Project 22703: SYRF

**Deliverable 7.1**

# Hybrid Systems

---

## 1 Introduction

Synchronous programs are frequently used for controlling physical devices in process control applications. To analyse such embedded systems one can proceed along two paths

- Verifying the properties of the controller, which can come in any of the flavours of synchronous languages
- Verifying the properties of the controlled system, often expressed in terms of state variables in a model of the physical world

The first path covers well-formedness requirements on the control program (constructive semantics in Esterel, clock calculi in Lustre and Signal), as well as proofs of some internal properties of the controller (e.g. the valve is not shut unless the right mode and event combinations are present). This is "what we can do" based on existing models and techniques from computer science.

The second path covers the properties which are not easily verifyable using models from computer science alone, but are nevertheless critical to the intended functioning of the system. This is "what we want to do" in interaction with other disciplines e.g. control engineering. When successful, analysing systems for this types of properties results in side effects such as generating design parameters or optimal interfaces (event generators) between the environment and the controller. However, it must be borne in mind that the research area is substantially underdeveloped, and a systematic characterization of how to combine the techniques from the different disciplines is not available. The aim of the work in case studies is to increase our understanding of how to apply proof tools from different disciplines in appropriate contexts.

In Section 2, we propose a general framework which can be adopted for verifying properties of systems controlled by synchronous programs, and discuss its application in the work package 7 of the project during the first year.

Section 3 presents the modelling work performed in the case study so far, sections 4 and 5 present the technical results on modelling and verification of hybrid systems during the first year of work package 7.

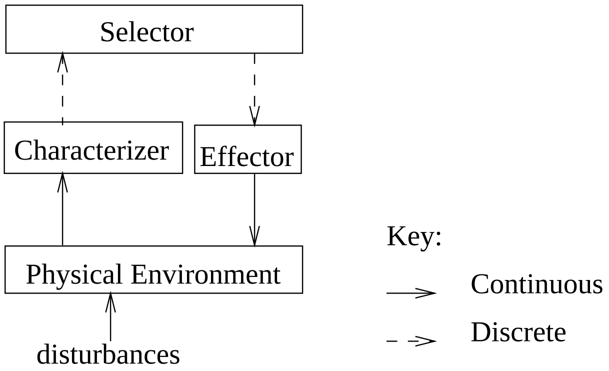


Figure 1: A generic architecture for decomposition of hybrid systems

## 2 General Framework

Having decided "what to do" the next step is to determine a way for "how to do it". A general architecture [4] for modelling hybrid systems can be found in Figure 1. In this figure, the interface between the controller and its physical environment consists of the characterizer and the effector. The former transforming continuous signals to discrete events, and the latter providing (continuous) input signals to the environment based on discrete reactions of the selector. In our part of the project the selector is initially assumed to consist of a synchronous program with an input/output automaton as a control structure. Extending the results to a composition of heterogenuous synchronous programs will be considered later.

The following 5 step plan roughly characterizes the verification process with hybrid models. Given a controller as a synchronous program, the following further activities are essential to verification of the embedded system.

1. Identifying and formalising the properties to prove – i.e. those critical to correct functioning of the controlled system based on informal requirements specifications.

2. Modelling the physical environment.

3. Applying abstraction techniques to obtain "expressive enough" mathematical models.

4. Composing the controller and environment models to obtain the model of the closed loop system.

5. Applying algorithmic or deductive methods for verifying the properties specified under 1.

This approach thus assumes that the interface has been synthesised according to some domain knowledge (which sensor types are available, what can be used as a control signal, etc.). The thresholds and parameter values used in an interface can, however, be adjusted as a result of the verification process.

## 2.1 Requirements specifications

The traditional safety and liveness properties rooted in computer science appear in hybrid systems in the following forms:

- Showing that undesirable states are unreachable in the physical state space
- Showing that some states in the physical model are reached, or some system outputs (typically warnings) emitted, within given time bounds

In other words, in the hybrid context, the typical computer science properties can be proved by formulation as reachability problems – solved in restricted cases by application of model checking, for example [6]. In fact, some specification languages, e.g. Duration Calculus give a standard form for both of these properties as invariances [7].

