I N F O R M A T I K

A Deductive Model Checking Approach for Hybrid Systems

Andreas Nonnengart

MPI–I–1999–2–006 October 1999

FORSCHUNGSBERICHT RESEARCH REPORT

M A X - P L A N C K - I N S T I T U T
F Ü R
I N F O R M A T I K

Im Stadtwald 66123 Saarbrücken Germany

Author's Address

Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
Email: Andreas.Nonnengart@mpi-sb.mpg.de
WWW: http://www.mpi-sb.mpg.de/~nonnenga/

Acknowledgements

Thanks to Harald Ganzinger, Tom Henzinger, Andreas Podelski, Giorgio Delzanno, and Supratik Mukhopadhyay for fruitful discussions.

Abstract

In this paper we propose a verification method for hybrid systems that is based on a successive elimination of the various system locations involved. Briefly, with each such elimination we compute a weakest precondition (strongest postcondition) on the predecessor (successor) locations such that the property to be proved cannot be violated. This is done by representing a given verification problem as a second-order predicate logic formula which is to be solved (proved valid) with the help of a second-order quantifier elimination method. In contrast to many "standard" model checking approaches the method as described in this paper does not perform a forward or backward reachability analysis. Experiments show that this approach is particularly interesting in cases where a standard reachability analysis would require to travel often through some of the given system locations. In addition, the approach offers possibilities to proceed where "standard" reachability analysis approaches do not terminate.

Keywords

Hybrid Systems, Verification, Model Checking, Quantifier Elimination, Location Elimination.

1 Introduction

Hybrid Systems are real-time systems that are embedded in analog environments. They contain discrete and continuous components and interact with the physical world through sensors and actuators. Due to the rapid development of computer technology, hybrid systems directly control much of what we depend on in our daily lives [AHH96]. Since they typically operate in safety-critical situations, the development of rigorous analysis techniques is of high importance. However, traditional program verification is hardly useful, for it allows us, at best, to merely approximate continuously changing environments by discrete sampling. Also, earlier verification techniques based on temporal logics [CE81, CES86, EMSS90, GH90, MP92, MW84, Non95, Non96, OL82, Pnu77, PH88, Sis85] lead only halfway towards what is actually demanded. Only recently have there been some attempts at developing a verification methodology for hybrid systems [ABL97, ACD90, ACH$^+$95, ACHH93, AD94, AH92, AHH96, AHS96, ANKS95, CHR91, GNRR93, Hen91, Hen95, Hen96, HNSY92, Ho95, Kop96, LLW95, Sha93, SUM96].

A common model for hybrid systems can be found in *hybrid automata*. Briefly, such hybrid automata are finite graphs whose nodes correspond to global states. A *computation* of such an automaton is a sequence of state changes (steps). Within each step the system state evolves continuously according to a dynamical law until a transition from one node to another one occurs. Transitions are instantaneous state changes that separate continuous state evolutions.

The paper is now organized as follows. We start with a formal definition of hybrid systems. After that we proceed with the formal definition of the syntax and the semantics of *Integrator Computation Tree Logic*, ICTL [AHH96], that lets us formulate temporal properties of the hybrid system under consideration. What follows is the introduction of the deductive model checking approach in general. This includes both the logical representation and the method to solve the verification problem. Some common generalizations are briefly examined in a subsequent section. In order to provide with some more intuition on the approach some examples follow which also allow us to compare the approach with standard reachability analysis methods. Finally, we conclude that paper with a brief summary and an outlook at what ought to be done in the near future.

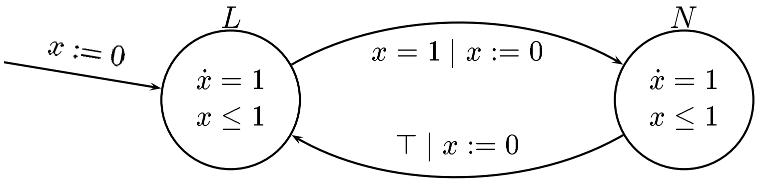
2 Hybrid Systems

2.1 Syntax

DEFINITION 2.1 (CONSTRAINT TERMS AND CONSTRAINT FORMULAS)
The set CT of Constraint Terms *over a fixed variable set X is defined as the smallest set containing X, and real-valued constants, and, moreover, is closed under addition, subtraction, and multiplication with real-valued constants.*

The set of CF of Constraint Formulas *(over the variable set X) is defined as the smallest set that is closed under conjunction and contains $\top$ (truth) and $\perp$ (falsity) as well as all atoms of the form $t_1 > t_2$, $t_1 \geq t_2$, $t_1 < t_2$, $t_1 \leq t_2$, and $t_1 = t_2$, where t_1 and t_2 are constraint terms taken from CT.*

As usual, we illustrate hybrid systems as graphs like



Nodes L and N represent discrete locations, whereas x is a data variable. Within each location we describe the location invariant ($x \leq 1$ in the example) and the continuous activity which describes how the values of the data variables change in time. In the above example the value of x increases by 1 per time unit (say, second), i.e., the first derivative of the function describing the behavior of x over time is the constant 1.
Edges are annotated with guards and discrete actions. Guards form a constraint on the data variables to hold if a transition via the corresponding edge is to be performed. The discrete action specifies how the data variables are to be changed after taking the transition. In the above example the guard of the edge from L to N is $x = 1$ and the corresponding action is to reset x to 0.
The above hybrid system thus describes the following behavior: it starts at location L with the data variable x set to 0. Within L and N the value of x increases by 1 every second (so x is a *clock*). The system leaves location L after exactly one second and resets x to 0. Similarly, it remains within N for at most one second and reenters L after resetting x to 0 again.

The following definition specifies what hybrid systems are in general.

DEFINITION 2.2 (HYBRID SYSTEMS)
Hybrid Systems *are tuples of the form $(X, \mathcal{L}, \mathcal{E}, dif, inv, guard, act)$, where*

- *X is a finite set of real-valued* data variables,
- *$\mathcal{L}$ is a finite set of* locations, *i.e., nodes of a graph,*

- $\mathcal{E} \subseteq \mathcal{L} \times \mathcal{L}$ *is a finite (multi)set of* transitions, *i.e., edges of the graph with nodes from* $\mathcal{L}$,

- *dif*: $\mathcal{L} \times X \mapsto CT$ *is a mapping that associates with each location and each data variable a constraint term (with free variables taken from* X*), representing the change of the data variable within this location over time,*

- *inv*: $\mathcal{L} \mapsto CF$ *is a mapping that associates with each location a constraint formula (with free variables taken from* X*), representing the location invariant,*

- *guard*: $\mathcal{E} \mapsto CF$ *is a mapping that associates with each edge a constraint formula (with free variables taken from* X*), representing the condition that has to be enabled in order to travel along the edge, and*

- *act*: $\mathcal{E} \times X \mapsto CT$ *is a mapping that associates with each edge and each data variable a constraint term (with free variables taken from* X*), representing the value of the variable after traveling along the edge.*

2.2 Semantics

We define a *state* of a hybrid system as a pair (L, ϕ) where $L \in \mathcal{L}$ is a location and $\phi\colon X \mapsto \mathbb{R}$ is a valuation of the data variables. ϕ naturally extends to (constraint) terms and (constraint) formulas. A state (L, ϕ) is called *admissible* if $\phi(inv(L))$ holds. Given two admissible states $\sigma = (L, \phi)$ and $\sigma' = (L', \phi')$ we say that σ' is *transition-reachable* from σ – denoted by $\sigma \overset{\mathrm{tr}}{\mapsto} \sigma'$ – if there exists a transition $T = (L, L') \in \mathcal{E}$ with source L and target L', and both $\phi(guard(T))$ and $\phi'(x) = \phi(act(T, x))$ for each $x \in X$. We call σ' *timely-reachable* from σ with delay δ – denoted by $\sigma \overset{\delta}{\mapsto} \sigma'$, where δ is a non-negative real number – if $L = L'$ and for each $x \in X$ there exists a differentiable function $f_x\colon [0, \delta] \mapsto \mathbb{R}$, with the first derivative $\dot{f}_x\colon (0, \delta) \mapsto \mathbb{R}$, such that (1) $f_x(0) = \phi(x)$ and $f_x(\delta) = \phi'(x)$ and (2) for all $\epsilon \in \mathbb{R}$ with $0 < \epsilon < \delta$: both $inv(L)[x_1/f_{x_1}(\epsilon), \ldots, x_n/f_{x_n}(\epsilon)]$ and $\dot{f}_x(\epsilon) = dif(L, x)[x_1/f_{x_1}(\epsilon), \ldots, x_n/f_{x_n}(\epsilon)]$ are true. σ' is *timely-reachable* from σ – denoted by $\sigma \overset{\star}{\mapsto} \sigma'$ – if there exists a non-negative $\delta \in \mathbb{R}$ such that $\sigma \overset{\delta}{\mapsto} \sigma'$. σ' is said to be *reachable* from σ if $(\sigma, \sigma') \in (\overset{\star}{\mapsto} \cup \overset{\mathrm{tr}}{\mapsto})^*$.
A *run* ρ of $\mathcal{H}$ with initial state $\sigma_0 = (L_0, \phi_0)$ is a maximal sequence of states represented as

$$\rho = \sigma_0 \mapsto_{f_0}^{t_0} \sigma_1 \mapsto_{f_1}^{t_1} \sigma_2 \mapsto_{f_2}^{t_2} \sigma_3 \mapsto_{f_3}^{t_3} \cdots$$

where $t_i \in \mathbb{R}^{\geq 0}$ and $f_i\colon [0, t_i] \mapsto (X \mapsto \mathbb{R})$, such that $f_i(0) = \phi_i$, and moreover, $inv(L_i)[X/f_i(t)(X)]$ holds for all $0 \leq t \leq t_i$, $(L_i, f_i(t_i)) \overset{\mathrm{tr}}{\mapsto} \sigma_{i+1}$ and for all $0 \leq t' \leq t' + \delta \leq t_i : (L_i, f_i(t')) \overset{\delta}{\mapsto} (L_i, f_i(t' + \delta))$. The set of states

contained in such a run ρ is given as $\mathrm{States}(\rho) = \{(L_i, f_i(t)) \mid t \in \mathbb{R}, 0 \leq t \leq t_i\}$. The set of all runs of a hybrid system $\mathcal{H}$ with initial state σ is denoted by $\mathrm{runs}(\mathcal{H}, \sigma)$. A *position* π of a run $\rho = \sigma_0 \mapsto^{t_0}_{f_0} \sigma_1 \mapsto^{t_1}_{f_1} \sigma_2 \mapsto^{t_2}_{f_2} \sigma_3 \mapsto^{t_3}_{f_3} \cdots$ is a pair $\pi = (i, r) \in \mathbb{N} \times \mathbb{R}$ such that $0 \leq r \leq t_i$. We denote the set of positions of a run ρ as $\mathrm{pos}(\rho)$. Positions are ordered lexicographically, i.e., $(i, r) < (j, s)$ if and only if $i < j$ or ($i = j$ and $r < s$). Also, $(i, r) \leq (j, s)$ if and only if $(i, r) < (j, s)$ or ($i = j$ and $r = s$). By $\rho(\pi)$ with $\pi = (i, r)$ we denote the state $(L_i, f_i(r))$. Thus $\mathrm{States}(\rho) = \{\rho(\pi) \mid \pi \in \mathrm{pos}(\rho)\}$.
A run is said to be *non-zeno* if $\sum t_i$ diverges. In the sequel we shall assume that the runs of the hybrid system under consideration are all non-zeno.[1]

For the simple hybrid system from page 2 it is quite easy to find the set of reachable states. It contains exactly all states of the form (L, ϕ) or (N, ϕ) where ϕ maps x to an arbitrary real value between 0 and 1. Intuitively, it should thus be possible to prove that the value of the data variable x always remains smaller than 1. But there are much more interesting properties of the above system that we want to be able to prove. As we noted already, the system will always remain within location L for exactly one second, whereas it can only remain within location N for at most one second. Thus, the accumulated time spent in location N can never exceed one half of the overall running time of the system. Such properties should be provable as well. This, however, demands for a requirement language that lets us formulate these kinds of properties. One such language can be found in ICTL [AHH96] as described in the section to follow.

3 Integrator Computation Tree Logic ICTL

3.1 ICTL Syntax

We describe properties of a hybrid system with data variables X and locations $\mathcal{L}$, in terms of ICTL formulas, where

- every constraint formula over X is an ICTL formula,
- every location name from $\mathcal{L}$ is an ICTL formula,
- if Φ and Ψ are ICTL formulas, so are $\neg\Phi$, $\Phi \wedge \Psi$, $\Phi \vee \Psi$, $\Phi \rightarrow \Psi$, and $\Phi \equiv \Psi$,
- if Φ and Ψ are ICTL formulas, so are $AG\ \Phi$, $AF\ \Phi$, $EG\ \Phi$, $EF\ \Phi$, $\Phi\, EU\, \Psi$, and $\Phi\, AU\, \Psi$,

[1] The assumption of non-zenoness implies that hybrid systems are deadlock-free, i.e., there is no reachable state that has no successor. So-called livelocks, however, are not excluded. This means that we absolutely allow states which have only themselves as future alternatives. The latter case just states that the situation does not change in time, whereas the former case (deadlock) would claim that time itself has come to an end.

- if Φ is an ICTL formula, z is a new data variable, and $\{L_1, \dots, L_n\} \in \mathcal{L}$ is a subset of the location names then $z^{\{L_1,\dots,L_n\}}.\Phi$ is an ICTL formula (and z is added to the set X).

Intuitively, the temporal operators AG , AF , EG , EF , EU, AU, mean "always", "inevitably", "possibly always", "possibly", "possibly until", and "inevitably until" respectively. Their formal semantics with respect to hybrid systems is defined below.

