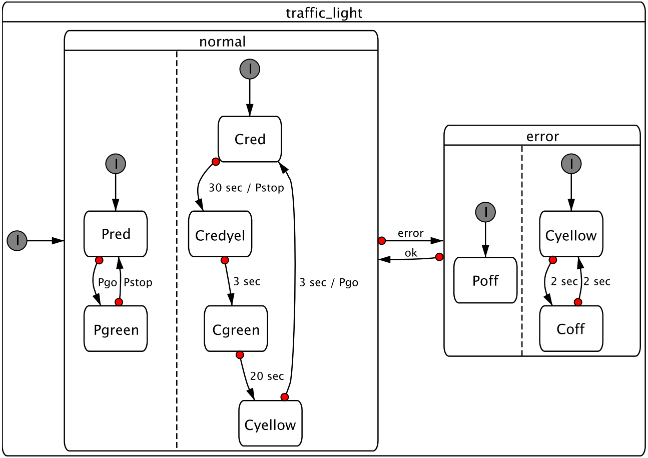
Improvements for Developing Statecharts

Steffen Prochnow

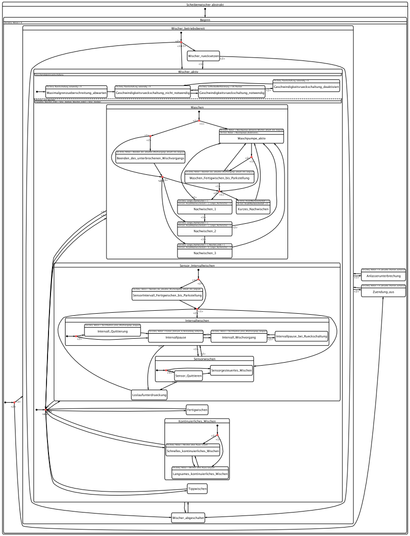
Distributed and Complex Systems Group
VERIMAG Laboratory

DCS Seminar, March, 2009

Simple Statecharts Example



Statechart Example: Window Wiper



Daimler Chrysler AG, Research REI/SM

Motivation and Purpose

Motivation:

- Statecharts possess high complexity (combinations of components, dependencies, system dynamics, concurrency)
- tools for modeling Statecharts provide restricted facilities to enter and understand complex system behavior

Purpose:

- formulation of improvements for easy modeling, analyzing and understanding complex Statecharts
- establishment of these improvements in a highly configurable tool for modeling and simulation
- experimental analysis of automatic Statecharts layout and Statechart editing techniques

Outline

- Motivation and Purpose
- Difficulties with Statecharts and their Improvements
- An Empirical Study
- SPEEDS
- Summary and Conclusion

Difficulties with Statecharts

1st Observation:

Graphical models are nice to browse,
but hard to write.

Creating Graphical Models

State of the Practice

- WYSIWYG editors to create graphical models
- some editors offer alignment tools (ArgoUML)
- in initial phase, often resort to paper and pencil
- creating and maintaining graphical models is time consuming

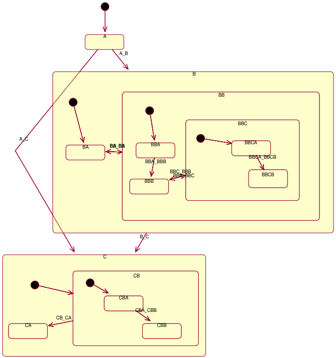
Editing Action Sequence

1. if needed, create free space.
2. focus on a Statechart element for modification resp. supplementation.
3. apply an editing schema.
4. if needed, rearrange the modified chart to improve readability.

