable Formal Framework for Testing Real-Time Systems", *in: The 17th IFIP Intl. Conf. on Testing of Communicating Systems (TESTCOM'05)*, *LNCS*, *3502*, Springer, 2005.

[217] M. Krichen, S. Tripakis, "State identification problems for timed automata", *in: The 17th IFIP Intl. Conf. on Testing of Communicating Systems (TESTCOM'05)*, *LNCS*, *3502*, Springer, 2005.

[218] Y. Lakhnech, S. Bensalem, S. Berezin, S. Owre, "Incremental Verification by Abstraction", *in: Tools and Algorithms for the Construction and Analysis of Systems*, T. Margaria, W. Yi (ed.), *LNCS*, *2031*, Springer-Verlag, p. 98–112, 2001.

[219] O. Maler, B. Krogh, M. Mahfoudh, "On Control with Bounded Computational Resources", *in: FTRTFT'02*, *LNCS*, 2469, p. 147–164, 2002.

[220] O. Maler, D. Nickovic, "Monitoring Temporal Properties of Continuous Signals", *in: Proc. FORMATS/FTRTFT'04*, Springer, 2004.

[221] O. Maler, A. Pnueli, "On Recognizable Timed Languages", *in: FOSSACS*, I. Walukiewich (ed.), Springer, 2004.

[222] O. Maler, "Control from Computer Science", *in: Proc. IFAC Symposium on Non-linear Control*, IFAC, 2001.

[223] O. Maler, "On Optimal and Sub-optimal Control in the Presence of Adversaries", *in: Proc. Wodes'04*, 2004.

[224] F. Maraninchi, L. Morel, "Arrays and Contracts for the Specification and Analysis of Regular Systems", *in: Fourth International Conference on Application of Concurrency to System Design (ACSD)*, Hamilton, Ontario, Canada, 2004.

[225] F. Maraninchi, L. Morel, "Logical-Time Contracts for the Development of Reactive Embedded Software", *in: 30th Euromicro Conference, Component-Based Software Engineering Track (ECBSE)*, Rennes, France, 2004.

[226] F. Maraninchi, Y. Rémond, E. Rutten, "Effective programming language support for discrete-continuous mode-switching control systems", *in: 40th IEEE Conference on Decision and Control (CDC)*, 2001.

[227] L. Mazaré, "Decidability of Opacity with Non-Atomic Keys", *in: The second international Workshop on Formal Aspects in Security and Trust (FAST'04)*, 2004.

[228] L. Mazaré, "Satisfiability of Dolev-Yao Constraints", *in: Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA'04)*, 2004.

[229] L. Mazaré, "Using Unification For Opacity Properties", *in: Workshop on Issues in the Theory of Security (WITS'04)*, 2004.

[230] J. Mikáč, P. Caspi, "Flush: a system development tool based on Scade/Lustre", *in: Formal methods for industrial critical systems, FMICS05*, 2005.

[231] J.-F. Monin, "Proof pearl: from concrete to functional unparsing", *in: Proceedings of TPHOL'04*, 2004.

[232] L. Morel, "Efficient Compilation of Array Iterators for Lustre", *in: Electronic Notes in Theoretical Computer Science*, F. Maraninchi, A. Girault, E. Rutten (ed.), *65*, Elsevier, 2002.

[233] M. Moy, F. Maraninchi, L. Maillet-Contoz, "LusSy: A Toolbox for the Analysis of Systems-on-Chip at the Transactional Level", *in: Proc. ACSD'05*, 2005.

[234] M. Moy, F. Maraninchi, L. Maillet-Contoz, "Pinapa: The Extraction Tool for SystemC descriptions of Systems-on-Chip", *in: Proc. EMSOFT'05*, 2005.

[235] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, N. Jain, O. Maler, "Verification of Timed Automata via Satisfiability Checking", *in: FTRTFT*, W. Damm, E.-R. Olderog (ed.), *LNCS*, *2469*, Springer, p. 225–244, 2002.

[236] I. Ober, S. Graf, I. Ober, "Validating Timed UML models by simulation and verification", *in: Workshop SVERTS on Specification and Validation of UML models for Real Time and Embedded Systems, a satellite event of UML 2003, San Francisco, October 2003*, 2003.

[237] I. Ober, S. Graf, I. Ober, "Model Checking of UML Models via a Mapping to Communicating Extended Timed Automata", *in: 11th International SPIN Workshop on Model Checking of Software, 2004*, *LNCS 2989*, 2004.

[238] I. Ober, S. Graf, Y. Yushtein, "Timing analysis and validation of the embedded MARS bus manager", *in: Intl Workshop on Modeling and Analysis of Real Time Embedded Systems, MARTES'05, associated with MoDELS'05*, 2005.

[239] I. Ober, "An ASM semantics for UML Derived from the meta-model and incorporating actions", *in: Abstract State Machines - Advances in Theory and Applications.*, *LNCS*, *2589*, Proceedings 10th International Workshop, ASM 2003, 2003.

[240] G. Pace, N. Halbwachs, P. Raymond, "Counter-example generation in symbolic abstract model-checking", *in: 6th International Workshop on Formal Methods for Industrial Critical Systems, FMICS'2001*, Inria, Paris, 2001.

[241] C. Pachon, "Une approche pour la génération automatique de tests de robustesse", *in: Proceedings of Majecstic'03, Marseille, France*, 2003.

[242] A. Puri, S. Tripakis, P. Varaiya, "Problems and Examples of Decentralized Observation and Control for Discrete Event Systems", *in: Workshop on Supervisory Control for Discrete Event Systems, SCODES*, 2001.

[243] P. Raymond, Y. Roux, "Describing non-deterministic reactive systems by means of regular expressions", *in: Synchronous Languages Applications and Programming - SLAP02*, *65.5*, Electronic Notes in Theoretical Computer Science, 2002.

[244] Robby, M. B. Dwyer, J. Hatcliff, R. Iosif, "Space-Reduction Strategies for Model Checking Dynamic Software", *in: Workshop on Software Model Checking (SoftMC03)*, *Electronic Notes in Theoretical Computer Science*, *89*, 3, 2003.

[245] G. Salagnac, S. Yovine, D. Garbervetsky, "Fast Escape Analysis for Region-based Memory Management", *in: Proceedings of "1st Int. Workshop on Abstract Interpretation for Object Oriented Languages (AIOOL'05)"*, *Electronic Notes in Theoretical Computer Science*, Elsevier Science, Paris, France, 2005.

[246] R. B. Salah, M. Bozga, O. Maler, "On Timing Analysis of Combinational Circuits", *in: FORMATS'03*, *Lecture Notes in Computer Science*, *2791*, Springer, 2003.

[247] N. Scaife, P. Caspi, "Integrating model-based design and preemptive scheduling in mixed time- and event-triggered systems", *in: Euromicro conference on Real-Time Systems (ECRTS'04)*, Catania, Italy, 2004.

[248] N. Scaife, C. Sofronis, P. Caspi, S. Tripakis, F. Maraninchi, "Defining and translating a "safe" subset of Simulink/Stateflow into Lustre", *in: 4th ACM International Conference on Embedded Software (EMSOFT'04)*, 2004.

[249] R. Sengupta, S. Tripakis, "Decentralized Fault Diagnosis for Discrete Event Systems", *in: IEEE Conference on Decision and Control, invited session "Decentralized Approaches for Failure Diagnosis"*, 2002.

[250] J. Sifakis, "Modeling real-time systems–challenges and work directions", *in: EMSOFT 2001, lake Tahoe*, *LNCS 2211*, 2001.

[251] J. Sifakis, "Scheduler Modeling Based on the Controller Synthesis Paradigm.", *in: Formal Techniques in Real-Time and Fault-Tolerant Systems, 7th International Symposium, FTRTFT 2002*, *Lecture Notes in Computer Science*, *2469*, Springer, p. 107–110, 2002.

[252] J. Sifakis, "Embedded Systems - Challenges and Work Directions.", *in: Principles of Distributed Systems, 8th International Conference, OPODIS 2004, Grenoble, France, December 15-17, 2004, Revised Selected Papers*, *Lecture Notes in Computer Science*, *3544*, Springer, p. 184–185, 2004.

[253] J. Sifakis, "Modeling Real-Time Systems.", *in: Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS 2004), 5-8 December 2004, Lisbon, Portugal*, IEEE Computer Society, p. 5–6, 2004.

[254] J. Sifakis, "A Framework for Component-based Construction Extended Abstract", *in: Third IEEE International Conference on Software Engineering and Formal Methods (SEFM'05)*, p. 293–300, 2005.

[255] J.-P. Talpin, D. Potop-Butucaru, B. Caillaud, "Compositional synthesis of latency-insensitive systems from multi-clocked synchronous specifications", *in: EMSOFT 2005*, New York, 2005.

[256] S. Tripakis, C. Sofronis, N. Scaife, P. Caspi, "Semantics-preserving and memory-efficient implementation of inter-task communication under static-priority or EDF schedulers", *in: Embedded Software (EMSOFT'05)*, ACM, 2005.

[257] S. Tripakis, S. Yovine, "Timing Analysis and Code Generation of Vehicle Control Software using Taxys", *in: Electronic Notes in Theoretical Computer Science*, K. Havelund, G. Rosu (ed.), *55*, Elsevier Science Publishers, 2001.

[258] S. Tripakis, "Undecidable problems of decentralized observation and control", *in: IEEE Conference on Decision and Control (CDC'01)*, 2001.

[259] S. Tripakis, "Automated Composition of Module Chains", *in: ETAPS'02 Workshop on Software Composition (SC'02)*, Volume 65, issue 4 of ENTCS, Elsevier, 2002.

[260] S. Tripakis, "Decentralized control of discrete event systems with bounded or unbounded delay communication", *in: 6th International Workshop on Discrete Event Systems (WODES'02)*, IEEE CS Press, 2002.

[261] S. Tripakis, "Description and Schedulability Analysis of the Software Architecture of an Automated Vehicle Control System", *in: Embedded Software (EMSOFT'02)*, *LNCS*, *2491*, Springer, 2002.

[262] S. Tripakis, "Fault Diagnosis for Timed Automata", *in: Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'02)*, *LNCS*, *2469*, Springer, 2002.

[263] S. Tripakis, "Automated Module Composition", *in: Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, *LNCS*, *2619*, Springer, 2003.

[264] S. Tripakis, "Folk theorems on the determinization and minimization of timed automata", *in: Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, *LNCS*, *2791*, Springer, 2004.

[265] S. Tripakis, "Decentralized observation problems", *in: 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005 (CDC-ECC'05)*, IEEE, 2005.

[266] S. Tripakis, "Two-phase distributed observation problems", *in: 5th Intl. Conf. on Application of Concurrency to System Design (ACSD'05)*, IEEE CS Press, 2005.

[267] M. Zennaro, R. Sengupta, S. Tripakis, J. Ko, "A Service Network Architecture for a Multi-Vehicle Search Mission", *in: IEEE Conference on Decision and Control*, 2001.

14.5 Research Reports and Internal Publications

[268] Agedis, "AML: Agedis Language Specification", *Research Report n^o D-2.2*, Agedis IST-1999-20218, 2001.

[269] O. B. S. Ali, *Simulation des Systèmes Continus Ouverts*, Mémoire, DEA Informatique: Système et Communication, Université Joseph Fourier, Grenoble, 2003.

[270] E. Asarin, O. Maler, S. Y. (Eds), "Theory and Practice of Timed Systems, TPTS'02", *Research Report* , 2002.

[271] L. Bozga, Y. Lakhnech, M. Perin, "Pattern-based Abstractions for Secrecy", *Research Report* , Verimag, 2001.

[272] M. Bozga, R. Iosif, Y. Lakhnech, "Counting Aliases", *Research Report n° TR-2004-17*, Verimag, 2004.

[273] M. Bozga, R. Iosif, "Quantitative Verification of Programs with Lists", *Research Report n° TR-2005-2*, Verimag Tech Report, 2005.

[274] M. Bozga, Y. Lakhnech, "IF-2.0: Common Language Operational Semantics", *Technical report*, Verimag, 2002.

[275] M. Bozga, O. Maler, "Timed Automata Approach for the AXXOM Case Study", *Research Report* , Verimag, 2003.

[276] M. Bozga, "Common Description Language IF-2.0", *Research Report n° D-2.1*, Advance IST-1999-29082, 2001.

[277] T. Dang, O. M. (Eds.), "Workshop on Formal Verification of Analog Circuits, FAC'05", *Research Report* , Edinburgh, 2005.

[278] L. Danthony-Gonnord, "Du calcul des durées aux automates symboliques", *Mémoire de DEA* , Universités Paris VI/VII, 2003.

[279] A. Donze, *Etude d'un modèle de contrôleur hybride*, Mémoire, DEA Informatique: Système et Communication, Université Joseph Fourier, Grenoble, 2003, http://www-verimag.imag.fr/ maler/Papers/dea-donze.ps.

[280] A. Donzé, "On Temporal Differences Algorithms For Continuous Systems", *Research Report n° TR-2005-8*, Verimag Technical Report, 2005.

[281] S. Graf, F. de Boer, P. Combes, J. Hooman, H. Kugler, M. Kyas, D. Lesens, I. Ober, A. Votintseva, Y. Yushtein, M. Zenou, "Omega Final Project Report", *Deliverable of the omega ist project*, 2005, 60 pages, a short version is to be published in SoSym.

[282] S. Graf, I. Ober, "Timed Extensions for SDL", *Document temporaire, itu, secteur de standardisations des télécommunications, groupe d'étude 17*, 2001.

[283] S. Graf, "States and events in the context of timed systems", *contribution to the workshop ST.EVE on state based versus event based approach, a satellite event of fm 2003*, Verimag, 2003.

[284] R. Janvier, Y. Lakhnech, L. Mazare, "Completing the Picture: Soundness of Formal Encryption in the Presence of Active Adversaries", *Research Report n° TR-2004-19*, Verimag, Centre Équation, 38610 Gières, 2005.

[285] R. Janvier, Y. Lakhnech, L. Mazaré, "(De)Compositions of Cryptographic Schemes and their Applications to Protocols", *Research Report nº TR-2005-3*, Verimag, Centre Équation, 38610 Gières, 2005.

[286] L. M. Jeremy W. Bryans, Maciej Koutny, P. Y. Ryan, "Opacity Generalised to Transition Systems", *Research Report nº TR-2004-25*, Verimag, Centre Équation, 38610 Gières, 2004.

[287] M. Krichen, S. Tripakis, "State identification problems for finite-state transducers", *Research Report nº TR-2005-5*, Verimag, 2005, Available at http://www-verimag.imag.fr/TR/TR-2005-5.pdf.

[288] Y. Lakhnech, L. Mazare, "Computationally Sound Verification of Security Protocols Using Diffie-Hellman Exponentiation", *Research Report nº TR-2005-7*, Verimag, Centre Équation, 38610 Gières, 2005.

[289] Y. Lakhnech, L. Mazaré, "Probabilistic Opacity for a Passive Adversary and its Application to Chaum's Voting Scheme", *Research Report nº TR-2005-4*, Verimag, Centre Équation, 38610 Gières, 2005.

[290] D. Merchat, "Représentation d'ensembles numériques, en vue de la vérification et du test de programmes", *Mémoire de DEA* , Université Joseph Fourier, 2001.

[291] I. Ober, S. Graf, D. Lesens, "A case study in UML model-based validation: The Ariane-5 launcher software", *Research Report nº TR-2005-9*, Verimag, Centre Équation, 38610 Gières, 2005, submitted to SoSym.

[292] M. Périn, "Consistency of static and dynamic diagrams in multiple views graphical design methods", *Saladin technical report*, University of L'Aquila, Italy, 2001.

[293] S. Shapero, A. Donzé, "Search-based control of the simplified model of the ABB case study", *Research Report nº 2005-6*, Verimag, 2005.

[294] S. Tripakis, "Software Architecture of PATH's Automated Vehicle Control", *Research Report* , California PATH, 2001, Technical Report.

[295] H. Venturini, J.-F. Monin, M. Bozga, "A Translation From Ambient-Calculus to IF, a Language for Asynchronous Real-Time Systems", *Research Report nº TR-2005-10*, VERIMAG, Centre Équation, 38610 Gières, 2005.

14.6 Miscellaneous

[296] *Autofor, Méthodes formelles pour les automatismes de sécurité* (CD-ROM, Akita Ed.), 2001.

[297] S. Bensalem, K. Havelund, "Deadlock Analysis of Multi-threaded Java Programs", To appear in Formal Method and System Design, Special Issue, 2004.

[298] O. Maler, D. Nickovic, A. Pnueli, "Real-time Temporal logic: past, present, future", submitted for publication, 2005.

[299] F. Maraninchi, N. Halbwachs, "Travaux d'étude sur la spécification ELMU-A380", Rapport de contrat de conseil Airbus, 40 pages, 2002.

[300] F. Maraninchi, L. Morel, Y. Rémond, "Contrat d'étude ELMU", Rapport de contrat de conseil Airbus, 44 pages, 2001.

[301] J.-F. Monin, "Formalisation of the Join-calculus in Type Theory", International Workshop on Formal Methods and Security, 2004.

[302] R. B. Salah, M. Bozga, O. Maler, "Automatic Abstraction of Real-Time Components", submitted for publication, 2005.

14.7 Distributed Tools

1. PINAPA : *Open source SystemC front-end*
   `http://www.greensocs.com/projects-pinapa.html`

2. Simulink2Lustre and Stateflow2Lustre : *Simulink and Stateflow (Mathworks Inc. trademark) to Lustre translators.*
   `http://www-verimag.imag.fr/SYNCHRONE/index.php?page=SimulinkStateflow2Lustre/SimulinkStateflow2Lustre`

3. Lustre V4 / Lesar : *Full Lustre distribution including compilers, model-cheker*
   `http://www-verimag.imag.fr/~raymond/tools/lv4-distrib.html`

4. Reglo : *Regular expressions to Lustre compiler*
   `http://www-verimag.imag.fr/~raymond/tools/reglo-manual/reglo-manual.html`

5. Lurette : *Black-box testing tool for reactive programs (Lustre)*
   `http://www-verimag.imag.fr/SYNCHRONE/index.php?page=lurette/lurette`

6. Lucky : *Simulation of non-deterministic reactive systems (see Lurette)*
   `http://www-verimag.imag.fr/SYNCHRONE/index.php?page=lurette/lucky`

7. BDDC : *A basic symbolic calculator over Boolean expressions*
   `http://www-verimag.imag.fr/~raymond/tools/bddc-manual/bddc-manual-pages.html`

8. Nbac : *Abstract interpretation tool for numerical Lustre programs*
   `http://www.irisa.fr/prive/bjeannet/nbac/nbac.html`

9. Gloups : *Lustre to PVS proof obligations generator*
   `http://www-verimag.imag.fr/SYNCHRONE/index.php?page=gloups`

10. LuDic : *Lustre debugger*
    `http://www-verimag.imag.fr/SYNCHRONE/LUDIC_Home_page.html`

11. IF Toolbox : *Full IF Toolbox distribution including front ends, application programming interfaces and simulation, model checking and static analysis tools*
    http://www-verimag.imag.fr/~async/IF

12. uml2if : *Compiler translating UML (the OMEGA profile) into IF*
    http://www-verimag.imag.fr/~async/IF

13. sdl2if : *Compiler translating SDL specifications into IF*
    http://www-verimag.imag.fr/~async/IF

14. SMI (Symbolic Model Interface): *Symbolic model-checking tool, working on communicating, extended automata, and interfaced with several BDD packages.*
    http://www-verimag.imag.fr/~async/SMI

15. Hermes: *Verification of secrecy properties for unbounded security protocols.*
    http://www-verimag.imag.fr/~lbozga/hermes/hermes.php

16. Tgv: *Model-based test-case generation for black-box conformance testing (developed in collaboration with Irisa/Vertecs)*
    http://www.irisa.fr/pampa/VALIDATION/TGV/TGV.html

17. Aldebaran: *Equivalence checking and minimization tool for communicating systems, represented by labelled transition systems*
    http://www.inrialpes.fr/vasy/cadp.html

18. OpenKronos: *A model-checker for timed Buchi automata*
    http://www-verimag.imag.fr/~tripakis/openkronos.html

19. Kronos: *A TCTL model-checker for timed automata*
    http://www-verimag.imag.fr/TEMPORISE/kronos/

20. RTSpin: *A real-time extension of Spin*
    http://www-verimag.imag.fr/~tripakis/rtspin.html

Part V

Projects and Industrial Contracts

Chapter 15

Projects

15.1 European Projects

15.1.1 ADVANCE

`http://liafa.jussieu.fr/~haberm/ADVANCE/main.html`
Starting date: Octomber 2000, **Duration**: 39 months
Partners: LIAFA - University of Paris 7 (coordinator), France Telecom, University of Liege, Technical University of Munich, University of Uppsala, Verimag and Weizmann Institute

Telecommunication protocols are critical systems of very high complexity. The goal of the project has been to develop a new generation of validation tools based on advanced an novel techniques allowing to tackle problems that arise in the validation of modern protocols, and that are beyond the scope of the state-of-the-art methods and tools. The objectives of this project were (1) to develop a general methodology for the specification of automated validation of telecommunication protocols, taking into account important aspects such as complex timing constraints, use of unbounded data structures, like counters and queues, parametrization, mobility, etc. (2) to provide tool support for this methodology, and (3) to prove it substantially advances the state of the art by applying it to industrial relevant case studies.

The major outcome of the Advance project has been a validation environment which integrates powerful abstraction techniques and efficient validation techniques based on reachability analysis for finite and infinite state systems. This environment includes several specialized tools (abstraction tools, symbolic analysis tools, etc) developed by different partners which are all interconnected through a common format for the description of extended automata based on the IF language (see sections 7.1.1 and 5.2.1).

15.1.2 AGEDIS

`http://www.agedis.de`
Starting date: December 2000, **Duration**: 39 months

Partners: IBM Haifa Research Laboratory (co-ordinator), IBM Hursley, Imbus AG, Intrasoft International, France-Telecom, Oxford University and VERIMAG

The AGEDIS project is a three-year research and development on the part of a consortium of seven industrial and academic bodies, partially funded by the European Commission under the 5th Framework agreement (IST-1999-20218). The aim of this project is to provide a methodology and tools for the automatization of *model-based software testing*, with particular emphasis on distributed applications. The expected benefits are both an improvement of the software quality and a time and cost reduction of the testing phase.

The main project results are:

- The definition of an *UML profile*, called AML (Agedis Modeling Language), dedicated to the behavioral specification and test of distributed systems. Thus, in addition to the system specification, an AML model also include some *test directives*. A test directive may contain some coverage requirements (on variable, transition or state combination of the specification), test constraints (invariants) and test purposes (abstract scenario to be observed during the test execution). The AML action language is the Verimag IF intermediate format.

- The design and implementation of a complete *test environment*. This environment relies on three main components: an AML compiler (able to translate an AML description into plain IF), a test generator (able to produce a set of abstract test cases), and a test engine (able to execute these test cases on a distributed Java or C++ application). This environment also includes a graphical user interface, an AML interactive simulator, and several tools to manage the results obtained during a test campaign.

- The experimentation of the test environment on three case studies provided by the industrial partners of the consortium: a Java programming interface to a messaging protocol, a web-based public procurement service, and a middleware for message distribution in a telecommunication service).

From the dissemination point of view, the AGEDIS consortium also produced a quarterly newsletter and held an European conference on model-driven software engineering (ECMDSE). It also created an institutional package of the tool environment, freely licensed to non-commercial bodies.

During the project Verimag was in charge of the test generator design and development. This generator is based on the TGV tool (co-developed with IRISA Vertecs), but it also integrates some features from the Gotcha tool (developed by IBM Haifa). Verimag also largely contributed to the definition of the UML profile, and on the industrial experiments.

15.1.3 ARTIST and ARTIST2

`http://www.artist-embedded.org/FP6/`

The long-term objectives of the ARTIST2 Network of Excellence on Embedded Systems Design is to build a durable European research community on Embedded Systems Design.

This will be achieved by integrating teams and competencies from 7 essential topics called clusters: Modelling and Components, Hard Real-Time, Adaptive Real-Time, Compilers and Timing Analysis, Execution Platforms, Control for Embedded Systems, Testing and Verification. The NoE will act as a Virtual Centre of Excellence in the area of Embedded Systems Design.

Integration will be achieved around a Joint Programme of Activities (JPA), aiming to create critical mass from the selected European teams.

The ARTIST2 Network of Excellence on Embedded Systems Design implements an international and interdisciplinary fusion of effort to create a unique European virtual Centre of Excellence on Embedded Systems Design. This interdisciplinary effort in research is mandatory to establish Embedded Systems Design as a discipline, combining competencies from electrical engineering, computer science, applied mathematics, and control theory. The ambition is to compete on the same level as equivalent centres in the USA (Berkeley, Stanford, MIT, Carnegie Mellon), for both the production and transfer of knowledge and competencies, and for the impact on industrial innovation.