3.2 ICTL Semantics with respect to Hybrid Systems

Given a hybrid system $\mathcal{H}$, by $\mathcal{H}^{z^{\{L_1,\dots,L_n\}}}$ we mean the extended system we obtain from adding the new clock z which is initialized with 0 and which is supposed to run with slope 1 within locations $L_1, \dots, L_n$ and with slope 0, i.e., it is stopped, for all other locations. Notice that this implies that the value of the new clock z will never get below 0. Formally:
Let $\mathcal{H} = (X, \mathcal{L}, \mathcal{E}, \mathit{dif}, \mathit{inv}, \mathit{guard}, \mathit{act})$. Then

$$\mathcal{H}^{z^{\{L_1,\dots,L_n\}}} = (X \cup \{z\}, \mathcal{L}, \mathcal{E}, \mathit{dif}', \mathit{inv}', \mathit{guard}, \mathit{act})$$

where $\mathit{inv}'(L) = \mathit{inv}(L) \wedge 0 \leq z$ and

$$\mathit{dif}'(L, x) = \begin{cases} \mathit{dif}(L, x) & \text{if } x \neq z \\ 1 & \text{if } x = z \text{ and } L \in \{L_1, \dots, L_n\} \\ 0 & \text{otherwise} \end{cases}$$

for all data variables $x \in X$ and locations $L \in \mathcal{L}$.[2]
As usual, we define the valuation $\phi[z/0]$ as $\phi[z/0](x) = \begin{cases} \phi(x) & \text{if } x \neq z \\ 0 & \text{otherwise.} \end{cases}$

Definition 3.1
Given a hybrid system $\mathcal{H} = (X, \mathcal{L}, \mathcal{E}, \mathit{dif}, \mathit{inv}, \mathit{guard}, \mathit{act})$ and a state $\sigma = (L, \phi)$, the semantics of ICTL with respect to $\mathcal{H}$ and σ is defined as:

$\mathcal{H}, \sigma \models c$ *iff* $\models \phi(c)$, *provided c is a constraint formula*

$\mathcal{H}, \sigma \models N$ *iff* *locations N and L are identical*

$\mathcal{H}, \sigma \models \neg\Phi$ *iff* $\mathcal{H}, \sigma \not\models \Phi$

$\mathcal{H}, \sigma \models \Phi \wedge \Psi$ *iff* $\mathcal{H}, \sigma \models \Phi$ & $\mathcal{H}, \sigma \models \Psi$
and similarly for the other boolean connectives

$\mathcal{H}, \sigma \models AG\ \Phi$ *iff* $\forall \rho\ (\rho \in \mathrm{runs}(\mathcal{H}, \sigma) \Rightarrow \forall \pi\ (\pi \in \mathrm{pos}(\rho) \Rightarrow \mathcal{H}, \rho(\pi) \models \Phi))$

[2] Actually, the function *act* would also have to be changed accordingly. However, if we take the convention that we only describe the action on data variables that change their value by taking the transition, it becomes unneccessary to add something like $\mathit{act}(T, z) = z$.

$$
\begin{array}{lll}
\mathcal{H},\sigma \models EF\,\Phi & \textit{iff} & \exists\rho\,(\rho\in \mathrm{runs}(\mathcal{H},\sigma) \;\;\&\\
& & \quad \exists\pi\,(\pi\in \mathrm{pos}(\rho)\;\&\;\mathcal{H},\rho(\pi)\models\Phi))\\
\mathcal{H},\sigma \models EG\,\Phi & \textit{iff} & \exists\rho\,(\rho\in \mathrm{runs}(\mathcal{H},\sigma) \;\;\&\\
& & \quad \forall\pi\,(\pi\in \mathrm{pos}(\rho)\Rightarrow\mathcal{H},\rho(\pi)\models\Phi))\\
\mathcal{H},\sigma \models AF\,\Phi & \textit{iff} & \forall\rho\,(\rho\in \mathrm{runs}(\mathcal{H},\sigma) \;\;\Rightarrow\\
& & \quad \exists\pi\,(\pi\in \mathrm{pos}(\rho)\;\&\;\mathcal{H},\rho(\pi)\models\Phi))\\
\mathcal{H},\sigma \models \Phi\,EU\,\Psi & \textit{iff} & \exists\rho\,(\rho\in \mathrm{runs}(\mathcal{H},\sigma) \;\;\&\\
& & \quad \exists\pi\,(\pi\in \mathrm{pos}(\rho)\;\&\;\mathcal{H},\rho(\pi)\models\Psi\;\&\\
& & \qquad \forall\pi'\,((\pi'\in \mathrm{pos}(\rho)\;\&\;(0,0)\le\pi'\le\pi)\;\;\Rightarrow\\
& & \qquad\quad \mathcal{H},\rho(\pi')\models\Phi)))\\
\mathcal{H},\sigma \models \Phi\,AU\,\Psi & \textit{iff} & \forall\rho\,(\rho\in \mathrm{runs}(\mathcal{H},\sigma) \;\;\Rightarrow\\
& & \quad \exists\pi\,(\pi\in \mathrm{pos}(\rho)\;\&\;\mathcal{H},\rho(\pi)\models\Psi\;\&\\
& & \qquad \forall\pi'\,((\pi'\in \mathrm{pos}(\rho)\;\&\;(0,0)\le\pi'\le\pi)\;\;\Rightarrow\\
& & \qquad\quad \mathcal{H},\rho(\pi')\models\Phi)))\\
\mathcal{H},\sigma \models z^{\mathcal{N}}.\Phi & \textit{iff} & \mathcal{H}^{z^{\mathcal{N}}},(L,\phi[z/0])\models\Phi, \qquad \textit{where } \mathcal{N}\subseteq\mathcal{L}
\end{array}
$$

LEMMA 3.2
For the ICTL operators AG and EF the above semantics can be changed to

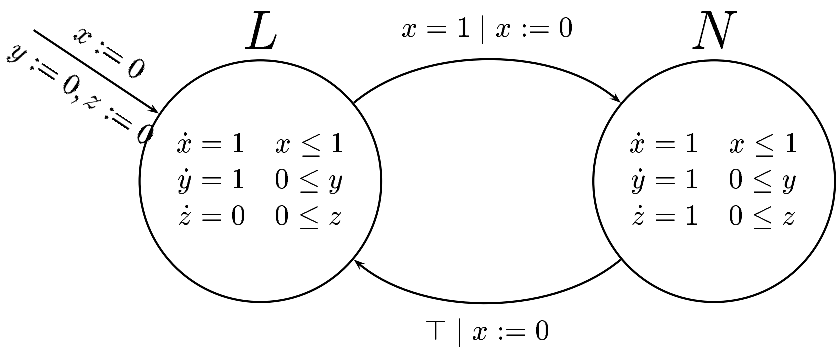
$$
\begin{array}{lll}
\mathcal{H},\sigma \models AG\,\Phi & \textit{iff} & \mathcal{H},\sigma'\models\Phi \textit{ for every } \sigma' \textit{ reachable from } \sigma\\
\mathcal{H},\sigma \models EF\,\Phi & \textit{iff} & \mathcal{H},\sigma'\models\Phi \textit{ for some } \sigma' \textit{ reachable from } \sigma
\end{array}
$$

Clearly, from the above definition it follows that $AG\;\Phi \leftrightarrow \neg EF\;\neg\Phi$ and $EF\;\Phi \leftrightarrow \top\,EU\,\Phi$. Also, $EG\;\Phi \leftrightarrow \neg AF\;\neg\Phi$ and $AF\;\Phi \leftrightarrow \top\,AU\,\Phi$. All temporal operators can thus be described in terms of the two *Until*-operators.

EXAMPLE 3.3
Recall the property we wanted to express and prove for our example hybrid system on page 2. In terms of ICTL this property can be formulated as

$$y^{\{L,N\}}.z^{\{N\}}.AG\;2z \le y$$

i.e., we assume two additional clocks y and z, where y counts overall time (it runs with slope 1 in each location) and z accumulates the time spent in location N (it acts as a usual clock in location N but is stopped in location L). The picture of our hybrid system then changes to



For this extension we want to prove that $AG\ 2z \le y$, i.e., we have to check whether the constraint $2z \le y$ holds for all reachable states.

4 Deductive Model Checking

The general idea behind the deductive model checking approach is as follows. Our ultimate aim is to automate what is described in Definition 3.1. To this end consider a hybrid system $\mathcal{H}$, some location of $\mathcal{H}$, say L, and an ICTL formula Φ. For some valuations of the data variables in L, Φ is true and for the others Φ is false. Let us collect the former in the set $\hat{\phi} = \{\phi \mid \mathcal{H}, (L, \phi) \models \Phi\}$. Now, suppose we were able to describe this set $\hat{\phi}$ as a (finite) constraint formula, say $\lceil\Phi\rceil^L$. Then, checking whether $\mathcal{H}, (L, \phi) \models \Phi$ holds can be reformulated as to checking whether $\phi(\lceil\Phi\rceil^L)$ is valid. And in case of merely linear constraints this could even be decided. Our intermediate goal, therefore, is to find $\lceil\Phi\rceil^L$, the *characteristic constraint formula* for Φ in location L.[3] In order to achieve this, we consider the structure of the ICTL formula Φ. In the simplest case Φ is either a constraint formula or a location name. The latter case simply reduces to $\top$ or $\bot$, depending on whether or not this location name is identical to L. In the former case, we can assume Φ to be its own characteristic constraint formula and so $\lceil\Phi\rceil^L = \Phi$. Also, there are no difficulties with boolean connectives as long as the characteristic constraint formulas for the respective components are known. For instance, $\lceil\Phi \wedge \Psi\rceil^L = \lceil\Phi\rceil^L \wedge \lceil\Psi\rceil^L$ which is a finite constraint formula provided both $\lceil\Phi\rceil^L$ and $\lceil\Psi\rceil^L$ are. Evidently, the more complicated and more interesting cases are those where Φ has a temporal operator as a top symbol. However, it is in general not possible to find a corresponding finite characteristic constraint formula, for otherwise the validity of ICTL formulas for arbitrary (linear) hybrid systems would be decidable which, unfortunately, is not the case. Therefore we cannot

[3]The reader who is familiar with standard model checking approaches for hybrid systems probably notices a small change in perspective since we do not compute the set of states that fulfill Φ, but (representatives of) constraint formulas instead. This view is far from artificial; in fact, it is crucial for the proposed approach.

expect to find a characteristic constraint formula as easy as for the simple cases above. Instead of attempting to construct $\lceil \Phi \rceil^L$ directly, we describe it as a formula of the second-order predicate calculus and try to simplify this description to a constraint formula if possible. How this can be done is described later. At this stage we are more concerned with the construction of the *characteristic constraint formula* for Φ at L.

4.1 First-order Theories of Reachability and Inevitability

Here we restrict our view to *linear* hybrid systems, where $dif(L, x)$ is a constant, say k_L^x, for each location L and data variable x. We extend this to sets of data variables $X = \{x_1, \ldots, x_n\}$ in the natural way such that a term like $X + k_L^X \delta$ represents the sequence $x_1 + k_L^{x_1}\delta, \ldots, x_n + k_L^{x_n}\delta$.

DEFINITION 4.1
An interpretation $\Im = (\mathcal{D}, \Im_{\mathcal{L}}, \phi)$ *for a first-order theory associated with a hybrid system* $\mathcal{H}$ *with locations* $\mathcal{L}$ *has a fixed domain* $\mathcal{D}$ *(the reals or the rationals, say), a valuation* ϕ *for the data variables in* X*, and a meaning function* $\Im_{\mathcal{L}}$ *for the locations in* $\mathcal{L}$ *such that* $\Im_{\mathcal{L}}(L) \in \mathcal{D}^n$*, where* n *is the number of data variables in* X*. A* model *of a formula* Φ *is an interpretation satisfying this formula.*
We often also speak of a model as a set of ground atoms of the form $\{L(\Im(t_1), \ldots, \Im(t_n)) \mid \Im \models L(t_1, \ldots, t_n)\}, t_i$ *are constraint terms*$\}$*, where* $\Im$ *is a model in the above sense. Interpretations (models) are partially ordered by set-inclusion. A* minimal model *of* Φ *is a model of* Φ *such that none of its proper subsets is also a model of* Φ*. We denote the set of minimal models for a formula* Φ *by* minMod(Φ)*. In case there exists only one unique minimal model we shall also refer to this one as* minMod(Φ)*.*

DEFINITION 4.2
Let $\mathcal{H} = (X, \mathcal{L}, \mathcal{E}, dif, inv, guard, act)$ *be a hybrid system. For each* $L \in \mathcal{L}$ *we define the first-order theory*[4]

$$\forall X\; L(X) \rightarrow \begin{cases} inv(L) \;\wedge \\ \forall \delta\; (\delta \geq 0 \wedge inv(L)[X/X + k_L^X \delta] \rightarrow L(X + k_L^X \delta)) \;\wedge \\ \displaystyle\bigwedge_{T=(L,N)\in\mathcal{E}} guard(T) \rightarrow N(act(T, X)) \end{cases}$$

as the local reachability theory *of* L *in* $\mathcal{H}$*,* $\mathcal{R}_{\mathcal{H}}^L$ *for short. By the* reachability theory of $\mathcal{H}$ *– which we call* $\mathcal{R}_{\mathcal{H}}$*, or simply* $\mathcal{R}$ *if* $\mathcal{H}$ *is clear from the context – we understand the conjunction of all local reachability theories, i.e.,*

$$\mathcal{R}_{\mathcal{H}} = \bigwedge_{L \in \mathcal{L}} \mathcal{R}_{\mathcal{H}}^L$$

[4]For readability let us abbreviate $L(act(T, x_1), \ldots, act(T, x_n))$ with $L(act(T, X))$.

Hence, for each location $L \in \mathcal{L}$ we introduce an n-ary predicate with the same name, where n is just the number of data variables in $\mathcal{H}$, i.e., $n = |X|$. Then, (location) atoms $L(\alpha_1, \ldots, \alpha_n)$ correspond to states (L, ϕ), where $\phi(x_i) = \alpha_i$ with $X = \{x_1, \ldots, , x_n\}$. For simplicity we abbreviate this with $L(\alpha_1, \ldots, \alpha_n) \cong (L, \phi)$. This notion of *correspondence* between atoms and states naturally extends to sets of atoms (interpretations, models) and sets of states (e.g., members of a run).

The purpose of such a reachability theory is to have a logical representation of the reachable states. It is constructed in a way such that for each possible state (denoted by an atom $L(X)$ that corresponds to this very state) the possible *immediate future states* can be determined. For instance, given the atom $L(X)$ the local reachability theory of L in $\mathcal{H}$ tells us that the corresponding state satisfies the location invariant $inv(L)$ and also that there are potential timed successors $(\forall \delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X \delta] \rightarrow L(X + k_L^X \delta)))$, and, finally, potential transition successors $(\bigwedge_{T=(L,N)\in\mathcal{E}} guard(T) \rightarrow N(act(T, X)))$. The exact connection between reachability theories and reachable states is given by the following lemma.