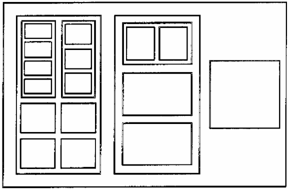
Alternatives to Accelerate Editing

1. add-on to traditional editors
   - create quick-and-dirty graphical model (WYSIWYG) ...
   - ... then apply automated layout
2. macro-based modeling
   - direct manipulation of Statechart structure using keyboard macros
   - employs Statechart production rules
   - ensures syntax-consistency
   - automated layout of graphical elements
3. text-based modeling
   - model synthesis from textual description
   - automated placement of graphical elements
   - separate content from layout

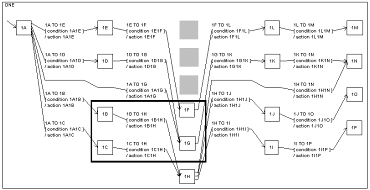
Statechart Layout



Autolayout from Rational Rose

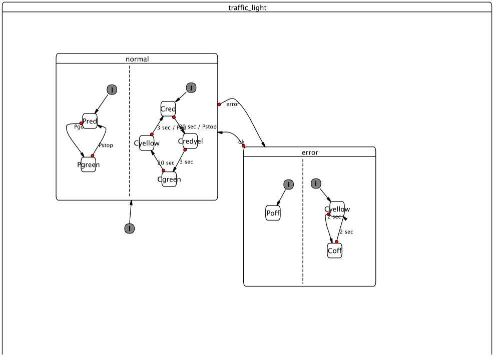


Blobs (Harel and Yashchin, 2002)

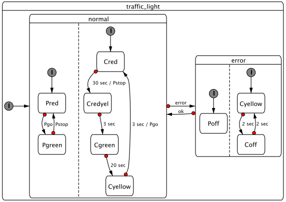


Vista (Castelló, Mili, and Tollis, 2002)

Statechart Layout



(a) original layout



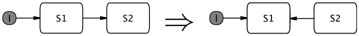
(b) after auto-layout

- Steffen Prochnow and Reinhard von Hanxleden.
  Comfortable modeling of complex reactive systems.
  In *Proceedings of Design, Automation and Test in Europe (DATE'06)*, Munich, Germany, March 2006.

Macro-Based Modeling

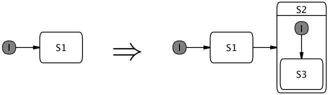
- identified nine main Statechart editing schemata
- three categories: 1. *creation*, 2. *modification*, 3. *deletion*

(a) insertion of a simple successor state

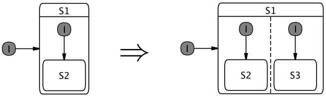


(b) modification of transition direction

(c) deletion of a Statechart element

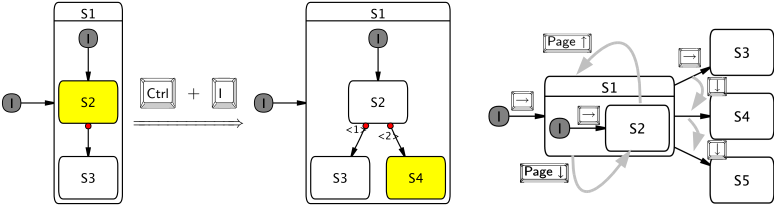


(d) insertion of hierarchical successor state



(e) insertion of a parallel region

Macro-Based Modeling



(a) example of applying the “insert simple successor state” schema

(b) navigation with key strokes

- Steffen Prochnow and Reinhard von Hanxleden.
  Statechart development beyond WYSIWYG.
  In *Proceedings of the ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS'07)*, Nashville, TN, USA, October 2007.

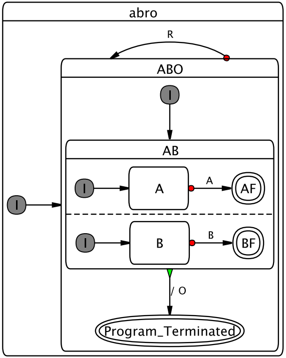
Text-Based Modeling

KIel Statechart extension of doT:

- implicit declarations in *dot*,
- hierarchy construction in *Argos*,
- orthogonal construction in *Esterel*, and
- ability to describe different Statechart dialects

```
statechart abro[model="Esterel Studio";version="5.0"]{
 input A;
 input B;
 input R;
 output O;
 {
  ->ABO;
  ABO{
   AB{
     ->A;
     A->AF[type=sa;label="A"];
     AF[type=final];
    ||
     ->B;
     B->BF[type=sa;label="B"];
     BF[type=final];
   };
   ->AB;
   AB->Program_Terminated[type=nt;label="/ O"];
   Program_Terminated[type=final];
  };
  ABO->ABO[type=sa;label="R"];
 };
};
```

(a) KIT description representation



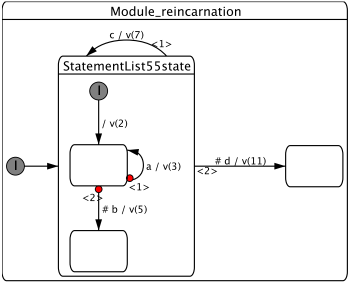
(b) SSM representation.