Building the embedded systems design scientific community is an ambitious programme. To succeed, ARTIST2 builds on the achievements and experience from the ARTIST1 FP5 Accompanying Measure on Advanced Real-Time Systems.

The Joint Programme of Activities includes research both within the clusters, and between clusters. Intra-cluster research aims to create critical mass and excellence on each essential topic. Inter-cluster research aims to integrate the area as a whole. The implementation of the JPA includes research platforms and mobility of personnel. A central mission for the NoE is spreading excellence in the area, including Education and Training, Dissemination and Communication, Industrial Liaison, and International Collaboration. Through these actions, we hope to promote the emergence of this new multi-disciplinary area, and forge a scientific community.

15.1.4 ASSERT

`http://www.assert-online.net`
Starting date: September 2004, **Duration**: 36 months
Partners: European Space Agency (coordinator), ALCATEL-Space, ALENIA SPAZIO SpA, ASTRIUM (EADS) SAS, Axlog Ingenierie, BSSE, CS-Systèmes d'Information, DASSAULT Aviation, DIT/UPM university of Madrid, Dutch Space BV, EADS Corporate Research Center, EADS-Space Transportation (D), MBDA France, EADS-Space Transportation (F), Ecole Nationale Superieure des Télécommunications, European Software Institute, ESTEREL Technologies, ETH -Swiss Federal Institute of Technology, Institut National de Recherche en Informatique et Automatique, INTECS HRT, CNRS-LAAS, CNRS-VERIMAG, PROVER, SciScys, SoftwCare, SynSpace, Terma A/S, TNI-Valiosys, Univeristy of Padua, Vienna University of Technology

ASSERT (Automated proof based System and Software Engineering for Real-Time) is an integrated project (IP) co-sponsored by the European Commission under the Information Society Technology (IST) priority within the 6th Framework Programme (FP6).

The project addresses the strategic objective of "Embedded Systems".

The ASSERT main goal is to improve the system-and-software development process for critical embedded real-time systems, in the Aerospace and Transportation domains by:

- Identifying and developing proven critical system families, architecture, using a proof based development process supported by formal notations, component models, and innovative processes and tools.

- Developing associated building blocks that can be composed, tailored and verified in open frameworks that shall be reused and shared by European teams across multi domain projects.

Within ASSERT, Vérimag is coordinator of the cluster "Software development and verification tools".

15.1.5 AMETIST

`http://ametist.cs.utwente.nl/`
Starting date: April 2002, **Duration**: 36 months
Partners: University of Nijmegen, Bosch, Cybernetix, Axxom, Terma, University of Aalborg, University of Dortmund, VERIMAG, Weizmann Institute, LIF Marseille, University of Twente

The goal of the IST-2001-35304 5th framework project AMETIST (Advanced Methods for Timed Systems) was to advance the timed automaton technology, both in expressive power and computational capabilities in order to be able to handle complex and dynamic scheduling and resource allocation problems. The problem is coordinated by the university of Nijmegen (F. Vaandrager) with Verimag (O. Maler) serving as a scientific coordinator. Other partners include the academics Weizmann, Aalborg, Twente and Marseille and the industrial partners Terma, Cybernetix and Axxom, a company that markets scheduling solutions. The major achievements of Verimag during the project are:

- Development of new algorithms for problems of optimal scheduling (job-shop, precedence graphs); the algorithm use the concept of *non-lazy schedule* to reduce the search space to a finite number of paths

- Using these results within the IF toolset to find feasible schedules for a realistic-size case study provided by the Axxom partners

- New models and algorithms for problems of scheduling under uncertainty, both discrete and continuous; The algorithms find optimal and sub-optimal strategies either by dynamic programming on timed automata or by heuristic search on a game graph

- Development of a series of increasingly more powerful SAT solvers for difference logic as a preparation for doing bounded model-checking on timed automata

15.1.6 CC

`http://www.dii.unisi.it/~hybrid/cc/`
Starting date: January 2002, **Duration**: 39 months
Partners: VERIMAG, ETH Zurich, CWI, Lund University, Parades GEIE, ABB Coroporate Research, EDF, Univ. of Siena

The IST-2001-33520 5th framework project CC (Computation and Control) was the major hybrid systems project in Europe so far. Coordinated by Verimag (O. Maler), the only computer science partner, the project brought together leading teams in the European control arena such as ETH Zurich, Lund, CWI as Siena, as well as several industrial partners (PARADES, ABB, EDF). Within the Grenoble site, we had a genuine collaboration with local laboratories of control (LAG, C. Canudas) and applied mathematics (LMC, J. Della Dora). The major Verimag-related achievements in the project include:

- Extensions of **d/dt** to treat parameters, non-linearities and differential-algebraic equations; development of new abstraction techniques
- Treating case studies coming from the three industrial partners:
  1. Water-level controller for a nuclear plant (EDF); the work on formalizing the requirements for the plant turned out to be very useful in the context of the PROSYD project
  2. Automatic synthesis for an idle-speed controller in a car (PARADES)
  3. Finding stabilizing sequences after a breakdown for a power distribution network (ABB)
- The main contribution of the project to Verimag was in understanding different computational approaches used for solving optimal control problems, such as dynamic programming, bounded horizon optimization and heuristic search;

In the general hybrid systems context, the project made some big steps in reducing the gap between academic control and applications, especially by insisting on the computational content of the results. To quote the final review: *CC has raised the level of science, applications and software tools in the area of hybrid systems and it has laid foundations for further advances in all these aspects of research.*

15.1.7 INTERVAL

`http://www-interval.imag.fr`
Starting date: March 2000, **Duration**: 26 months
Partners: University of Lübeck, VERIMAG, France Telecom R&D, Ericsson, Solinet, Teletel, Telelogic

The aim of the IST-1999-11557 INTERVAL project was to take into account real-time requirements such as timing constraints, during the whole development process of real-time

systems. The project focussed on the definition of timed extensions to existing standardised languages (SDL, MSC and TTCN). These timed extensions have been formalised and submitted to the standardisation bodies at ITU-T and ETSI [BGK+00, 142, DGN02].

Existing tools supporting modelling and validation of these languages have been adapted, prototyped or extended by taking into account the time extensions [139, 142, GH02, HKN01]. New functionalities have been provided for the specification, simulation, validation and testing of real-time systems. The technology has been validated and the impact of the timed extensions on quality and productivity evaluated on two telecom applications, provided by the end-user partners: an embedded MTP2 application and a Multicast Transfer Protocol.

Verimag focussed on the real-time extensions of SDL and the validation of real-time extended SDL specifications with the IF tool (see section 5.2.3).

15.1.8 NEVA

Starting date: January 2005, **Duration**: 48 months
Partners: ACE (NL), Bull (FR), Certess (FR), LETI / CEA (FR), METASymbiose (FR), Philips (NL), Silicomp (FR), STMicroelectronics (FR), TIMA (FR), VERIMAG (FR)

With circuit size potentially reaching one billion transistors by end 2008, traditional bus-based single-clock architectures become unusable for commercial circuits. Starting from successful design approaches (e.g. Multi-Processors, Asynchronous Design) proven during MEDEA+ phase-1, NEVA intends to raise 3 main innovations up to industrial level: communication-centric design for fast simulation and execution, infrastructures for real-time applications, and a complete design flow to implement asynchronous techniques. Datastream applications, mainly from the Video field, will be used as drivers, with a target computing power of 1 GOPS per chip.

15.1.9 Next TTA

`http://www.vmars.tuwien.ac.at/projects/nexttta/`
Starting date: January 2002, **Duration**: 25 months
Partners: Audi, Austriamicrosystems, Budapest University Of Technology And Economics, TTTech Computertechnik, Technische Universität Darmstadt, University Of Ulm, Technische Universität Wien University Of York, Verimag

Objectives: The European IST project NEXT-TTA "High-Confidence Architecture for Distributed Control Applications" concerned the evolution of the Time-Triggered Architecture (TTA) into a full-fledged methodology and tool-chain for the development of distributed control applications, such as those found in the automotive or the avionic industries. For more information check the project's home page.

Verimag role and achievements: The role of Verimag in the project has been to use Lustre as a high-level design language for TTA-based applications. So far, no such language has existed for TTA. Our goal in the project has been to develop an end-to-end tool-chain from high-level design in Lustre all the way down to a running implementation on TTA. In the course of the project this goal has been partially achieved. We now provide a tool-chain which can automatically generate distributed code for a multi-periodic Lustre program on a multi-processor architecture. The processors are assumed to run the OSEKtime operating system, use non-preemptive scheduling, and communicate via the TTP protocol. In particular:

- We have proposed a set of extensions to the Lustre language, in order to facilitate its implementation on a distributed, real-time architecture such as TTA. These extensions can be seen as "pragmas" and do not change the logical semantics of the program. The extensions include primitives to express assumptions on periodicity of inputs and on worst-case execution times, real-time requirements such as deadlines of the form "output must be produced at most x time units after input", as well as primitives to indicate which part of the program should be allocated to which processor.

- We have proposed methods to partition a Lustre program into tasks and schedule these tasks on the multi-processor architecture, respecting the TTP constraints on bus communication. We have also proposed methods to refine the partition based on feedback from the scheduler for cases where a schedule is infeasible. Finally, we have proposed methods to automatically generated "glue code" (interface code) so that the final running system preserves the logical semantics of the original Lustre program.

- The automotive industry largely employs the Matlab/Simulink/Stateflow (TM Mathworks) tool to design control software. Given this context, we have developed methods and tools to translate existing Simulink/Stateflow designs into Lustre (the Stateflow part has been done as part of the follow-up project RISE).

- Our main industrial partner in the project, Audi, was particularly interested in verification techniques, at different levels: verification of design, verification of schedules, verification of distributed run-time systems. We have provided methods to address these verification problems. For design verification, we used our Simulink-to-Lustre translator and the Lustre model-checker Lesar. For schedule and run-time system verification, new techniques and prototype tools have been developed.

- The above have been implemented in prototype tools, to be integrated in the Lustre software distribution. The tools have been used, along with existing tools such as Lustre to C code generators, in a number of case studies provided by Audi. One of these concerned the implementation of a steer-by-wire controller (camera based) for a prototype Audi vehicle. We used the Simulink-to-Lustre translator and a Lustre-to-C code generator to automatically generate code from Audi's Simulink design.

This code was then integrated and run on the prototype vehicle. A demonstration of the system was performed for the reviewers and project partners during the final project review on January 22, 2004 at the Audi testbed in Ingolstadt, Germany.

The activities of this project continue in the follow-up IST project Rise.

Related papers: [157, 158, 248, 256].

15.1.10 OMEGA

`http://www-omega.imag.fr`
Starting date: January 2002, **Duration**: 38 months
Partners: VERIMAG, Grenoble, CWI, University of Kiel, University of Nijmegen, OFFIS, Weizmann Institute, EADS Space Transportation, Fance Telecom R&D, Israeli Aircraft Industries (IAI) and the Dutch National Aerospace Laboratory (NLR)

Verimag was the coordinator of the IST 5th framework project OMEGA. The aim of this project was to increase the efficiency and competitiveness of Software development by providing a framework for tighter integration of software validation into the software development process with the aim of contributing to the reduction of the cost and time of the software development process in the context of real-time and embedded systems.

To this aim, the project has

- Developed a UML profile adequate for the development of real-time embedded systems and usable with some major UML CASE tools, including means for the expression of functional and non functional specifications and requirements of such systems. The consortium has also provided a formal semantics allowing a consistent use of the different notations provided by the profile

- Developed a set of tools, and methods allowing the validation of different aspects, in particular timing related ones, of models adhering to the constraints of the Omega profile. The tools address both checking of internal consistency of specifications and requirements and consistency of specification models with requirements.

- Completed four different experiments of the usage of the Omega profile and tool set on four different case studies addressing different application domains in collaboration between the tool providers and the users and provided methodological support for the usage of the developed profile and tools.

An overview on the results of OMEGA can be found in the following overview papers available on the projects web page [281, 194].

An overview of the contribution of Verimag to OMEGA can be found in section 5.2.3. The main roles of Verimag in OMEGA were beside the project coordination, the definition of the real-time extension of the profile [239, 197, 196, 198, 199, 58, 200], the development of verification tools for this profile [143, 236, 237, 144, 59] and the development of methods and theories for scheduling which has been done jointly with the Ametist

and other projects [212, 211, 275, 85, 151, 41] and components [42, 189, 190]. We have also worked on the evaluation of the developed tools [59, 76, 291] where some additional publications are planned in a near future.

Finally, we have participated in the dissemination of the results by organizing a number of conferences, symposia and workshops [2, 3, 193, 12].

15.1.11 PROSYD

`http://www.prosyd.org/`
Starting date: January 1, 2004, **Duration**: 36 months
Partners: IBM, STM, Infineon, Weizmann Institute, IRC-IRST Trento, TU Graz, Verimag, Accelera The goal of the 6th framework project 507219 PROSYD (Property-based

System Design) is to promote the use of PSL (Property Specification Language) across the semi-conductor industry as the lingua franca for describing system properties throughout the whole development cycle. PSL has been accepted as a standard by the ACCELERA consortium and the projects attempts to improve its usability by various means such as development of methodologies, libraries of properties and synthesis-from-property algorithms. The project is coordinated by IBM with participation of industrial partners Infineon and ST and academics Weizmann, Trento, Graz and Verimag.

The role of Verimag (O. Maler) in the project is to explore the extension of PSL toward analog and mixed signals based on its experience in hybrid systems. The main achievements so far include:

- Definition of STL (signal temporal logic) for defining properties of dense-time real-valued signals and development of a monitoring procedure that checks whether sampled simulation traces satisfy the properties; the procedure has been tested against Simulink-generated traces
- Exhaustive verification of some small analog circuits using **d/dt** and other tools.

15.1.12 RISE

`http://www.esterel-technologies.com/rise/`
Starting date: August 2003, **Duration**: 31 months
Partners: Audi Aktiengesellschaft, GERMANY, Esterel Technologies, FRANCE (coordinator), TTTech Computertechnik Ag, AUSTRIA, CNRS-Verimag

The RISE project, terminated in march 2005, was funded by the EC under IST-2001-38117 contract in the frame of FP5 work programme.

The objective of the project is to provide support for the development of embedded reactive software, in particular for cars, by developing:

- Concepts and methods for the development of such embedded reactive software
- An integrated toolset supporting the concepts and the method, covering the analysis, design and development phases of such embedded reactive software.

It is based on synchronous techniques both for the applications and for the middleware. Three levels of architecture are required for the deployment of such systems:

- A safe communication infrastructure
- Safe computers, on top of the infrastructure
- Safe applicative software, embedded in those computers

The synchronous paradigm is the best-known approach to build reliable, deterministic systems. In this project, the synchronous paradigm is used both for the communication infrastructure and the applications running in the computing units. Computers are beyond the scope of this project.

For the specification and implementation of the applicative software, we use Lustre/SCADE. For the communication infrastructure, the Time-Triggered Architecture (TTA) is suited to high-dependability distributed systems. The TTA is based on the synchronous paradigm, where an autonomous communication subsystem establishes a fault-tolerant global time base and exchanges autonomously and predictably the information among nodes that are connected to replicated communication channels. The foundations of this infrastructure are addressed in the Next TTA project (see section 8.2.2). RISE is complementary of Next TTA: Next TTA focuses on the communication infrastructure (hardware and middleware), while RISE focuses on the development of the application software in the Computing Units, on top of the middleware.

Verimag participated in two tasks: 1) translating imperative features of Simulink/Stateflow into Lustre; 2) studying event-triggered mechanisms and their implementation on preemptive systems (see sections 8.2.1 and 8.2 in the description of the research activities).

The related publications are [156, 248, 247].

15.1.13 SafeAir II

`http://www.safeair2.org`
Starting date: July, 2002, **Duration**: 28 months
Partners: HISPANO-SUIZA (coordinator), IAI, MBDA, REGIENOV, TNI, The Weizman Institute of Science, OFFIS, CNRS-Verimag, Infineon Technologies, CEAT.

The SafeAir-II project, terminated in November 2004, was partly funded by the EC under IST-2001-34363 contract in the frame of FP5 work programme.

SafeAir II aimed to foster the future adoption of the Avionics Systems Development Environment (ASDE) methodology, developed with the SafeAir-I project, in actual real size industrial applications in:

- Defining the adaptation process of industrial ASDE exploitation.
- Connecting ASDE to industry-standard requirements traceability tools,
- Generating test automatically,
- Modifying/Qualifying the process step by step (from the classical “V” cycle to the Enhanced “Y” Cycle) and controlling the result with the preceding step.

Verimag participation was mainly focused on automatic testing: the new version of Lurette (see §7.2.1.1) was designed, and used on real applications [207].

15.1.14 SPEEDS

notyetexisting
Starting date: Mai 2006, **Duration**: 36 months
Partners: Airbus Germany, Airbus France, Daimler Chrysler, Israeli Aircraft Industries, Bosch, INRIA, OFFIS, PARADES, Verimag, TNI, I-Logix, Extessy, Knorr Bremse, Magna Powertrain, SAAB, Esterel Technologies

SPEEDS is an Integrated Project of IST 6th framework programme on "Speculative and Exploratory Design in Systems Engineering".

SPEEDS is a concerted effort to define the new generation of end-to-end methodologies, processes and supporting tools for safety-critical embedded system design that will enable European systems industry to evolve from model-based design of hardware/software systems, towards integrated component based construction of complete virtual system models.

SPEEDS aims at improving substantially the competitiveness of the European industry in this critical economic sector by marrying design competence with deep technical insights and theoretical foundations. SPEEDS partners are companies active in the entire supply chain: OEMs, suppliers, and tool vendors, supported by leading European research institutions.

The technical approach concerns the following topics:

- A framework for system and software modelling adapted for the construction of complex embedded systems by the composition of heterogeneous subsystems. The framework defines "rich-component" models to represent both functional and non-functional aspects so that efficient implementations can be derived from abstract models.

- The modelling framework will have a well defined semantic underpinning for allowing the integration of models provided by different tools and for allowing early model based system exploration

- The project will develop novel formal analysis tools and techniques to assess precisely properties of the system that will allow to explore architectural alternatives of implementation platforms and enable correct-by-construction designs. Compositionality and abstractions will make this approach scalable for large systems.

- tool-supported process controlled speculative design aims at minimizing the risk of concurrent design activities by establishing formal contracts between inter- and intra-company design groups.

15.1.15 TYPES

`http://www.cs.chalmers.se/Cs/Research/Logic/Types/`
Starting date: September 2004, **Duration**: 36 months
Partners: Chalmers University, project coordinator, (subsites: Bergen University, University of Helsinky, University of Stockholm) Université Paris 7, (subsite : Université de Savoie), INRIA Sophia (subsites INRIA Futurs, University of Bologna, Dassault Aviation, University of Minho), Université Paris-Sud (subsites Université de Grenoble, France Telecom), LMU München, TU München (subsites University of Bamberg), Nijmegen University, University of Bialystok, Toyal Holloway University of London (4 subsites of UK) University of Manchester, University of Edinburgh, University of Torino, University of Udine, University of Warsaw, IoS Tallin

The Types project is a coordination action in EU's 6th framework programme. The objectives of the TYPES project are the following:

- develop the technology of formal reasoning and computer programming based on Type Theory
- improve the languages and computerised tools for reasoning
- apply the technology in several domains such as analysis of programming languages, certified software, formalisation of mathematics and mathematics education.

15.1.16 VHS

`http://www-verimag.imag.fr/VHS/main.html`
Starting date: April 1998, **Duration**: 42 months
Partners: Univ. of Nijmegen, Univ. of Aalborg, Univ. of Kiel, Weizmann Institute, CWI, Univ. of Dortmund, Univ. of Ghent, LAG, ETHZ, VERIMAG(coordinator), Nylstar, Krupp-Uhde and Sidmar

The 4th framework LTR project 26270 VHS (Verification of Hybrid Systems), was the first European project on hybrid systems dominated by the computer science approach. Coordinated by Verimag (O. Maler), it had 13 partners, some coming from verification (Nijmegen, Aalborg, Kiel, Weizmann) others from control (CWI, Dortmund, Ghent, LAG, ETHZ), as well as some industrial partners as case-study providers (Nylstar, Krupp-Uhde and Sidmar). The project major goal was to check whether there are some useful applications for the emerging hybrid systems technology. The principal application domains chosen were related to industrial plants (chemical and other). The importance of the project goes beyond its formal results, as it was an opportunity for theoretical computer scientists to see how the "real" world beyond computers looks like. We mention below some of the project achievements, emphasizing those related to Verimag. Major contributors were P. Niebert, S. Yovine, T. Dang and Y. Lakhneche (then at Kiel).

- Modeling using timed automata of several case studies, including an academic chemical batch plant and the Sidmar steel plant, and finding schedules for them. These

works indicated a shift of focus in the use of timed automata from *verifying qualitative* properties of a given system to *synthesizing optimal schedules*. This work triggered a lot of new ideas in applying shortest path algorithms on timed automata to find optimal schedules, work which was continued in the follow-up project AMETIST

- Studying the way controllers for production plants are designed and implemented on programmable logic controller (PLCs). Studying the semantics and programming environments for the IEC programming languages and the development of verification tools for them was a good opportunity to observe different software cultures

- Fruitful exchanges with the control and optimization community and their optimization-based approach to model-predictive control (which is similar to bounded model-checking), which led to the follow-up project CC

- Development of the hybrid verification tool **d/dt**

The project has drawn a lot of attention in the hybrid systems community and after its successful termination, culminating in the special issue [70] two follow-up project were spawned, CC on the continuous control aspects, and AMETIST, dedicated to applying timed-automata technology to scheduling.

15.2 National Projects

15.2.1 ARESA

Starting date: May 2006, **Duration**: 36 months
Partners: France Télécom, LSR-IMAG, Vérimag, TIMA, CITI (INSA de Lyon), Coronis Systems

This project is a RNRT project (French National Research Network in Telecommunications) accepted in september 2005.

Objectives: In this project, we aim at studying, designing and validating new software and hardware architectures, and also new protocols and new organization structures for networks of smart sensors. In particular, we look at:

- new event-based hardware and software architectures optimized for low consumption
- new level 3 protocols that take energy into account
- new protocols adequate for data fusion and data aggregation.
- new network structures allowing auto-configuration, auto-organization and auto-repair
- the needs of a development environment for applications based on sensor networks
- methods and tools for the global modeling and anlysis of sensor networks.
- the validation of all these proposals using a case-study from industry.

Verimag role Verimag is in charge of the methods and tools for the global modeling and analysis of sensor networks. We favor an approach for the formal modeling and analysis of ad-hoc sensor networks, at various levels of abstraction. It is *global* because it takes into account all the following aspects: a precise modeling of the hardware that implements a single node; the protocol layers; the application code; an abstract model of the physical environment as viewed by the sensors. The global model is *executable*, to enable validation by simulations, but we also aim at analyzing the global model with various formal validation tools (automatic test, runtime verification techniques, model-checking and abstract interpretations). Each technique or tool may need particular abstractions of the model.

15.2.2 openTLM

Starting date: mid 2006, **Duration**: 48 months
Partners: CEA-LETI, INRIA, KEESDA, SILICOMP, ST MICROELECTRONICS, THOMSON, TIMA, VERIMAG.

This project is a Minalogic project. It has been accepted in december 2005, and is now submitted for funding.

Objectives: The project is dedicated to the design and development of software embedded in systems-on-a-chip. The Register Transfer Level (RTL) used to be the entry point of the design flow of hardware systems, but the simulation environments for such models do not scale up well. Developing and debugging embedded software for these low level models before getting the physical chip from the factory is no longer possible at a reasonable cost. New abstraction levels, such as the *Transaction Level Model (TLM)*, have emerged. The TLM approach uses a component-based approach, in which hardware blocks are modules communicating with so-called *transactions*. The TLM models are used for early development of the embedded software, because the high level of abstraction allows a fast simulation. TLM is an OCSI (Open SystemC Initiative www.systemc.org) standard. SystemC is the language that has become a de-facto standard for the design of SoCs at the transaction level.

Open TLM aims at providing developers with a integrated development environment for the design and validation of systems-on-a-chip. This environment includes design and debugging tools, simulation tools, and validation tools.

Verimag role In this project, Verimag is in charge of the validation work package. This includes:

- The connection of verification tools to SystemC
- Test and Runtime-Verification techniques for SoCs
- The formalization of the various levels of abstraction that appear in the TLM design flow (pure functional models, then models with timing information, etc.) and the study of the comparison between these levels.

15.2.3 ALIDECS

`http://www-verimag.imag.fr/SYNCHRONE/alidecs`
Starting date: September 2004, **Duration**: 36 months
Partners: VERIMAG, LIP6-Paris, INRIA, Evry-LAMI

This project addresses large size critical embedded systems, for which reuse is becoming crucial. The objective is to study an integrated development environment for the construction and use of safe embedded components. The use of an appropriate programming language being one of the key points contributing to safety in computer systems, we will favour a "language" approach for all the following aspects:
— managing the whole lifecycle, from early specification phases to efficient embedded code;
— modular composition mechanisms, to be used as well for: programs, their specifications, the description of their environments;
— early validation while assembling components;
— early validation of the systems' dynamic behaviour, based on simulations of programs and specifications, all along the lifecycle; executing incomplete programs is one of the key objectives.