Lemma 4.3

If the conjunction $L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}$ has a model at all then it has a unique minimal model which corresponds to the set of states that are reachable from (L, ϕ) in the hybrid system $\mathcal{H}$. More formally:

$$\mathrm{minMod}(L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}) \cong \{\sigma \mid ((L, \phi), \sigma) \in (\overset{\star}{\mapsto} \cup \overset{\mathrm{tr}}{\mapsto})^*\}$$

Proof: First note that a reachability theory is Horn in the location predicates, that all other symbols have fixed interpretation, and that the only boolean connective within constraint formulas is the logical conjunction. Therefore, if the theory has a model at all then it has a unique minimal model.

What remains to be shown is that for every location L' and every data variable valuation ϕ'

$$\begin{array}{c} L'(\phi'(x_1), \ldots, \phi'(x_n)) \in \mathrm{minMod}(L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}) \\ \Leftrightarrow \\ ((L, \phi), (L', \phi')) \in (\overset{\star}{\mapsto} \cup \overset{\mathrm{tr}}{\mapsto})^* \end{array}$$

For the direction from left to right take any proof of $L'(\phi'(x_1), \ldots, \phi'(x_n))$ from $L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}$. The claim then follows by an easy induction on the length of this proof.

As for the other direction, consider the reachable state (L', ϕ') that is included in some run $\sigma_0 \mapsto_{f_0}^{t_0} \sigma_1 \mapsto_{f_1}^{t_1} \sigma_2 \mapsto_{f_2}^{t_2} \sigma_3 \mapsto_{f_3}^{t_3} \cdots$ with $\sigma_0 = (L, \phi)$ and $\phi' = f_i(t)$ for some $0 \leq t \leq t_i$. A simple induction on i then shows that the ground atom represented by $L'(f_i(0))$ belongs to the minimal model of $L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}$. Moreover, since $\forall X\ L'(X) \rightarrow \forall \delta\ (\delta \geq 0 \wedge inv(L')[X/X +$

$k^X_{L'}\delta] \rightarrow L'(X + k^X_{L'}\delta))$ is a clause of the theory under consideration we also know that the ground atom represented by $L'(f_i(t))$ is a member of the minimal model and we are done. $\Box$

EXAMPLE 4.4
For our simple example from page 6 the reachability theory is given by

$$\forall x, y, z\ L(x,y,z) \ \rightarrow \left\{ \begin{array}{l} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \ \wedge \\ \forall \delta\ \delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z \ \rightarrow \\ \quad L(x + \delta, y + \delta, z) \ \wedge \\ x = 1 \rightarrow N(0, y, z) \end{array} \right.$$

$$\wedge$$

$$\forall x, y, z\ N(x,y,z) \ \rightarrow \left\{ \begin{array}{l} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \ \wedge \\ \forall \delta\ 0 \leq \delta \leq 1 - x \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ \quad N(x + \delta, y + \delta, z + \delta) \ \wedge \\ L(0, y, z) \end{array} \right.$$

The reachability theory will ultimately be responsible for the temporal operators *AG* and *EF*. However, it is not well suited for the other temporal operators we are interested in. We therefore, in addition, define a similar first-order theory; this time for these other temporal operators, though. Just as the reachability theory provides us with some information about the states that *can* be reached, the *inevitability theory* to be defined below tells us something about the states that are inevitable or unavoidable. It does so by stating between which possible future alternatives the system *must* choose. The following specifies this.

DEFINITION 4.5 (INEVITABILITY THEORY)
Let $\mathcal{H} = (X, \mathcal{L}, \mathcal{E}, \mathit{dif}, \mathit{inv}, \mathit{guard}, \mathit{act})$ *be a hybrid system. For each* $L \in \mathcal{L}$ *we define the first-order theory*

$$\forall X\ L(X) \ \rightarrow \ \mathit{inv}(L)$$

$$\forall X\ L(X) \ \rightarrow \left\{ \begin{array}{l} \forall \delta\ \delta \geq 0 \rightarrow L(X + k^X_L \delta) \ \vee \\ \exists \delta \left\{ \begin{array}{l} \delta \geq 0 \ \wedge \\ \forall \delta'\ 0 \leq \delta' \leq \delta \rightarrow L(X + k^X_L \delta') \ \wedge \\ \displaystyle\bigvee_{T=(L,N)\in\mathcal{E}} \begin{array}{l} \mathit{guard}(T)[X/X + k^X_L\delta] \ \wedge \\ \quad N(\mathit{act}(T,X)[X/X + k^X_L\delta]) \end{array} \end{array} \right. \end{array} \right.$$

as the local inevitability theory *of* L *in* $\mathcal{H}$, $\mathcal{I}^L_{\mathcal{H}}$ *for short. By the* inevitability theory of $\mathcal{H}$ *– which we call* $\mathcal{I}_{\mathcal{H}}$, *or simply* $\mathcal{I}$ *if* $\mathcal{H}$ *is clear from the context – we understand the conjunction of all local inevitabiity theories, i.e.,*

$$\mathcal{I}_{\mathcal{H}} = \bigwedge_{L\in\mathcal{L}} \mathcal{I}^L_{\mathcal{H}}$$

The first part of any local inevitability theory is trivial. It just guarantees the mere fact that for each location predicate the corresponding location invariant is supposed to hold. The second part is more complicated and more interesting. Note that, given an arbitrary state represented by the location predicate $L(X)$, either the system remains forever in this location, i.e., $\forall\delta\ \delta \geq 0 \rightarrow L(X+k_L^X\,\delta)$, or it will sooner or later leave this very location. In the latter case we know that there is a time delay δ after which one of the guards of the outgoing edges is true and until then the system remains within location L. This is exactly what is expressed by the complicated second part of the local inevitability theories.

The importance of the inevitability theory is made precise in the lemma below.

LEMMA 4.6
Each minimal model of $L(\phi(X)) \wedge \mathcal{I}_\mathcal{H}$ corresponds to the members of one of the possible runs[5] *of $\mathcal{H}$ with initial state (L, ϕ). Also, the members of any possible run of $\mathcal{H}$ correspond to a model of $\mathcal{I}_\mathcal{H} \wedge L(\phi(X))$. Formally:*

- $$\begin{array}{l}\forall\Im\ \Im \in \mathrm{minMod}(L(\phi(X)) \wedge \mathcal{I}_\mathcal{H}) \ \Rightarrow \\ \qquad\qquad \exists\rho\ \rho \in \mathrm{runs}(\mathcal{H}, (L, \phi))\ \&\ \Im \cong \mathrm{States}(\rho)\end{array}$$
- $$\begin{array}{l}\forall\rho\ \rho \in \mathrm{runs}(\mathcal{H}, (L, \phi)) \ \Rightarrow \\ \qquad\qquad \{L'(\phi'(X)) \mid (L', \phi') \in \mathrm{States}(\rho)\} \models L(\phi(X)) \wedge \mathcal{I}_\mathcal{H}\end{array}$$

Proof: Consider the systematic construction of a minimal model for the theory $L(\phi(X)) \wedge \mathcal{I}_\mathcal{H}$. Evidently, this leads to run of $\mathcal{H}$ with initial state (L, ϕ).
On the other hand, consider an arbitrary run of $\mathcal{H}$ with initial state (L, ϕ). It is easy to see that the atoms that correspond to states of this run are closed under $L(\phi(X)) \wedge \mathcal{I}_\mathcal{H}$. □

EXAMPLE 4.7
For our simple example from page 6 the inevitability theory is given by

$$\begin{array}{l}
\forall x,y,z\ L(x,y,z)\ \rightarrow\ x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\
\forall x,y,z\ L(x,y,z)\ \rightarrow \left\{\begin{array}{l}
\forall\delta\ \delta \geq 0 \rightarrow L(x+\delta, y+\delta, z)\ \vee \\
\exists\delta\ \delta \geq 0\ \wedge \\
\quad \forall\delta'\ (0 \leq \delta' \leq \delta \rightarrow L(x+\delta', y+\delta', z))\ \wedge \\
\quad x+\delta = 1 \wedge N(0, y+\delta, z)
\end{array}\right. \\
\forall x,y,z\ N(x,y,z)\ \rightarrow\ x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\
\forall x,y,z\ N(x,y,z)\ \rightarrow \left\{\begin{array}{l}
\forall\delta\ \delta \geq 0 \rightarrow N(x+\delta, y+\delta, z+\delta)\ \vee \\
\exists\delta\ \delta \geq 0\ \wedge \\
\quad \forall\delta'\ (0 \leq \delta' \leq \delta \rightarrow N(x+\delta', y+\delta', z+\delta'))\ \wedge \\
\quad L(0, y+\delta, z+\delta)
\end{array}\right.
\end{array}$$

[5]Recall that we only consider non-zeno runs of hybrid systems. Zeno runs could even lead to inconsistencies in the inevitability theory.

which can be simplified to

$$
\begin{array}{lcl}
\forall x, y, z\ L(x,y,z) & \rightarrow & x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\
\forall x, y, z\ L(x,y,z) & \rightarrow & \forall \delta'\ (0 \leq \delta' \leq 1 - x \rightarrow L(x+\delta', y+\delta', z)) \wedge \\
 & & N(0, y+1-x, z) \\
\forall x, y, z\ N(x,y,z) & \rightarrow & x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\
\forall x, y, z\ N(x,y,z) & \rightarrow & \exists \delta\ \delta \geq 0 \wedge \\
 & & \forall \delta'\ (0 \leq \delta' \leq \delta \rightarrow N(x+\delta', y+\delta', z+\delta')) \wedge \\
 & & L(0, y+\delta, z+\delta)
\end{array}
$$

4.2 The Deductive Approach for Linear Hybrid Systems

Suppose we are given a hybrid system $\mathcal{H}$, its reachability theory $\mathcal{R}$ together with an initial state (L, ϕ), and a property $AG\ c$ to be proved, where c is a constraint formula over the data variables X. Then we have to show that c holds for all the reachable states of $\mathcal{H}$, i.e., it is true for each atom of the minimal model of the corresponding reachability theory. Trivially, this means that there exists a model (namely the minimal model) whose elements all satisfy the constraint c. On the other hand, since the minimal model is by definition a subset of any model of the theory, we know that having such a model means that also for the minimal model it holds that each of its elements satisfy c. Altogether, we know that $AG\ c$ holds at (L, ϕ) for $\mathcal{H}$ if and only if there exists a model of its reachability theory (together with the atom that corresponds to the initial state) such that c holds for all its elements, or, more formally,[6] if $L(\phi(X)) \wedge \mathcal{R} \wedge \bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \rightarrow c$ has a first-order model. This latter statement, however, can be formulated in terms of second-order logic, namely

$$\exists L_1, \ldots, L_n\ L(\phi(X)) \wedge \mathcal{R} \wedge \forall X\ (L_1(X) \vee \ldots \vee L_n(X) \rightarrow c)$$

since the existence of a model is tantamount to the existence of suitable interpretations for the free symbols involved.

EXAMPLE 4.8
Recall that we wanted to prove $AG\ 2z \leq y$ for the extended example system on page 6. According to the above observations this means that we have to

[6] Recall that the location names are the only predicate symbols that have a free interpretation.

prove the validity of

$$\exists L, N \left[\begin{array}{lll} L(0,0,0) & & \\ \forall x,y,z & L(x,y,z) & \to x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\ \forall x,y,z & L(x,y,z) & \to x = 1 \to N(0,y,z) \\ \forall x,y,z & L(x,y,z) & \to \forall \delta\, 0 \leq \delta \leq 1-x \wedge 0 \leq y+\delta \wedge 0 \leq z \to \\ & & \qquad L(x+\delta, y+\delta, z) \\ \forall x,y,z & N(x,y,z) & \to x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\ \forall x,y,z & N(x,y,z) & \to L(0,y,z) \\ \forall x,y,z & N(x,y,z) & \to \forall \delta\, 0 \leq \delta \leq 1-x \wedge \\ & & \qquad 0 \leq y+\delta \wedge 0 \leq z+\delta \to \\ & & \qquad\quad N(x+\delta, y+\delta, z+\delta) \\ \forall x,y,z & L(x,y,z) & \to 2z \leq y \\ \forall x,y,z & N(x,y,z) & \to 2z \leq y \end{array}\right]$$

As for the general case, assume that we have to show that $AG\ \Phi$, where Φ is an arbitrary ICTL formula. That means we have to verify that Φ holds for every state that is reachable from the initial state, say (L, ϕ), within the hybrid system $\mathcal{H}$. If Φ were a constraint formula or a location name, we would know what to do from the observations above. Nevertheless, even if Φ is not a constraint formula, we have a description of its characteristic constraint formula, namely $\lceil \Phi \rceil_{\mathcal{H}}^{L(X)}$ for each location name L. I.e., proving that $\mathcal{H}, (L, \phi) \models AG\ \Phi$ holds can be reduced to showing the validity of $\phi(\lceil \Phi \rceil_{\mathcal{H}}^{L(X)})$. This, and similar reflections on the other temporal operators, leads to the following definition.