Language Transformation

```
module reincarnation :
input a, b, c, d;
output v := 1 : combine integer
        with *;

loop
  weak abort
    emit v(2);
    loop
      await
      case a do
        emit v(3)
      case immediate b do
        emit v(5);
        halt
      end await
    end loop
  when
  case c do
    emit v(7)
  case immediate d do
    emit v(11);
    halt
  end weak abort
end loop
end module
```

(a) Esterel source



(b) Synthesized SSM

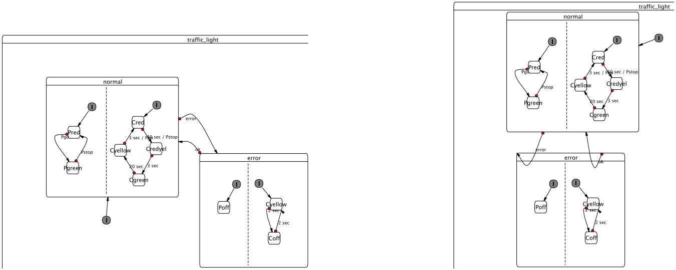
- Steffen Prochnow, Claus Traulsen, and Reinhard von Hanxleden.
  Synthesizing Safe State Machines from Esterel.
  In *Proceedings of ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'06)*, Ottawa, Canada, June 2006.

Difficulties with Statecharts

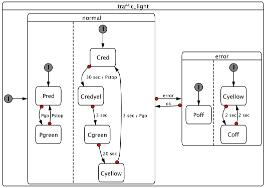
2nd Observation:

Graphical languages are appealing,
but not effective enough.

Use of Statechart Normal Form



(a) Statecharts with manually placed elements



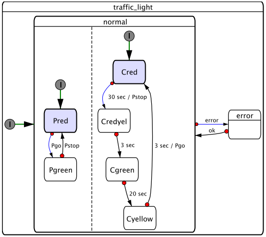
(b) automatic layout confirming to the Statechart Normal Form

Difficulties with Statecharts

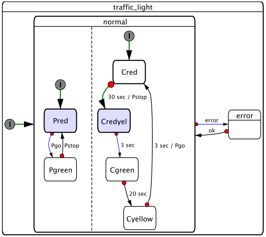
3rd Observation:

Graphical languages are good for understanding structures, but limited for analyzing dynamics.

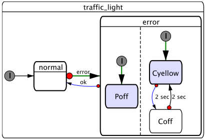
Dynamic Focus-and-Context Simulation



(a) entering the initial state



(b) cars get a red/yellow light



(c) entering the error state

Statistics for this example:

- 15 states
- 10 state configurations
- 2 views

Difficulties with Statecharts

4$^{\text{th}}$ Observation:

Graphical languages are easy to model syntactically correct, but limited for avoiding modeling errors.

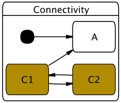
Checking Statechart Style

Statechart style guide:

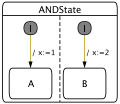
- operational instructions for humans and configuration for automated analysis
- set of 41 wellformedness-, syntactic, and semantic rules
- defines a subset of the language Statechart

Statechart style checking:

- based on defined style guide
- allows to express new rules in OCL or in Java
- theorem prover for more advanced checks



Connectivity



Race Conditions

Syntactic rules

Dwelling

Semantic rules

- Steffen Prochnow, Gunnar Schaefer, Ken Bell, and Reinhard von Hanxleden.
  Analyzing robustness of UML State Machines.
  In *Proceedings of the Workshop on Modeling and Analysis of Real-Time and Embedded Systems (MARTES'06), held in conjunction with the 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML 2006)*, Genua, Italy, October 2006.