The project benefits from the competences of the research teams involved, concerning: synchronous languages and tools; semantics and implementation techniques for embedded software; validation and simulation tools for complex computer systems.

15.2.4 APRON

`http://www.cri.ensmp.fr/apron`
Starting date: September 2004, **Duration**: 36 months
Partners: CRI/Ecole des Mines de Paris (François Irigoin, coordinator), Ecole Normale Supérieure (Patrick Cousot), Ecole Polytechnique (Radhia Cousot) Vérimag (Nicolas Halbwachs), INRIA/IRISA (Bertrand Jeannet)

APRON ("Analyse de PROgrammes Numériques") is a project funded by the French Department of Research within the ACI "Sécurité Informatique". It aims at improving the performances and accuracy of abstract interpreters devoted to the analysis of of numerical properties of programs.

15.2.5 AS "Verification of Quantitative Properties"

`http://http://www-verimag.imag.fr/AS22/`
Starting date: January 2002, **Duration**: 12 months
Partners: LSV (Philippe Schnoebelen, Alain Finkel), LIAFA (Ahmed Bouajjani), IRCCYN (Christophe Mauras), Vérimag (Nicolas Halbwachs, Sergio Yovine)

This project was a "specific action" of the CNRS, devoted to automatic verification of infinte state programs, with an emphasis on quantitative properties: real time systems, systems with counters and numerical variables, hybrid systems, ...

15.2.6 AVERROES

`http://www-verimag.imag.fr/AVERROES/`
Starting date: Period: October 2002, **Duration**: 42 months
Partners: France Telecom R&D and Verimag, project coordinators, CRIL Technology, LABRI Université de Bordeaux, LORIA, PCRI (LIX LRI Université Paris-Sud, INRIA Futurs), LSV ENS Cachan

The objectives of the AVERROES project are the following:

- extend specification and verification technology to quantitative properties, dealing with time, probabilities and complexity
- develop general and specialized tools (model checkers, interactive proof assistants and simulation tools)
- experimentation on embedded systems for telecommunication and avionics

15.2.7 CORTOS

`http://www.lsv.ens-cachan.fr/aci-cortos/`
Starting date: 2003, **Duration**: 36 months
Partners: LSV ENS Cachan (coordinator), IRCCYN Nantes, VERIMAG

The main objectives of the project ACI CORTOS "Control and Observation of Real-Time Open Systems" are to advance our understanding of observation and control problems for timed systems and provide new methods for automatic synthesis of controllers or monitors. Most of the work done in the project is around the models of timed or hybrid automata.

15.2.8 CTRL-a

`http://www-verimag.imag.fr/~altisen/CTRL-a/`
Starting date: September 2003, **Duration**: 12 months
Partners: VERIMAG/synchrone, INRIA Rhône-Alpes/PopArt

Participants: K. Altisen, F. Maraninchi, D. Stauch

This small project was devoted to aspect-oriented programming for reactive systems.

For reactive systems, there are a lot of simple and expressive formal models that can be used, based on traces and automata. Moreover, the main construct in the programming languages of the domain is parallel composition, and the notion of *transverse* modification is quite natural. We proposed: 1) a semantical definition of an aspect, allowing one to study its impact on the original program; 2) some operational constructs that can constitute the basis of a weaving mechanism.

We also studied the relationship between such a weavinf mechanism and general controller synthesis techniques.

Publications: [91, 92].

15.2.9 DYNAMO

http://www-verimag.imag.fr/~iosif/projects/ACI/aci-dynamo.html
Starting date: May 2003, **Duration**: 36 months
Partners: VERIMAG, LIAFA

Verifying concurrent embedded software is a hot research topic. Growing complexity and the often critical character of these systems make necessary the development of methods and tools for ensuring safety operation. The automatic verification of complex software poses many difficult problems, which are out of the range currently available techniques and tools. The sources of complexity are: dynamic data structures, parameters and variables over unbounded domains, concurrency, and dynamic control structures.

DYNAMO aims at developing effective methods to check and analyze in an automatic way programs including combinations of the aspects referred to above. We intend to apply the techniques to verifying JAVA programs, a programming language where all the aforementioned difficulties are present.

15.2.10 EDEN

Starting date: March 2002, **Duration**: 39 months
Partners: Axalto, Trusted Logic (project co-ordinator), CEA-LIST, CEA-LETI and VERIMAG

The objective of this project is to develop a methodology with tool support for the devlopement of application cetified at the highest assurance level of the Common Criteria.

15.2.11 EVA

http://www-eva.imag.fr/
Starting date: November 2000, **Duration**: 36 months
Partners: Trusted Logic (co-ordinator), LSV ENS Cachan, VERIMAG

The objectives of the EVA project are the following:

- Design a specification language for cryptographic protocols with a rigorous semantics.
- Develop a repository of security protocols. This lead to SPORE, the Security Protocols Open Repository.
- Develop verification tools for cryptographic protocols.

15.2.12 EXPRESSO

http://www.telecom.gouv.fr/rntl/FichesA/Expresso.htm
Starting date: May 2001, **Duration**: 24 months

Partners: Aonix, EDF, INRIA/IRISA, Silicomp, Thalès, VERIMAG

The objective of the Expresso project was to develop a methodology and the tools associated which will make it possible to carry out real time applications for critical embedded systems likely to be certified compliant to safety standards such as DO178-B.

15.2.13 MADEJA

Starting date: October 2003, **Duration**: 36 months
Partners: Silicomp,VERIMAG

Garbage collection (GC) is not used in real-time embedded systems. The reason is that temporal behavior of dynamic memory reclaiming is extremely difficult to predict. Several GC algorithms have been proposed for real-time embedded applications. However, these approaches are not portable (as they impose restrictive conditions on the underlying execution platform), do require aditional memory, and/or do not really ensure predictable execution times.

An appealing solution to overcome the drawbacks of GC algorithms, is to allocate objects in regions which are associated with the lifetime of a computation unit (typically a thread or a method). Regions are freed when the corresponding unit finishes its execution. This approach is adopted, for instance, by the Real-Time Specification for Java (RTSJ). These region-based approaches define API's which can be used to explicitly and manually handle allocation and deallocation of objects within a program. However, care must be taken when objects are mapped to regions in order to avoid dangling references. Thus, programming using such API's is error-prone, mostly because determining objects' scope is difficult.

The MADEJA approach consists in *automatically* transforming a program so as (a) to replace (whenever possible) allocations by calls to the region-based memory allocator, and (b) to place appropriate calls (i.e., guaranteeing absence of dangling references) to the deallocator. Such an approach requires to analyze the program behavior to determine the lifetime of dynamically allocated objects.

15.2.14 MASH

`http://www-verimag.imag.fr/~/projects/mash`
Starting date: November 2000, **Duration**: 24 months
Partners: VERIMAG, LMC

MASH aimed at a fundamental study of the basic mathematical techniques for modeling and analysis of hybrid systems. On the one hand, it was a question of studying the fundamental problem of the existence of trajectories of a hybrid system using the framework of computer algebra and symbolic computation. In particular, we studied the relationship between hybrid automata and differential-alebraic inequations. In addition,

the project aimed at the study and the development of algorithms for computing, exactly or approximately, the set of reachable states, as well as the associated decidability questions. The idea was to exploit the geometrical properties of the dynamic systems considered.

15.2.15 OpenEmBeDD

`http://openembedd.inria.fr`
Starting date: April 2006, **Duration**: 36 months
Partners: Airbus, Anyware Technologies, CEA-List, CS, France Telecom, INRIA, LAAS, THALES, Verimag.

The OpenEmbeDD project is partly financed by the French government (National Software Engineering Research Network - RNRT).

The aim of the project is to build an open source platform under Eclipse implementing the principles of model driven engineering (MDE) for embedded systems of modelling standards based on the synchronous and asynchronous paradigm, allowing also mixed approaches. It represents a continuation of an effort started in the CARROLL initiative towards a free and open source software engineering framework for embedded systems.

The MARTE UML profile for Modelling and Analysis of Real Time and Embedded systems which is being standardised at OMG, will be the modelling standard adopted.

Verimag will participate in the definition of an executable formal semantics for the chosen modelling standard in terms of a framework for the composition of heterogeneous components developed at Verimag.

Verimag will also connect validation tools, in particular the IF platform for the verification of functional and non functional properties and apply them to the case studies provided by the industrial users of the project.

15.2.16 PERSIFORM

`http://www-persiform.imag.fr`
Starting date: November 2004, **Duration**: 24 months
Partners: France Telecom, Verimag, IRISA (Rennes) et INT (Evry).

The PERSIFORM project is partly financed by the French government (National Telecommunication Research Network - RNRT). The aim of the project is to develop an integrated methodology, supported by tools, allowing to take into account performance-related issues (and particularly issues related to architecture and resources) very early in the design of complex systems.

The project builds upon several technological foundations. There are the methods of performance engineering. We focus particularly on techniques that provide the best cost/result ratio for large systems, that is stochastic modeling and analysis by intensive discrete simulation. The other important foundations are the formal and semi-formal languages used in functional modeling and analysis, like UML, SDL or MSC. The project aims to bridge the gap between functional and performance models, by allowing a unified,

model-centered approach in which aspects pertaining to these two types of models are seamlessly put together, generated from one another, or verified for coherence.

Concretely, the research directions of PERSIFORM are:

- The identification of the data (at the right abstraction level) relevant for performance analysis and dimensioning, and their integration in the form of annotations into a functional model.
- The definition of a set of functional modeling notations to be used with the PERSIFORM methodology and their semantics. These notations will constitute a specialized profile of an existing modeling language such as UML. The coherence between different modeling styles and views provided in the profile will be studied, where we will consider a workflow view and a per component design view.
- The definition of model transformation methods allowing the generation of performance simulation models from annotated functional models. For some types of views the transformation is not trivial, since it requires a change of perspective (e.g., from a modular behavior description to a global workflow description) and a change of abstraction level (necessary for efficient discrete simulation).
- The definition of a methodology and of a set of tools supporting editing of annotated functional models, transformations to performance models, simulation (using existing commercial simulators like HyPerformix Workbench) and interpretation of results.

15.2.17 POTESTAT

`htpp://www-lsr.imag.fr/POTESTAT/`
Starting date: September 2004, **Duration**: 36 months
Partners: Landes IRISA, Vertecs IRISA, LSR-IMAG, VERIMAG

POTESTAT is a three-year public research action (*ACI sécurité*), partially funded by the French government, and gathering five academic teams.

The aim of this project is to study the conformance testing of an open service implementation (smartcard, distributed software, or heterogeneous network) with respect to a given *security policy*. Most of the time the security of such systems is organized from pragmatic elements, based on well-known vulnerabilities and their associated solutions. Verifying if such security policies are correctly and effectively implemented in the actual system is therefore an important issue. The current practice usually consists in auditing the administrative procedures and the system configuration. However, even if some tools are already available for specific tests (like password crackers), there is no effective approach to correctly analyze the whole system conformance.

In the framework of the POTESTAT project we plan to tackle this problem according to the following items:

- Formal modelization of security policies, allowing a *test directed* analysis.

- Definition of a conformance notion between a system configuration and some security policies elements. The goal is to obtain a test theory similar to the one existing for instance in the protocol testing area (like the Z.500 norm).

- Definition of methods to test this conformance notion, including the testability problems, the execution environment, code analysis and test selection.

15.2.18 PROUVE

`http://www.lsv.ens-cachan.fr/prouve/`
Starting date: November 2003, **Duration**: 36 months
Partners: CRIL Technology Systèmes Avancés, France Telecom R&D, LSV ENS Cachan, LORIA Nancy, VERIMAG.

The objectives of the PROUVÉ project is to develop new results and tools for the verification of security protocols that go beyond secrecy and authentication.

15.2.19 ROSSIGNOL

`http://www.cmi.univ-mrs.fr/~lugiez/aci-rossignol.html`
Starting date: December 2003, **Duration**: 36 months
Partners: LIF Marseille, INRIA Futurs (LIX and LSV ENS cachan) and Verimag

The objective of this project is to study the theory of the semantics and verification of security protocols. In particular, it focusses on equational theories, the relationship between formal and complexity-theoretic semantics of cryptograhic protocols and probabilistic aspects of these protocols.

15.2.20 SODA

Starting date: October 2000, **Duration**: 36 months
Partners: VERIMAG, France Telecom

Embedded real-time systems are subjected to strong timing constraints whose non-observance can involve the violation of the requirements of safety, safety and reliability. They are thus critical systems as well by the consequences of their use as by complexity of their design, deployment and evolution. Real-time embedded system's development is characterized by a very strong integration of software components. The composed system is thus constrained by the individual requirements of each component, plus the requirements of synchronization imposed by integration itself and the environment of execution. This poses a certain number of challenges, in particular schedulability, which requires from now on a methodology making it possible to connect, in a integrated manner, since the design phase, the behavior of the system at the functional level, with the non-functional aspects relative to its implementation and execution.

The objective of the project is to develop the theoretical and practical tools to synthesize schedulers in a systematic way. With this intention, we propose an original and innovating approach, based on a compositional methodology allowing to build a model of the system including: the behavior of the applicatication, annotated with information concerning the execution time of the components, the behavior of the environment of execution, the synchronization constraints, and the scheduling policies.

15.2.21 TAXYS

`http://www-verimag.imag.fr/~yovine/projects/taxys`
Starting date: September 1999, **Duration**: 36 months
Partners: Alcatel Business Systems (ABS), VERIMAG, France Telecom

The basic objective of the project is to develop a programming environment aimed at real-time embedded applications on single-processor architectures. The main feature of this environment is to lead to structurally secure implantations with respect to time constraints. Solving this question implies going through a measurement of the processing time requested by each application. The ultimate ambition of the project is to extend the basic objective to validating the coexistence, on one single-processor machine, of several individually validated applications (composing). This extension will provide answers to the access control issues under QoS guarantee constraints, which are one of the technical roadblocks to the emergence of upgradeable multi-service terminals.

15.3 Industrial Contracts

15.3.1 Airbus 2002

Dates: 2001,2002

The goal of the contract was to study and experiment the extensions of Lustre/Scade (mainly arrays, see §6), in view of case studies coming from the flight software of the A380.

15.3.2 Airbus 2004

Dates: 2003

Goal: Supplying the testing tool Lurette, with support on case studies.

15.3.3 ANACONDA

Starting date: October 2003, **Duration**:
Partners: VERIMAG, STMicroelectronics

Principal Investigator Sergio Yovine
Participants Christophe Rippert (MC), Ismail Assayad (PhD student), Olivier Quévreux (Research ingeneer), Fran cois-Xavier Défaut (Research Engineer)
Funding STMicroelectronics

The ANACONDA 15.3.3 project is a VERIMAG-STMicroelectronics joint effort to develop a theoretically sound and automated model-based design, analysis, and implementation tool-chain for generating correct-by-construction code for embedded multi-processor architectures.

15.3.4 ADHOC

Starting date: October 2004, **Duration**:
Partners: VERIMAG/synchrone, VERIMAG/DCS, FranceTelecom R&D, LSR-IMAG Grenoble, TIMA Grenoble, INSA Lyon

Principal investigators: F. Maraninchi, L. Mounier
Participants: F. Maraninchi, L. Mounier, L. Samper

The Adhoc project is a cooperation between FranceTelecom R&D and several laboratories in Grenoble, on ad-hoc sensor networks.

The main tasks are the following:

- How to build small network nodes guaranteeing low consumption: this part is investigated by the a team of the TIMA laboratory working on asynchronous circuits (CIS, M. Renaudin)
- How to design efficient protocols: this part is investigated by the a team of the LSR laboratory working on networks (Drakkar, A. Duda)
- How to organize the network into groups of nodes, in such a way that the protocols are more efficient (INSA Lyon)
- How to understand the global behaviour of a sensor network, and the interactions between level of abstractions (the node, the various levels of protocols, the application) that have to be taken into account in order to guarantee an efficient behaviour. This part is investigated by Verimag.

15.3.5 Esterel Technologies

Dates: 2005

Goal: Expertise and evaluation of the specifications of Scade-V6 (including arrays and state machines).

15.3.6 SocTLM

Starting date: July 2002, **Duration**:
Partners: VERIMAG/synchrone, STmicroelectronics, Software Platform Group

Principal investigator: F. Maraninchi
Participants: F. Maraninchi, M. Moy, C. Helmstetter, J, Cornet, M. Shahbaz.

The SoCTLM project is part of the strong cooperation between Verimag and STmicroelectronics. It is devoted to the modeling and analysis of Systems on a chip at the so-called *transactional level* (TLM).

In this project, we develop the tool Pinapa (a SystemC front-end) and various connections to verifications tools (Leasr from Verimag, SMV, ...). We also study runtime verification techniques and testing. See chapter 11 for more details.

Related publications: [233, 234].

15.3.7 FVTA

Starting date: July 1, 2002
Termination: July 1, 2005

This is a research grant from Intel to Verimag (O. Maler) entitled "Bridging the Gap between Formal Verification and Timing analysis". Within this project we develop new techniques for analyzing circuits with bi-bounded delays. The first class of techniques is based on abstraction and hiding of internal clocks, first for combinational circuits whose input changes only once, and later for "open" systems with inputs that can change any time. The other class of techniques is based on extended SAT with timing constraints. The project mentors from Intel are Avi Efrati and Ken Stevens. The project also gave an opportunity to visit Intel labs in Haifa and discuss with various groups doing formal verification, timing and power analysis, analog design and instruction scheduling.

Part VI

Summary and Perspectives

Chapter 16

Summary

Over the past four years, VERIMAG has grown considerably by nearly 50%, from a total of 49 in 2001 to 72 in 2005, including 6 new permanent members. Our vision for embedded systems has guided us in selecting new work directions. VERIMAG has been a driving force for the emergence of embedded systems as a discipline in Europe. It has been strongly involved in the creation of the Artist2 Network of Excellence, which federates leading European researchers into a common research and dissemination work programme (http://www.artist-embedded.org/FP6/).

16.1 Research

Our scientific policy includes two main components:

16.1.1 Strengthen and Extend Existing Background and Competencies

Leveraging on existing strengths, VERIMAG has pursued coverage of the entire development lifecycle for embedded systems. The synchronous approach, which initially focused on the LUSTRE language and its validation, now covers aspects such as specification of synchronous systems, reactive components, as well as distributed implementation techniques.

Our validation techniques for asynchronous specification languages, are now accessible to higher-level languages such as UML and SDL, due to the successful development of the IF format and its associated tools. IF bridges the gap between domain-specific user languages and state of the art validation techniques developed at VERIMAG.

Our work on timed and hybrid systems has found numerous applications, including real-time systems modeling, scheduling and schedulability analysis, testing for real-time systems, modeling of digital and analog circuits and control design.

16.1.2 Opening to New Research Directions

VERIMAG has invested considerable resources into the development of new research directions.

Our work on security, started four years ago, has seen rapid development and has acquired a good level of visibility, through publications in high-standard conferences and journals (e.g., CAV, Fossacs, ESOP, JLAP), as well as involvement in more than five R&D projects. Highlights of recent results include the relationship between the Dolev-Yao and the complexity-theoretic models for cryptographic protocols and development of the Hermes tool. It is important to emphasize that these results have been obtained with a small team of researchers, consisting of only two permanent members until September 2004, when a third professor joined the group.

Activities on components have known rapid development, in both theoretical and applied work, mainly through tight collaboration with industry. Our strong involvement in the IST Omega project has raised a number of interesting theoretical perspectives and research directions for component-based descriptions of real-time systems. Thanks to the strong cooperation with STMicroelectronics since 2002, on the modeling and analysis of systems-on-a-chip at the transactional level, we identified a set of theoretical problems related to the component-based development of SoCs: hardware and software components, several levels of granularity. We started to develop a set of tools around SystemC to study these problems on industrial case-studies.

The results on new implementation techniques for synchronous languages significantly broaden the application domain of synchronous programming: we are able now to generate code for distributed architectures, and to handle multicyle systems and urgent sporadic events, properly. An important achievement of our research effort on model-based and correct-by-construction implementation approaches has been the development of pre-industrial compilation chains for generating code without semantic breakdowns.

In timed and hybrid systems, new application domains have been investigated, most notably analog circuits, but also planning for autonomous robots and systems biology which could benefit a lot from hybrid analysis methods. In terms of techniques, we invested a lot in "lightweight" alternative for formal verification, such as random and guided search in the space of trajectories and automatic construction of property monitors for Boolean and analog signals. Another new technical domain is SAT modulo theories where we have developed solvers that compete favorably with the state-of-the-art.

Work on fine-grain QoS control has produced very interesting experimental results in tight collaboration with a team from STMicroelectronics. It also provides a solid theoretic framework for studying adaptive systems.

16.2 Transfer and Industrialization

Our policy for transfer and industrialization is implemented through long-term partnerships with leading industrial players in the area. We note that our collaboration with Airbus, which was initially very strong and productive, has drastically decreased. The reason for this seems to be that SCADE/Lustre is now a mature technology, and no

longer requires strong support from VERIMAG. Another reason may be the organization of research activities in France on a regional basis. Regionalisation can partially explain the spectacular development of our relations with STMicroelectronics. These were inexistent 4 years ago. Today STMicroelectronics is VERIMAG's main industrial partner. A joint research laboratory was set up lest year (see next section). The research topics include modeling with SystemC and TLM, property-aware compilation and fine-grain QoS control. Consistent development of tools and technology, integrating research results according to a long-term research vision, is an important element of our R&D strategy. It is a means for attracting industrial collaboration and being accepted into top R&D projects. Our industrial relations are structured mainly around 2 toolsets, which are the results of continuous investment for more than ten years: the Lustre/Scade toolbox (See chapter 2) and the IF toolbox (See Sections 5.2 and 7.1.1).

16.3 Visibility and Infrastructure

In this section, we summarize the recent activities dedicated to make the laboratory a major actor in its domain, and to provide it with the infrastructures needed for its development.

16.3.1 The European Network of Excellence ARTIST

VERIMAG is the scientific coordinator of the ARTIST2 Network of Excellence (`http: www.artist-embedded.org/FP6/`), about the design of Embedded Systems. The strategic goal of this network is the emergence of a stable European research community in this multidisciplinary domain. This community is organised in "clusters" dedicated to 6 main topics: Modelling and Components, Hard Real-Time, Adaptive Real-Time, Compilers and Timing Analysis, Execution Platforms, Control for Embedded Systems, Testing and Verification. Today, 36 European teams and more than 200 researchers are working within ARTIST2.

The ARTIST2 network is working as a European virtual laboratory. Its mi-term goal is to become a competitor of large US centers (Berkeley, Stanford, MIT, Carnegie Mellon), both concerning the production and transfer of scientific knowledge and competence, and concerning the impact on industrial innovation.

The different teams are integrated around a common activity programme, which aims at strengthening the critical mass within the selected teams. ARTIST2 is made of a double kernel, made of (1) ambitious research activities directed towards the progress of the state of the art, and (2) connected development activities, consisting, for instance, in developing platforms around research activities.

ARTIST2 acts also in favor of the dissemination of knowledge, by organizing schools, workshops, and technical meetings open to a large attendance. An internet gateway will be open soon, which should become the main source of information about the evolution of techniques devoted to the design of embedded systems. This gateway will broadcast informations concerning standards, books, conferences, research papers, etc.

ARTIST2 is playing an important role in structuring the European community of research on Embedded Systems, and especially in launching the European platform ARTEMIS in the framework of the 7th PCRD, and in setting up a European association for Embedded Systems in cooperation with the Hycon and HiPEAC networks of excellence.

16.3.2 The Center for the Technologies of Software (CTL)

VERIMAG was the main promoter of the CTL project, conducted in the framework of the 12th "Contrat de Plan Etat-Région", with the support of CNRS for the first equipment. The goal is to build 1200 m^2 of building on the site of Gieres, to host activities for the valorisation and transfer in the domain of Software. The total cost of the project is around 3 millions euros.

Since November 2005, CTL hosts two platforms: 1) the MultiCom platform, managed by the CLIPS laboratory, which concerns innovative projects in man-machine interfaces; 2) the platform "Systèmes et Logiciels Critiques" (PSLC), animated by VERIMAG, which concerns innovative projects in cooperation with industry. PSLC hosts teams of the joint laboratory with STMicroelectronics.

The CTL offers also premises for meetings and workshops, in particular a 100 places amphitheatre, and a video-conference room.