Definition 4.9
The characteristic constraint formula $\lceil \Phi \rceil_{\mathcal{H}}^{L(X)}$ associated with the ICTL formula Φ, the hybrid system $\mathcal{H} = (X, \mathcal{L}, \mathcal{E}, dif, inv, guard, act)$, and the location $L \in \mathcal{L}$ is recursively defined by

$$\begin{array}{lcl} \lceil c \rceil_{\mathcal{H}}^{L(X)} & = & c \\ \lceil L' \rceil_{\mathcal{H}}^{L(X)} & = & \begin{cases} \top \text{ if } L \text{ and } L' \text{ are identical} \\ \bot \text{ otherwise} \end{cases} \\ \lceil \neg\Phi \rceil_{\mathcal{H}}^{L(X)} & = & \neg \lceil \Phi \rceil_{\mathcal{H}}^{L(X)} \\ \lceil \Phi \wedge \Psi \rceil_{\mathcal{H}}^{L(X)} & = & \lceil \Phi \rceil_{\mathcal{H}}^{L(X)} \wedge \lceil \Psi \rceil_{\mathcal{H}}^{L(X)} \\ & & \textit{and similarly for the other boolean connectives} \\ \lceil z^{\mathcal{N}}.\Phi \rceil_{\mathcal{H}}^{L(X)} & = & \left(\lceil \Phi \rceil_{\mathcal{H}^{z^{\mathcal{N}}}}^{L(X,z)} \right)_0^z, \text{ where } \mathcal{N} \subseteq \mathcal{L}^{7} \\ \lceil AG\ \Phi \rceil_{\mathcal{H}}^{L(X)} & = & \exists L_1, \ldots, L_n\ L(X) \wedge \mathcal{R}_{\mathcal{H}} \wedge \bigwedge_{N \in \mathcal{L}} \forall X\, N(X) \to \lceil \Phi \rceil_{\mathcal{H}}^{N(X)} \\ \lceil EG\ \Phi \rceil_{\mathcal{H}}^{L(X)} & = & \exists L_1, \ldots, L_n\ L(X) \wedge \mathcal{I}_{\mathcal{H}} \wedge \bigwedge_{N \in \mathcal{L}} \forall X\, N(X) \to \lceil \Phi \rceil_{\mathcal{H}}^{N(X)} \end{array}$$

[7] As usual, the notation A_y^x means A with every occurrence of x replaced by y.

The temporal operators EF and AF are to be treated as $\neg AG\,\neg$ and $\neg EG\,\neg$ respectively. For the Until operators see Subsection 4.3 on page 15.

Intuitively, such a characteristic constraint formula describes the necessary and sufficient condition on the data variables such that the ICTL formula Φ holds for the hybrid system $\mathcal{H}$ in location L. This, however, is exactly what we need for our deductive model checking approach. The following (main) theorem makes this more precise.

THEOREM 4.10
Given a hybrid system $\mathcal{H}$ with data variables X, an initial state (L, ϕ) and an ICTL formula Φ. Then

$$\mathcal{H}, (L, \phi) \models \Phi \qquad \textit{iff} \qquad \models \phi\left(\lceil \Phi \rceil_{\mathcal{H}}^{L(X)}\right)$$

Proof: By induction on the structure of Φ.
For Φ being a constraint formula c or a location name L the theorem holds trivially. Also in case of a boolean connective there are no problems at all. Therefore, let us only consider the more complicated cases.

$$\begin{aligned}
&\mathcal{H}, (L, \phi) \models z^{\mathcal{N}}.\Psi \\
&\quad \text{iff} \quad \mathcal{H}^{z^{\mathcal{N}}}, (L, \phi[z/0]) \models \Psi \text{ (Definition 3.1)} \\
&\quad \text{iff} \quad \models \phi[z/0]\left(\lceil \Psi \rceil_{\mathcal{H}^{z^{\mathcal{N}}}}^{L(X,z)}\right) \text{ (induction hypothesis)} \\
&\quad \text{iff} \quad \models \phi\left(\lceil z^{\mathcal{N}}.\Psi \rceil_{\mathcal{H}}^{L(X)}\right) \text{ (Definition 4.9)}
\end{aligned}$$

$$\begin{aligned}
&\mathcal{H}, (L, \phi) \models AG\,\Psi \\
&\quad \text{iff} \quad \mathcal{H}, \sigma \models \Psi \text{ for every } \sigma \text{ reachable from } (L, \phi) \text{ (Lemma 3.2)} \\
&\quad \text{iff} \quad \forall \sigma\ ((L, \phi), \sigma) \in (\overset{\star}{\mapsto} \cup \overset{\mathtt{tr}}{\mapsto})^* \Rightarrow \mathcal{H}, \sigma \models \Psi \\
&\quad \text{iff} \quad \forall N, \phi'\ N(\phi'(X)) \in \text{minMod}(L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}) \Rightarrow \mathcal{H}, (N, \phi') \models \Psi \\
&\qquad\qquad \text{(Lemma 4.3)} \\
&\quad \text{iff} \quad \exists \Im\ \Im \models L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}\ \&\ \forall N, \phi'\ (N, \phi') \in \Im \Rightarrow \mathcal{H}, (N, \phi') \models \Psi \\
&\quad \text{iff} \quad \exists \Im\ \Im \models L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}\ \&\ \forall N, \phi'\ (N, \phi') \in \Im \Rightarrow\ \models \phi'\left(\lceil \Psi \rceil_{\mathcal{H}}^{N(X)}\right) \\
&\qquad\qquad \text{(induction hypothesis)} \\
&\quad \text{iff} \quad \exists \Im\ \Im \models L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}}\ \&\ \Im \models \textstyle\bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \rightarrow \lceil \Psi \rceil_{\mathcal{H}}^{N(X)} \\
&\quad \text{iff} \quad \exists \Im\ \Im \models L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}} \wedge \textstyle\bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \rightarrow \lceil \Psi \rceil_{\mathcal{H}}^{N(X)} \\
&\quad \text{iff} \quad \models \exists L_1, \ldots, L_n\ L(\phi(X)) \wedge \mathcal{R}_{\mathcal{H}} \wedge \textstyle\bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \rightarrow \lceil \Psi \rceil_{\mathcal{H}}^{N(X)} \\
&\quad \text{iff} \quad \models \phi\left(\lceil AG\,\Psi \rceil_{\mathcal{H}}^{L(X)}\right)
\end{aligned}$$

$\mathcal{H}, (L, \phi) \models EG\, \Psi$

iff $\exists \rho\, (\rho \in \mathrm{runs}(\mathcal{H}, (L, \phi))\ \&\ \forall \pi\, (\pi \in \mathrm{pos}(\rho) \Rightarrow \mathcal{H}, \rho(\pi) \models \Psi))$
(Definition 3.1)

iff $\exists \Im\, \Im \models L(\phi(X)) \wedge \mathcal{I}_{\mathcal{H}}\ \&\ \forall N, \phi'\ N(\phi'(X)) \in \Im \Rightarrow \mathcal{H}, (N, \phi') \models \Psi))$
(Lemma 4.6)

iff $\exists \Im\, \Im \models L(\phi(X)) \wedge \mathcal{I}_{\mathcal{H}}\ \&\ \forall (N, \phi')\ (N, \phi') \in \Im \Rightarrow \models \phi' \left(\lceil \Psi \rceil_{\mathcal{H}}^{N(X)} \right)$
(induction hypothesis)

iff $\exists \Im\, \Im \models L(\phi(X)) \wedge \mathcal{I}_{\mathcal{H}}\ \&\ \Im \models \bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \to \lceil \Psi \rceil_{\mathcal{H}}^{N(X)}$

iff $\exists \Im\, \Im \models L(\phi(X)) \wedge \mathcal{I}_{\mathcal{H}} \wedge \bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \to \lceil \Psi \rceil_{\mathcal{H}}^{N(X)}$

iff $\models \exists L_1, \dots, L_n\ L(\phi(X)) \wedge \mathcal{I}_{\mathcal{H}} \wedge \bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \to \lceil \Psi \rceil_{\mathcal{H}}^{N(X)}$

iff $\models \phi \left(\lceil EG\, \Psi \rceil_{\mathcal{H}}^{L(X)} \right)$

Finally, $\mathcal{H}, (L, \phi) \models EF\, \Psi$ iff $\mathcal{H}, (L, \phi) \not\models AG\, \neg\Psi$ iff $\not\models \phi \left(\lceil AG\, \neg\Psi \rceil_{\mathcal{H}}^{L(X)} \right)$ iff[8] $\models \phi \left(\lceil EF\, \Psi \rceil_{\mathcal{H}}^{L(X)} \right)$ and also $\mathcal{H}, (L, \phi) \models AF\, \Psi$ iff $\mathcal{H}, (L, \phi) \not\models EG\, \neg\Psi$ iff $\not\models \phi \left(\lceil EG\, \neg\Psi \rceil_{\mathcal{H}}^{L(X)} \right)$ iff $\models \phi \left(\lceil AF\, \Psi \rceil_{\mathcal{H}}^{L(X)} \right)$. □

4.3 *Until*-Formulas

The *Until*-operators $\Phi\, EU\, \Psi$ and $\Phi\, AU\, \Psi$ give rise to a slight complication of the deductive model checking approach described in this paper. Let us first illustrate their treatment with the help of a special case, namely Φ being a constraint formula. In order to check a property of the form $c\, EU\, \Psi$ in state σ_0 for the hybrid system $\mathcal{H}$ we have to find out whether there exists a run $\rho = \sigma_0 \mapsto_{f_0}^{t_0} \sigma_1 \mapsto_{f_1}^{t_1} \sigma_2 \mapsto_{f_2}^{t_2} \cdots$ such that $\mathcal{H}, (L_i, f_i(t)) \models \Psi$ for some $0 \leq t \leq t_i$ and for all states "inbetween" the constraint c holds. The reachability theory (together with the initial state) is only helpful in determining whether such a Ψ is about to hold. It does not tell us, though, what happens inbetween. In order to overcome this problem, we introduce the notion of a c-safe transition. Intuitively, c-safe transitions preserve the constraint c. Now, the set of states reachable via c-safe transitions is definitely a subset of the set of reachable states. Moreover, if a state with property Ψ is reachable via c-safe transitions then there exists a prefix of at least one run of the hybrid system such that each transition within this prefix is c-safe – which guarantees that the states occurring in this prefix have property c – and which ends with a state having property Ψ. In other words,

[8]Note that this "if and only if" holds because $\phi \left(\lceil AG\, \neg\Psi \rceil_{\mathcal{H}}^{L(X)} \right)$ contains no free symbols whatsoever, and therefore is either $\top$ or $\bot$.

if a state with property Ψ is reachable via c-safe transitions then $c\,EU\,\Psi$ holds. The other direction holds trivially anyway. Hence, what remains to be done is to describe the reachability theory for c-safe transitions. This, however, is actually very simple, for we just have to add c as an additional location invariant for all locations of the hybrid system. For instance, the local reachability theory of L then changes to

$$\forall X\ L(X) \rightarrow \begin{cases} inv(L) \wedge c\ \wedge \\ \forall \delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X \delta] \wedge c[X/X + k_L^X \delta]\ \rightarrow \\ \qquad L(X + k_L^X \delta))\ \wedge \\ \quad \bigwedge\limits_{T=(L,N)\in\mathcal{E}} guard(T) \rightarrow N(act(T,X)) \end{cases}$$

Note that adding this constraint to the invariants of all locations ensures that c is also preserved for edge-transitions, (L, N) say. Also note, that the above change in the reachability theory allows us to describe the operator $c\,EU\,\Psi$ where the interval in which c is supposed to hold includes the two interval borders. This might not be very satisfactory for many interesting problems. Thus, if we want to exclude the left border, we have to change the local reachability theory for L to

$$\forall X\ L(X) \rightarrow \begin{cases} inv(L)\ \wedge \\ \forall \delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X \delta] \wedge c[X/X + k_L^X \delta]\ \rightarrow \\ \qquad L(X + k_L^X \delta))\ \wedge \\ \quad \bigwedge\limits_{T=(L,N)\in\mathcal{E}} guard(T) \wedge c[X/act(T,X)] \rightarrow N(act(T,X)) \end{cases}$$

The difference to the earlier local reachability theory is that c is no longer forced to hold for the initial state, but is guaranteed to hold after time and edge transitions. Excluding the right border of the interval can be done by considering $(c \vee \Psi)\,EU\,\Psi$ instead of $c\,EU\,\Psi$. The latter way to describe reachability theories therefore seems to be the most general one.

For the general case, we have to consider ICTL formulas of the form $\Phi\,EU\,\Psi$ where Φ is not necessarily a constraint formula. This complicates matters again a bit because the additional invariant to hold is $\lceil\Phi\rceil_{\mathcal{H}}^{L(X)}$ for location L and thus differs for each location.

Definition 4.11
We define the reachability theory $\mathcal{R}_{\mathcal{H}}(c_{L_1}, \ldots, c_{L_n})$, where $n = |\mathcal{L}|$, for the

hybrid system $\mathcal{H}$ under the constraints $c_{L_1}, \dots, c_{L_n}$ as:

$$\mathcal{R}_{\mathcal{H}}(c_{L_1}, \dots, c_{L_n}) =$$
$$\bigwedge_{L_i \in \mathcal{L}} \forall X\ L_i(X) \rightarrow \begin{cases} \mathit{inv}(L_i)\ \wedge \\ \forall \delta\ (\delta \geq 0 \wedge \mathit{inv}(L_i)[X/X + k^X_{L_i}\delta] \wedge \\ \quad c_{L_i}[X/X + k^X_{L_i}\delta]) \rightarrow L_i(X + k^X_{L_i}\delta)\ \wedge \\ \displaystyle\bigwedge_{T=(L_i,L_j)\in\mathcal{E}} \begin{array}{l} \mathit{guard}(T) \wedge c_{L_j}[X/\mathit{act}(T,X)] \rightarrow \\ L_j(\mathit{act}(T,X)) \end{array} \end{cases}$$

Evidently, by the above Definition, $\mathcal{R}_{\mathcal{H}} = \mathcal{R}_{\mathcal{H}}(\top, \dots, \top)$.

DEFINITION 4.12
The characteristic second-order formula associated with the ICTL-formula $\Phi\, EU\, \Psi$ in location L for the hybrid system $\mathcal{H}$ is defined as:

$$\lceil \Phi\, EU\, \Psi \rceil^{L(X)}_{\mathcal{H}} = \neg \exists L_1, \dots L_n \begin{cases} L(X) \wedge \mathcal{R}_{\mathcal{H}}\left(\lceil \Phi \rceil^{L_1(X)}_{\mathcal{H}}, \dots, \lceil \Phi \rceil^{L_n(X)}_{\mathcal{H}}\right) \wedge \\ \displaystyle\bigwedge_{N \in \mathcal{L}} \forall X\ N(X) \rightarrow \lceil \Psi \rceil^{N(X)}_{\mathcal{H}} \end{cases}$$

How to describe the operator AU in terms of EU can be found in [AHH96].

5 Second-Order Quantifier Elimination

So far, we have defined how to obtain a second-order characteristic constraint formula from a given verification problem (a hybrid system with initial state and a property to be checked). This second-order formula is now to be proved valid. To this end we make use of the *Elimination Theorem* [NS95, NS99, NOS99] that allows us to transform a given second-order formula into an equivalent first-order formula if possible.