The KIEL Modeling Tool

Kiel **I**ntegrated **E**nvironment for **L**ayout

- employs generic concept of Statecharts
- several layout heuristics
- micro-step simulation and visualization
- has been used in teaching "System Modeling and Synchronous Languages"
- URL: `http://rtsys.informatik.uni-kiel.de/~rt-kiel`

Outline

An Empirical Study

Goals

1. investigation of differences in editing using an WYSIWYG Statechart editor, the KIEL macro editor, and the KIT editor
2. comparison of the readability of Statechart layouts created by the KIEL layouter and other Statechart layouts

Subjects

- graduate-level students attending the lecture "Model-Based Design and Distributed Real-Time Systems"
- conducted two series of experiments, at beginning and end of semester

Support

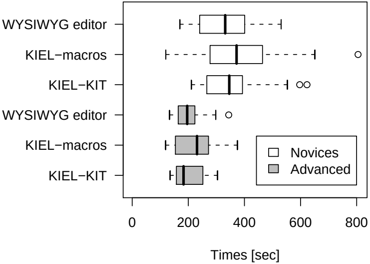
- ... in experiment design and statistical analysis by Prof. Golz (Department of Psychology, CAU Kiel)

Statechart Editing

Statechart Creation: Hypotheses

Novices will need less time to create a Statechart using the WYSIWYG editor. Advanced will need less time using the KIT editor.

Results

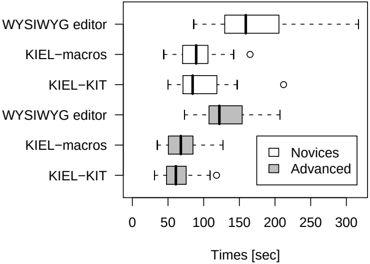


Distribution of times for creating a new Statechart

Statechart Modification: Hypothesis

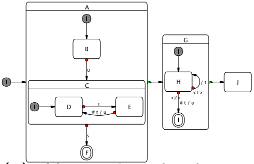
Statechart modification using the KIT editor or the KIEL macro editor is faster than using the WYSIWYG editor.

Results

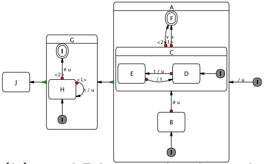


Distribution of times for modifying an existing Statechart

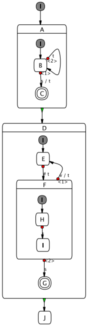
Statechart Layout Alternatives



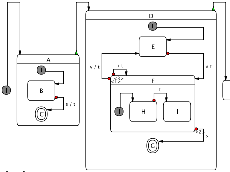
(a) Alternating dot layout (ADL)



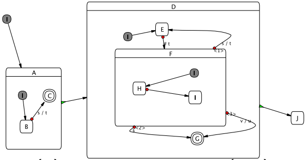
(b) ADL backwards (ADBL)



(c) Linear layer layout (LLL)



(d) Alternating linear layout (ALL)



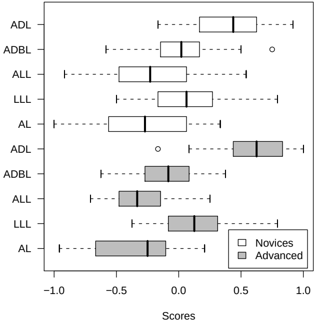
(e) Arbitrary layout (AL)

Statechart Layout Assessments

Statechart Aesthetics: Hypotheses

We expect the best scores for Statecharts laid out according certain layout styles realized by the KIEL Statechart layouter.

Results

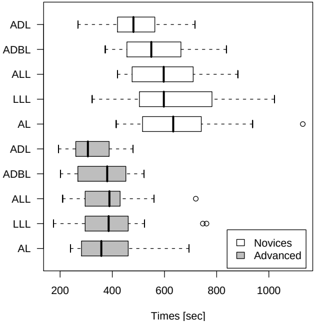


Distribution of subjective Statechart layout scores

Statechart Aesthetics: Hypotheses

Well arranged Statecharts, as laid out by the KIEL Statechart layouter are better (faster) understandable than arbitrary layouts.

Results



Distribution of Statechart comprehension times

Related Work

Statechart layout

- David Harel and Gregory Yashchin.
  An algorithm for blob hierarchy layout.
  The Visual Computer, 18:164–185, 2002.