16.3.3 The joint laboratory with STMicroelectronics

During the last four years, STMicroelectronics (ST) has become VERIMAG's first industrial partner. In September 2005, this partnership between VERIMAG and ST gave rise to a joint laboratory in the domain of embedded software. The laboratory is made of more than 15 people (researchers, engineers, post-docs and PhD students), distributed among 4 common projects:

- Automatic generation of controlled software;
- SoCTLM, construction and validation of Systems-on-a-chip at the transaction level;
- ANACONDA, automatic synthesis of application software and systems for multiprocessor, ensuring by construction the given time, memory, and consumption requirements;
- Development of a highly optimized code support for embedded software debugging.

Beside the specific funding of projects, the joint laboratory receives from ST a basic yearly funding of 100 Keuros.

The activities of the projects of the joint laboratory are integrated in the project "Atelier du Futur" of the "pôle de compétitivité" Minalogic, and in the RNTL project Galogic.

16.3.4 The "Pôle de Compétitivité" Minalogic

Verimag played an active role in setting up the EmSoC (Embedded Systems on Chip) cluster of the "Pôle de Compétitivité" Minalogic (http://www.minalogic.org/). The latter

is a regional coordination structure for R&D in nanoelectronics and embedded software. It gathers together more than 52 partners including large companies, PMEs, academic institutions and local authorities. It has been granted by the French government the label of world-class importance.

Verimag has actively contributed to the structuring of the embedded software community as the chair of the EmSoC's "Operational Committee". This includes 2 academic and 4 industrial members. The role of this committee is the coordination of industrial and academic research teams for the definition of innovative R&D projects as well as the evaluation of project proposals. Successfully evaluated projects get a label giving access to support from national R&D Programmes. So far, 7 projects have been labeled and 4 received support.

16.3.5 ETAPS 2002

In 2002, VERIMAG organized the "European Joint Conferences on Theory and Practice of Software", ETAPS'2002 (`http://www-etaps.imag.fr/`). Since 1998, ETAPS takes place every spring, with an increasing success. It is made of 5 main conferences, and many satellite events, in the domain of the foundations of Software Engineering.

In 2002, ETAPS was very successful, with 13 workshops (1-3 days each), 9 tutorials (1 or 1/2 day each), and more than 600 attendants, from the whole world.

16.3.6 Awards

- Joseph Sifakis received the "silver medal" of the CNRS.
- Paul Caspi and Nicolas Halbwachs received in 2004 the prize Michel Monpetit of the French Academy of Science.

Chapter 17

Perspectives

17.1 General Perspectives

We intend to preserve a good balance between basic and applied research, in tight collaboration with selected industrial partners. It anticipates evolutions in the organization of research on a regional basis. VERIMAG is actively involved in local initiatives such as the "Pôle de Compétitivité" Minalogic federating R&D in Nanotechnology and Embedded Systems on Chip. The laboratory coordinates the academic activities of the EmSoC (Embedded Systems on Chip) cluster within Minalogic.

VERIMAG is also actively involved in defining a new embedded systems curriculum and in implementing it in an international five year Master's Degree.

Excellence and relevance in research are VERIMAG's founding principles. We will continue our effort for better integration of existing activities, by encouraging synergy between teams and approaches. A main objective is the consolidation of the new topics and their extension for better coverage of the development lifecycle. Below is a non-exhaustive list of work directions:

- Work on components will be continued to further explore theoretical aspects related to heterogeneity. Bridging the gap between synchronous and asynchronous computation paradigms is now a feasible objective, due to our strong background and existing work in both fields. On the application side, we will pursue our work on component-based modeling by using SystemC and TLM, the IF toolbox and its extensions. We will apply our results on new applications such as sensor networks, RTOS and middleware modeling.

- An important milestone in our work in security is the transfer of our methodology and tools towards the development of certified applications. From the theoretic point of view, we plan to study a semantic model and its associated verification techniques, that take into account physical attacks. In particular, we would like to develop a symbolic verification framework that takes into account these attacks, and to study its soundness and completeness with respect to the computational model for cryptography.

- We plan to study methods for building adaptive systems which meet both safety-critical and best-effort properties. So far, our work mainly focused on correct-by-construction implementation, handling timing properties, and homogeneous architectures. A major challenge is to automatically synthesize implementations that also comply with memory and energy constraints, as well as heterogeneous architectures. Indeed, one of the most important advantages of model-based implementation is the ability to perform early prototyping and simulation. This approach paves the way to developing architecture exploration methods, thus achieving optimal deployment.

We plan to continue our policy of leveraging on industrial collaboration to obtain the resources necessary for sustaining research. This policy relies on a selected set of strategic alliances with companies such as STMicroelectronics, France Telecom, and Schneider Electric. It will take into account the organization of research on a regional basis, as well as initiatives such as Minalogic and its EmSoC cluster.

We expect that the joint research lab with STMicroelectronics will be renewed. In this context, we will continue existing projects and will try to set up new ones on topics such as RTOS and middleware component-based modeling, as well as modeling and analysis using UML in conjunction with the IF toolbox. For this toolbox, there are also concrete plans to develop an open source version, specialized for voice and convergent services, in collaboration with France Telecom.

In the framework of the restructuring of reserach laboratories requested by the CNRS, we first tried, at the beginning of 2005, a fusion with the TIMA laboratory, in order to build the kernel of a "big laboratory" in the domain of embedded systems. Because of the connections between TIMA and some laboratories in micro-electronics, and in accordance with the requests from the CNRS, this project evolved then towards a "very big laboratory" ranging from micro- and nano-technologies to embedded software. This project was driven till an advanced stage, before encountering serious obstacles — health problems of the planned director, strong opposition of the CEA, diffulties for reaching some internal consensus —, and was finally given up, at the beginning of 2006. So, VERIMAG will continue as it is, at least for the next period.

An important challenge for this next period, is that, in spite of its modest size, VERIMAG play its whole role within the new environment of "big laboratories". We will cooperate with other laboratories working in Information Technologies in Grenoble, to set up a coordination aiming at building a consistant and clear image of this community. On the other hand, we will continue our animation activities around the domain of embedded systems, at the local level (EmSoc cluster of the "pôle de compétitivité Minalogic), the regional level (regional project EmSoC-recherche), and the European level (ARTIST2).

17.2 Perspectives of the "Synchrone" team

In the "Synchrone" team, we will pursue the ongoing redirection of work, from the study of a particular language and its specific application domain, towards a wider use of the synchronous methodology and tools for the development of embedded components. The

existing industrial relations (Airbus, STMicroelectronics, FranceTelecom R&D, etc.) will be continued.

17.2.1 Research directions

Apart from the basic work around the LUSTRE toolbox, the mid-term research directions are the following:

- Using LUSTRE/SCADE as a *development method* for embedded applications on various architectures: the most recent projects showed that LUSTRE is more than a programming language. Now, it can be realistically considered as an actual development method, from a high level description with clean semantics, to various implementations, whose correctness can be proved with respect to the initial specification. On the other hand, languages that become standard, like Simulink/Stateflow or SystemC , can now be translated into LUSTRE, in order to take advantage of the associated tools.
- Verification tools and methods: the work on the kernel of validation tools will be continued. For data-dominated programs, the approach based on interactive techniques must be further explored, since the actual problems are often not tractable by fully automatic means. For control-dominated programs, the main data types are Booleans, and, e.g., counters. Dedicated techniques must be improved. Finally, the use of LUSTRE as a modelling language raises new verification problems concerning, for instance, the synchronous modelling of asynchrony and non-determinism. Mixed enumerative/symbolic techniques can be considered.
- Formal methods and tools for modelling, early execution and simulation of heterogeneous systems: this topic is a longer term perspective. We intend to use the team's expertise in dedicated languages, implementation techniques, models of concurrency and time, and validation techniques. The goal is to define methods and tools connecting industrial tools aiming at rapid simulation, and semantically sound formal methods.
- Components for embedded systems: the ongoing researches concerning the notion of embedded component will be enhanced. The short-term perspectives concern two topics that must be considered together: 1) the proposal of a simple mathematical model for heterogeneous components; 2) the study of actual contexts in which the notion of components plays a central role. The confrontation of these two directions should permit the identification of interesting sub-models, and the understanding of the conditions allowing formal validation before deployment.

Concerning the new recently-adressed application domains, the perspective are the following:

- Modeling and analysis of Systems on chip: the formalisation of our relations with STMicroelectronics, together with the extension of our cooperation with Silicomp-AQL, allow us to consider this application domain as a long-term reserach topic. For the Synchrone team, it is an ideal cooperation framework to improve our knowledge of the general domain of model-driven development. We are expecting support for theses on subjects like component-based formal validation, or formal criteria for

comparing different levels of abstraction. The success of the first CIFRE thesis with STmicroelectronics gives us good perspectives for such cooperation on relatively theoretical subjects.
In the near future, the work will continue in the framework of the openTLM project of the "pôle de compétitivité" Minalogic, approved in December 2005, and submitted for funding in February 2006.

- Modeling of ad-hoc sensors networks: The new project ARESA (RNRT, starting in 2006) makes possible the continuation of our work on this topic for at least 3 or 4 years. After a first exploratory year, we progressively constructed a global modeling approach, as our understanding of the domain progressed.
  A first track to be explored concerns a formal framework for modeling/abstraction/validation, in which such a complex hardware/software system as a sensor network could be modeled by a component-based structure (e.g., the air, the environment, the hardware part of the nodes, the software part, the protocols, ...). For each global evaluation (e.g., energy consumption), the possible component abstraction should be identified, giving worst-case evaluations. Then, the replacement of a component by a more abstract one, in the global model, should ensure a global worst-case evaluation.
  This approach applies to a more general domain than that of sensor networks, but this particular domain is very rich and deserves a careful study, for a good understanding of the needs concerning the global modeling.
  Interesting connections exists with the so-called "transaction-level modeling" of systems on chips. Small grain simulators of sensor networks will promptly encounter complexity problems which are well-known in integrated systems: simulating 1000 nodes at processor RTL level is too expensive in time. The hardware industry developed more and more efficient simulators, by raising the abstraction level of the simulation. We think that exhaustive simulation of sensor networks is hopeless if it does not rely on an abstract formal framework making such abstractions possible.

17.2.2 New projects

- **Proofer**
  Partners : RATP, Alcatel, Prover Technology and Verimag.
  The Proofer projects aims at automatically proving railway switching systems installed by Alcatel on the RATP network. Although such verification have already been performed in the past, they were always partial in the sense that they only concern safety invariants resulting from an informal failure analysis. Such proofs did not need any modeling of the environment. Here, the goal is a complete industrial verification.
- **ARESA**, Embedded systems and sensor networks
  Partners : France Télécom, LSR-IMAG, Vérimag, TIMA, CITI (INSA Lyon), Coronis Systems.
  The goal of the project is to study, design and validate new hardware-software architectures, and new communication protocols, for intelligent sensor networks. A

consequence should be the production of networks that are more open and easier to use by a user equipped with a mobile terminal. More specifically, the project aims at:
  - exploring new event-based software and hardware architectures, optimized for a very low energy consumption;
  - proposing new protocols at level 3, taking into account energy consumption;
  - designing new applicative protocols, dedicated to data fusion and agregation;
  - studying new network structures, allowing self-configuration, self-organization, and self-repairing;
  - studying the needs of a development environment for applications based on sensor networks, and providing global modeling and validation tools;
  - validating all these proposals on an applicative scenario.
- **openTLM**, Partners : CEA LETI, INRIA, KEESDA, SILICOMP, ST MICROELECTRONICS, THOMSON, TIMA, Verimag.
  This project conserns the design and development of embedded sofware on systems-on-chips (SoCs). While technology progress makes possible the construction of faster ans faster chips, containing more and more components, an important bottleneck nowadays is the designers productivity. A new level of abstraction, called transaction level modeling (TLM), has been standardized by the OSCI (Open SystemC Initiative www.systemc.org) organization, based on a proposal by ST Microelectronics.
  Open TLM aims at providing these designers with a toolbox for developing and validating these models. This toolbox should propose design tools, a simulation environment, and verification tools at transaction level.

17.3 Perspectives of the "Distributed and Complex Systems" team

Over the course of the last quadriennal (4-year) period, the DCS team has opened up major work in three new research directions. During the four coming years, the priority will be to reinforce these. Another objective will be to continue the transfer of the IF environment. Discussions are currently underway to incorporate it into a commercial offering. The main research directions that the DCS team will pursue in the medium term are:

17.3.1 Methods and tools for Component-based Adaptive Real-time Engineering

Work on components will continue in existing directions: development of the theoretical framework and implementation of the supporting tools for the BIP platform. New work on the adaptivity of component-based systems will be implemented. This will extend existing results already obtained in collaboration with STMicroelectronics, on controlled quality-of-service multimedia systems. These results have led to the implementation of a prototype that takes as input both a model of the application code, and a model of the

target processor, and generates a controlled implementation with dynamic optimization of the data processing, while also preserving the temporal deadlines for the application actions (Theses of Jacques Combaz and Loic Strus). Here a brief description of the work directions :

Theoretical Framework: We will formalize this framework by a *component algebra* whose terms are used for the description of components BIP. The operators of this algebra satisfy particular properties such as incrementality which are not generally satisfied by operators in process algebras. The algebra uses two classes of operators: connectors and priorities. For each of these two classes are defined specific calculii making it possible to simplify and compose operator expressions. Composition of the components is expressed in terms of operations on the connectors and on the priorities.

This formalization sould allow the proposal for a taxonomy of real time systems. We will study in particular the relationship between timed and untimed systems, various execution models (such as synchronous and asynchronous) and various models of interaction. In addition, this formalization leads to a concept of expressivity which characterizes the structure of the systems that we will study.

In the longer term, we will introduce into BIP concepts and constructs to describe dynamic structural changes, in particular the dynamic creation and destruction of components, as well as mobility.

BIP Platform: We will continue ongoing work to develop an execution platform for real-time systems. If the results are positive, this platform will be used in parallel with the IF platform (dedicated to simulation/validation) within three projects on the components which now starting: IST/SPEEDS, RNTL/OpenEmbedd, ITEA/SPICES. The expected results are the creation of an editor for BIP and the development of code generators for the platform and THINK code. Connection with THINK must enable us to generate implementations for various target platforms.

Adaptive Components: Adaptivity is the capacity for systems to modify their behavior according to dynamic changes in their execution or external environments, to satisfy certain properties. This is considered as the means for guaranteeing by construction the essential properties of systems relating to resource management and robustness. Indeed, it is very difficult to make systems' behavior predictable both because of their structure as well as because they are open to their environments, which have rich and unforeseeable dynamics. Design methods based on hypotheses of worst case operation, are prohibitively costly for embedded systems, integrated in consumer products (eg: mobile phones).

The work studies and implements methods for constructing systems from adaptive components. The results already obtained are very promising. They make it possible to satisfy safety and optimality constraints by controlling quality parameters for multimedia systems (eg: algorithms for video encoding). Control is carried out by a hierarchical controller composed of a sequencer and a quality manager that cooperate to determine, for a given system execution state, the next action and its quality. The experimental

results are very interesting. The ratio 'overhead due to control' over 'quality improvement' remains very small.

Our work has the following goals:

- Studying a general framework for systems based on adaptive components which generalizes in several directions the results obtained, in particular the symbolic characterization of the quality manager via a precalculated 'areas graph' and the on line estimation of the controller parameters. Lastly, we will apply the BIP theoretical framework to the composition of quality controllers.

- The creation of a tool which improves and extends the existing tool by implementing the theoretical results. This work will be done in close cooperation with a team from STMicroelectronics and will be partially completed within the RNTL/Galogic project which begins this year.

17.3.2 Software Verification

We are considering continuing our work on software verification for programs having mutable data structures. We are particularly interested in properties which combine topological and quantitative aspects of memory. The objective will be to develop methods for verification, and to implement these in verification tools and to show their practical applications using real case studies. The team has published several results on this in 2006: TACAS, CAV and ICALP. Currently, we are studying the extension of separation logic with arithmetic constraints that allow us to specify quantitative properties.

17.3.3 Security

This new topic is expected to expand in coming years following the existing work directions and including the following topics:

Group Protocols: A very important class of cryptographic protocols relates to group protocols, which are divided into two distinct sub-classes: those where the number of participants is static, and those where it is dynamic. In the second sub-class, the participants can join or leave the group after the protocol execution has started. For example, in the case of key exchange protocol, the problem is to be able to generate a new key efficiently while guaranteeing that participants who do not make parties or are not or no longer a part of the group cannot have access to the information about this key. Several application domains use this type of protocol: teleconferencing, sensor networks, etc.

The problem of verifying group protocols has received little attention thus far. We are considering developing verification algorithms for these protocols.

Complexity-theoretic Proof Assistance: There exist two approaches for the verification of cryptographic protocols: the symbolic approach, and the complexity-theoretic approach. The symbolic approach is based on the hypothesis of perfect cryptography. This

hypothesis allows modeling the exchange of data by the protocols as algebraic terms. It allows also to consider a single adversary, which is easy to model. In the complexity-theoretic approach, the data are represented by sequences of bits. The adversaries correspond to probabilistic algorithms which execute in polynomial time. This latter approach is more realistic because the model is close to the implementation of the protocols. There is no theory of proof of correctness of protocols within the complexity-theoretic framework. We are considering developping such theory. The goal is to develop a theory for the refinement of probabilistic programs which contain calls to non-specified procedures. These procedures correspond to the adversaries.

Physical Attacks: The majority of ongoing work on modeling and verification of cryptographic protocols make the hypothesis that the adversary has access only to the messages exchanged. In 1996 appeared the first attack called "side-channel". These are primarily related to analyzing the electrical consumption of a component or of injecting miscalculations there, in order to extract secret information (key). Since then, many theoretical and practical advances have appeared, for both attacks and countermeasures.

However, the majority of this work remains empirical; little research has been done for developping an adequate semantic model and the appropriate verification methodes. We plan to study these problems in leveraging on our detailed knowledge of both the symbolic and the calculation-based models.

17.3.4 Modeling and verification of autonomous systems.

Complex systems having a capacity for autonomous decision-making pose the problem of developpement and validation of the models used for planning and controle of execution. Currently, the majority of these systems, for example autonomous exploration robots, which integrate fonctionnalities for complex perception and actions, associated with mechanisms for deliberation (planning/controle of execution), follow relatively integration simple rules.

At the functional level, these systems' autonomy requirements include a great deal of flexibility in the use of resources. It is thus interesting to implement a finer degree of control over execution at the level of these functional modules, based on a model of temporal constraints of these operations and their interactions. This model would be then used during the execution to both guarantee that the constraints remain satisfied, but also to check the actions that the systeme must execute to remain coherent with this model. In the same way, on a decisional level, the use of proven techniques (temporal automatons) could allow the study of problems related to planning and execution control in a new light. Thus, we could increase the robustness of the deliberative layer of the robot by studying and deploying a planning based also on a temporal model of the actions, coupled with execution control capable of reparing the plans produced when there has been a failure at execution time. Such an approach would allow verification of the correct dynamic behavior of the system at execution time, and also to increase the robustness and the adaptivity under real conditions. It seems interesting to us to study these two objectives together,

because they are based on similar models and implement the Resolution Algorithms of a similar order.

17.3.5 Projects accepted for 2007-2011

- IST/SPEEDS (2006-2009): Speculative and Exploratory Design in Systems Engineering.
  SPEEDS is an Integrated Project of IST 6th framework programme on "Speculative and Exploratory Design in Systems Engineering".

  SPEEDS is a concerted effort to define the new generation of end-to-end methodologies, processes and supporting tools for safety-critical embedded system design that will enable European systems industry to evolve from model-based design of hardware/software systems, towards integrated component based construction of complete virtual system models.

  SPEEDS aims at improving substantially the competitiveness of the European industry in this critical economic sector by marrying design competence with deep technical insights and theoretical foundations. SPEEDS partners are companies active in the entire supply chain: OEMs, suppliers, and tool vendors, supported by leading European research institutions.

- ITEA SPICES (2006-2009) : Support for Predictable Integration of mission Critical Embedded Systems.
  Support for Predictable Integration of mission Critical Embedded Systems; propose a 'reconciled' approach, by means of new methods and tools to derive from extended AADL (Architecture Analysis & Design Language, formerly known as Avionics Architecture Description Language) descriptions, component-based predictable implementations of mission-critical embedded systems associated with certification issues.

- RNTL AVERILES (2006-2009).
  Partners : Liafa (Univ. Paris 7), LSV (ENS Cachan) and EDF (Saclay).
  The goal of this project is to develop advanced techniques for the analysis and verification of complex embedded software. The project aims in particular to handle multi-task programs with dynamic allocation of memory and pointers handling. The goal is to develop an approach able to verify: (1) the absence of intrinsic defects in memory handling and (2) properties on the structure of the memory for this type of complex embedded software. Examples of such programs are those used by EDF in energy production systems.

- RNRT ARESA (2006-2009) : Embedded Systems and Sensor Networks.
  Partners : France Télécom, LSR-IMAG, Vérimag, TIMA, CITI (INSA Lyon), Coronis Systems.
  See Section 17.2.2 for a description of the projnect objectives.

- ARASSIA AMAES (2006-2009)

- RNTL EDEN2 (2006-2009).
  Partners : AXALTO, CEA-LETI, CEA-LIST, TRUSTED-LABS.
  The main objective of EDEN 2 is to industrialise the méthodology and tools developed in the exploratory RNTL project EDEN :

  1. Improvement of the tool-chain interfaces
  2. Study and implementation of technical improvement in the specification language and associated tools

  The second objective is to evaluate the methodology and tools in the framework of a CC evaluation:

  1. Taking into account all the levels of software architecture suggested by the CC (version 3.0).
  2. Update the software supplying evaluations, proof and test plans for version 3.0 of the CC.
  3. Validation of an industrial case study by a CC-type evaluation.

- RNTL OpenEmBeDD (2006-2009).
  Partners : Airbus, Anyware Technologies CEA/LIST ̆2013 Commissariat à l̆2019Energie Atomique, CS SI, France Telecom, LAAS - CNRS, THALES AEROSPACE (THALES DAE), THALES Research & Technology.
  In the field of real-time embedded systems, the points of rupture in the current design flow are obstacles for the generalized use of proven formal techniques (technological obstacle), which have themselves been developed outside of standards for representation - which constitutes a methodological obstacle to their use. The project tackles these methodological and technological obstacles. In particular by:

  1. contribution to UML standards for real-time embedded systems,
  2. implementation of innovative interoperable technologies,
  3. mastering the methodological chain,
  4. simulation of real-time models.

  An open solution for interoperable generic tools effectively implementing tool chains for real-time embedded systems will make it possible for industrial partners to better capitalize on their know-how, and to thus contribute to necessary gains in productivity.

- RNRT Politess (2006-2009) : Security POLIcies for networked information systems: modeling, deployment, TESt and Surveillance.
  Partners : GET- Équipes de Rennes et Évry, LSR, IRISA, France Télécom - R&D, Leirios Technologies, SAP Research, Silicomp-AQL
  POLITESS: Modelling and deriving configurations and tests for security policies Security policies define the rules that information systems and networks should enforce.

However, it is hard to ensure that a given set of interconnected machines and systems actually satisfies the requirements of a policy. POLITESS investigates methods to derive configurations of policy enforcement points, as well as methods of actually checking or testing whether a system actually conforms to a given policy. It is based on a formal model for policies. Real-size web service applications will be used as case studies.

- RNTL GALOGIC (2006-2009) : Automatic Generation of Controled Software.
  Partners :
  This project aims for the automatic generation of controlled software, to simplify the adaptation of software to a physical platform. We propose to develop a methodology and the tools for generating fine grain controllers for real-time software applications. The objective for these tools is to integrate the temporal dimension into the entire design flow, from specification implementation on a pysical platform.

17.4 Perspectives of the "Timed and Hybrid Systems" team

The group intends to maintain its position on the interface between informatics and other scientific and engineering disciplines, providing new ideas, algorithms and, eventually, tools to aid in the design and analysis of systems in the broad sense of the term. We exploit our privilleged position as academic scientists to pursue long-term research goals with innovative scientific ingredients, without insisting too much on immediate industrial applicability, which might involve spending too much time on activities with marginal scientific interest. Having made this choice, which is not the only legitimate one, we still do not adopt an elitistic attitude of theory for the sake of theory, but rather view applications as a source of inspiration for new research directions.

Much of the last four years was invested in fighting the complexity barriers that prevented the application of timed and hybrid models to real-life problems. Toward the end of the period we had some breakthroughs in this respect, most notably the ability to verify linear systems with hundreds of state variables, and the very recent discovery that allows us to get rid of some of the state explosion in timed automata. These results give hope for scaling up our techniques and attracting the attention of practitioners and tool vendors. Below we detail some of our plans and expectations for the coming years.

17.4.1 Hybrid Systems

Having solved the reachability problem for linear systems of high dimension, what remains to be done is to extend the technique toward more complex systems, namely piecewise-affine and non-linear systems. For the former, which are hybrid automata with affine dynamics in every mode, we need to find a good algorithm for approximating the intersection of a zonotope with a hyper-plane, and solve other problems related to the accumulation of errors for systems that switch too often. For non-linear systems we intend to explore

several directions, the first of which is the "hybridization" technique that transforms any non-linear system into a piecewise-affine one. Here the problem is that the number of discrete modes grows exponentially with the number of variables that need to be linearized. An alternative approach would be to use specialized methods that work directly on the non-linear system, replacing polytopes by more "curved" classes of geometric objects inspired by computer-aided geometric design. At the end of the process we will be in a position to provide designers and analyzers with new powerful ways to explore complex dynamical systems.