NOTATION 5.1
As usual, by $\Phi^{\overline{x}}_{\overline{y}}$ we mean Φ with each x_i in the sequence $\overline{x}$ replaced by the corresponding y_i from the sequence $\overline{y}$. With $\Phi\left[P(\overline{\alpha})/\Psi^{\overline{x}}_{\overline{\alpha}}\right]$ we refer to Φ with every occurrence of the predicate symbol P replaced by the formula Ψ. The argument sequence $\overline{\alpha}$ here allows us to name the argument list of the respective occurrences.

THEOREM 5.2 (ELIMINATION THEOREM)
Let Φ and Ψ be two first-order formulas which are positive with respect to the predicate symbol P. Then

$$\exists P\ \left(\forall \overline{x}\, (P(\overline{x}) \rightarrow \Phi) \wedge \Psi\right) \quad \equiv \quad \Psi\left[P(\overline{\alpha})/\left(\nu P(\overline{x}).\Phi(P)\right)^{\overline{x}}_{\overline{\alpha}}\right]$$

$$\textit{where} \quad \nu P(\overline{x}).\Phi(P) = \bigwedge_{i \leq \omega} \Phi^i(\top) \quad \textit{with} \quad \Phi^0(\top) = \top, \Phi^{n+1}(\top) = \Phi(\Phi^n(\top))$$

The proof of this Theorem can be found in [NS95] (but also see [NS99, NOS99]). There, in addition, some generalizations and dual forms are examined. For the purpose of this paper, however, the above form suffices. Note that evaluating such a greatest fixpoint, means to successively compute each $\Phi^i(\top)$ until we reach one that is entailed by its predecessor $\Phi^{i-1}(\top)$. The monotonicity of Φ with respect to P (P occurs only positively within Φ) then guarantees that each further iteration would also be implied by $\Phi^{i-1}(\top)$. In fact, for simplicity, it is often not necessary to fully compute each $\Phi^i(\top)$. It suffices to consider only those conjuncts in $\Phi^{i-1}(\top)$ that are not already subsumed by one of its predecessors.

The above Elimination Theorem is fairly general for it does not take the special appearance of the reachability and inevitability theories into account. Yet, in many interesting cases – namely those where a location predicate is to be eliminated for which no edge transition to itself exists – we can provide with a special case of the Elimination Theorem whose application does not require the computation of fixpoints. This special case is given below.

Corollary 5.3 (Simplification Lemma)
Suppose that Ψ contains L only positively and that Φ has no mention of L at all. Then

$$\exists L \left[\Psi \wedge \forall X\ L(X) \rightarrow \left\{ \begin{array}{l} inv(L) \wedge \Phi\ \wedge \\ \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta] \rightarrow L(X + k_L^X\delta)) \end{array} \right\}\right]$$

$$\Leftrightarrow$$

$$\Psi\left[L(\overline{\alpha})\ /\ (inv(L) \wedge \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta] \rightarrow \Phi[X/X + k_L^X\delta]))\tfrac{X}{\overline{\alpha}}\right]$$

Proof: By applying the Elimination Theorem. Recall that Φ is supposed to have no mention of L. We are thus able to compute the fixpoint of the right-hand side of the implication sign as:

$$\begin{array}{rcl}
\Gamma^1(\top) & = & inv(L) \wedge \Phi \\
\Gamma^2(\top) & = & \Gamma^1(\top) \wedge \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta]\ \rightarrow \\
 & & \qquad inv(L)[X/X + k_L^X\delta] \wedge \Phi[X/X + k_L^X\delta]) \\
 & = & \Gamma^1(\top) \wedge \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta] \rightarrow \Phi[X/X + k_L^X\delta]) \\
\Gamma^3(\top) & = & \Gamma^2(\top) \wedge \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta] \rightarrow \\
 & & \qquad \forall\delta'\ (\delta' \geq 0 \wedge inv(L)[X/X + k_L^X\delta + k_L^X\delta']\ \rightarrow \\
 & & \qquad\quad \Phi[X/X + k_L^X\delta + k_L^X\delta'])) \\
 & = & \Gamma^2(\top) \wedge \forall\delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X\delta] \rightarrow \\
 & & \qquad \forall\delta'\ (\delta' \geq 0 \wedge inv(L)[X/X + k_L^X(\delta + \delta')]\ \rightarrow \\
 & & \qquad\quad \Phi[X/X + k_L^X(\delta + \delta')]))
\end{array}$$

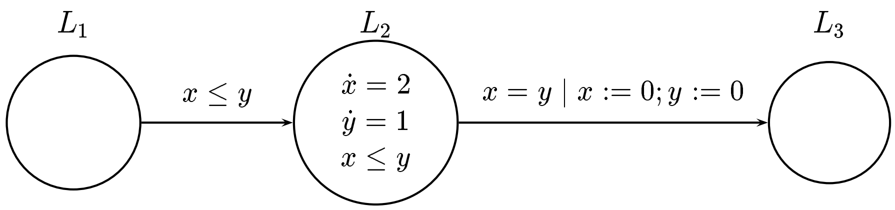
At this stage it is easy to see that $\forall X\ \Gamma^2(\top) \rightarrow \Gamma^3(\top)$ and therefore we are done with the fixpoint computation and the result (after simplification)

$inv(L) \wedge \forall \delta\ (\delta \geq 0 \wedge inv(L)[X/X + k_L^X \delta] \rightarrow \Phi[X/X + k_L^X \delta])$. This final result (the "computed" fixpoint) has to be substituted for every occurrence within the formula Ψ where the free variables have to be instantiated accordingly. $\square$

The above Lemma is useful because it can safe us a lot of fixpoint computations. It states that it is almost trivial to eliminate a location predicate from a (reachability or inevitability) theory provided the location has no self-loop (Φ contains no L in the preliminaries of the Lemma). Evidently, applications of the Simplification Lemma (and also the Elimination Theorem) usually introduce new edges and therefore it is very unlikely that all eliminations can be performed only with the help of the above Lemma. However, it is obvious that many eliminations are just of the above kind.

The purpose of both the Simplification Lemma and the Elimination Theorem, is to successively eliminate existentially quantified (location) predicates. I.e., each elimination reduces the number of locations of the hybrid system by one. Such eliminations result in new properties and new transitions that, in a sense, represent paths through the eliminated location.

As an illustration let us assume that we have to verify that $AG\ x+y \leq 10$ holds for a hybrid system that contains the following sub-system.



Suppose that we are now about to eliminate location L_2. According to the approach presented in this paper this means that we have to compute – in fact, find a first-order equivalent for – the second-order formula

$$\exists L_2 \left[\begin{array}{l} \forall x,y\ L_1(x,y) \rightarrow x \leq y \rightarrow L_2(x,y)\ \wedge \\ \forall x,y\ L_2(x,y) \rightarrow \left\{ \begin{array}{l} x \leq y\ \wedge \\ x + y \leq 10\ \wedge \\ \forall \delta\ (\delta \geq 0 \wedge x + 2\delta \leq y + \delta\ \rightarrow \\ \qquad L_2(x + 2\delta, y + \delta))\ \wedge \\ x = y \rightarrow L_3(0,0) \end{array} \right\} \end{array} \right]$$

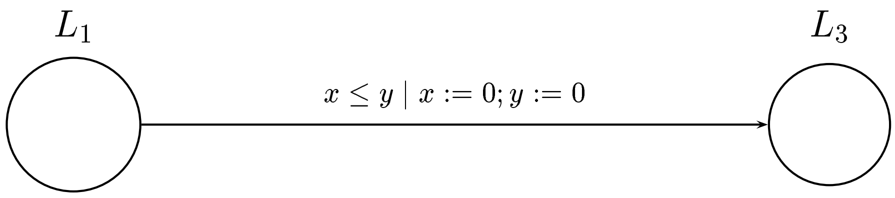
The five conjuncts of the above second-order formula describe the transition from L_1 to L_2, the location invariant for L_2, the property to be proved, the time transition for location L_2, and the edge transition from L_2 to L_3 respectively.

Now, what we would expect as the result of eliminating L_2? Evidently, location L_2 will vanish. And also, we will have to introduce a new edge from

location L_1 to location L_3 which in a sense represents the sub-path through L_2. The guard for this new edge should be $x \leq y$ which is inherited from the edge between L_1 and L_2. The discrete action for the new edge should be $x := 0; y := 0$ which is inherited from the edge between L_2 and L_3. But this cannot be all, and indeed this is not all that is computed by the elimination. As a concrete example suppose that we are in location L_1 with $x = 4$ and $y = 5$. In the new system, i.e., after eliminating L_2, we can see that the guard of the new edge holds and therefore we can make a transition to L_3 while resetting both x and y to 0. Moreover, the property to be proved, namely $x + y \leq 10$ is never violated. In the original system, however, we could also perform the transition from L_1, this time with destination L_2, though. We can leave L_2 only when x and y have an equal value, namely 6, which is reached after exactly one time unit. After leaving L_2 we reach L_3 with both x and y reset to 0. But note, in the original system the property to be proved $(x + y \leq 10)$ has been violated in location L_2, e.g., when both data variables had the value 6.
It is thus not sufficient to merely add the new edge; we also have to find the necessary and sufficient condition on the data variables in L_1 such that the property to be proved cannot be violated within location L_2. And indeed, this is what the Elimination Theorem (and also the Simplification Lemma in this case) allows us to compute. According to the Simplification Lemma and some further simplifications based on variable eliminations in quantified constraint formulas we can see that the above second-order formula is equivalent to

$$\forall x, y\ L_1(x, y) \to x \leq y \to L_3(0, 0)$$
$$\forall x, y\ L_1(x, y) \to x \leq y \to 2y \leq x + 5$$

The first formula describes just the new edge to be introduced. The second formula, however, tells us about the necessary and sufficient condition on the data variables for location L_1 such that it would be impossible to violate $x + y \leq 10$ in location L_2.
Thus, what we achieved by eliminating location L_2 is, that we now can switch to the somewhat simpler system we obtain by replacing the sub-system from above by



For this simplified system we then have to show that $AG\ x+y \leq 10$ (inherited from the original problem) and also that $x \leq y \to 2y \leq x + 5$ for location L_1.

6 Examples

6.1 The Initial Simple Example

Recall the hybrid system of page 6 for which we wanted to prove that $AG\ 2z \leq y$. According to Example 4.8 on page 12 this means to check the validity of

$$\exists L, N \left[\begin{array}{lll} L(0,0,0) & & \\ \forall x,y,z & L(x,y,z) & \to x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\ \forall x,y,z & L(x,y,z) & \to x = 1 \to N(0,y,z) \\ \forall x,y,z & L(x,y,z) & \to \forall \delta\ \delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z \to \\ & & \qquad L(x+\delta, y+\delta, z) \\ \forall x,y,z & N(x,y,z) & \to x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \\ \forall x,y,z & N(x,y,z) & \to L(0,y,z) \\ \forall x,y,z & N(x,y,z) & \to \forall \delta\ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge \\ & & \qquad 0 \leq y + \delta \wedge 0 \leq z + \delta) \to \\ & & \qquad\qquad N(x+\delta, y+\delta, z+\delta) \\ \forall x,y,z & L(x,y,z) & \to 2z \leq y \\ \forall x,y,z & N(x,y,z) & \to 2z \leq y \end{array}\right.$$

I.e., we apply the Simplification Lemma and/or the Elimination Theorem successively to the existentially quantified location predicates L and N. For instance, applying the Simplification Lemma to the part of the above second-order formula that is concerned with the location predicate L, i.e.,

$$\exists L \left[\begin{array}{l} L(0,0,0) \ \wedge \\ \forall x,y,z\ N(x,y,z) \to L(0,y,z) \ \wedge \\ \forall x,y,z\ L(x,y,z) \to \left\{\begin{array}{l} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 2z \leq y \\ \forall \delta\ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z \ \to \\ \quad L(x+\delta, y+\delta, z)) \\ x = 1 \to N(0,y,z) \end{array}\right. \end{array}\right.$$

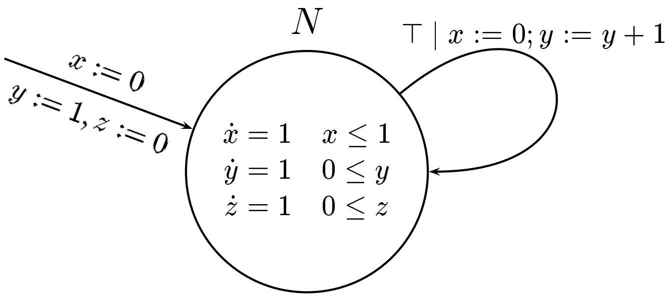
results in (after some easy simplifications, e.g. with Fourier's algorithm)

$$\begin{array}{l} N(0,1,0) \ \wedge \\ \forall x,y,z\ N(x,y,z) \to 0 \leq y \wedge 0 \leq z \wedge 2z \leq y \wedge N(0,y+1,z). \end{array}$$

It therefore remains to eliminate N in the resulting formula as given below.

$$\exists N \left[\begin{array}{l} N(0,1,0) \ \wedge \\ \forall x,y,z\ N(x,y,z) \to \left\{\begin{array}{l} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 2z \leq y \ \wedge \\ N(0,y+1,z) \ \wedge \\ \forall \delta\ 0 \leq \delta \leq 1 - x \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \to \\ \quad N(x+\delta, y+\delta, z+\delta) \end{array}\right. \end{array}\right.$$

Interestingly, this first elimination step resulted in a second-order formula which we could equally obtain from the attempt to prove $AG\ 2z \leq y$ for the hybrid system



In a sense, the new arrows – one for the initial situation and one describing a loop from N to itself – take over the responsibility of the old location L. Now we have to eliminate the remaining second-order quantification from the above formula. This time, however, we cannot apply the Simplification Lemma for the location N has got a self-loop after eliminating L. We therefore have to proceed with the more general Elimination Theorem. I.e., we have to evaluate $\nu N(x,y,z).\Gamma(N)$ where

$$\Gamma(N) = \begin{cases} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 2z \leq y \ \wedge \\ N(0, y+1, z) \ \wedge \\ \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ \qquad N(x+\delta, y+\delta, z+\delta)) \end{cases}$$

We do so by successively computing the $\Gamma^i(\top)$.

$$\begin{array}{lcl} \Gamma^0(\top) & = & \top \\ \Gamma^1(\top) & = & x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 2z \leq y \\ \Gamma^2(\top) & = & \Gamma^1(\top) \ \wedge \\ & & \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \rightarrow 2z + \delta \leq y) \\ & = & \Gamma^1(\top) \wedge 1 + 2z \leq x + y \\ \Gamma^3(\top) & = & \Gamma^2(\top) \ \wedge \\ & & \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ & & \qquad 1 + 2(z+\delta) \leq x + y + 2\delta) \\ & = & \Gamma^2(\top) \end{array}$$

Hence, $\nu N(x,y,z).\Gamma(N) \equiv x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 1 + 2z \leq x + y$ and a final instantiation with the values $0, 1, 0$ for the variables x, y, z respectively results in

$$0 \leq 1 \wedge 0 \leq 1 \wedge 0 \leq 0 \wedge 1 + 0 \leq 0 + 1 \equiv \top$$

Thus, we have finally proved that the original hybrid system indeed satisfies $AG\ 2z \leq y$.