Statechart layout framework

- Rodolfo Castelló, Rym Mili, and Ioannis G. Tollis.
  A framework for the static and interactive visualization for statecharts.
  Journal of Graph Algorithms and Applications, 6(3):313–351, 2002.

graph Layout

- Emden Gansner, Eleftherios Koutsofios, and Stephen North.
  Drawing graphs with dot.
  Technical report, AT&T Bell Laboratories, Murray Hill, NJ, USA, February 2002.

semantic focus-and-context

- Oliver Köth and Mark Minas.
  Structure, Abstraction, and Direct Manipulation in Diagram Editors.
  In *DIAGRAMS '02: Proceedings of the Second International Conference on Diagrammatic Representation and Inference*, pages 290–304, London, UK, 2002. Springer-Verlag.

Statechart style checking

- Martin Mutz.
  Eine durchgängige modellbasierte Entwurfsmethodik für eingebettete Systeme im Automobilbereich.
  Dissertation, Technische Universität Braunschweig, 2005.

empirical studies in graphical modeling

- T. R. G. Green and M. Petre.
  When visual programs are harder to read than textual programs.
  In *Human-Computer Interaction: Tasks and Organisation, Proceedings ECCE-6 (6th European Conference Cognitive Ergonomics)*, 1992.

Outline

SPEEDS

SPEEDS: **SPE**culative and **E**xploratory **D**esign in **S**ystem Engineering

- EU project
- supports tools for safety-critical embedded system design

Purpose:

- component-based construction and analysis of system models

Approach:

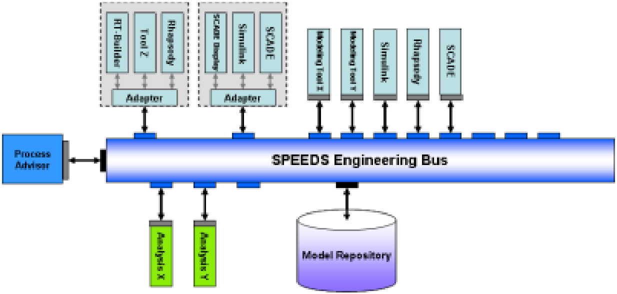
- **H**eterogeneous, **R**ich **C**omponent-based modeling of systems (HRC)
- rich Components: functional components with static interfaces + non-functional behavior constraints (e. g. the quality of services in terms of timing, resource, etc.)
- constraints: a pair of the form *(Assumption, Promise)* owned by a component
- a promise is guaranteed only if the corresponding assumption about the environment is fulfilled

SPEEDS Infrastructure

Consortium: potential end-users from aerospace and automotive industries, tool-vendors, and European research institutes:

- Airbus Deutschland GmbH
- Airbus France
- Israel Aerospace Industries Ltd. (formerly Israel Aircraft Industrie Ltd.)
- Carmeq
- Robert Bosch GmbH
- Magna Powertrain Engineering Center
- Knorr-Bremse Fekrendszerek Kft
- SAAB AB
- EADS
- Telelogic Israel Ltd.(formerly I-Logix)
- Esterel Technologies SA
- Extessy
- GEENSYS (formerly TNI)
- Université Joseph Fourier - VERIMAG
- PARADES GEIE
- Kuratorium OFFIS E.V.
- INRIA

SPEEDS Tool Interaction:



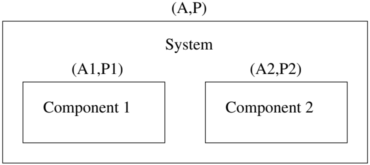
VERIMAG Contribution—Dominance Analysis

Purpose:

- verify compositional context dependent properties of systems

Approach:

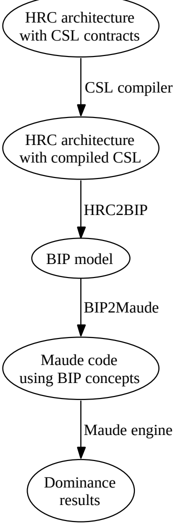
- verification of dominance between contracts