In parallel to this activity, focused on exhaustive verification, we will continue with the complementary direction of making simulation of *individial* trajectories a more systematic activitiy using a variaety of methods (probabilistic exploration, approximation metrics) to cover the reachable state space.

17.4.2 Timed Systems

Our efforts during the last years start bearing fruits with the discovery of new ways to reduce the explosion in the number of zones. We intend to implement new verification algorithms based on this result and combine them with the automatic abstraction technique that we have developed. As a result we expect to be able to handle in a compositional manner products of dozens of timed automata. Application domains can be digital circuits, performance evaluation of real-time software and scheduling problems.

Some of our insights concerning verification, synthesis and scheduling using timed automata will be exploited within the new ANR projeact AMAES (coordinated by S. Benaslem from the DCS group, together with LAAS and LIAFA). This project aims to to reapproach the AI planning community with ours, to collaborate on several problems including the application of timed automata to planning missions for autonomous robots, and for scheduling tasks inside the robot architecture. We also hope to learn from the AI partners about their heuristics for finding paths (plans) in high-dimensional state spaces.

Finally let us mention some close relatives of timed automata, the "linear" hybrid automata, where in each mode the continuous variables can have a different (but fixed) derivative. Although these automata can be used to approximate systems with "real" continuous dynamics, their advantage is in their capabability to model sophisticated flow systems where quantities flow between buffers at different rates subject to switchings that open or close channels. Such models can be useful for performance analysis of networks and other "stream processing" applications and architectures which are becoming very popular recently due to the progress in audio, video and radio technologies.

17.4.3 Monitoring

Monitoring, also known as runtime verification, is the process of checking simulation traces against high-level specifications such as those expressed in temporal logic. We have already invested a lot in building property monitors for real-time properties expressed in the logic MITL and used it also for monitoring continuous signals. We intend to continue with this line of research by attacking pragamtic issues such as monitor size or different modes

of interaction between the simulator and the monitor, as well as more conceptual issues such as the definition of other types of properties and performance measures which are more useful in the continuous domain such as distance from a reference signal. This work, which is part of the ongoing European project PROSYD, targets the application domain of analog circuits, but can be used in other domains, including the monitoring of safety-critical plants, enterprises, etc.

17.4.4 SAT modulo theories

The field of constraint solving is undergoing a dramatic change due to the improvement in the performance of Boolean SAT solvers, combined with a development of a generic scheme, DPLL(T), for solving Boolean combinations of constraints from a theory T. We have developed a generalization of the DPLL(T) framework that can treat a chain of progressively more complex theories, distributing the effort wisely among theories so that reasoning is always done in the cheapest theory possible. We have so far experimented with the theory of difference contraints (motivated by timing problems) and we intend to extend the solver to other theories such as linear algebra. We expect the new solver to be a major tool in hybrid optimization, and perhaps also serve as an alterntaive to the standard engine for computing successors in discrete, timed and hybrid dynamical system. The solver is developed in the framework of the RNTL project DECIDE! in which ILOG, the world leader in constraint solving is the other partner.

17.4.5 Applications

So far we have focused mainly on the following two aspects of timed and hybrid systems:

1. Modeling different real-life phenomena using timed and hybrid automata and formulating various design problems (monitoring, verification, synthesis, planning, scheduling, optimization) as computational problems on these models.

2. Identifying the major computational bottlenecks in solving these problems and developing progressively more powerful algorithmic solutions. The history of formal verification shows that efficient algorithmics is a necessary condition for gaining acceptance.

In the process we made use of both synthetical hypothetical applications (circuits, abstract scheduling problems, arbitrary linear systems) that represented the inherent computational difficulty of the respective domians, as well as some small-scale real examples encountered in various occasions, mostly in the context of European projects. We list below some of the potential application domains for our research results, along with a summary of actions we have already taken in each domain.

Verification of Analog Circuits: This is a new domain where not much has been done, and we are in a good position to influence its development. The monitoring techniques developed in the PROSYD project are currently applied to a FLASH memory example

given by ST, and can be very useful in this domain where simulation is very costly. Being able to do exhaustive verification of critical analog components may change, in the long run, the landscape of analog circuit validation. We have been contacted by the mobile electronics group in ST to establish a pilot project concerning the stability of a power management system for cellphones. We have also other points of contact at ST, Intel, IBM, Rambus, Cadence and Synopsis.

Control System Design: Control was one of the major driving forces behind the whole hybrid systems domain. So far only toy applications were reported in domains such as automotive, avionics and robotics. We hope that the increased performance of our tools will lead to more convincing examples. We gained some experience in the automotive domain within the CC project and we intend to continue working on more complex automotive problems provided by PARADES within the HYCON network. Our collaboration with LAAS and with Penn may lead to interesting robotics applications.

Biology and Chemistry: Chemical and biochemical reactions where state variables correspond to conentrations of substances, are very complex dynamical systems. In recent years hybrid automata were found useful in modeling switching phenomena that occur in such reactions. For biological systems, the type of analysis that our techniques provide, can demonstrate, for example, the robustness of certain cellular mechanisms under parameter variations. For industrial chemical processes, our techniques can potentially find better and cheaper ways to synthesize materials (it is estimated that more than 10% of the world's energy is invested in industrial chemical reactions). We intend to explore these directions together with local biologists, as well as with international collaborators in NYU (the thesis of C. Le Guernic will be co-supervised by Prof. B. Mishra) and Penn.

Power Grids: The stability of power grids is a very important problem with a lot of economic implications. We gained some experience in the problem working on very small problems provided by ABB, and we hope to treat larger problems with our enhanced tools.

Timing Analysis of Digital Circuits: Although our research in the domain has been sponsored by Intel, the gap between the size of circuits that we could treat and real-life design was several orders of magnitudes. With our recent results we hope to decrease that gap and find the right niche for our techniques, either in asynchronous circuits which are typically much smaller and cannot be analyzed using "static" technioques, or in a level higher than the gate level in which we worked so far.

Multiprocessors: The new trend in microprocessor architecture is the "multi-core": several processors on the same chip sharing the same memory. Another variant of this concept is the "stream processor" (oriented toward multi-media applications), which includes also some specialized devices such as DSPs and other dedicated hardware. The deployment of software on these architectures involves scheduling and timing problems

similar to those we have worked on. We are currently negotiating a project with ST on this topic in the framework of MINALOGIC.

Scheduling and Optimization in General: Within the AMETIST project we have demonstrated that our techniques can solve some real-life scheduling problems provided by the tool vendor Axxom. The development of our constraint solver within the DECIDE! project with ILOG may open new opprtunities in this direction.

17.4.6 Tools

Tools may have different levels of maturity ranging from an initial proof of concept down to industrial-strength tools. Most of our tools so far were closer to the first category as one may expect from tools developed within a thesis. During the next period we intend to push some of the tools closer to industrial usability, with the goal of extending the user base, getting feedback and visibility and having better assessment for the prospects of industrial transfer. We will sketch below the characteristics of the tools we expect to develop in the coming period, especially if our group obtains a research enginer.

- *The Hybrid Tool*: we intend to develop version 2.0 of `d/dt` that will include, among other things, the following features: a compositional input language, hybridization, zonotopes, algebraic representation, systematic and random simulation, controller synthesis, monitoring and interface with graphical packages. This tool will be publicly distributed with an extensive set of examples at all levels of complexity. The possibility of packaging it as a Matlab toolbox will be investigated.

- *The Timing Tool*: So far our work on modeling, analysis and abstraction of circuits has been done on an exploratory version of the IF toolset, which contains many additional ingredients for treating object-oriented software, for example. We intend to develop a more timing-specific tool, which will be based on some subset of IF, augmented with new developments such as convexity reduction, abstraction, translation from input languages (circuits, scheduling), MITL model checking, simulation and monitoring. This development is conditioned upon some feasibility proofs for convincing applications.

- *The Monitoring Tool*: This tool will check behaviors of timed and hybrid systems, generated by various simulators, against specifications expressed in various formalism such as temporal logic (including the industrial version PSL) and other performance measures. Large parts of the tool have already been implemented within PROSYD and can be integrated with the tools mentioned above. Since such a tool can be interfaced with practically any simulator in any domain, we prefer to keep it seperate.

- *The Constraints Tool*: A constraint solver based on SAT modulo theories, which can be plugged into theory-specific solvers. The tool will be available as a stand-alone program, as well as a collection of libraries to facilitate the development of add-ons.

17.5 Creation of a new team on the Implementation of Embedded Systems

Market trends for embedded systems are strongly oriented to mass-market products, with very high volumes, low-cost and still lower time-to-market. As a consequence, implementing modern real-time embedded systems is becoming an extremely complex activity that requires mastering several interdependent and sometimes conflicting dimensions, such as time, memory and parallelism.

Several new factors make embedded software programming difficult. On one hand, because more and more functions of embedded applications are implemented in software, multiprocessor architectures are becoming an appealing hardware solution to provide high computational power at low-cost. On the other, the use of high-level programming languages, such as Java, allow dynamic multi-threading and memory allocation. Therefore, mastering multiprocessing, multi-threading and dynamic memory is a key objective for embedded systems development.

A rigorous design and implementation approach taking into account these factors is missing today. Making it work requires understanding the dynamic behaviour of systems, and the nature of relationships between non-functional constraints. Moreover, it requires a deep knowledge about the structure and the properties of the embedded application. To overcome this issue, we propose to set up a research team with the objective to develop theoretically sound and automated design, analysis, and implementation methods and tools that take into account these dimensions.

17.5.1 Multi-processor architectures

Motivation — Many embedded applications such as video compression, HDTV and packet routing require higher and higher performance. For instance, as camera resolutions are improved, the volume of pixels to be processed increases, for instance, going from VGA to 720p formats multiplies the number of pixels/s by 3. This creates new challenges, in particular for parallelizing the processes in a better way. Video enhancements are also requiring increasing processing, in particular for motion estimation, motion compensation, and 3-dimensional from 2-dimensional information extraction. These requirements will progress from 100 MOPS to 1 GOPS.

This trend in embedded systems design is leading to a significant growth in the workload of embedded processors as software gradually shifts towards an increasing computational complexity, requiring the execution of real-time tasks. To achieve the required processing capacity, improvements in silicon technology have enabled SoC devices with multiple embedded processors. The hardware solution is therefore to ressort to multiprocessor chips, such as the Intel's IXP2XXX NP family and the Philips Wasabi chip. However, due to the complexity of developing software for complex multiprocessor architectures, developers are often unable to fully optimize the performance of their programs on these advanced chips.

Approach — The implementation process consists in exploiting platform capabilities (e.g., multi-threading, pipe-lining, dedicated devices, multi-processors, etc) in the best possible way to map the abstract model into the concrete execution model in such a way that the timing requirements and constraints are met.

Implementing embedded real-time software compromises the abstractions of the high-level programming model: components must be executed on one or a few processors (thus, sequentially rather than concurrently), computation and communication takes time, and so on. Therefore, implementation involves resolving a number of issues not always taken into account at application software level, such as resource allocation (e.g., distribution of tasks to different processors, scheduling policy, etc) or task communication and synchronization (e.g., shared memory, semaphores, queues, etc).

To ensure implementations meet design timing requirements, there is a pressing need to come up with rigourous techniques and tools which will bridge gaps in the design flow and help reducing validation costs, while guaranteeing implementation correctness with respect to timing issues.

For this, the team's project is to develop component-based formal modelling, timing verification, simulation, and implementation synthesis techniques which will pave the way for generating correct-by-construction implementations (with respect to timing constraints) for complex multi-processor target execution platforms from high-level application software designs.

This project is strongly supported by a tight cooperation with STMicroelectronics in a number of contracts (ANACONDA, NEVA, SCEPTRE) in the context of the joint research lab.

17.5.2 Dynamic memory management

Motivation — Dynamic memory management is a serious challenge for real-time embedded systems based on Java technology. Contrary to the standard Java paradigm, garbage collection is rarely used in such real-time environments, since the temporal behavior of dynamic memory collection (e.g. *pause times*) is usually difficult to predict and thus significantly complicates the implementation of real-time scheduling policies. On resource-limited platforms, such as smart cards, the implementation of efficient garbage collectors (GC) is furthermore hindered by hardware limitations, and embedded systems manufacturers frequently omit them completely (e.g., in JavaCard for instance).

Several GC algorithms have been proposed for real-time applications, but they typically require the programmer to provide a model of the dynamic memory management behaviour of the application, such as the maximum allocation and mortality rates of objects for instance, a difficult task at best (i.e. determine the maximum allocation rate is undecidable).

Approach — An appealing solution to the dynamic memory collection issue is to allocate objects in *regions*. With region-based memory management, allocation and deallocation of objects can be performed in a predictable time. This memory model is advocated by the Real-Time Specification for Java (RTSJ), which allows the programmer to specify

that a given computation must run in the context of a pre-allocated region. However, programming with RTSJ is usually deemed much more difficult than with standard Java, especially since the sizes of the various memory regions must be known when developing the application, and since the programmer must decide in which region to allocate data structures. Moreover, current RTSJ implementations, like the JamaicaVM for instance, require too many resources (in terms of memory space and processor time) to be used on resource-constrained platforms.

Instead of requiring the programmer to decide where to allocate objects, static analysis can be performed on the application to resolve object placement issues. This approach requires to compute the lifetime of dynamically allocated objects, in order to insert calls to the deallocator of a region as soon as all the included objects are no longer used, while guaranteeing that deallocation of a region will not create dangling references. Escape analysis techniques conservatively determine at compile time whether the lifetime of an object exceedes its allocating method. Many escape analysis algorithms have been proposed for Java, but they typically fail to produce results complete enough to suppress the need for a GC.

This failure highlights the limitations of automatic analysis tools and advocates the use of *semi-automatic* mechanisms that provide hints to the programmer on where to place object allocations in the application. Providing a guided-development environment that can determine whether a given dynamic object creation can easily be replaced by region-based allocation/deallocation permits non-expert programmers to write their applications without needing to know precisely how memory management is implemented. This requires a fast analysis algorithm so as to be able to provide hints to the programmer at implementation time, without slowing down software development.

For this, our project is to develop a simple and efficient static analysis algorithm, combined with a region allocation policy for real-time embedded Java applications. The goal of this work is to provide a static analysis mechanism efficient enough to be integrated in an assisted-development environment, and to implement region-based memory management primitives suited for resource-limited platforms such as smart cards.

Appendix A

Organization and Resources

A.1 Management and Administration

Joseph Sifakis is the Director of VERIMAG Laboratory. Nicolas Halbwachs is now the Deputy Director. Susanne Graf was Deputy Director until October 2004.

VERIMAG is organized into three teams:

- "Synchrone" Team (leader: F. Maraninchi)
- Distributed and Complex Systems Team (leader: Y. Lakhnech)
- Timed and Hybrid Systems Team (leader: O. Maler)

The VERIMAG General Assembly is composed of all the permanent staff and one elected student. It meets at least once per year.

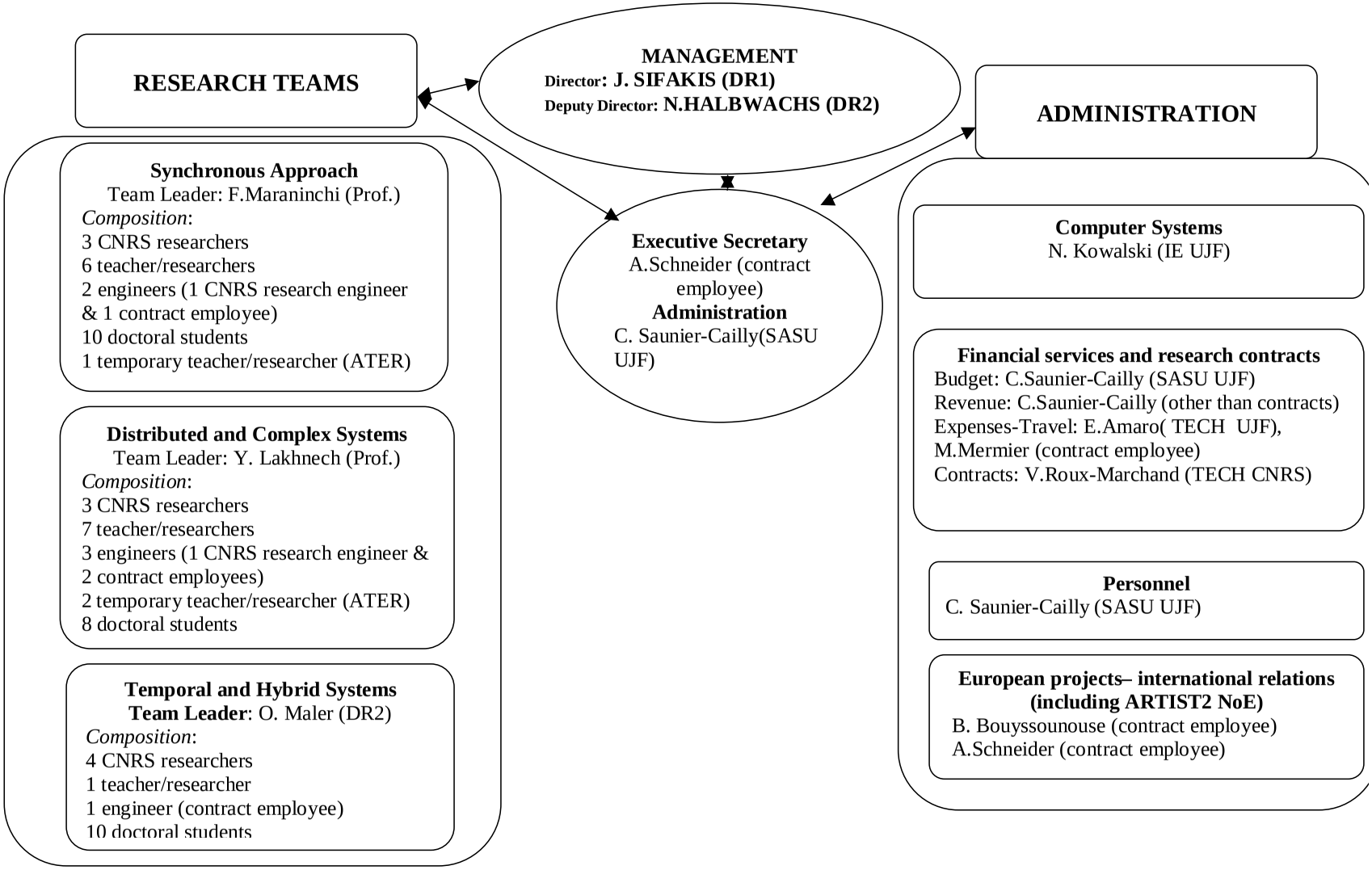
The work is coordinated in biweekly laboratory Council meetings. The Council is composed of the Director, the Deputy Director, and 15 members (7 are elected, and 6 are named by the Director).

The computer infrastructure is managed by N. Kowalski, and "Industrialization and Transfer" is managed by B. Bouyssounouse. The executive secretary is A. Schneider. Accounting and secretarial tasks are handled by C. Saunier and E. Amaro. An organisation chart of VERIMAG is presented on the following page.

In the rest of this annex, we provide a list of the staff as of end of January 2005, as well as an overview of the financial figures for the laboratory. This includes both "Consolidated" and "Non-Consolidated" views of the budget. The "Consolidated" budgets include salaries for permanent staff, which are paid by the CNRS, INPG, and UJF. The "Non-Consolidated" budgets include both institutional and contractual resources. Institutional resources are provided by CNRS, UJF, and INPG, and cover less than 10% of the overall "Non-Consolidated" budget for years 2002, 2003, and 2004.

At the end of this annex, we provide the list of contracts that were active at any time over this period.

ORGANISATION CHART VERIMAG Laboratory



A.2 List of the Staff - January 2005

A.2.1 Teachers/Researchers

CNRS

CASPI Paul	DR2
DANG Thao	CR2
GRAF Susanne	CR1
HALBWACHS Nicolas	DR2
IOSIF Radu	CR2
MALER Oded	DR2
RAYMOND Pascal	CR1
SIFAKIS Joseph	DR1
TRIPAKIS Stavros	CR1
YOVINE Sergio	CR1

UJF

BENSALEM Saddek	PR2
ENE Cristian	MC
FERNANDEZ Jean-Claude	PR2
LAGNIER Fabienne	MC
LAKHNECH Yassine	PR2
MONIN Jean-François	PR2
MOUNIER Laurent	MC
PARENT-VIGOUROUX Catherine	MC
PERIN Michaël	MC
RASSE Anne	MC
SCHOLL Pierre-Claude	PR1

INPG

ALTISEN Karine	MC
MARANINCHI Florence	PR2
NICOLLIN Xavier	MC
RIPPERT Christophe	MC

A.2.2 Engineers

CNRS

BOZGA Marius-Dorel	IR2
JAHIER Erwan	IR2

UJF

KOWALSKI Nicolas	IE2

A.2.3 Contract Employees

AKHAVAN Ana	ADR
BOUYSSOUNOUSE Bruno	
BOUZOUZOU Yussef	UJF
DEFAUT François-Xavier	FLORALIS
OBER Iulian	UJF
POTOP BUTUCARU Dumitru	UJF
REDROVAN Wilver	UJF
SCAIFE Norman	UJF
ZANCONI Marcelo	UJF

A.2.4 Administration

CNRS

ROUX-MARCHAND Valérie	TECH

UJF

AMARO Elisabeth	TECH
SAUNIER Christine	SASU

Contract Employees

SCHNEIDER Anne	FLORALIS
MERMIER Martine	UJF

A.2.5 Temporary Teacher/Researchers (ATER)

AMRANI Moussa	Univ.Savoie
BOZGA Liana	UJF
MOREL Lionel	INPG
PACHON Cyril	Univ.Savoie

A.2.6 PhD Students

ABED Nazha	Région Rhone Alpes
AMRANI Moussa	Contrat UJF
ASSAYAD Ismail	BDI
BASU Ananda	EGIDE
BEN SALAH Ramzi	EGIDE
BOZGA Liana	EGIDE
COMBAZ Jacques	CIFRE
CORNET Jerome	BDI
CURIC Adrian	MENRT
DANTHONY-GONNORD Laure	MENRT
DONZE Alexandre	Contrat UJF
GOTTELY Laurent	France TELECOM
HELMSTETTER Claude	BDI
JANVIER Romain	MENRT
KERBAA Abdelkarim	EGIDE
KOSSENTINI Chiheb	CIFRE
KRICHEN Moez	Région Rhone Alpes
LYBERIS Spiros	
MAZARE Laurent	MENRT
MERCHAT David	MENRT
MIKAC Jan	MENRT
MOREL Lionel	BDI
MOY Matthieu	CIFRE
NAHHAL Tarik	EGIDE
NAKHLI Chaker	EGIDE
NICKOVIC Dejan	EGIDE
PACHON Cyril	Contrat UJF
SALAGNAC Guillaume	Région Rhone Alpes
SAMPER Ludovic	CIFRE
SOFRONIS Christos	Région Rhone Alpes
STAUCH D	MENRT
VENTURINI Hugo	CIFRE