Now, let us change the property to be proved to $AG\ 3z \leq y$, i.e., we consider the same hybrid system as before (on page 6) but try to prove a property that does *not* hold.

Then the elimination of L does not make a real difference, we just have to substitute a 2 with a 3 in the final result. For the elimination of N, however, things change drastically. We have to compute $\nu N(x,y,z).\Gamma(N)$ where

$$\Gamma(N) = \begin{cases} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 3z \leq y \ \wedge \\ N(0, y+1, z) \ \wedge \\ \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ \qquad N(x+\delta, y+\delta, z+\delta)) \end{cases}$$

The various $\Gamma^i(\top)$ then result in

$$\begin{array}{rcl} \Gamma^0(\top) & = & \top \\ \Gamma^1(\top) & = & x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge 3z \leq y \\ \Gamma^2(\top) & = & \Gamma^1(\top) \wedge \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ & & \qquad\qquad 3z + 3\delta \leq y + \delta) \\ & = & \Gamma^1(\top) \wedge 2 + 3z \leq 2x + y \end{array}$$

Now, note that we ultimately have to instantiate the variables x, y, and z in the fixpoint result by 0, 1, and 0 respectively. Also note, that $\nu N(x,y,z).\Gamma(N) \rightarrow \Gamma^i(\top)$ for each i. If we take a look at $\Gamma^2(\top)$ we observe that its instantiation results in $\perp$ and therefore we know that $\nu N(x,y,z).\Gamma(N)$ must be equivalent to $\perp$, i.e., the property does not hold. It thus makes sense to check each $\Gamma^i(\top)$ after it has been generated for instantiation, for this might lead to considerable simplifications.

As a final little variant of the example let us exchange $2z \leq y$ with $az \leq y$, i.e., we introduce a parameter a to the property to be proved. Again, the elimination of L does not make a real difference to the earlier cases, we just have to substitute a 2 with an a in the elimination result. And again, for the elimination of N things change indeed. We have to compute $\nu N(x,y,z).\Gamma(N)$ where

$$\Gamma(N) = \begin{cases} x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge az \leq y \ \wedge \\ N(0, y+1, z) \ \wedge \\ \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ \qquad N(x+\delta, y+\delta, z+\delta)) \end{cases}$$

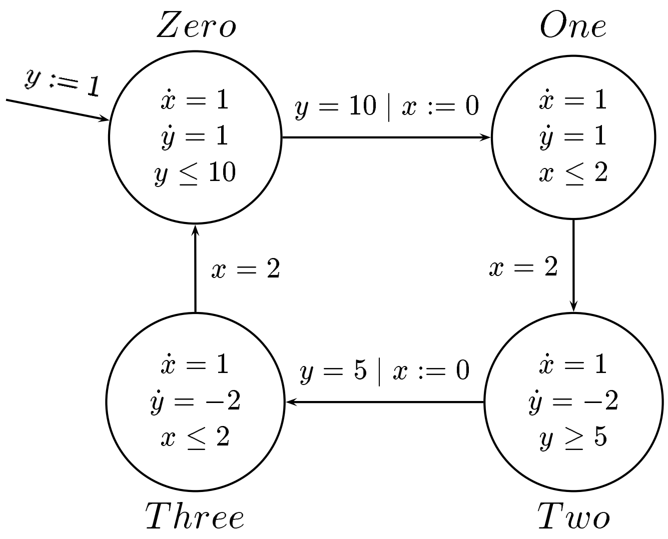
The various $\Gamma_i(\top)$ then result in

$$\begin{array}{rcl} \Gamma^0(\top) & = & \top \\ \Gamma^1(\top) & = & x \leq 1 \wedge 0 \leq y \wedge 0 \leq z \wedge az \leq y \\ \Gamma^2(\top) & = & \Gamma^1(\top) \wedge \forall \delta \ (\delta \geq 0 \wedge x + \delta \leq 1 \wedge 0 \leq y + \delta \wedge 0 \leq z + \delta \ \rightarrow \\ & & \qquad\qquad 3z + 3\delta \leq y + \delta) \\ & = & \Gamma^1(\top) \wedge a - 1 + 3z \leq y + (a-1)x \end{array}$$

Now, note that we ultimately have to instantiate the variables x, y, and z in the fixpoint result by 0, 1, and 0 respectively. Also note, that $\nu N(x,y,z).\Gamma(N) \rightarrow \Gamma^i(\top)$ for each i. If we take a look at $\Gamma^2(\top)$ we observe that its instantiation results in $a \leq 2$ and therefore we know that $\nu N(x,y,z).\Gamma(N)$ at least implies $a \leq 2$. By taking this additional knowledge into account, the fixpoint computation terminates with just this result $a \leq 2$. We therefore have shown that the example hybrid system has property $AG\ az \leq y$ if and only if the *parameter* a has a value less than or equal to two.

6.2 The Water Level Monitor

The hybrid system is given as follows:



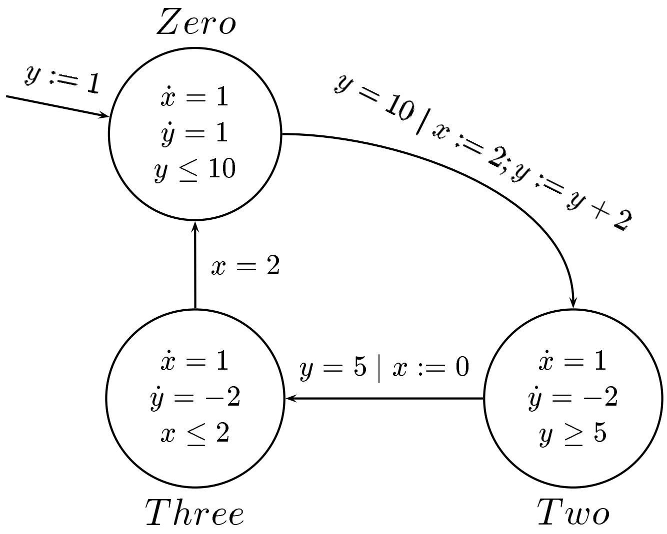
It is to be checked whether the water level (denoted by the data variable y) always remains between 1 and 12, i.e., we have to prove the ICTL property $AG\ (1 \leq y \wedge y \leq 12)$. According to the deductive model checking approach presented in this paper this means to prove the validity of the second-order formula

$$
\exists \begin{array}{l}\text{Zero}\\ \text{One}\\ \text{Two}\\ \text{Three}\end{array}
\left(\begin{array}{lll}
\text{Zero}(x,1) & & \\
\forall x,y\ \text{Zero}(x,y) & \rightarrow & \left\{\begin{array}{l} y \le 10 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge y+\delta \le 10\ \rightarrow \\ \quad \text{Zero}(x+\delta, y+\delta) \\ y = 10 \rightarrow \text{One}(0,y) \end{array}\right. \\
\forall x,y\ \text{One}(x,y) & \rightarrow & \left\{\begin{array}{l} x \le 2 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge x+\delta \le 2\ \rightarrow \\ \quad \text{One}(x+\delta, y+\delta) \\ x = 2 \rightarrow \text{Two}(x,y) \end{array}\right. \\
\forall x,y\ \text{Two}(x,y) & \rightarrow & \left\{\begin{array}{l} y \ge 5 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge y-2\delta \ge 5\ \rightarrow \\ \quad \text{Two}(x+\delta, y-2\delta) \\ y = 5 \rightarrow \text{Three}(0,y) \end{array}\right. \\
\forall x,y\ \text{Three}(x,y) & \rightarrow & \left\{\begin{array}{l} x \le 2 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge x+\delta \le 2\ \rightarrow \\ \quad \text{Three}(x+\delta, y-2\delta) \\ x = 2 \rightarrow \text{Zero}(x,y) \end{array}\right.
\end{array}\right.
$$

According to Lemma 5.3 this is equivalent to (by eliminating location One)

$$
\exists \begin{array}{l}\text{Zero}\\ \text{Two}\\ \text{Three}\end{array}
\left(\begin{array}{lll}
\text{Zero}(x,1) & & \\
\forall x,y\ \text{Zero}(x,y) & \rightarrow & \left\{\begin{array}{l} y \le 10 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge y+\delta \le 10\ \rightarrow \\ \quad \text{Zero}(x+\delta, y+\delta) \\ y = 10 \rightarrow \text{Two}(2,y+2) \end{array}\right. \\
\forall x,y\ \text{Two}(x,y) & \rightarrow & \left\{\begin{array}{l} y \ge 5 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge y-2\delta \ge 5\ \rightarrow \\ \quad \text{Two}(x+\delta, y-2\delta) \\ y = 5 \rightarrow \text{Three}(0,y) \end{array}\right. \\
\forall x,y\ \text{Three}(x,y) & \rightarrow & \left\{\begin{array}{l} x \le 2 \wedge 1 \le y \wedge y \le 12 \\ \forall\delta\ \delta \ge 0 \wedge x+\delta \le 2\ \rightarrow \\ \quad \text{Three}(x+\delta, y-2\delta) \\ x = 2 \rightarrow \text{Zero}(x,y) \end{array}\right.
\end{array}\right.
$$

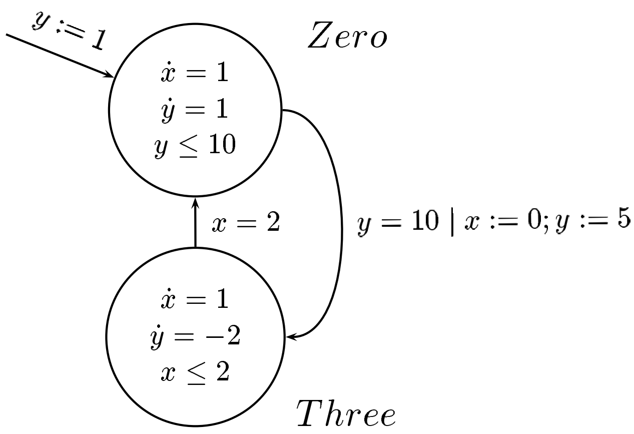
Note that this is exactly the formula we would have obtained from attempting to prove $AG\ (1 \le y \wedge y \le 12)$ for the hybrid system



Again by Lemma 5.3 this is equivalent to (by eliminating location Two)

$$\exists \text{Zero}, \text{Three} \left(\begin{array}{lcl} \text{Zero}(x, 1) & & \\ \forall x, y\ \text{Zero}(x, y) & \to & \left\{ \begin{array}{l} y \leq 10 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge y + \delta \leq 10\ \to \\ \quad \text{Zero}(x + \delta, y + \delta) \\ y = 10 \to \text{Three}(0, 5) \end{array} \right. \\ \forall x, y\ \text{Three}(x, y) & \to & \left\{ \begin{array}{l} x \leq 2 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge x + \delta \leq 2\ \to \\ \quad \text{Three}(x + \delta, y - 2\delta) \\ x = 2 \to \text{Zero}(x, y) \end{array} \right. \end{array} \right.$$

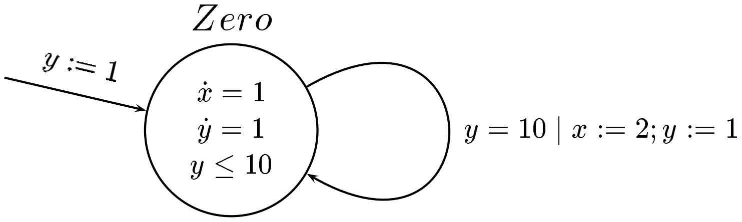
This second-order formula would have equally been obtained by proving $AG\ (1 \leq y \wedge y \leq 12)$ for the hybrid system



A final application of Lemma 5.3 then leads to (after eliminating location Three)

$$\exists \text{Zero} \left(\begin{array}{l} \text{Zero}(x,1) \\ \forall x, y \ \text{Zero}(x,y) \quad \rightarrow \quad \left\{ \begin{array}{l} y \leq 10 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta \ \delta \geq 0 \wedge y + \delta \leq 10 \ \rightarrow \\ \quad \text{Zero}(x+\delta, y+\delta) \\ y = 10 \rightarrow \text{Zero}(2,1) \end{array} \right. \end{array} \right)$$

Again, this would be exactly the formula we would get from the attempt to prove $AG\ (1 \leq y \wedge y \leq 12)$ for some simpler hybrid system, namely



This final second-order formula trivially reduces to $\top$, and that in fact again with the Simplification Lemma alone (since the self-loop is subsumed by the initial state), and so the desired property is proved valid.

7 Generalizations

7.1 Parameterization

The characteristic second-order formula we obtain from an ICTL-formula, a hybrid system and a ground initial state has no free symbols whatsoever unless the formula, the system, or the initial state are parameterized with constants over the reals. In this case the characteristic formula represents a constraint on these parameters. This constraint is the neccessary and sufficient condition on the parameters for the ICTL-formula to hold. As an example recall the parameterized system property from page 23.