$(A_1, P_1) \mid\mid (A_2, P_2)$ dominates (A, P) w. r. t. *System*

- A, A_1, A_2: Assumptions
- P, P_1, P_2: Promises
- $(A, P), (A_1, P_1), (A_2, P_2)$: Contracts

Dominance Tools



HRC: component specification

- models provided by project partners

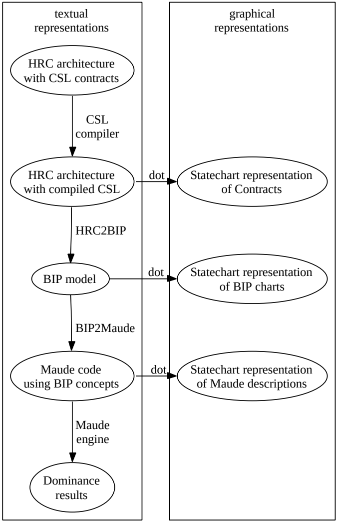
CSL: Constraint **S**pecification **L**anguage

Maude: language for equational and rewriting logic specification of systems

- used for checking deadlock freedom
- deadlock found: no dominance
- checking output: location of the problem

SPEEDS—Ongoing Work

- **bug fixing**
- improve expressiveness of Maude output
- visual debugging using graphical Statecharts representations
- code generation using BIP for hosted simulation
- syntactical simplification of transformed Statecharts



Summary and Conclusion

Summary

- KIEL: platform for experimenting with the pragmatics of graphical modeling
  - editing alternatives to WYSIWYG
  - paradigm to simulate complex Statecharts: semantic focus-and-context
  - uniform framework for robustness checking
  - empirical study
- SPEEDS: model transformation and dominance checking

Conclusion

- experimental studies show
  - efficiency of Statechart layout
  - good performance in modifying Statechart structure
- Statechart layout (Secondary Notation) can improve development process (comp. SPEEDS)
- improvements combine the best of two worlds

Thanks!
Questions or comments?

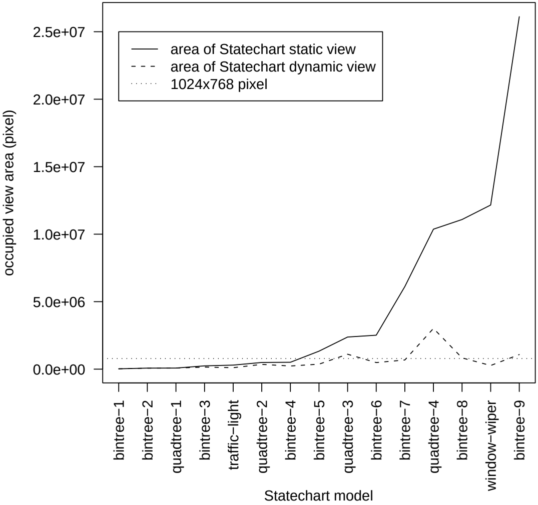
Appendix: Simulation and visualization performance of KIEL

MODEL	GRAPHICAL ELEMENTS			STATE SPACE		OCCUPIED AREA					COMPUTATION TIMES						
	States	Transitions	Total	Configurations	Views	Static view (pixel × pixel)		Dynamic view (pixel × pixel)		Reduction factor	Load (ms)	Next configuration (ms)		Layout (ms)		Display next view (ms)	
												min	max	min	max	min	max
traffic-light	15	25	40	10	2	652	460	398	268	0.36	828	31	67	1	24	213	356
window-wiper	36	82	118	80	12	4200	2895	870	311	0.02	1120	535	1497	1	50	497	1759
bintree-1	3	5	8	2	1	248	103	248	103	1	226	5	26	1	5	147	273
bintree-2	9	15	24	4	2	288	268	268	268	0.93	346	3	46	1	18	172	274
bintree-3	21	35	56	8	4	636	384	442	342	0.62	834	31	153	2	44	181	260
bintree-4	45	75	120	16	8	716	712	462	506	0.46	1446	36	188	1	28	264	436
bintree-5	93	155	248	32	16	1412	944	636	580	0.28	1772	41	250	1	32	272	532
bintree-6	189	315	504	64	32	1572	1600	656	744	0.19	2392	45	374	1	38	390	647
bintree-7	381	635	1016	128	64	2964	2064	830	818	0.11	2937	214	709	1	39	456	837
bintree-8	765	1275	2040	256	128	3284	3376	850	982	0.08	8636	555	1389	2	41	954	1802
bintree-9	1533	2555	4088	512	256	6068	4304	1024	1056	0.04	20695	1429	2640	2	48	1911	3504
quadtree-1	10	11	21	4	1	300	268	300	268	1	668	20	54	2	4	214	264
quadtree-2	50	55	105	64	4	696	712	696	506	0.71	1413	277	374	1	27	287	453
quadtree-3	210	231	441	16384	64	1488	1600	1488	744	0.47	1985	545	969	2	77	594	1834
quadtree-4	850	935	1785	1073741824	16384	3072	3376	3072	982	0.29	10186	2220	2852	3	127	3151	3633