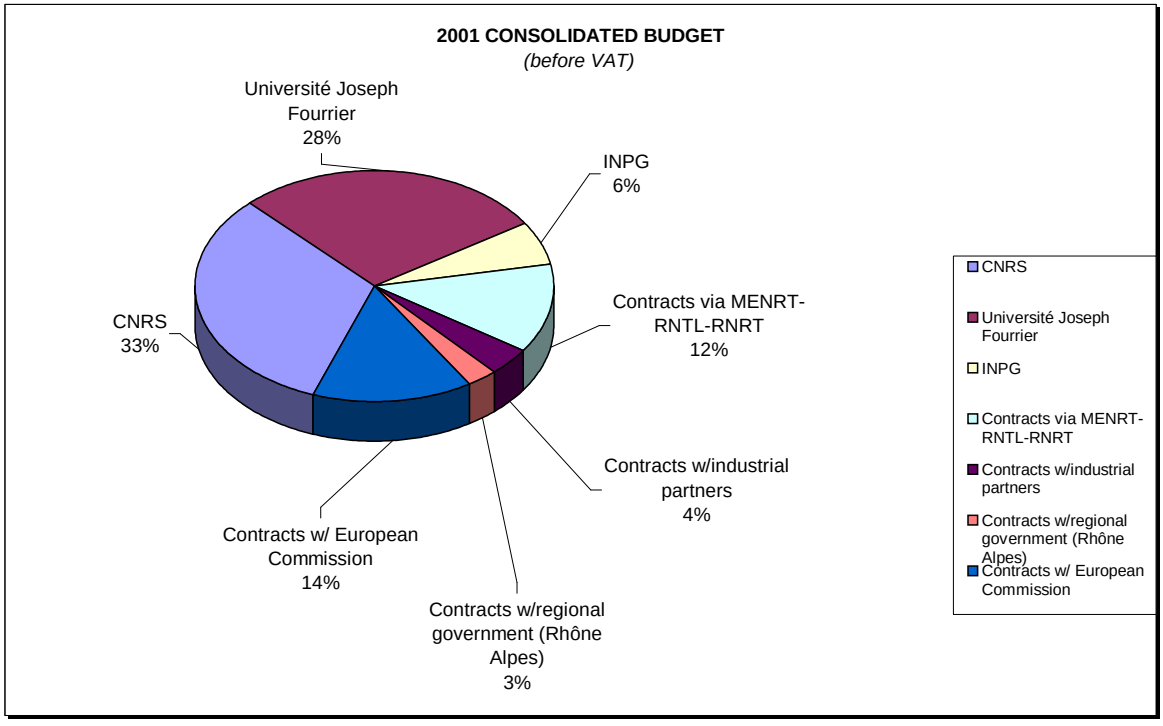
A.3 Budget

A.3.1 Budget for 2001

Consolidated Budget for 2001

CNRS(1)	673 534
Université Joseph Fourier(1)	582 656
INPG(1)	131 016
Contracts via MENRT-RNTL-RNRT	260 986
Contracts w/ industrial partners	81 230
Contracts w/ regional government (Rhône Alpes)	56 661
Contracts w/ European Commission	302 683
TOTAL	2 088 766

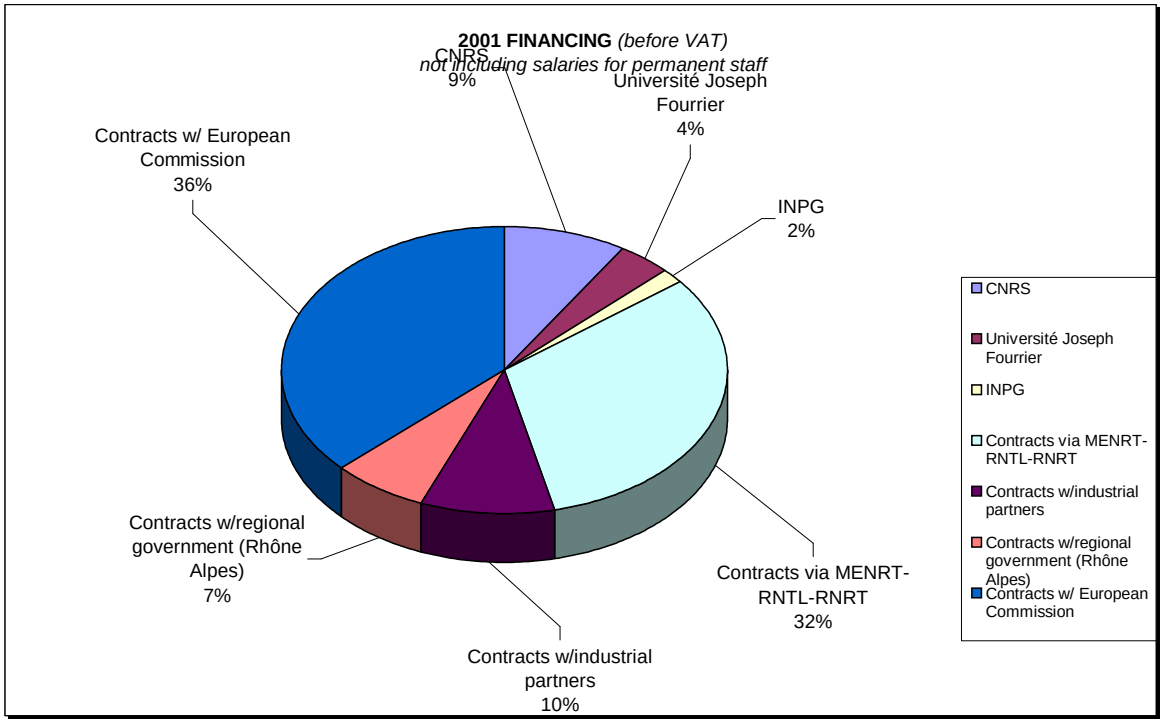
(1) including salaries for permanent staff



Financing for 2001 (before VAT)

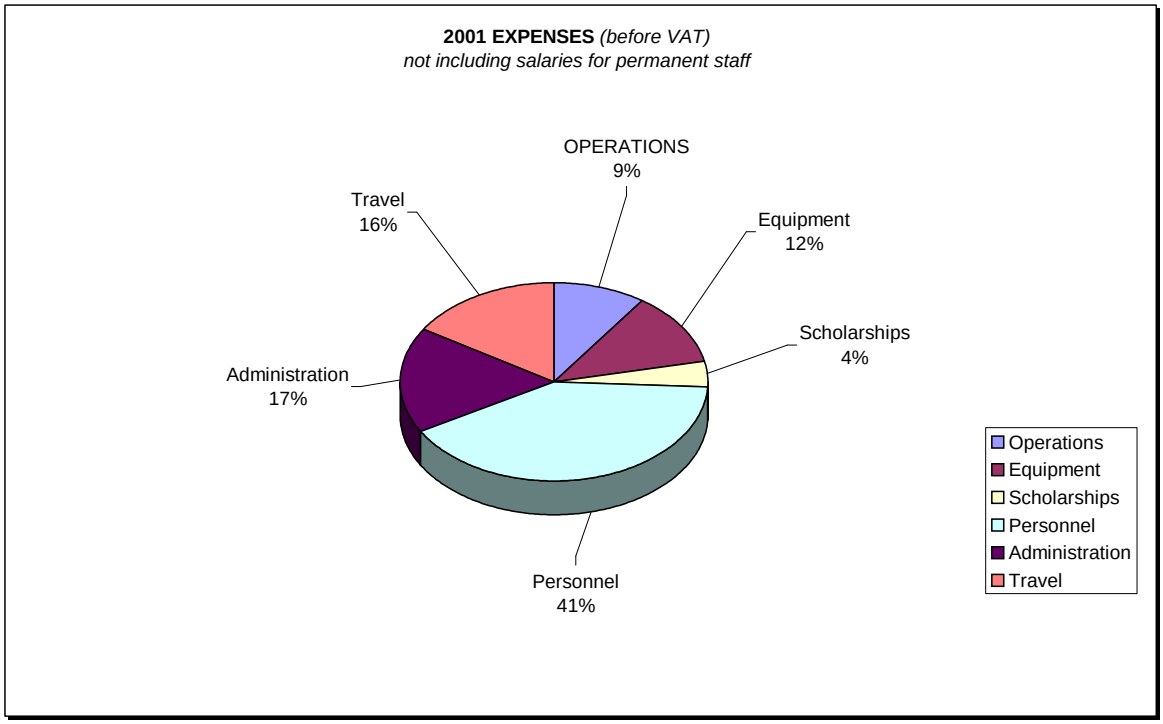
CNRS[(2)]	72 114
Université Joseph Fourier[(2)]	33 256
INPG[(2)]	14 256
Contracts via MENRT-RNTL-RNRT	260 986
Contracts w/industrial partners	81 230
Contracts w/regional government (Rhône Alpes)	56 661
Contracts w/ European Commission	302 683
TOTAL	821 186

(2) not including salaries for permanent staff



Expenses for 2001 (before VAT)

Operations	80 432
Equipment	97 522
Scholarships	33 883
Personnel	334 747
Administration	141 608
Travel	132 994
TOTAL	821 186

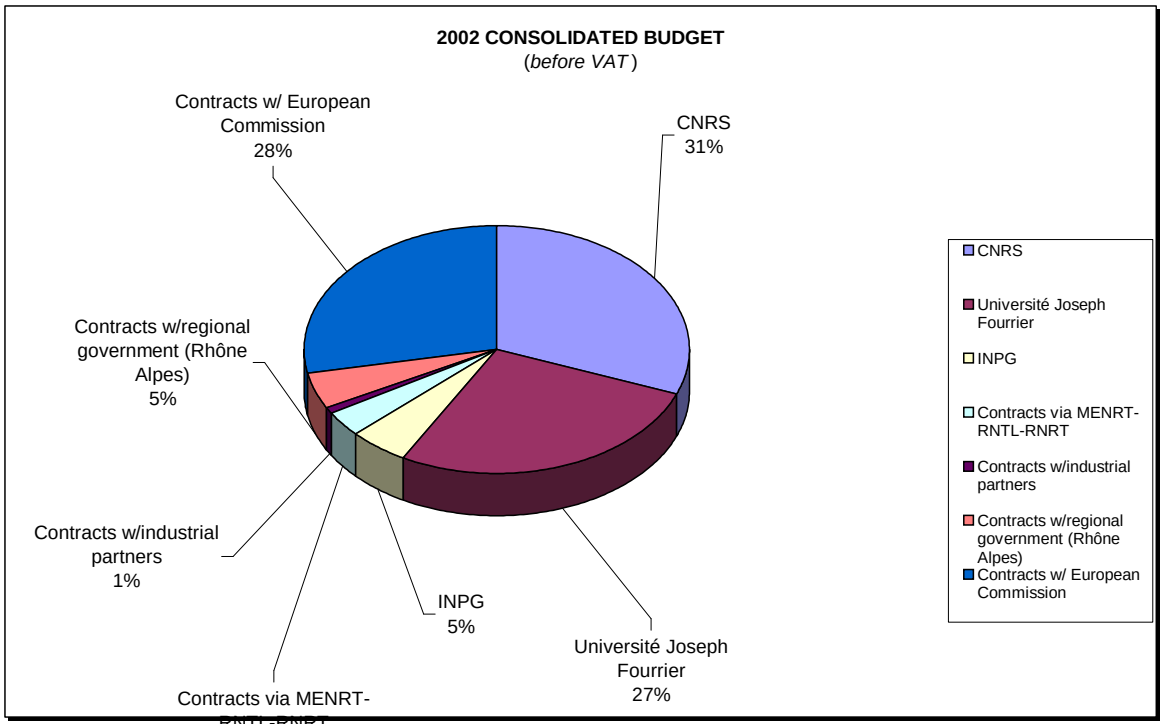


A.3.2 Budget for 2002

Consolidated Budget for 2002

CNRS(1)	787 903
Université Joseph Fourier(1)	704 000
INPG(1)	128 180
Contracts via MENRT-RNTL-RNRT	81 245
Contracts w/ industrial partners	20 792
Contracts w/ regional government (Rhône Alpes)	126 423
Contracts w/ European Commission	717 677
TOTAL	2 566 219

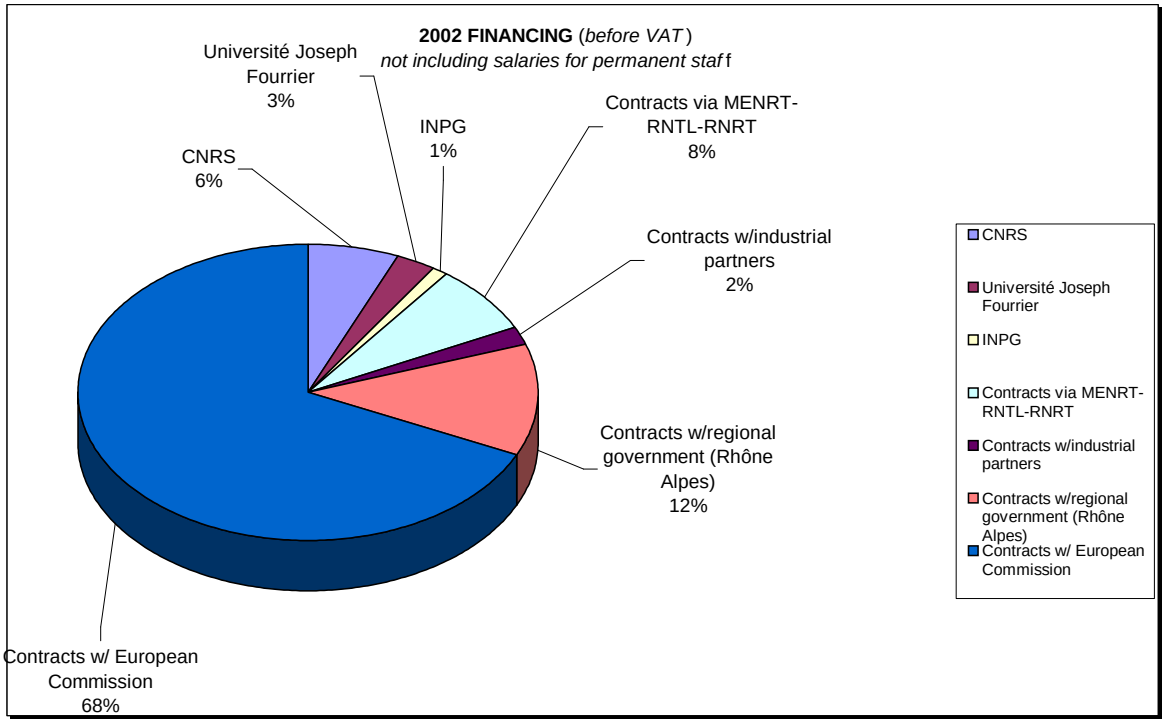
(1) including salaries for permanent staff



Financing for 2002 (before VAT)

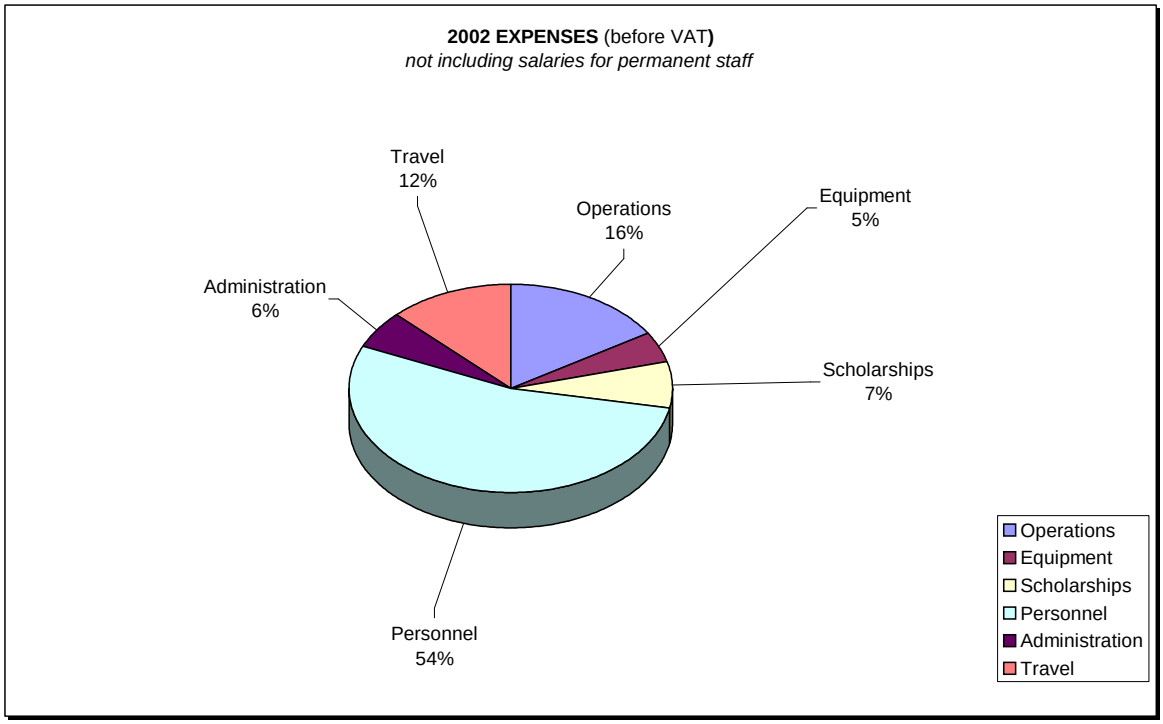
CNRS[(2)]	67 273
Université Joseph Fourier[(2)]	28 460
INPG[(2)]	11 420
Contracts via MENRT-RNTL-RNRT	81 245
Contracts w/ industrial partners	20 792
Contracts w/ regional government (Rhône Alpes)	126 423
Contracts w/ European Commission	717 677
TOTAL	1 053 289

(2) not including salaries for permanent staff



Expenses for 2002 (before VAT)

Operations	168 348
Equipment	51 993
Scholarships	75 043
Personnel	565 113
Administration	61 783
Travel	131 009
TOTAL	1 053 289

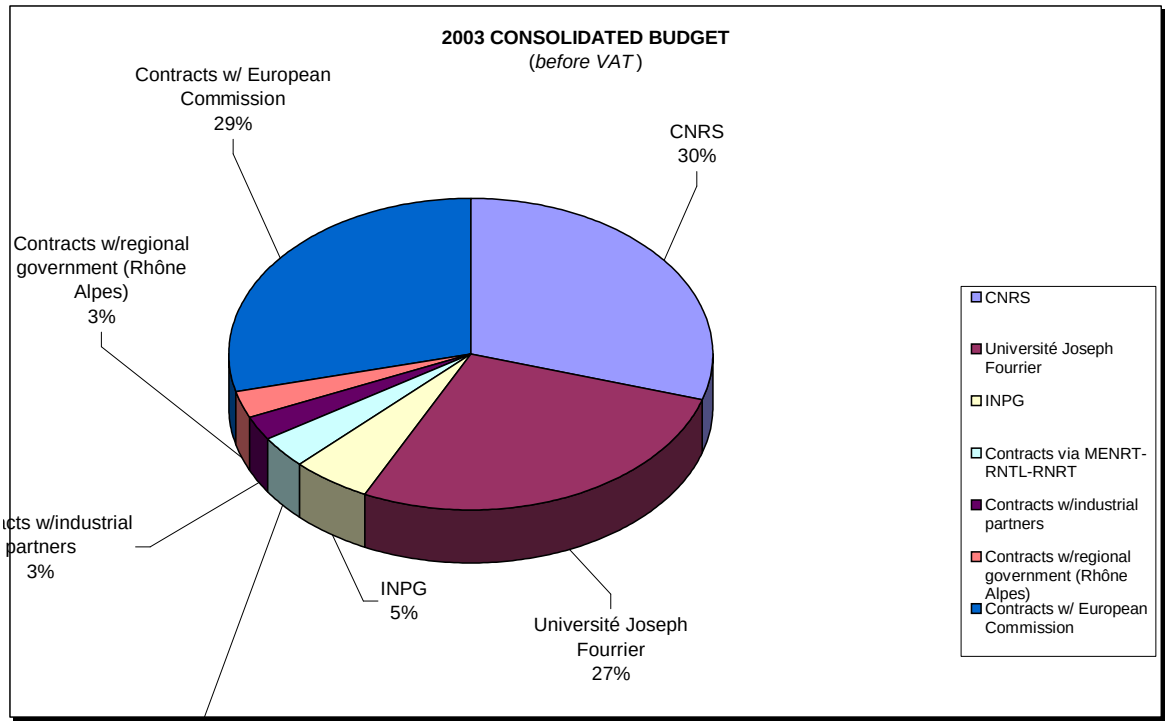


A.3.3 Budget for 2003

Consolidated Budget for 2003

CNRS(1)	775 210
Université Joseph Fourier(1)	713 569
INPG(1)	139 846
Contracts via MENRT-RNTL-RNRT	83 961
Contracts w/ industrial partners	65 522
Contracts w/ regional government (Rhône Alpes)	70 380
Contracts w/ European Commission	754 957
TOTAL	2 603 445

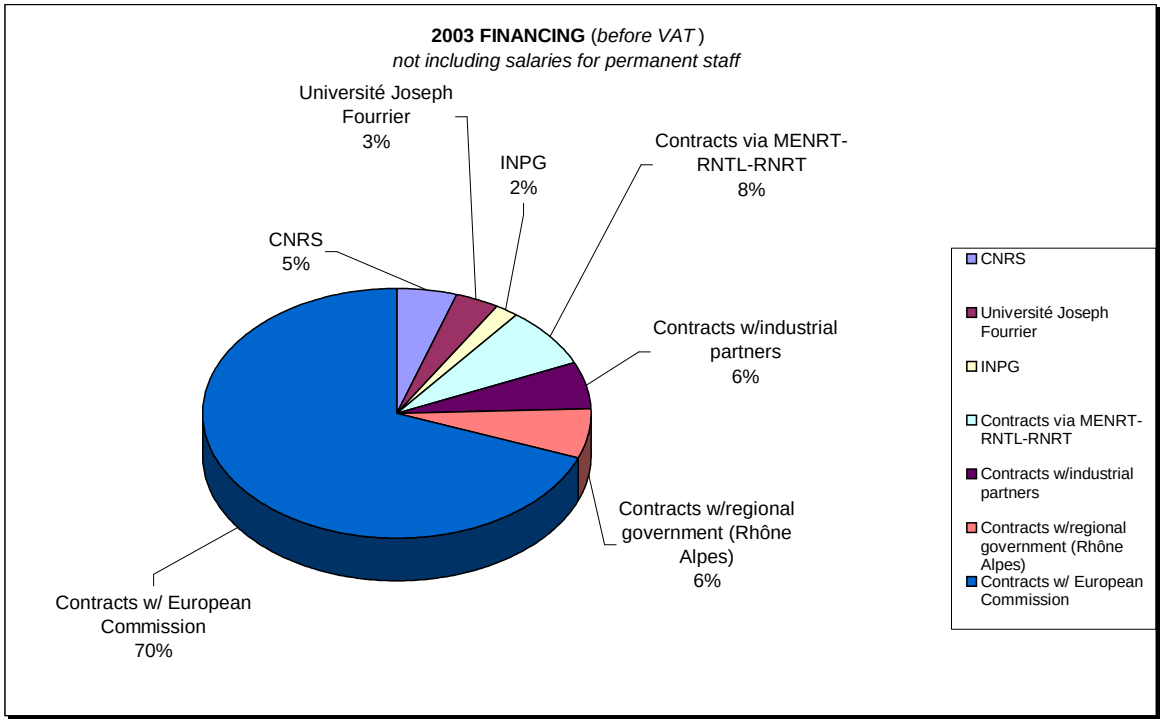
(1) including salaries for permanent staff



Financing for 2003 (before VAT)

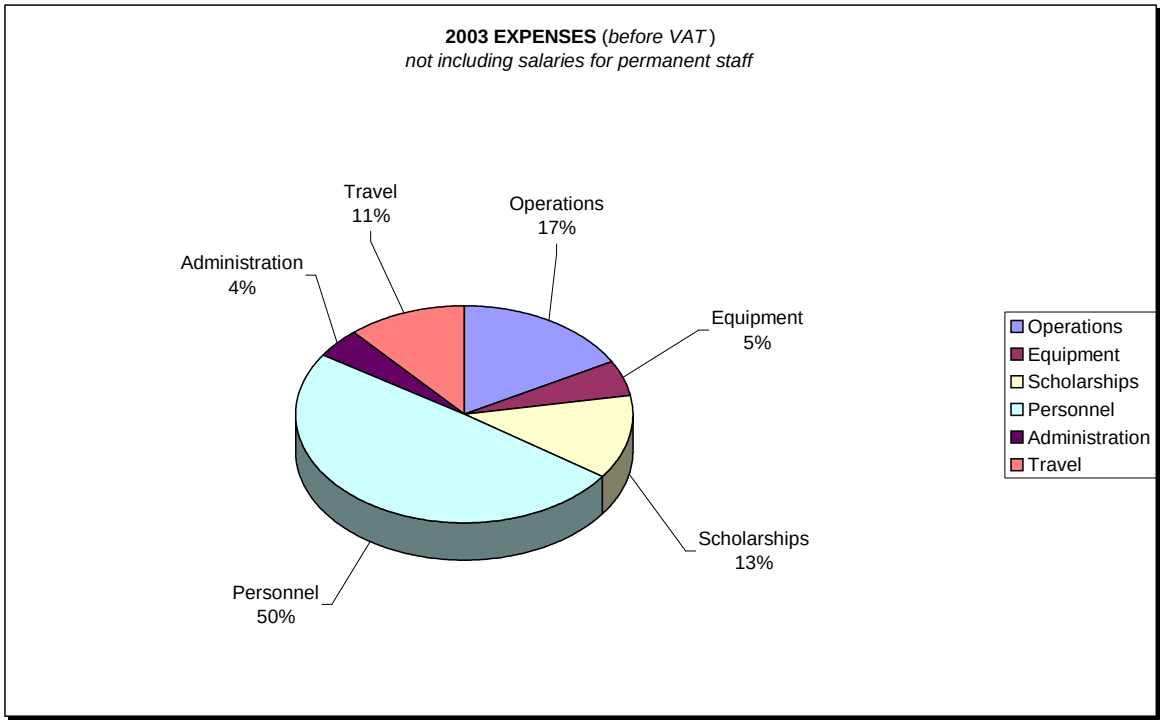
CNRS[(2)]	54 580
Université Joseph Fourier[(2)]	38 029
INPG[(2)]	23 086
Contracts via MENRT-RNTL-RNRT	83 961
Contracts w/ industrial partners	65 522
Contracts w/ regional government (Rhône Alpes)	70 380
Contracts w/ European Commission	754 957
TOTAL	1 090 515