7.2 Approximations

The fixpoint computations do not necessarily terminate in general. In the standard reachability analysis of hybrid systems one therefore often considers certain more or less strict approximations of the set of reachable sets. Evidently such approximations are also possible for the approach presented here. For instance, one might consider the convex hull of the constraint formulas that arise from the elimination of some of the locations. Also one might think of artificially terminating the fixpoint computations after a certain amount of iterations. In both cases we end up in defintions for the location to be eliminated that are unnecessarily "big". Similarly, there exist possibilities to approximate "smaller" candidates for the location to be eliminated. For further details the reader is refered to the relevant literature.

7.3 Rectangular Hybrid Systems

In this paper the approach is described merely in terms of linear hybrid systems, i.e. data variables are assumed to change their value by a certain constant amount (which might vary from data variable to data variable) per time unit. Nevertheless, the whole approach also works for rectangular hybrid systems, i.e. for systems within which the change in the data variables is only described by some interval over the reals. For instance, recall the definition of the reachability theory for some hybrid system (Definition 4.2). One part of it consists of the clause $\forall \delta\ (\delta \geq 0 \wedge inv(L)[x/x + k_L^x \delta] \rightarrow L(x + k_L^x \delta))$, where k_L^x denotes the real number that describes the change of x in L within one time unit. If, however, we are given an interval, say $[a, b]$, rather than a fixed number we have to change the corresponding part of the reachability theory to $\forall \delta, \beta\ (\delta \geq 0 \wedge a \leq \beta \leq b \wedge inv(L)[x/x + \beta\delta] \rightarrow L(x + \beta\delta))$. The non-linearity can easily be resolved and so we finally end up with linear formulas again. The Railroad-Gate-Controller from [AHH96] certainly is one of the most famous examples of a rectangular hybrid system.

8 Experimental Results

There exists a prototype implementation of the Elimination Approach (for proving safety-properties) written in Sicstus-Prolog with the CLP(Q,R)-library for constraint handling. Briefly, the overall procedure implemented works as follows: (i) read the problem file, (ii) compute the compound automaton (parameters are additional arguments), (iii) add the property to be proved (and also delete some of the time transitions in case this is required from some "urgent" or "as-soon-as-possible"-semantics), (iv) select one of the initial locations, (v) eliminate the selected location (thus possibly introducing new inital locations), (vi) if finished or trivial then stop; otherwise proceed with step (iv). The approach of selecting initial locations for elimination has the obvious advantage that it will never be attempted to eliminate an unreachable location. On the other hand, such a strategy takes away much of the freedom to choose whatever location we want for elimination. Another feature of the implementation is that it allows us to abstract from (some of the local) locations of a compound automaton. This makes it possible to perform a (forward or backward) reachability analysis (see below) which allows for a thorough comparison between reachability and elimination approaches.

Of major interest was the question whether there can be anything better (at least for safety properties) than forward reachability provided it at all terminates. After all, within forward reachability we compute exactly the set of reachable states; and in fact we need to know about all the reachable states for proving safety properties. Thus, forward reachability does not compute any redundant information. However, it sometimes performs redundant computations. This can happen whenever a reachability analysis requires more than one pass through the reachable locations before it terminates. Systems for which a single pass is sufficient are probably best examined by forward reachability.

We claim that the Elimination Approach presented in this paper can help us to avoid such redundant computations. This is the case for instance for the famous "audio-protocol"-example. For other, unfortunately rather trivial systems like the "Leaking Gas Burner" or the "Billiards"-example, the Elimination Approach showed a slightly better behavior than standard reachability analysis. However, in such cases, where safety properties can be proved in milliseconds anyway, this can hardly be called "evidence".

The lack of non-trivial hybrid system in the literature that require several passes through some of their locations made us compose our own examples. They are designed as simple as possible such that they may serve to illustrate the effect of the Elimination Approach compared to reachability analysis methods. Some such examples are given below.

8.1 Simulating Reachability Analysis

The Elimination Approach as described in this paper assumes that a predicate symbol is introduced for each of the locations of the (compound) automaton. This method therefore is neither a forward nor a backward analysis approach. However, if we put these location names into the argument list and introduce a single and unique dummy predicate symbol instead that replaces each of the older location names then it becomes obvious that the Elimination Approach – by eliminating the new dummy predicate – performs a backward reachability analysis. Also, if we perform this location abstraction but eliminate with the dual form of the Elimination Theorem, i.e., let P occur only negatively in Φ and in Ψ then

$$\exists P\ [\forall \overline{x}\ (P(\overline{x}) \vee \Phi) \wedge \Psi] \equiv \Psi \left[\overline{P}(\overline{\alpha}) / \left(\nu \overline{P}(\overline{x}).\Phi\right)\frac{\overline{x}}{\overline{\alpha}}\right]$$

In this case a forward reachability analysis is performed. For compound systems one can even perform something like a "mixed" approach by abstracting from only some of the local systems.

8.2 Some Further Examples

8.2.1 Railroad-Gate-Controller

This example is taken from [AHH96]. It describes the control system for a railroad gate that has to guarantee that the gate is closed whenever a train is near and that it is open in cases where it is safe to be open. The whole system consists of three component systems: a train, a gate, and the controller with three, four and again three locations respectively. This suggests that the composed system has at most 36 locations. However, this number is restricted by the synchronisation labels that forbid certain edge compositions. As it turns out, the composed automaton has 22 locations, but some of the guards denote what we call *impossible guards*, i.e., constraint formulas that will never become true because of the source location invariant. Such impossible guards usually cannot be discovered syntactically, but they obviously may reduce the number of reachable locations[9] considerably. In fact, this railroad-gate-controller example has only 7 reachable locations and there is only little non-determinism. This makes the example fairly trivial, despite it looks rather complicated at the first glance.

Both the Elimination Approach and forward reachability analysis prove the safety requirement $AG\ (x \leq 10 \rightarrow \text{Gate.closed})$ in about 0.5 seconds on a 333 MHz UltraSPARC. The dual version of the Elimination Approach and backward reachability analysis require 1.0 sec. and 1.3 sec. respectively.[10]

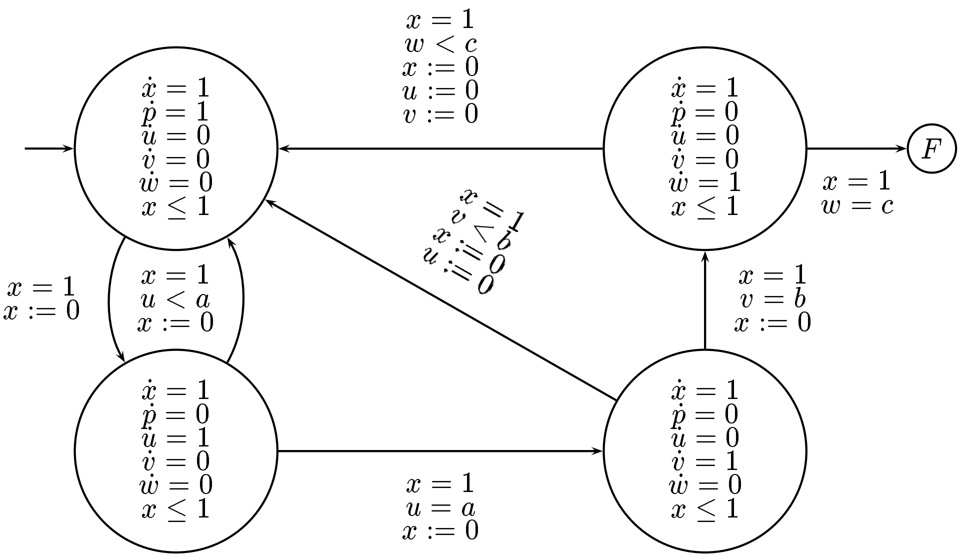
8.2.2 A Silly Multiplier

This is an example where three positive numbers a, b, and c are multiplied and the final product is stored in the data variable p. The multiplication is performed by successively adding 1 to p, similar to the nested for-loop

```
for (w:=0; w<c; w++)
    for (v:=0; v<b; v++)
        for (u:=0; u<a; u++)  {p:=p+1}
```

[9] Notice the difference between reachable locations and reachable states. A location is reachable if there exists a reachable state that has this very location as its first component. If a location is not reachable then there exists no reachable state with this location.

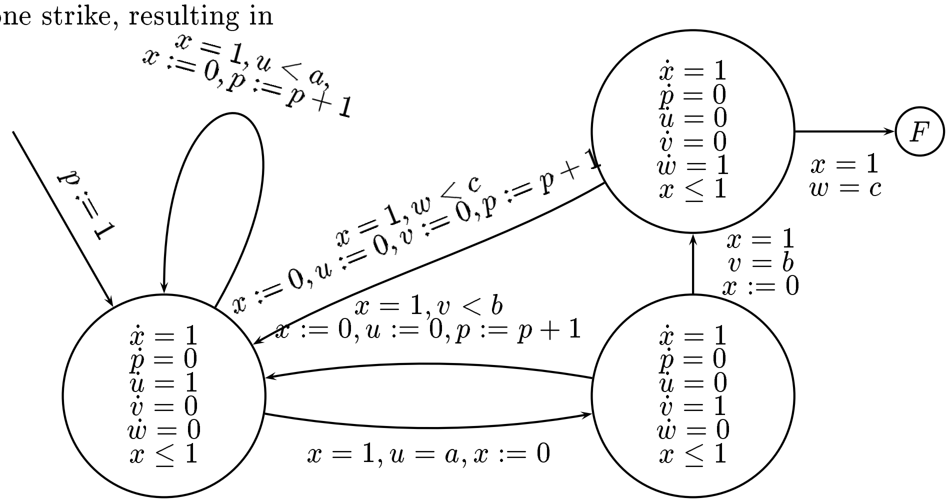
[10] Interestingly, forward reachability requires twice as many iterations as backward reachability, but also it is about twice as fast. This is explained by the fact that an iteration step in the backward analysis is far more complicated than an iteration step during forward analysis. It has to take much more states, even impossible ones, into account.



It is to be shown that the location F can be reached – after all, as soon as F is reached, the data variable p contains the multiplication result we are interested in. (Backward or forward) reachability analysis in a sense simulates the behavior of the multiplier. I.e., since this system is fully deterministic, it takes a walk through the whole computation. Evidently, this is very time consuming even if we only attempted to compute $10 \times 10 \times 10$; it takes approximately 8000 iterations.

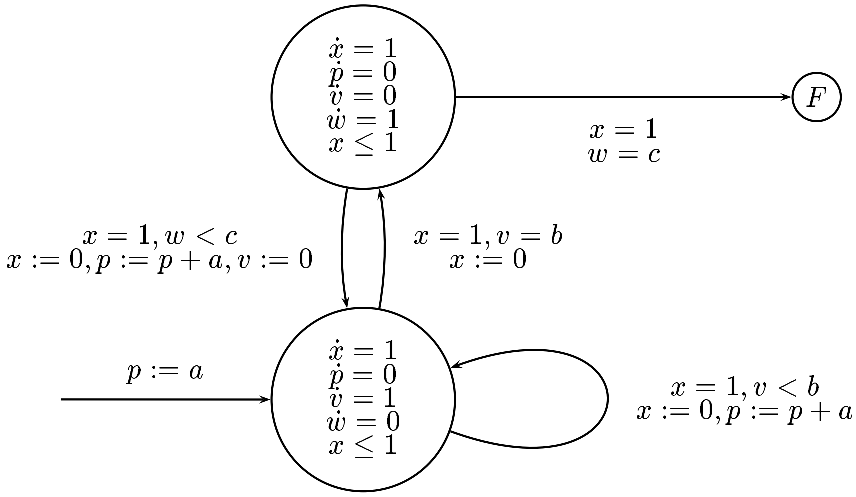
Now, compare this with the Elimination Approach.[11] (In the automata below irrelevant information within the locations or at the transitions are omitted for readability)

The Simplification Lemma allows us to eliminate the top left location in one strike, resulting in

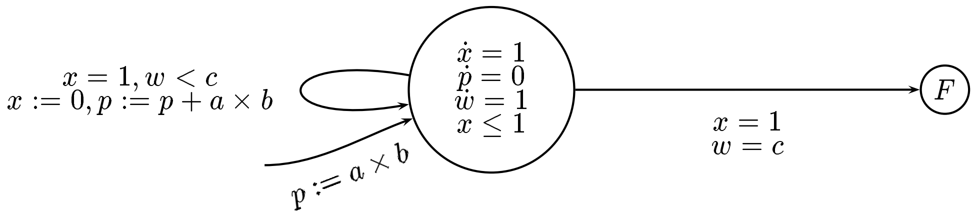


[11] The prototype implementation of the Elimination Approach is designed for safety properties only. Thus, in order to prove that location F can be reached we have to show that it is not the case that location F will never be reached.

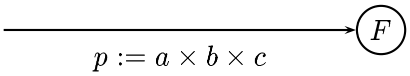
As the next canditate for a location elimination the prototype implementation chooses the bottom left location and after approx. $2a$ iterations it ends up with



The next step is to eliminate the new bottom location. After about $2b$ iterations it reaches



Now, in a final elimination the system attempts to get rid of the left location. This requires another $2c$ iterations and provides us with this final picture

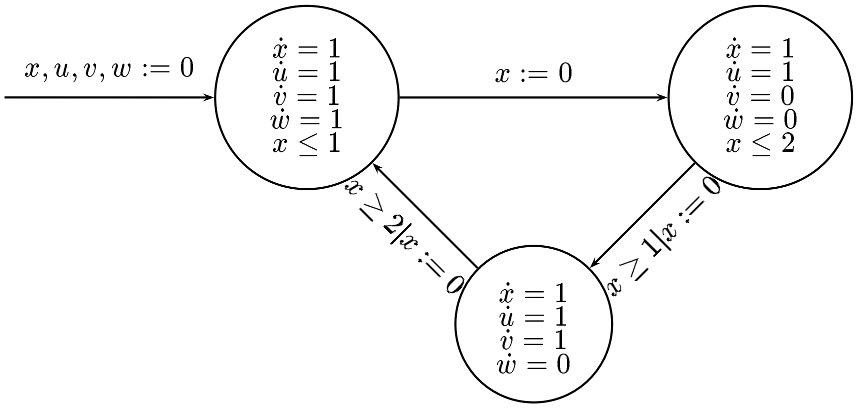


In this remaining trivial system there is only one location which, in particular, is also the initial location and which therefore is trivially reachable. The attempt to prove that F will never be reached thus fails. Therefore The location F can be reached and, while entering it, the data variable p will contain the product of the positive numbers a, b, and c.