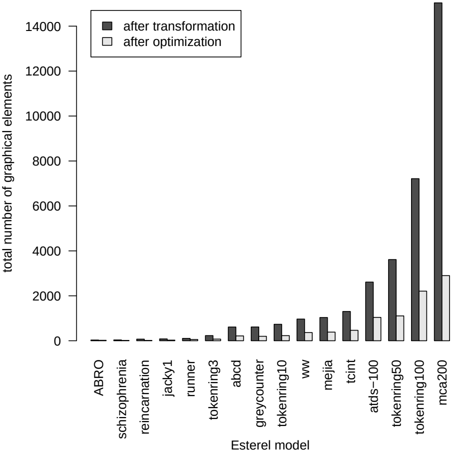
Appendix: Experimental results of SSM synthesis

model	Esterel	Safe State Machines											time	
		Before optimization					After optimization							
	Lines of code (loc)	States	Pseudo-states	Transitions	Total graphical elements (ges)	ges/loc	States	Pseudo-states	Transitions	Total graphical elements (ges)	ges/loc	SSM reduction factor	Transformation (ms)	Optimization (ms)
ABRO	7	12	8	12	32	4.57	8	4	8	20	2.86	0.63	861	56
schizophrenia	10	13	9	14	36	3.60	4	3	6	13	1.30	0.36	838	81
reincarnation	25	27	17	28	72	2.88	5	2	6	13	0.52	0.18	642	83
jacky1	27	31	19	33	83	3.07	11	5	12	28	1.04	0.34	913	93
runner	55	39	24	42	105	1.91	21	11	25	57	1.04	0.54	725	160
tokenring3	79	77	61	91	229	2.90	15	20	38	73	0.92	0.32	733	278
greycounter	82	211	148	254	613	7.48	42	50	106	198	2.41	0.32	789	593
abcd	101	231	130	250	611	6.05	78	41	97	216	2.14	0.35	943	731
tokenring10	247	245	194	294	733	2.97	43	62	122	227	0.92	0.31	1267	736
mejia	555	374	246	414	1034	1.86	127	76	181	384	0.69	0.37	3085	1266
tcint	687	475	285	543	1303	1.90	163	81	221	465	0.68	0.36	3382	1310
atds-100	948	961	558	1092	2611	2.75	352	184	504	1040	1.10	0.40	7046	2760
ww	1088	342	228	386	965	0.89	102	85	177	364	0.33	0.38	4470	1053
tokenring50	1207	1205	954	1454	3613	2.99	203	302	602	1107	0.92	0.31	6608	8148
tokenring100	2407	2405	1904	2904	7213	3.00	403	602	1202	2207	0.92	0.31	20910	25528
mca200	7269	5159	3931	5947	15037	2.07	179	925	1794	2898	0.40	0.19	61510	100594