Other properties include stability and controllability typically analysed by control engineers. In the approach we adopt, certain forms of stability and controllability will also be formulated in terms of reachability:

- If the system arrives in a given mode, then the state variables of the physical environment will remain within given bounds.
- Given an initial set of states, the system will arrive in a an appropriate (stable) mode within a time bound.

Successful application of formal proofs should exploit combinations of proof techniques- from both areas. We explore the use of control engineering techniques in local subproofs and computer science techniques in global reachability proofs.

## 2.2 Heterogenous models

Work on earlier case studies indicates that modelling engineering systems (step 2) is a non-trivial part of the modelling process [6, 7]. There is a spectrum of modelling languages used in different disciplines (for mechanical, electrical, hydraulics and other components in a system). Systematic development of mathematical models from physical models is facilitated by the energy-based (non domain-specific) language of bond graphs resulting in systems of differential and algebraic equations (DAE) [8]. In this project we do not cover this aspect of modelling but import systematically developed models according to these principles.

## 2.3 Abstraction and verification

Once a mathematical model of the physical environment is available, two different strategies can be adopted based on applying steps 3 and 4 in different orders. In the project we are exploring methods in both directions.

**A**pproach 1: applying step 3 first, from a DAE model with $n$ state variables, we derive an abstract model of the physical system as an input/output automaton, the states of which represent flow regions in the $n$-dimentional state space of the physical system, and the input of which are control actions issued by the synchronous program (and the possible effects of disturbances). Going back to Figure 1, this amounts to modelling the environment, the characterizer, and the effector together as one discrete automaton. Arbitrary abstractions of the continuous model can be highly non-deterministic. If the abstraction is parametrized by a particular safety property, then we get a model which is "deterministic enough" with respect to reachability of a particular region. The idea is to develop a model which determines how the uncontrolled environment can be moved from safe to unsafe regions by continuous flows initialized by control actions. Such an abstract automaton can then be composed with the controller automaton (step 4) to prove that the unsafe regions are unreachable (or not). In section 4 we describe the relation between abstract interpretation and the derivation of the abstract environment models.

**A**pproach 2: applying step 4 first, we get a hybrid model of the composed system. When the continuous part of the model consists of engineering models plugged in "as they are". Then we are faced with two options. Either we proceed with verification of systems which may possibly have non-linear developments in continuous phases, or we abstract the model to a form for which algorithmic methods exist. In the former case, analysis by deductive techniques requires a great deal of user knowledge and interaction [2]. In the latter case, we derive time-linear models of the continuous change. This leads to algorithmic analysis of hybrid models in cases where environment dynamics is simple enough to be modelled by differential equations of the form $\dot{x} = c,\ c \in \mathbb{R}$ – in so called linear hybrid automata (LHA) [1]. Since the problem of verifying LHA is undecidable, we have incorporated abstraction techniques in symbolic computations of reachable states of LHA – leading to decidable verfication by algorithmic methods. This topic is further discussed in section 4 and is also related to task 3.3.1.

Since the LHA models of physical systems are deemed to be too coarse to accomodate a large class of applications, work on general models for hybrid systems for the purpose of deductive reasoning is further pursued in work package 7.1. In section 5 we describe a restriction of hybrid transition systems which suffice for modelling synchronous programs, their timing constraints, and a wide range of physical systems. These models will be used as underlying models for deductive proofs of properties.

## 3 The air control case study

The air control case study was provided by one of the industrial partners in the project (SAAB aircraft AB) in the form of a textual document describing certain requirements on the system, and an implemented controller in an automaton based design tool (ASA). In the appendix to this report we describe how far we have reached in application of the general framework in section 2 to this case study. To summarise:

- We have casted the given design into a controller which is a combination of Argos, Lustre and Esterel code (see section 4 of deliverable 2.1 for workpackage 2).

- We have identified critical properties of the closed loop system to be considered for formal verification in later phases of the project (step 1).

- We have identified a model for the physical environment and the physical assumptions that go with it (step 2).