(2) not including salaries for permanent staff



Expenses for 2003 (before VAT)

Operations	186 293
Equipment	55 188
Scholarships	136 471
Personnel	539 433
Administration	48 046
Travel	125 084
TOTAL	1 090 515

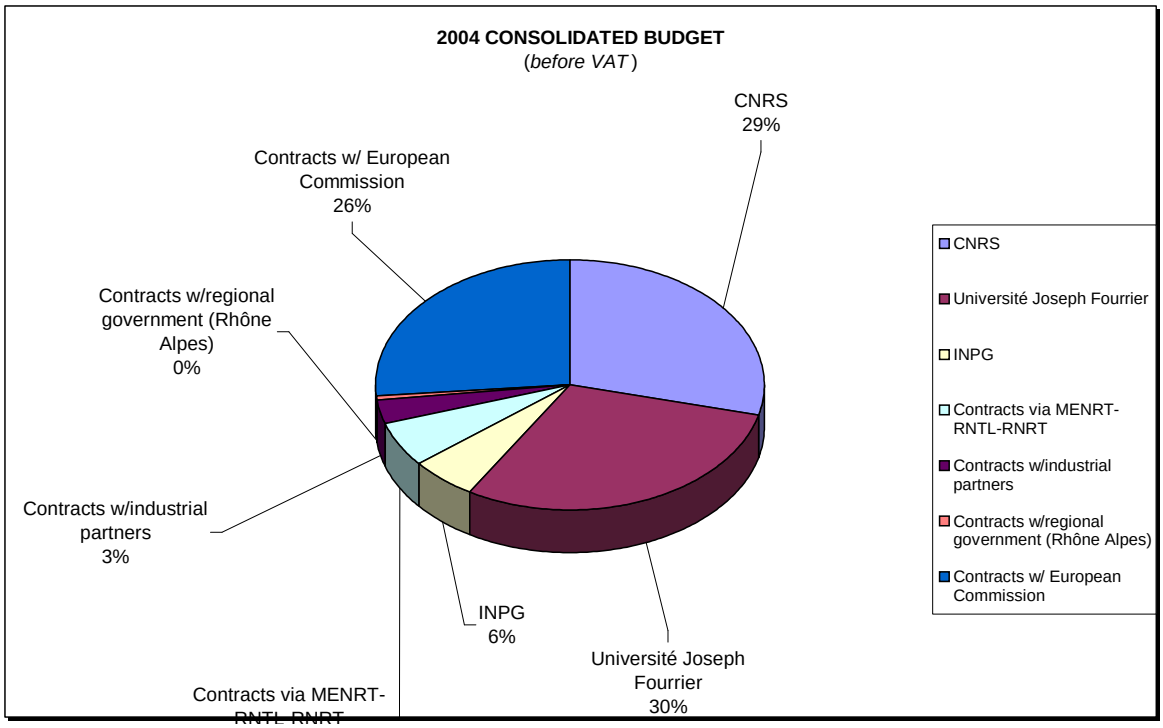


A.3.4 Budget for 2004

Consolidated Budget for 2004

CNRS(1)	754 786
Université Joseph Fourier(1)	769 457
INPG(1)	145 478
Contracts via MENRT-RNTL-RNRT	154 465
Contracts w/ industrial partners	79 309
Contracts w/ regional government (Rhône Alpes)	11 961
Contracts w/ European Commission	685 528
TOTAL	2 600 985

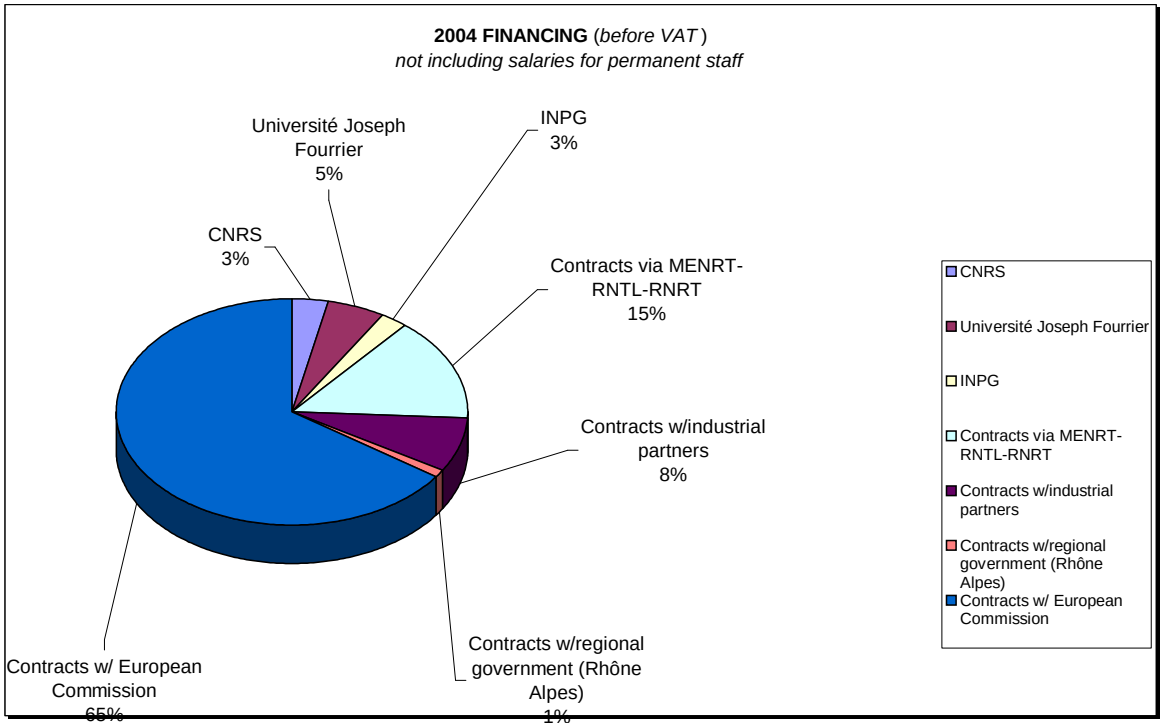
(1) including salaries for permanent staff



Financing for 2004 (before VAT)

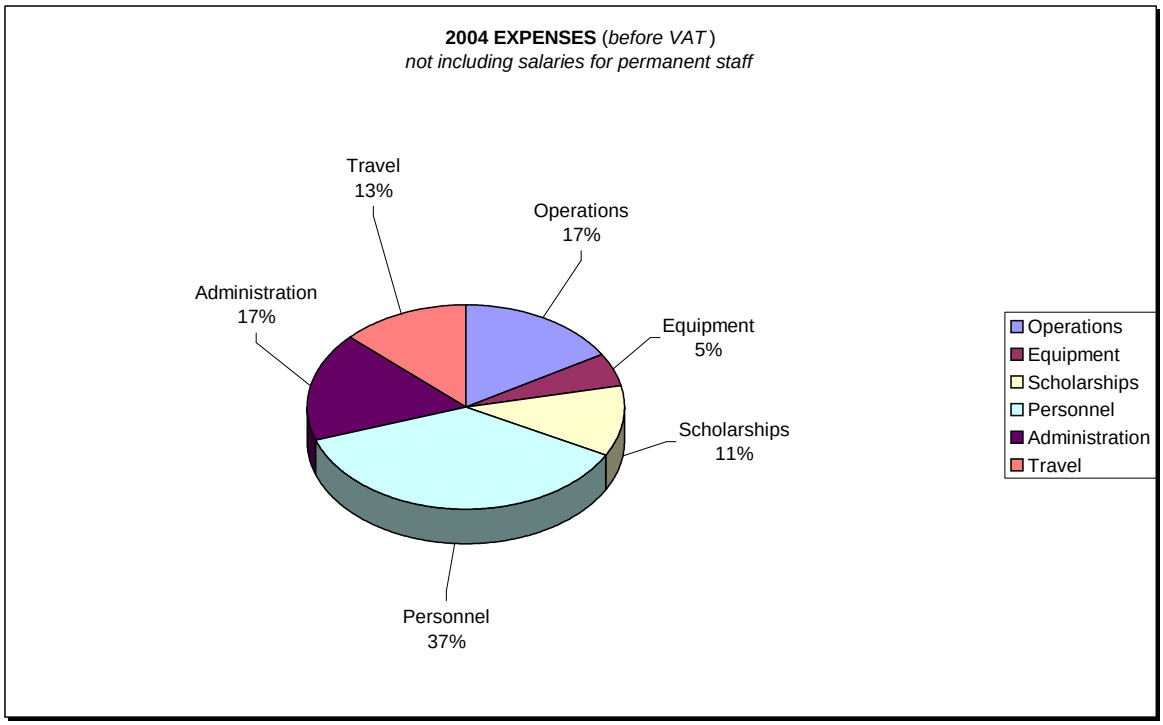
CNRS(2)	34 156
Université Joseph Fourier(2)	54 917
INPG(2)	28 718
Contracts via MENRT-RNTL-RNRT	154 465
Contracts w/ industrial partners	79 309
Contracts w/ regional government (Rhône Alpes)	11 961
Contracts w/ European Commission	685 528
TOTAL	1 049 055

(2) not including salaries for permanent staff



Expenses for 2004 (before VAT)

Operations	173 352
Equipment	54 218
Scholarships	116 240
Personnel	387 351
Administration	180 433
Travel	137 461
TOTAL	1 049 055



Europeean Comission Contracts (before VAT)

Contract	Leader	Start Date	End Date	Total
VHS	MALER Oded	05/1998	11/2001	294 457
CRISYS	CASPI Paul	11/1997	01/2001	146 429
INTERVAL	GRAF Susanne	03/2000	02/2002	76 225
ADVANCE	LAKHNECH Yassine	10/2000	09/2003	240 107
AGEDIS	LAKHNECH Yassine	11/2000	10/2003	228 865
NEXT TTA	CASPI Paul	01/2002	02/2004	397 289
SAFEAIR II	HALBWACHS Nicolas	07/2002	11/2004	115 929
OMEGA	GRAF Susanne	01/2002	02/2005	529 884
RISE	CASPI Paul	08/2002	02/2005	124 384
ARTIST 1	SIFAKIS Joseph	04/2002	03/2005	725 178
CC	MALER Oded	01/2002	04/2005	472 584
AMETIST	MALER Oded	04/2002	06/2005	271 116
ASSERT	HALBWACHS Nicolas	09/2004	08/2007	175 586
TYPES	MONIN Jean-François	09/2004	08/2007	9 100
PROSYD	MALER Oded	01/2004	12/2007	300 000
ARTIST 2	SIFAKIS Joseph	09/2004	08/2008	670 150
	Total for EEC Contracts			4 777 283,56

Industrial Contracts (before VAT)

Contract	Leader	Start Date	End Date	Total
AIRBUS I	MARANINCHI Florence	10/2000	03/2001	30 489
AEROMATRA LANCEUR	HALBWACHS Nicolas	03/2000	03/2001	60 674
INTEL	MALER Oded	05/2002	12/2002	27 864
AIRBUS II	MARANINCHI Florence	10/2001	01/2002	60 980
HONEYWELL		02/2003	12/2003	23 342
ISD	SIFAKIS Joseph	03/2000	02/2002	6 098
AIRBUS III	RAYMOND Pascal	03/2003	11/2003	20 000
RATP	CASPI Paul	01/2005	03/2006	36 800
AVERROES	MONIN Jean-François	05/2004	11/2006	100 000
INTEL 2	MALER Oded	09/2003	12/2003	29 160
INTEL 3	MALER Oded	07/2004	12/2004	27 098
CIFRE -STM- Combaz	FERNANDEZ J.Claude	10/2002	09/2005	18 400
CiFRE -STM- Moy	MARANINCHI Florence	10/2002	09/2005	18 288
CIFRE-STM-Bauer	YOVINE Sergio	09/2003	09/2004	12 000
CIFRE-STM-Kossentini	CASPI Paul	06/2004	06/2006	37 200
	Total for Industrial Contracts			508 396

French Ministry of Research (MENRT) Contracts (before VAT)

Contract	Leader	Start Date	End Date	Total
PREDIT	CASPI Paul	02/1999	02/2002	35 063
ACI Blanche	LAKHNECH Yassine	10/1999	10/2002	65 553
EDEN	LAKHNECH Yassine	11/2002	05/2005	239 383
ACI Coopération européenne	SIFAKIS Joseph	06/2003	06/2005	10 033
CORTOS	TRIPAKIS Stavros	07/2003	07/2006	25 084
DYNAMO	YOVINE Sergio	08/2003	08/2006	45 987
ACI ROSSIGNOL	LAKHNECH Yassine	09/2003	09/2006	18 060
PERSIFORM	SIFAKIS Joseph	11/2004	11/2006	132 554
PROUVE	LAKHNECH Yassine	11/2003	11/2006	143 520
ALIDECS	MARANINCHI Florence	09/2004	09/2007	42 642
APRON	HALBWACHS Nicolas	09/2004	09/2007	50 167
POTESTAT	MOUNIER Laurent	09/2004	09/2007	36 789
	Total for MENRT Contracts			713 315

RNTL & RNRT Contracts (before VAT)

Contract	Leader	Start Date	End Date	Total
TAXIS	YOVINE Sergio	09/1999	09/2002	80 036
EXPRESSO	YOVINE Sergio	06/2001	06/2003	168 914
EVA	LAKHNECH Yassine	11/2000	11/2003	83 085
	Total for RNTL & RNRT Contracts			80 036

Regional Contracts (before VAT)

Contract	Leader	Start Date	End Date	Total
Région 3	SIFAKIS Joseph	07/1999	07/2002	8 922
PSLC 2	SIFAKIS Joseph	03/2001	03/2004	146 586
SODA	YOVINE Sergio	06/2001	06/2004	2 549
NANO 2008 (2004)	YOVINE Sergio	01/2004	12/2004	92 548
SODA - année 3	YOVINE Sergio	07/2002	07/2005	4 250
Systèmes embarqués	SIFAKIS Joseph	12/2002	12/2005	75 000
MADEJA 1	YOVINE Sergio	06/2003	06/2006	4 180
MADEJA 2	YOVINE Sergio	06/2004	06/2007	10 000
Ing. à base de composants embarqués 2	SIFAKIS Joseph	06/2004	06/2007	5 000
Ing. à base de composants embarqués 1	SIFAKIS Joseph	06/2003	06/2006	5 000
Programme thématique thèse ABED	TRIPAKIS Stavros	01/2005	10/2008	6 270
	Total for Regional Contracts			360 305

Appendix B

External References

B.1 External References

[ABJ98] P. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels. In *10th Intern. Conf. on Computer Aided Verification (CAV'98)*. LNCS 1427, Springer-Verlag, June 1998.

[AC04] M. Abadi and V. Cortier. Deciding knowledge in security protocols under equational theories. In *31th International Colloquium on Automata, Languages and Programming (ICALP'04)*, 2004.

[ACT06] K. Altisen, F. Cassez, and S. Tripakis. Monitoring and fault diagnosis with digital clocks. In *Application of Concurrency to System Design (ACSD'06)*, Turku, Finland, June 2006.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[ADG+03] C. Artho, D. Drusinsky, A. Goldeberg, K. Havelund, M. Lowry, C. Pasareanu, G. Rosu, and W. Visser. Experiments with test case generation and runtime analysis. In *10th International Workshop on Abstract State Machines*, Sicily, Italy, 2003.

[AGS00] K. Altisen, G. Gössler, and J. Sifakis. A methodology for the construction of scheduled systems. In M. Joseph, editor, *proc. FTRTFT 2000*, volume 1926 of *LNCS*, pages 106–120. Springer-Verlag, 2000.

[AMS06a] K. Altisen, F. Maraninchi, and D. Stauch. Aspect-oriented programming for reactive systems: a proposal in the synchronous framework. *Science of Computer Programming*, 2006.

[AMS06b] K. Altisen, F. Maraninchi, and D. Stauch. Larissa: Modular design of man-machine interfaces with aspects. In *Fifth International Symposium on Software Composition (an ETAPS satellite event)*, March 2006.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *IFIP International Conference on Theoretical Computer Science (IFIP TCS2000)*, Sendai, Japan, 2000. Springer-Verlag, Berlin Germany.

[BÓ2] Alexis Bés. A survey of arithmetical definability. *A Tribute to Maurice Boffa. Bulletin de la Société Mathématique de Belgique*, 1 - 54, 2002.

[BA01] Mike Bond and Ross J. Anderson. Api-level attacks on embedded systems. *IEEE Computer*, 34(10):67–75, 2001.

[BAL97] H. Ben-Abdalla and S. Leue. Expressing and analysing timing constraints in message sequence chart specifications. Technical report, U. of Waterloo, 1997.

[BB88] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *ISDN*, 14(1):25–29, January 1988.

[BBLS00] Kai Baukus, Saddek Bensalem, Yassine Lakhnech, and Karsten Stahl. Abstracting ws1s systems to verify parameterized networks. In S. Graf and M. Schwartzbach, editors, *TACAS'00*, volume 1785 of *LNCS*, 2000.

[BBM00] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology-EUROCRYPT 2000, Lecture Notes in Comput. Sci., Vol. 1807*, pages 259–274. Springer, 2000.

[BCG+97] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, and B. Tabbara. *Hardware-Software Co-Design of Embedded Systems: The Polis Approach*. Kluwer Academic Publishers, 1997.

[BCG99] A. Benveniste, B. Caillaud, and P. Le Guernic. From synchrony to asynchrony. In J.C.M. Baeten and S. Mauw, editors, *CONCUR'99*. LNCS 1664, Springer Verlag, 1999.

[BCO04] J. Berdine, Cristiano Calcagno, and P. O'Hearn. A decidable fragment of separation logic. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, pages 97 – 109, 2004.

[Bel76] A. P. Beltyukov. Decidability of the universal theory of natural numbers with addition and divisibility. *Zapiski Nauch. Sem. Leningrad Otdeleniya Mathematical Institute*, 60:15 – 28, 1976.

[BFd+96] A. Belinfante, J. Feenstra, R. de Vries, J. Tretmans, N. Goga, L. Feijs, and S. Mauw. Formal test automation: a simple experiment. In *International Workshop on the Testing of Communicating Systems (IWTCS'99)*, pages 179–196, 1996.

[BFG+99] M. Bozga, J.Cl. Fernandez, L. Ghirvu, S. Graf, J.P. Krimm, and L. Mounier. IF: An intermediate representation and validation environment for timed asynchronous systems. In *Proceedings of Symposium on Formal Methods 99, Toulouse*, number 1708 in LNCS. Springer Verlag, September 1999.

[BFG00] Marius Bozga, Jean-Claude Fernandez, and Lucian Ghirvu. Using Static Analysis to Improve Automatic Test Generation. In S. Graf and M. Schwartzbach, editors, *Proceedings of TACAS'00 (Berlin, Germany)*, volume 1785 of *LNCS*, pages 235–250. Springer-Verlag, March 2000.

[BFN04] S. Bardin, A. Finkel, and D. Nowak. Toward symbolic verification of programs handling pointers. In *Proc. 3rd Int. Workshop on Automated Verification of Infinite-State Systems (AVIS 2004), Barcelona, Spain*. Electronic Notes in Theoretical Computer Science, 2004.

[BGK+00] Marius Bozga, Susanne Graf, Alain Kerbrat, Laurent Mounier, Iulian Ober, and Daniel Vincent. SDL for Real-Time: What is Missing ? In *Proceedings of SAM'00: 2nd Workshop on SDL and MSC (Grenoble, France)*, pages 108–122. IMAG, June 2000.

[BGS00] S. Bornot, G. Gössler, and J. Sifakis. On the construction of live timed systems. In S. Graf and M. Schwartzbach, editors, *Proc. TACAS 2000*, volume 1785 of *LNCS*, pages 109–126. Springer-Verlag, 2000.

[BHH+97] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thumer. Towards a formalization of the unified modeling language. In *Proc. of ECOOP'97*, pages 344–366. LNCS, Springer-Verlag, 1997.

[BHRZ03] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In R. Cousot, editor, *Static Analysis: Proceedings of the 10th International Symposium*, volume 2694 of *Lecture Notes in Computer Science*, pages 337–354, San Diego, California, USA, 2003. Springer-Verlag, Berlin.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In Yvo G. Desmedt, editor, *Proc. CRYPTO 94*, pages 341–358. Springer, 1994. Lecture Notes in Computer Science No. 839.

[Bla01] B. Blanchet. Abstracting cryptographic protocols by prolog rules. In *International Static Analysis Symposium*, volume 2126 of *LNCS*, pages 433–436, 2001.

[Bla03] B. Blanchet. Escape analysis for java(tm). theory and practice. *ACM Transactions on Programming Languages and Systems*, 6(25):713–775, November 2003.

[BLP+02] Felice Balarin, Luciano Lavagno, Claudio Passerone, Alberto L. Sangiovanni-Vincentelli, Marco Sgroi, and Yosinori Watanabe. Modeling and designing heterogeneous systems. In *Concurrency and Hardware Design, Advances in Petri Nets*, pages 228–273. Springer-Verlag, 2002.

[BLP06] L. Bozga, Y. Lakhnech, and M. Périn. Pattern-based abstraction for verifying secrecy in protocols. *STTT: International Journal on Software Tools for Technlogy Transfer*, 8:57–76, February 2006.

[BLS00] Kai Baukus, Yassine Lakhnech, and Karsten Stahl. Verifying universal properties of parameterized networks. In M. Joseph, editor, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1926 of *LNCS*, pages 291–304. Springer-Verlag, 2000.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT '00: Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer-Verlag, 2000.

[BOB] Richard Bornat, Peter O'Hearn, and Josh Berdine. http://www.cs.mdx.ac.uk/staffpages/r_bornat/movies/circular_list_dance.mpg.

[Bon01] Mike Bond. Attacks on cryptoprocessor transaction sets. In Çetin Kaya Koç et al. [cKKNP01], pages 220–234.

[Bon04] Mike Bond. *Understanding Security APIs*. PhD thesis, Cambridge University, 2004.

[BPPS00] V. Bertin, M. Poize, J. Pulou, and J. Sifakis. Towards validated real-time software. In *Proc. 12th Euromicro Conference on Real-Time Systems*, pages 157–164, 2000.

[BPW03] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *Proceedings of the 10th ACM conference on Computer and communication security*, pages 220–230. ACM Press, 2003.

[BPZ05] I. Balaban, A. Pnueli, and L. Zuck. Shape analysis by predicate abstraction. In *Proc. 6th Intl. Conference on Verification, Model Checking and Abstract Interpretation*, pages 165 – 180, 2005.

[BR00] Thomas Ball and Sriram K. Rajamani. Boolean programs: A model and process for software analysis. Technical report, Microsoft Research, February 2000.

[BR03] Mihir Bellare and Phillip Rogaway. *Introduction to Modern Cryptography*. 2003.

[BRZH02] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the parma polyhedra library. In M. V. Hermenegildo and G. Puebla, editors, *9th International Symposium on Static Analysis, SAS'02*, Madrid, Spain, September 2002. LNCS 2477.

[BS98] G. Berry and E. Sentovich. Embedding synchronous circuits in GALS-based systems. In *Sophia-Antipolis conference on Micro-Electronics (SAME 98)*, October 1998.

[BS00] S. Bornot and J. Sifakis. An algebraic framework for urgency. *Information and Computation*, 163, 2000.

[BST98] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *International Symposium: Compositionality - The Significant Difference*, LNCS 1536, 1998.

[BT01] E. Brinksma and J. Tretmans. Testing transition systems: An annotated bibliography. In *MOVEP 2000*, volume 2067 of *LNCS*. Springer, 2001.

[BW94] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *CAV'94*, Stanford (Ca.), 1994. LNCS 818, Springer Verlag.

[CAR04] Feng Chen, Marcelo Amorim, , and Grigore Roşu. A Formal Monitoring-based Framework for Software Development and Analysis. In *Proceedings of the Sixth International Conference on Formal Engineering Methods (ICFEM'04)*, LNCS. Springer, November 2004.

[CC97] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. *Conference Record of the 4th ACM Symposium on Principles of Programming Languages*, pages 238–252, 1997.

[CDR04] F. Chen, M. D'Amorim, and G. Roşu. *Monitoring-Oriented Programming: A Tool-Supported Methodology for Higher Quality Object-Oriented Software.* Technical Report UIUCDCS-R-2004-2420, University of Illinois at Urbana-Champaign, March 2004.

[CE69] R. B. Crittenden and C. L. Vanden Eynden. A proof of a conjecture of Erdös. *Bulletin of American Mathematical Society*, (75):1326 – 1329, 1969.