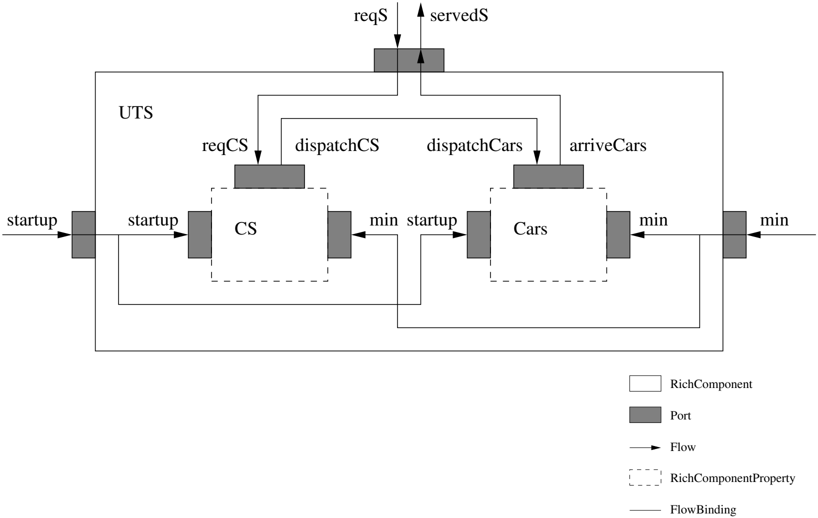
Appendix: Static and Dynamic View Areas



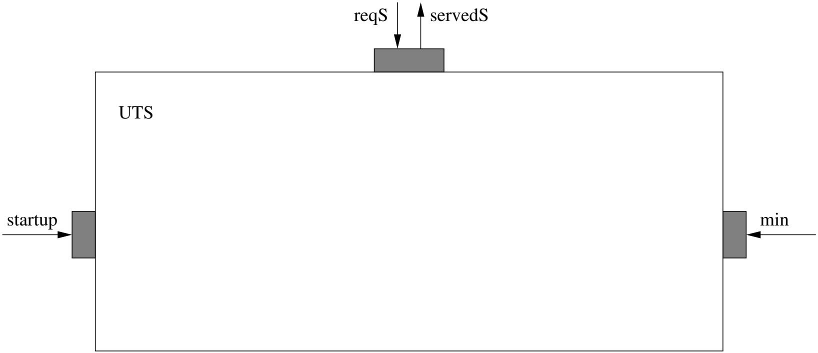
Appendix: Model Sizes before and after Optimization



Dominance Example—UTS

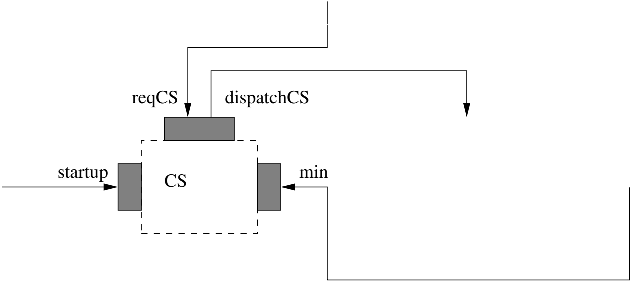


Dominance Example—UTS



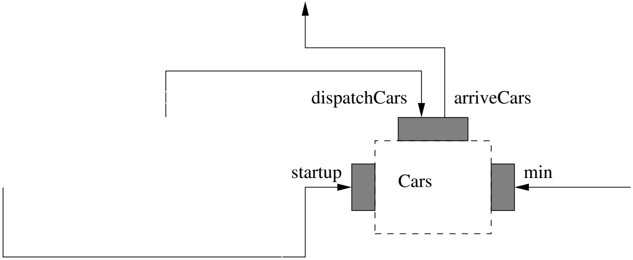
Assumption: `whenever [reqS] occurs [3#reqS] does not occur during following [(10,min)]`
Promise: `whenever [reqS] occurs [servedS] occurs within [(15,min)]`

Dominance Example—UTS

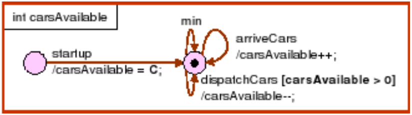


Assumption: `whenever [reqCS] occurs [2#reqCS] does not occur during following [(2,min)]`
Promise: `whenever [reqCS] occurs [dispatchCS] occurs within [(2,min)]`

Dominance Example—UTS



Assumption:



Promise: `whenever [reqCS] occurs [dispatchCS] does not occur during following [(2,min)]`

Dominance Example—UTS

Analysis Idea:
$(A_{CS}, P_{CS}) \mid\mid (A_{Car}, P_{Car})$ dominates (A_{UTS}, P_{UTS}) w. r. t. UTS