As for a concrete example: in order to compute the product $10 \times 10 \times 10$ the prototype implementation of the Elimination Approach requires about 0.8 seconds on a 333 MHz UltraSparc, whereas forward reachability analysis (utilizing the same implementation) needs some 380 seconds. The more sphisticated symbolic model checker HyTech, version 1.04, required some 12.3 seconds on the same machine (forward reachability).

8.2.3 A Long Loop

The following example again demonstrates the effect on long loops. In contrast to the multiplier example, however, the long loop is not inherent in the system; it comes from the property to be proved.

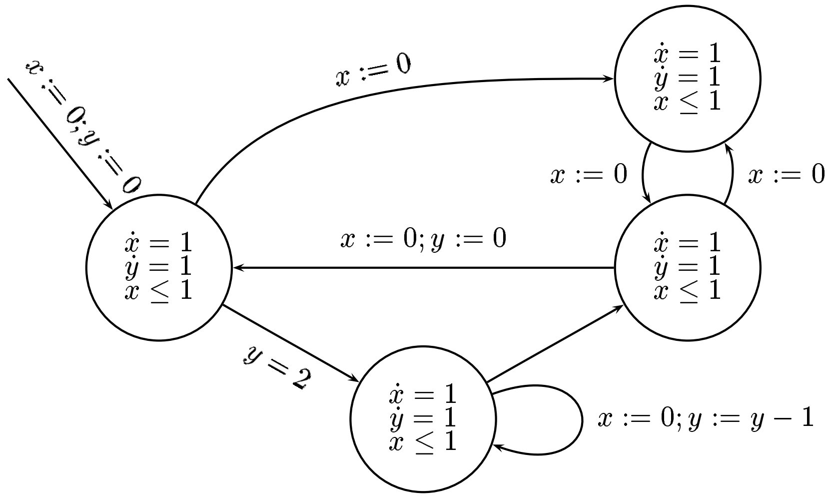


Suppose that, for some reason, we want to show that $AG\ (u \geq 154 \rightarrow 5.9 \times w \leq u + v)$. Although the system is fairly simple, the property to be proved requires a reachability analysis to somehow (backward) simulate the system over a rather long period of time. In fact, forward reachability does not terminate within a reasonable amount of time and backward reachability requires some 95 seconds on a 333 MHz UltraSparc. Unfortunately, HyTech version 1.04, the Berkeley symbolic model checker for embedded systems runs into a library overflow error after about 60 seconds.

The implementation of the Elimination Approach, on the other hand, first eliminates the two top locations within a fraction of a second (this requires only the Simplification Lemma) and, as an intermediate result, comes up with a system that consists of merely one remaining location that has a transition that leads to itself. It therefore has to be eliminated with the Elimination Theorem and the implemented system does so in about 7.5 seconds on a 333 MHz UltraSparc.

8.2.4 Where Reachability Fails

The particularity about the next example is that it contains an "impossible" location, i.e., one of the locations – the bottom one – is unreachable because the guard $(y = 2)$ of the transition that may lead to this very location can impossibly become true.



In a sense, forward reachability analysis detects this impossible transition, although rather indirectly, for it never tries to compute states which involve this location. Nevertheless, forward reachability does not terminate, since it derives more and more new reachable states that involve the two rightmost locations. At the first glance, backward reachability might have a better chance. Suppose we were about to prove that $x \leq y$ is an overall invariant of the system. If there were not the bottom location, backward reachability would have no problem to detect that the invariant indeed holds. However, this invariant does not hold for the bottom location and the only reason why this is non-critical for the whole system is the mere fact that this location is not reachable anyway. It is thus simply not necessary to try and prove the invariant for this very location. However, backward reachability cannot find out by itself that there is an impossible transition and therefore neither terminates.

Now, what does the Elimination Approach (or actually its prototype implementation) do with this example? After about 0.1 seconds (on a 333 MHz UltraSparc) it has eliminated the top three locations and ends up with the remaining bottom location, however, without any newly generated inital transition. This means that there exists a trivial model for the remaining set of formulas, namely the one that assigns $\perp$ (false) to the remaining location predicate, and the system terminates with success. The Elimination Approach thus allows us to solve this problem in a tiny fraction of a second.

8.3 Final Conclusion

The Elimination Approach has been tested on quite a lot of examples taken from the relevant literature, the various verifier distributions, and also self-made. Some of them are small and trivial like the Water Level Monitor, or the Leaking Gas Burner. Unfortunately, it seems that almost all non-trivial examples that can be found in the literature are designed such that a forward reachability analysis terminates after a single run through the reachable locations. Only those standard examples for which the properties to be proved force a reachability analysis to travel several (even many) times through the reachable locations showed how valuable the Elimination Approach can be. For instance, take the Billiards example from [ACH+95] and modify the movement of the white ball such that it is pushed almost vertically (or almost horizontically). Then any reachability analysis will have to perform many iterations through the fixpoint computation (one for each bounce) and it will take quite some time to come up with the desired result. Two of the self-made examples from above are also along these lines. Both the Silly Multiplier and the Long Lasting Loop require many iterations in a reachability analysis. The Elimination Approach, however, is insensitive to this fact. It simply eliminates the involved locations one by one and therefore never has to visit these locations again.

References

[ABL97] Luca Aceto, Augusto Burgueño, and Kim G. Larsen. Model checking via reachability testing for timed automata. BRICS Report Series RS-97-29, BRICS, Department of Computer Science, University of Aarhus, 1997.

[ACD90] R. Alur, C. Courcoubetis, and D. L. Dill. Model checking for real-time systems. In *Proceedings of the 5th Annual Symposium on Logic in Computer Science*, pages 414–425, 1990.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifaksi, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[ACHH93] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, pages 209–229. Springer Verlag, Lecture Notes in Computer Science, vol. 736, 1993.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AH92] R. Alur and T. A. Henzinger. Logics and models of real-time: A survey. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, pages 74–106. Springer Verlag, New York, LNCS 600, 1992.

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[AHS96] R. Alur, T. A. Henzinger, and E. Sontag, editors. *Hybrid Systems III.* Lecture Notes in Computer Science, Springer Verlag, 1996.

[ANKS95] P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors. *Hybrid Systems II.* Lecture Notes in Computer Science, vol. 999, Springer Verlag, 1995.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Workshop Logic of Programs*. Springer Verlag, Lecture Notes in Computer Science, vol. 131, 1981.

[CES86] E. M. Clarke, , E. A. Emerson, and A. P. Sisla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Programming Languages and Systems*, 8(2):244–263, 1986.

[CHR91] Zhao Chaochen, C. A. R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40:269–276, 1991.

[EMSS90] E. A. Emerson, A. Mok, A. P. Sistla, and J. Srinivasan. Quantitative temporal reasoning. In *CAV 90: Computer Aided Verification*, pages 163–145. Lecture Notes in Computer Science, vol. 531, Springer Verlag, New York, 1990.

[GH90] Dov Gabbay and I.M. Hodkinson. An axiomatization of the temporal logic with until and since over the real numbers. *Journal of Logic and Computation*, 1(2):229–260, 1990.

[GNRR93] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems.* Springer Verlag, Lecture Notes in Computer Science, vol. 736, 1993.

[Hen91] T. A. Henzinger. *The Temporal Specification and Verification of Real-Time Systems.* PhD thesis, Stanford University, Stanford, Ca., 1991.

[Hen95] T. A. Henzinger. Hybrid automata with finite bisimulations. In *ICALP 95: Automata, Languages, and Programming*, pages 324–335. Springer Verlag, Lecture Notes in Computer Science, vol. 944, 1995.

[Hen96] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th LICS*, pages 278–292. IEEE Comp. Soc. Press, 1996.

[HNSY92] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. In *Proceedings of the 7th Annual Symposium on Logic in Computer Science*, pages 394–406. IEEE Computer Society Press, New York, 1992.

[Ho95] Pei-Hsin Ho. *Automatic Analysis of Hybrid Systems.* PhD thesis, Cornell University, 1995.

[Kop96] P. Kopke. *The Theory of Rectangular Hybrid Systems.* PhD thesis, Cornell University, 1996.

[LLW95] François Laroussinie, Kim G. Larsen, and Carsten Weise. From timed automata to logic – and back. BRICS Report Series RS-95-2, BRICS, Department of Computer Science, University of Aarhus, 1995.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer Verlag, New York, 1992.

[MW84] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Prog. Lan. Syst.*, 6(1):68–93, 1984.

[Non95] Andreas Nonnengart. *A Resolution-Based Calculus for Temporal Logics.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, December 1995.

[Non96] Andreas Nonnengart. Resolution-based calculi for modal and temporal logics. In Slaney McRobbie, editor, *Proceedings of the 13th CADE*, pages 598–612. Springer Verlag, LNAI 1104, 1996.

[NOS99] Andreas Nonnengart, Hans Jürgen Ohlbach, and Andrzej Szałas. Elimination of predicate quantifiers. In Hans Jürgen Ohlbach and Uwe Reyle, editors, *Logic, Language and Reasoning – Essays in Honour of Dov Gabbay*, page ??? Kluwer, Dordrecht, Netherlands, 1999. ISBN: 0-7923-5687-X.

[NS95] Andreas Nonnengart and Andrzej Szałas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. Technical Report MPI-I-95-2-007, Max-Planck-Institute for Computer Science, Saarbrücken, Germany, March 1995. Available at: http://www.mpi-sb.mpg.de/∼nonnenga.

[NS99] Andreas Nonnengart and Andrzej Szalas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. in: [Orł99], 1999.

[OL82] J. S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Trans. Prog. Lan. Syst.*, 4(3):455–495, 1982.

[Orł99] Ewa Orłowska, editor. *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, volume 24 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, c/o Springer Verlag, 1999. ISBN: 3-7908-1164-5.

[PH88] A. Pnueli and E. Harel. Applications of temporal logic to the specification of real-time systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 84–93. Lecture Notes in Computer Science, vol. 331, Springer Verlag, New York, 1988.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, New York, 1977.

[Sha93] N. Shankar. Verification of real-time systems using PVS. In Costas Courcoubetis, editor, *Proceedings of the CAV '93*, pages 280–291. Springer Verlag, LNCS 697, 1993.

[Sis85] E. M. Sistla, A. P. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[SUM96] Henny B. Sipma, Tomás E. Uribe, and Zohar Manna. Deductive model checking. In *Proceedings of the 8th International Conference on Computer Aided Verification*, pages 208 – 219. Springer Verlag, LNCS 1102, 1996.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Im Stadtwald
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages
MPI-I-1999-3-002	J. Raskin, P. Schobbens	The Logic of Event Clocks
MPI-I-1999-3-001	S. Vorobyov	New Lower Bounds for the Expressiveness and the Higher-Order Matching Problem in the Simply Typed Lambda Calculus
MPI-I-1999-2-005	J. Wu	Symmetries in Logic Programs
MPI-I-1999-2-004	V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes	Decidable fragments of simultaneous rigid reachability
MPI-I-1999-2-003	U. Waldmann	Cancellative Superposition Decides the Theory of Divisible Torsion-Free Abelian Groups
MPI-I-1999-2-001	W. Charatonik	Automata on DAG Representations of Finite Trees
MPI-I-1999-1-007	C. Burnikel, K. Mehlhorn, M. Seel	A simple way to recognize a correct Voronoi diagram of line segments
MPI-I-1999-1-006	M. Nissen	Integration of Graph Iterators into LEDA
MPI-I-1999-1-005	J.F. Sibeyn	Ultimate Parallel List Ranking ?
MPI-I-1999-1-004	M. Nissen, K. Weihe	How generic language extensions enable “open-world” desing in Java
MPI-I-1999-1-003	P. Sanders, S. Egner, J. Korst	Fast Concurrent Access to Parallel Disks
MPI-I-1999-1-002	N.P. Boghossian, O. Kohlbacher, H.-. Lenhof	BALL: Biochemical Algorithms Library
MPI-I-1999-1-001	A. Crauser, P. Ferragina	A Theoretical and Experimental Study on the Construction of Suffix Arrays in External Memory
MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems

MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid *E*-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-2-002	F. Jacquemard, C. Meyer, C. Weidenbach	Unification in Extensions of Shallow Equational Theories
MPI-I-98-1-031	G.W. Klau, P. Mutzel	Optimal Compaction of Orthogonal Grid Drawings
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph
MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	*q*-gram Based Database Searching Using a Suffix Array (QUASAR)
MPI-I-98-1-023	C. Burnikel	Rational Points on Circles
MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns
MPI-I-98-1-020	C. Rüb	On Wallace's Method for the Generation of Normal Variates
MPI-I-98-1-019		2nd Workshop on Algorithm Engineering WAE '98 - Proceedings
MPI-I-98-1-018	D. Dubhashi, D. Ranjan	On Positive Influence and Negative Dependence
MPI-I-98-1-017	A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, E. Ramos	Randomized External-Memory Algorithms for Some Geometric Problems
MPI-I-98-1-016	P. Krysta, K. Loryś	New Approximation Algorithms for the Achromatic Number
MPI-I-98-1-015	M.R. Henzinger, S. Leonardi	Scheduling Multicasts on Unit-Capacity Trees and Meshes
MPI-I-98-1-014	U. Meyer, J.F. Sibeyn	Time-Independent Gossiping on Full-Port Tori
MPI-I-98-1-013	G.W. Klau, P. Mutzel	Quasi-Orthogonal Drawing of Planar Graphs
MPI-I-98-1-012	S. Mahajan, E.A. Ramos, K.V. Subrahmanyam	Solving some discrepancy problems in NC*
MPI-I-98-1-011	G.N. Frederickson, R. Solis-Oba	Robustness analysis in combinatorial optimization
MPI-I-98-1-010	R. Solis-Oba	2-Approximation algorithm for finding a spanning tree with maximum number of leaves
MPI-I-98-1-009	D. Frigioni, A. Marchetti-Spaccamela, U. Nanni	Fully dynamic shortest paths and negative cycle detection on diagraphs with Arbitrary Arc Weights
MPI-I-98-1-008	M. Jünger, S. Leipert, P. Mutzel	A Note on Computing a Maximal Planar Subgraph using PQ-Trees