[CGS+99] Jong-Deok Choi, M. Gupta, Maurice Serrano, Vugranam C. Sreedhar, and Sam Midkiff. Escape analysis for Java. In *ACM OOPSLA'99*, volume 34(10) of *SIGPLAN*, pages 1–19. ACM, October 1999.

[CH78] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. 5th ACM Symposium on Principles of Programming Languages*, 1978.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345 – 363, 1936.

[CJ97] John A. Clark and Jeremy L. Jacob. A survey of authentication protocol literature. Version 1.0, University of York, Department of Computer Science, November 1997.

[CJ98a] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In *CAV'98*, Vancouver (B.C.), 1998. LNCS 1427, Springer Verlag.

[CJ98b] Hubert Comon and Yan Jurski. Multiple Counters Automata, Safety Analysis and Presburger Arithmetic. In *Proceedings of the 10th International Conference on Computer Aided Verification*, volume 1427, pages 268 – 279. Lecture Notes in Computer Science, 1998.

[CK03] Edmund Clarke and Daniel Kroening. Hardware verification using ANSI-C programs as a reference. In *Proceedings of ASP-DAC 2003*, pages 308–311. IEEE Computer Society Press, January 2003.

[cKKNP01] Çetin Kaya Koç, David Naccache, and Christof Paar, editors. *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*. Springer, 2001.

[CKL04] Edmund Clarke, Daniel Kroening, and Flavio Lerda. A tool for checking ANSI-C programs. In Kurt Jensen and Andreas Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 168–176. Springer, 2004.

[CKRT03] Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *Proceedings of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS 2003)*, pages 261–270. IEEE, Computer Society Press, 2003.

[CLS03] Hubert Comon-Lundh and Vitaly Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *Proceedings of the eighteenth annual IEEE symposium on Logic In Computer Science*. IEEE Computer Society Press, June 2003.

[CO01] C. Calcagno and P.W. O'Hearn. On garbage and program logic. In *Foundations of Software Science and Computation Structures*, pages 137 – 151, 2001.

[Con71] J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, 1971.

[CR04] Sigmund Cherem and Radu Rugina. Region analysis and transformation for Java programs. In *ISMM'04*, SIGPLAN. ACM, October 2004.

[CT05] A. Cavarra and J. Trost. Agedis Modelling Language Tutorial. Public Deliverable 9.1.1 of the Agedis Project, april 2005.

[CW04] V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. Research Report RR-5341, INRIA, October 2004.

[dAH01] L. de Alfaro and T.A. Henzinger. Interface theories for component-based design. In *EMSOFT'01*. Springer, LNCS 2211, 2001.

[DC00] C. Dumas and P. Caspi. A PVS proof obligation generator for Lustre programs. In *7th International Conference on Logic for Programming and Automated Reasoning*, volume 1955 of *Lecture Notes in Artificial Intelligence*, 2000.

[DG] Rolf Drechsler and Daniel Grosse. Formal verification of LTL formulas for systemc designs.
`http://www.informatik.uni-bremen.de/grp/ag-ram/doc/konf/iscas03_verification_systemc.pdf.`

[DGG97] D. Dams, R. Gerth, and O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions in Programming Languages and Systems*, 19(2), 1997.

[DGN02] Zhen Ru Dai, Jens Grabowski, and Helmut Neukirchen. Timed ttcn-3 - a real-time extension for ttcn-3. In *TestCom*, volume 210 of *IFIP Conference Proceedings*, pages 407–424. Kluwer, 2002.

[DJPV03] Werner Damm, Bernhard Josko, Amir Pnueli, and Angelika Votintseva. Understanding UML: A formal semantics of concurrency and communication in real-time UML. In Frank de Boer, Marcello Bonsangue, Susanne Graf, and Willem-Paul de Roever, editors, *Proceedings of the 1st Symposium on Formal Methods for Components and Objects (FMCO 2002)*, volume 2852 of *LNCS Tutorials*, pages 70–98, 2003.

[DJPV05] Werner Damm, Bernhard Josko, Amir Pnueli, and Angelika Votintseva. A discrete-time uml semantics for concurrency and communication in safety-critical applications. *Science of Computer Programming*, 2005. (to appear).

[DPS99] C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific Publishing Company, 1999.

[DY83] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[DY96] C. Daws and S. Yovine. Reducing the Number of Clock Variables of Timed Automata. In *Proceedings of RTSS'96 (Washington, DC, USA)*, pages 73–82. IEEE Computer Society Press, December 1996.

[End72] H. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[FGK+96] Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. Cadp (cæsar/aldebaran development package): A protocol validation and verification toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *LNCS*, pages 437–440. Springer Verlag, August 1996.

[FJJ+96] Jean-Claude Fernandez, Claude Jard, Thierry Jéron, Laurence Nedelka, and César Viho. An experiment in automatic generation of test suites for protocols with verification technology. *SCP*, 1996. Special issue on Industrially Relevant Applications of Formal Analysis Techniques. Also available as Inria Research Report RR-2923.

[FJJV96] J.C. Fernandez, C. Jard, T. Jéron, and C. Viho. Using On-the-fly Verification Techniques for the Generation of Test Suites. In *Proceedings of CAV'96*, number 1102 in LNCS. Springer, 1996.

[Flo67] R. W. Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science*, 19:19–32, 1967.

[FP] Judith N. Froscher and Charles N. Jr. Payne. The handbook for the computer security certification of trusted systems. NRL Technical Memorandum 5540.

[FR74] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. *Complexity of Computation (AMS-SIAM Proceedings)*, 7(41), 1974.

[FS00] A. Finkel and G. Sutre. An algorithm constructing the semilinear post* for 2-dim reset/transfer vass. In *25th Int. Symp. Math. Found. Comp. Sci. (MFCS'2000)*, Bratislava, Slovakia, August 2000. LNCS 1893, Springer Verlag.

[G31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173 – 198, 1931.

[GGP03] Uwe Glässer, Reinhard Gotzhein, and Andreas Prinz. The formal semantics of sdl-2000: Status and perspectives. *Computer Networks*, 42(3):343–358, 2003.

[GH96] Y. Gurevich and J. Huggins. The Railroad Crossing Problem: An Experiment with Instantaneous Actions and Immediate Reactions. In *Proceedings*

of CSL'95 (Computer Science Logic), volume 1092 of *LNCS*, pages 266–290. Springer, 1996.

[GH02] Jens Grabowski and Dieter Hogrefe. Sdl- and msc-based specification and automated test case generation for inap. *Telecommunication Systems*, 20(3-4):265–290, 2002.

[GL00] Jean Goubault-Larrecq. A method for automatic cryptographic protocol verification. In *International Workshop on Formal Methods for Parallel Programming: Theory and Applications*, volume 1800 of *LNCS*, 2000.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, April 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ron L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[God97] Patrice Godefroid. Model checking for programming languages using VeriSoft. In ACM, editor, *Conference record of POPL '97, the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages: papers presented at the symposium, Paris, France, 15–17 January 1997*, pages 174–186, New York, NY, USA, 1997. ACM Press.

[God05] Patrice Godefroid. Dynamic partial-order reduction for model checking software. In ACM, editor, *Conference record of POPL '05, the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, New York, NY, USA, 2005. ACM Press.

[GS00] David Gay and Bjarne Steensgaard. Fast escape analysis and stack allocation for object-based programs. In *CC'00*, pages 82–93. Springer-Verlag, 2000.

[Gur91] Y. Gurevich. Evolving Algebras. A Tutorial Introduction. *Bulletin of EATCS*, 43:264–284, 1991.

[Gur95] Y. Gurevich. Evolving Algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.

[Hal93] N. Halbwachs. *Synchronous programming of reactive systems.* Kluwer Academic Pub., 1993.

[Har87] D. Harel. Statecharts : A visual approach to complex systems. *Science of Computer Programming*, 8:231–275, 1987.

[Hav99] Klaus Havelund. Java pathfinder: A translator from java to promela. September 30 1999.

[HKN01] Dieter Hogrefe, Beat Koch, and Helmut Neukirchen. Some implications of msc, sdl and ttcn time extensions for computer-aided test generation. In *SDL Forum*, volume 2078 of *Lecture Notes in Computer Science*, pages 168–181, 2001.

[HLR92] N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying critical systems by means of the synchronous data-flow programming language LUSTRE. *IEEE Transactions on Software Ingeneering, Special Issue on the Specification and Analysis of Real-Time Systems*, September 1992.

[HM06] N. Halbwachs and L. Mandel. Simulation and verification of asynchronous systems by means of a synchronous model. In *Sixth International Conference on Application of Concurrency to System Design, ACSD 2006*, Turku, Finland, June 2006.

[HMG06] N. Halbwachs, D. Merchat, , and L. Gonnord. Some ways to reduce the space dimension in polyhedra computations. *Formal Methods in System Design*, to appear, 2006.

[HNO97] L. Hammond, B. A. Nayfeh, and K. Olukotun. A single-chip multiprocessor. *IEEE Computer*, September 1997.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–585, October 1969.

[Hol99] G. J. Holzmann. The model-checker SPIN. *IEEE Trans. on Software Engineering*, 23(5), 1999.

[HP98] D. Harel and M. Politi. *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*. McGraw-Hill, 1998.

[HPV00a] Klaus Havelund, Seungjoon Park, and Willem Visser. Java pathfinder - second generation of a java model checker. May 27 2000.

[HPV00b] Klaus Havelund, John Penix, and Willem Visser, editors. *The Temporal Rover and the ATG Rover*, volume 1885 of *Lecture Notes in Computer Science*. Springer, 2000.

[HR] K. Havelund and G. Roşu. Runtime verification workshop series, starting 2001. ENTCS Vols. 55, 70(4), and 89(2).

[HvdZ03] J. Hooman and M.B. van der Zwaag. A Semantics of Communicating Reactive Objects with Timing. In *Proceedings of SVERTS'03 (Specification and Validation of UML models for Real Time and Embedded Systems)*, San Francisco, October 2003.

[ISO92] ISO/IEC. Conformance Testing Methodology and Framework, iso/iec9646, 1992.

[IT03] ITU-T. Recommendation Z.100. Specification and Description Language (SDL). Technical report, International Telecommunication Union – Standardization Sector, 2003.

[JHR99] B. Jeannet, N. Halbwachs, and P. Raymond. Dynamic partitioning in analyses of numerical properties. In A. Cortesi and G. Filé, editors, *Static Analysis Symposium, SAS'99*, Venice (Italy), September 1999. LNCS 1694, Springer Verlag.

[JJJR94] C. Jard, T. Jeron, G.-V. Jourdan, and J.-X. Rampon. A general approach to trace checking in distributed computing systems. In *14th International Conference on Distributed Computing Systems, Poznan, Pologne*, pages 396–403. IEEE Computer Society Press, June 1994.

[JLRM94] M. Jourdan, F. Lagnier, P. Raymond, and F. Maraninchi. A multiparadigm language for reactive systems. In *5th IEEE International Conference on Computer Languages*, Toulouse, May 1994. IEEE Computer Society Press.

[JM99] T. Jéron and P. Morel. Test generation derived from model-checking. In *Computer-Aided Verification (CAV'99)*, number 1633 in LNCS, pages 108–122, 1999.

[Jou94] M. Jourdan. *Etude d'un environnement de programmation et de vérification des systèmes réactifs, multi-langages et multi-outils.* PhD thesis, UJF, September 1994.

[KS94] N. Klarlund and M. I. Schwartzbach. Graphs and decidable transductions based on edge constraints. In *Proc. 19th Colloquium on Trees and Algebra in Programming*, pages 187 – 201, 1994.

[Kur94] R.P. Kurshan. *Computer-Aided Verification of Coordinating processes, the automata theoretic approach.* Princeton Series in Computer Science. Princeton University Press, 1994.

[Lan92] William Landi. Undecidability of static analysis. *ACM Letters on Programming Languages and Systems (LOPLAS)*, 1(4):323 – 337, December 1992.

[Lau04] Peeter Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *IEEE Symposium on Security and Privacy*, pages 71–85, 2004.

[LGS+95] C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6(1), 1995.

[Lip76] Leonard Lipshitz. The diophantine problem for addition and divisibility. *Transaction of the American Mathematical Society*, 235:271 – 283, January 1976.

[LKK+99] Insup Lee, Sampath Kannan, Moonjoo Kim, Oleg Sokolsky, and Mahesh Viswanathan. Runtime assurance based on formal specifications. In *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications.*, Las Vegas, June 28-July 1st 1999.

[Low95] G. Lowe. An attack on the Needham-Schroeder public-key authentification protocol. *Information Processing Letters*, 56(3):131–133, 1995.

[Low02] Gavin Lowe. Analysing protocols subject to guessing attacks. In *Proc. of the Workshop on Issues in the Theory of Security (WITS'02)*, 2002.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *STTT*, 1(1-2):134–152, 1997.

[LRG01] David Lugato, Nicolas Rapin, and Jean-Pierre Gallois. Verification and tests generation for SDL industrial specifications with the AGATHA toolset. In *Real-Time Tools Workshop affiliated to CONCUR 2001, Aalborg, Denmark*, 2001.

[LTC+03] A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, and R. Guerrieri. A vliw processor with reconfigurable instruction set for embedded applications. *IEEE J. of Solid-state Circuits*, 11(37), 2003.

[LX00] E. Lee and Y. Xiong. System-level types for component-based design. Technical report, Technical Memorandum UCB/ERL M00/8, University of California, Berkeley, 2000.

[LY96] D. Lee and M. Yannakakis. Principles and methods of testing finite state machines - A survey. *Proceedings of the IEEE*, 84:1090–1126, 1996.

[Mar97] F. Maraninchi. Modélisation et validation des systèmes réactifs : un langage synchrone à base d'automates. Document d'habilitation à diriger des recherches, Université Joseph Fourier, Grenoble, May 1997.

[Mat70] Yuri Matiyasevich. Enumerable sets are diophantine. *Journal of Sovietic Mathematics*, (11):354 – 358, 1970.

[MC01] W. Milam and A. Chutinan. Model composition and analysis challenge problems. Technical report of Mobies project. Available at: http://vehicle.me.berkeley.edu/mobies, 2001.

[McM93] K. L. McMillan. *Symbolic Model Checking.* PhD thesis, Boston, 1993.

[Mea03] C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE Journal on Selected Areas in Communication*, 21(1):44–54, January 2003.

[Mea04] C. Meadows. Ordering from satan's menu: a survey of requirements specification for formal analysis of cryptographic protocols. *Sci. Comput. Program.*, 50(1-3):3–22, 2004.

[MG00] F. Maraninchi and F. Gaucher. Step-wise + algorithmic debugging for reactive programs: Ludic, a debugger for lustre. In *AADEBUG'2000 – Fourth International Workshop on Automated Debugging*, Munich, August 2000.

[Mil78] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *TCS*, 25(3), July 1983.

[Min67] M. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.

[MMM06] Matthieu Moy, Florence Maraninchi, and Laurent Maillet-Contoz. LusSy: an open tool for the analysis of systems-on-a-chip at the transaction level. *Design Automation for Embedded Systems*, 2006. special issue on SystemC-based systems.

[Moo56] E.F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies*, number 34. Princeton University Press, 1956.

[mop] Mop website. http://fsl.cs.uiuc.edu/mop.

[MP06] Florence Maraninchi and Marc Pouzet, editors. *SLAP, Synchronous Languages, Applications, and Programming*, ENTCS, Electronic Notes in Theoretical Computer Science, April 2006.

[MR97] F. Maraninchi and Y. Rémond. Compositionality criteria for defining mixed-styles synchronous languages. In *International Symposium: Compositionality - The Significant Difference*, Malente (Holstein, Germany), September 1997. Springer Verlag.

[MR98] F. Maraninchi and Y. Rémond. Mode-automata: About modes and states for reactive systems. In *European Symposium On Programming*, Lisbon (Portugal), March 1998. Springer Verlag, LNCS 1381.

[MR00a] F. Maraninchi and Y. Rémond. Applying formal methods to industrial cases: The language approach (the production-cell and mode-automata). In *5th International Workshop on Formal Methods for Industrial Critical Systems (FMICS)*, Berlin, April 2000. `http://www.inrialpes.fr/vasy/fmics`.

[MR00b] F. Maraninchi and Y. Rémond. Running-modes of real-time systems: A case-study with mode-automata. In *12th Euromicro Conference on Real-Time Systems*, Stockholm (Sweden), June 2000. IEEE Computer Society Press.

[MRR00] F. Maraninchi, Y. Rémond, and Y. Raoul. Matou: An implementation of mode-automata. In *International Conference on Compiler Construction*, Berlin (Germany), March 2000. Springer Verlag.

[MSDH01] R. Münzenberger, F. Slomka, M. Dörfel, and R. Hofmann. General approach for the specification of real-time systems with SDL. In *Proceedings of SDL forum 2001*, LNCS 2078, 2001.

[MTMC99] A. Mitschele-Thiehl and B. Müller-Clostermann. Performance engineering of SDL/MSC systems. *Computer Networks 31(17): 1801-1815*, 1999.

[MW04] D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proceedings of the Theory of Cryptography Conference*, pages 133–151. Springer, 2004.

[MYRS05] R. Manevich, E. Yahav, G. Ramalingam, and M. Sagiv. Predicate abstraction and canonical abstraction for singly-linked lists. In *Proc. 6th Intl. Conference on Verification, Model Checking and Abstract Interpretation*, pages 181 – 198, 2005.

[OMG02] OMG. Response to the OMG RFP for Schedulability, Performance and Time, v. 2.0. OMG document ad/2002-03-04, March 2002.

[OMG03a] OMG. Standard UML Profile for Schedulability, Performance and Time, v. 1.0. OMG document formal/2003-09-01, September 2003.

[OMG03b] OMG. UML 2.0 superstructure proposal v.2.0. Technical report, OMG document ad/03-01-02, January 2003.

[ORY99] P.W. O'Hearn, J.C. Reynolds, and H. Yang. The logic of bunched implications. *Bulletin of Symbolic Logic*, pages 215 – 244, 1999.

[Pan01] P.K. Pandya. Specifying and deciding quantified discrete-time duration calculus formulae using dcvalid. In *Real-Time Tools, RTTOOLS'2001*, Aalborg, August 2001.

[Pha94] M. Phalippou. Test Sequence Generation using Estelle or SDL Structure Information. In D. Hogrefe and S. Leue, editors, *Proceedings of FORTE/PSTV'94 (Bern, Switzerland)*, pages 405–420, October 1994.

[Pre29] Mojzesz Presburger. Über die Vollstandigkeit eines gewissen Systems der Arithmetik. *Comptes rendus du I Congrés des Pays Slaves*, Warsaw 1929.

[PYAB+05] Paul P. Youn, B. Adida, M. Bond, J. Clulow, J. Herzog, L. Jonathan, a. Lin, R. L. Rivest, and Ross Anderson. Robbing the bank with a theorem prover. Technical Report UCAM-CL-TR-644, University of Cambridge, Computer Laboratory, August 2005.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[Ram94] G. Ramalingam. The undecidability of aliasing. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1467 – 1471, 1994.

[RdBJ00] V. Rusu, L. du Bousquet, and T. Jéron. An approach to symbolic test generation. In *Integrating Formal Methods (IFM'00)*, number 1945 in LNCS, pages 338–357, 2000.

[Rey02] John C. Reynolds. Separation logic: A logic for shared mutable data structures. In *Proc 17th IEEE Symposium on Logic in Computer Science*, 2002.

[RHR91] C. Ratel, N. Halbwachs, and P. Raymond. Programming and verifying critical systems by means of the synchronous data-flow programming language LUSTRE. In *ACM-SIGSOFT'91 Conference on Software for Critical Systems,* New Orleans, December 1991.

[RMJ04] V. Rusu, H. Marchand, and T. Jéron. Verification and symbolic test generation for safety properties. Technical Report 5285, INRIA, august 2004. Available at `http://www.inria.fr/rrrt/rr-5285.html`.

[RMT+04] V. Rusu, H. Marchand, V. Tschaen, T. Jeron, and B. Jeannet. From safety verification to safety testing. In *Testing of Communicating Systems (TestCom04)*, number 2978 in LNCS, 2004.

[Rob49] Julia Robinson. Definability and decision problems in arithmetic. *The Journal of Symbolic Logic*, 14(2):98 – 114, June 1949.

[Ros36] B. Rosser. Extensions of some theorems of Gödel and Church. *The Journal of Symbolic Logic*, 1:87 – 91, 1936.

[RT01] M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *IEEE Computer Security Foundations Workshop*, 2001.

[Rug04] R. Rugina. Quantitative shape analysis. In *Proc. 11th Intl. Static Analysis Symposium (SAS04)*, pages 228 – 245, 2004.

[RWNH98] P. Raymond, D. Weber, X. Nicollin, and N. Halbwachs. Automatic testing of reactive systems. In *19th IEEE Real-Time Systems Symposium*, Madrid, Spain, December 1998.

[SAM06] D. Stauch, K. Altisen, and F. Maraninchi. Interference of Larissa aspects. In *FOAL : Foundations of Aspect-Oriented Languages workshop (an AOSD'06 satellite event)*, March 2006.

[SF02] T. Sakunkonchak and M. Fujita. Verification of synchronization in SpecC description with the use of difference decision diagrams. In *FDL*, 2002.

[SMM+06] Ludovic Samper, Florence Maraninchi, Laurent Mounier, Erwan Jahier, and Pascal Raymond. On the importance of modeling the environment when analyzing sensor networks. In *3rd International Workshop on Wireless Ad-hoc and Sensor Networks (IWWAN'06)*, New York, USA, June 2006.

[SMMM06] Ludovic Samper, Florence Maraninchi, Laurent Mounier, and Louis Mandel. Glonemo: Global and accurate formal models for the analysis of ad-hoc sensor networks. In *InterSense: First International Conference on Integrated Internet Ad hoc and Sensor Networks*, Nice, France, May 2006. IEEE.

[Son99] D. Song. Athena: A new efficient automatic checker for security protocol analysis. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW '99)*, pages 192–202, Washington - Brussels - Tokyo, June 1999. IEEE.

[SRA02] Koushik Sen, Grigore Roşu, and Gul Agha. Online efficient predictive safety analysis of multithreaded programs. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS'04)*, Lecture Notes in Computer Science. Springer, 2002. to appear.

[SRA03] K. Sen, G. Roşu, and G. Agha. Runtime safety analysis of multithreaded programs. In *ACM SIGSOFT Conference on the Foundations of Software Engineering.European Software Engineering Conference (FSE / ESEC'03)*, Helsinki, Finland, 2003.

[SRW98] M. Sagiv, T. Reps, and R. Wilhelm. Solving shape-analysis problems in languages with destructive updating. In *ACM Transactions on Programming Languages and Systems*, pages 1 – 50, 1998.

[SRW02] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, pages 217 – 298, 2002.

[SVAR04] Koushik Sen, Abhay Vardhan, Gul Agha, and Grigore Roşu. Efficient decentralized monitoring of safety in distributed systems. In *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. IEEE/ACM, 2004. Edinburgh, Scotland, UK.

[Tho90] W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Methods and Semantics*, pages 134–191. Elsevier Science Publishers B. V., 1990.

[TR04] P. Thati and G. Roşu. Monitoring Algorithms for Metric Temporal Logic. In *Workshop on Runtime Verification (RV'04)*, volume to appear of *ENTCS*. Elsevier, 2004.

[Tre99a] J. Tretmans. Testing concurrent systems: A formal approach. In *CONCUR'99*, number 1664 in LNCS, pages 46–65, 1999.

[Tre99b] J. Tretmans. Testing concurrent systems: A formal approach. In *CONCUR'99*, volume 1664 of *LNCS*. Springer, 1999.

[TT97] Mads Tofte and Jean-Pierre Talpin. Region-based memory management. *Information and Computation*, February 1997.

[URS03] T. Ungerer, B. Robic, and J. Silc. A survey of processors with explicit multithreading. *ACM Computing Surveys*, 1(35):29–63, March 2003.

[Var93] P. Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207, February 1993.

[War03] B. Warinschi. A computational analysis of the needham-schroeder(-lowe) protocol. In *Proceedings of 16th Computer Science Foundation Workshop*, pages 248–262. ACM Press, 2003.

[WB98] P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *Proceedings of CAV'98 (Vancouver)*, volume 1427 of *LNCS*, pages 88–97. Springer, June 1998.

[Woo81] A. Woods. Some problems in logic and number theory and their connections. Technical report, Phd Thesis, University of Manchester, 1981.

[WR99] John Whaley and Martin Rinard. Compositional pointer and escape analysis for Java programs. In *ACM OOPSLA'99*, volume 34(10) of *SIGPLAN*, pages 187–206. ACM, October 1999.

[Yov97] S. Yovine. KRONOS: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer*, 1(1-2), December 1997.

[ZSM04] T. Zhang, H. Sipma, and Z. Manna. Decision procedures for recursive data structures with integer constraints. In *Proc. 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, pages 152 – 167, 2004.