The last two items will be covered in Annex 7.1.A to this deliverable. In continuation of this work package, hybrid modelling and analysis of the case study at different levels of abstraction will be considered.

# 4 Abstracted models

## 4.1 Abstractions for linear relation analysis

Using the forward computation described in work package 3 (Deliverable 3.3.1, section 2.4), we can compute the set of numerically reachable states in each location of an LHA. However, since determining reachability in LHA is in general undecidable, for some applications neither forward nor backward computations terminate. Annex 3.3.A of work package 3 proposes the use of convex hulls as an approximation to the union operator on polyhydra, as well as application of a widening operator which guarantees that the reachability computation terminates after a finite number of iterations.

The suggested widening operator $\nabla$ on polyhydra, provides an upper approximation of both of its operands. $P \nabla Q$ is obtained by removing those constraints in $P$ which are not satisfiable by $Q$, efter rewriting $P$ to a normal form which maximizes the number of inequlities satisfied by $Q$.

Let $X_\ell^{(n)} = F(X^{(n-1)})$ be the $n$-th step of the reachability computation at location $\ell$. Then, if the widening operator is applied in location $\ell$, the above step is replaced by $X_\ell^{(n)} = X_\ell^{(n-1)} \nabla F(X^{(n-1)})$. Since $P \nabla Q$ is a subset of $P$ and the number of inequalities in $P$ are finite, the computation for this location is guaranteed to converge after a finite number of iterations, approximating the original computation.

This method, as implemented by the Polka tool, has been applied to several classical examples as illustrated i Annex 3.3.A (including a water levels monitor, Fischer's mutual exclusion protocol, and a task scheduler).

## 4.2 Discrete models of continuous systems

Given a continuous state system (CSS), Lemmon and Antsaklis propose using an abstraction of the system – a discrete event systems model (DES plant) – for analysing the stability of the system with respect to infinitesimal perturbations in the CSS model [3]. In this workpackage we use a similar approach, not for stability analysis, but for showing safety properties in composition with a synchronous controller.

In the Lemmon-Antsaklis approach, the $n$-dimentinal CSS is represented by the tuple

$$\mathcal{P}_c = \langle \bar{X}, \Phi_t^{\bar{r}_1}, \ldots, \Phi_t^{\bar{r}_p} \rangle$$

where $\bar{X} \subset \mathbb{R}^n$ is the plant state space, and where $\Phi_t^{\bar{r}_i} : \bar{X} \to \bar{X}$ (for $i \in \{1, \ldots, p\}$) is the $i$-th family (indexed by time, t) of transition operators generated by the differential equation

$$\dot{\bar{x}}(t) = f(\bar{x}(t), \bar{r}_i)$$

in which $\bar{r}_i$ is assumed to be a constant vector in $\mathbb{R}^m$, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a Lipcshitz continuous function.

In this framework, the part of the plant-controller interface which corresponds to our characterizer, transforms the CSS plant's state trajectory, $\bar{x}(t)$, into a sequence of plant symbols, $\tilde{x}[k]$, $k \in \mathbb{N}$. They denote the time for the $k$-th characterization by $\tau_x[k]$ and assume that the controller chooses a (new) control action corresponding to some $\bar{r}_i$ at time $\tau_r[k]$. The output of the controller is of course a discrete symbol $\tilde{r}_i$ which is translated (via the effector) to the corresponding $\bar{r}_i$. In their framework, $\tau_x[k] \leq \tau_r[k] \leq \tau_x[k+1]$. Dealing with synchronous programs, we can initially assume $\tau_x[k] = \tau_r[k]$.

The discrete plant model is then represented by the labelled digraph $\mathcal{P}_d = \langle \tilde{X}, A \rangle$ where the set of vertices, $\tilde{X}$ is the set of plant symbols, and each arc in $A \subseteq \tilde{X} \times \tilde{X}$ is labelled by a timed plant event $\langle \tilde{x}, \tau_x \rangle$, and a timed controller event $\langle \tilde{r}, \tau_r \rangle$. It is easy to see that in a synchronous framework, such a plant model corresponds to an I/O automaton. It can be combined with a synchronous controller and checked for reachability of undesirable states.

Our work so far indicates that the model proposed by Lemmon-Antsaklis can be seen as abstract interpretation of a base interpretation which represents the CSS. It is therefore the case that if the composed system is safe in the abstract model then it is safe in the concrete model. The model is, however, not yet general enough to deal with plants with unmodelled inputs (e.g. disturbances). Note that the CSS receives all its inputs from the controller, and if the controller is deterministic then the composed system is deterministic. The work in progress explores whether the restriction to such plants (giving no-input closed loop systems) is too restrictive for proofs of safety in the context of our case studies. Extending these techniques to non-deterministic I/O automata is the next step.

# 5 Time-deterministic hybrid transition systems

*Hybrid Transition Systems* were proposed in 1993 as an attempt to capture both mode-switching physical systems and non-deterministically timed computer systems in a single formalism – treating both discrete and continuous elements of state as functions from time to appropriate domains. Based on several applications of the model to realistic case studies, in this project we have proposed a restriction and small modifications to the formalism, and proved certain compositionality properties.

In the modified model, the continuous dynamics is left as before, but discrete transitions are forced to take place with a deterministic time delay. In this way, engineering models of physical systems can still be plugged in with no additional effort, and the discrete control part can model synchronous programs. In Annex 7.1.B [5] to this deliverable we present the restricted model, its operational semantics, and parallel composition operator. Parallel composition is shown to be associative and commutative. We discuss compositionality in the sense of "if $H_1$ and $H_2$ satisfy a property $P$ then $H_1 \| H_2$ also satisfies $P$". Since composition is defined as a syntactic operation, an interesting property $P$, is of course, that of having well-defined semantics in terms of a (non-empty) set of runs.

We know that composing two systems each having a run does not necessarily result in a system with a run. This is due to Zeno time sequences which can not be excluded in the definition of parallel composition. However, our experience with physical modelling for zero delay systems, and the (positive) fixed delay assumption on controller programs make this property less relevant from a practical point of view.

We therefore proceed by showing that other properties of systems are preserved under parallel composition. To begin with, we define systems which do not have causal conflicts in the sense that they do not allow infinite number of discrete transitions to be taken in a single time point. Then we show that this property is preserved by parallel composition.

Next, we show that if two open systems are composed, with reasonable assumptions on their input functions, then if their composition has a run, then every element in that run appears in the run of some subsystem, when restricted to the variables of that subsystem.

## References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Journal of Theoretical Computer Science*, 138:3–34, 1995.

[2] B. Dutertre and V. Stavridou. Formal Requirements Analysis of an Avionics Control System. *IEEE Transactions on Software Engineering*, 23(5), 1997. To appear.

[3] M. Lemmon and P. J. Antsaklis. Inductively Inferring Valid Logical Models of Continuous-state Dynamical Systems. *Journal of Theoretical Computer Science*, 138:201–210, 1995.

[4] M. Morin, S. Nadjm-Tehrani, P. Österling, and E. Sandewall. Real-Time Hierarchical Control. *IEEE Software*, 9(5):51–57, September 1992.

[5] S. Nadjm-Tehrani. Time-Deterministic Hybrid Transition Systems. In *Hybrid Systems V, Proceedings of the fifth international workshop on hybrid systems, LNCS, To appear.* Springer Verlag, 1998.

[6] S. Nadjm-Tehrani and J-E. Strömberg. JAS-95 Lite: Modeling and Formal Analysis of Dynamic Properties. Technical Report LITH-IDA-R-96-41, Dept. of Computer and Information Science, Linköping University, December 1996. Available from `http://www.ida.liu.se/~snt/activities.html`.

[7] S. Nadjm-Tehrani and J-E. Strömberg. Verification of Dynamic Properties in an Aerospace application. *Formal Methods in System Design*, 1998. To appear.

[8] J.-E. Strömberg and S. Nadjm-Tehrani. Hybrid Systems Verification Combining Duration Calculus and Bond Graphs. In *the invited session on Hybrid Dynamic Systems, Proc. IFAC-IFIP-IMACS Conference on Control of Industrial Systems*, pages 481–486. IFAC, 1997.