Diploma Thesis

Algebraic Methods and Abstractions for Automated 3-valued Reasoning on Hybrid Automata

Kerstin Bauer

January 6, 2008

Supervisors:
Prof. Dr. Gerhard Pfister
Prof. Dr. Klaus Schneider
Dr. Raffaella Gentilini

AG Algebra, Geometrie und Computeralgebra
Department of Mathematics
University of Kaiserslautern

Contents

Chapter 1

Preface

Hybrid automata are a well developed formalism proposed in the early nineties by T. Henzinger for the modeling and the automated reasoning on hybrid systems, i.e. dynamical systems characterized by the interaction between discrete and continuous dynamics. The latter are ubiquitous in a variety of mathematical and engineering application fields (real-time systems, embedded hardware/software, aeronautics, robotics.,...). As originally envisioned in [Hen96, HKPV98] hybrid automata have aspired to combine well established formal tools arising from Mathematics, Logic, and Computer Science, in particular finite automata and differential equations. In hybrid automata, each discrete mode is represented by a location, while the corresponding continuous dynamic is expressed via a system of differential equations. As a result of the above formulation, hybrid automata expose an immediately understood trade-off between their representation fidelity and the solvability of related decidability problems addressing properties such as the reachability issue (is a given target set of states reachable from the initial conditions?), which is fundamental for the safety-analysis of the underlying hybrid system.

To date, along the detection of the exact border between decidability and undecidability for hybrid automata [HKPV98, AHLP00, Mil00, LPS00] a major effort of the related research community is devoted on the development of techniques for the symbolic analysis of undecidable - and yet reasonably expressive - hybrid automata [GTT03, PAM$^+$05, TK02, RS05]. Most of the methods developed so far focus on the over-approximation of the set of reachable states with applications to the safety certification of the modeled hybrid systems. In particular, a variety of abstraction methods based on the notion of simulation-preorder have been explored by many authors [GTT03, RS05, TK02]. In general, the simulation preorder from the abstraction to the hybrid automaton allows for preservation only of valid formulas in the universal fragment of a branching time temporal logic. Some techniques for the inference of polynomial invariants by means of Gröbner bases techniques have been proposed in [SSM04, RCT05], applying to linear and algebraic hybrid automata. Few authors address the problem of underapproximated reachability analysis by means of bounded reachability techniques and semi-algebraic representation of sets of states [PAM$^+$05]. Recently, a novel framework has been proposed in [GSM07] which allows to (1) combine over- and underapproximated reachability analysis on hybrid automata and (2) both prove and provide counter-example to general reactive system properties expressed by means of the computational tree temporal logic, on hybrid automata. The work in [GSM07] is based on the definition of improv-

ing abstractions of the original hybrid automaton, and of a corresponding three-valued semantics for the temporal logic CTL .

The diploma thesis has two objectives. The first objective is to use algebraic methods for the analysis of algebraic hybrid automata, based on Gröbner bases techniques. After giving a brief review of the state of the art in that research field, an algorithm for generating polynomial invariants, which has been proposed in [SSM04, San05], will be improved and made incremental. Furthermore, it will be shown, that Gröbner bases techniques are not the tool of choice for a combined over- and underapproximated reachability analysis on hybrid automata, since this technique only supports overapproximations of continuous and discrete pre- and postimages of sets of states.

The second objective of this thesis is to develop a three-valued semantics for the μ-calculus extending the ideas of the framework sketched in [GSM07] for combined over- and underapproximated reachability and three-valued CTL model checking on hybrid automata. This is done in two steps: In a first step, we develop a general parametric semantic framework, where preservation results for general μ-calculus formula only depend on the concrete instantiation of the modal operators. In a second step, these results are applied to two kinds of concrete abstractions of hybrid automata, allowing combined over / underapproximated reachability analysis: namely, the discrete bounded bisimulation abstraction introduced in [GSM07] and abstractions based on may / must transitions, which extend to the hybrid domain to the modal abstractions for discrete systems developed in [SG04].

The thesis is structured as follows: In Chapter 2, hybrid automata, CTL and μ-calculus on hybrid automata, three-valued logics and the basics of computer algebra are introduced. Chapter 3 will give an overview of the state of the art on methods based on Gröbner bases techniques together with the improvement of the invariant generation proposed in [SSM04, San05]. Furthermore, the inappropriateness of the use of Gröbner bases techniques for combined over- and underapproximated reachability analysis on hybrid automata will be shown. In Chapter 4, notions of abstraction frameworks for hybrid automata are presented. On this ground in Chapter 5 the theoretical background for a three-valued semantics for the μ-calculus is developed and applied to the concrete abstractions based on discrete bounded bisimulation and may / must relations.

This thesis has evolved with the advice, feedback and help of many people. I would like to express my gratitude especially to Prof. Schneider, Prof. Pfister and Dr. Gentilini for giving me the opportunity to work on this subject and supporting me with advice and suggestions throughout my time. I also want to thank everybody whom I have not mentioned before including my family. The support of the 'Promotionsprogramm des Fachbereichs Informatik der TU Kaiserslautern' is gratefully acknowledged.

Chapter 2

Preliminaries

2.1 Hybrid Automata

Hybrid systems are dynamical systems which combine discrete and continuous dynamics. A formal way to model hybrid systems is provided by *hybrid automata*, introduced in [Hen96]. In hybrid automata, the discrete part of the hybrid system is modeled by a finite control graph, whose vertices (locations) denote the different discrete states of the system, and where edges (discrete transitions) represent the discrete dynamics of the system. The continuous part of the hybrid system is encoded via a (finite) set of real-valued variables, continuously flowing according to appropriate sets of differential equations in each location of the hybrid automaton. The switches between locations (and consequently flow rules) are constrained by *edge guards* and *location invariants*.

Figure 2.1 depicts a simple example of a hybrid automaton, modelling a heating system. Such a hybrid automaton has two locations, corresponding to the two discrete control modes **on** and **off** of the modeled heating controller. The temperature, i.e. the continuous part of the system, is represented by the real valued variable x. The system starts with the temperature of 20 degrees at the location **off**. While the heating is off, the temperature in the hybrid automaton falls via the differential rule $\dot{x} = 0.1$. According to the edge guard $x < 20$, the location **off** *may* be left, when the temperature has fallen below 20. Because of the location invariant $x > 18$ the location **off** *must* be left, when the temperature falls below 18 degrees. The permanence in the control mode **on** has similar constraints.

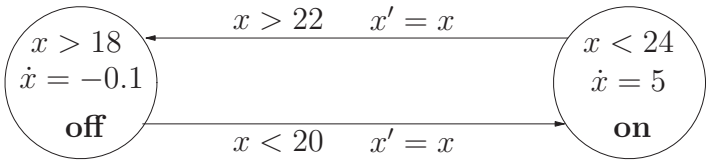


Figure 2.1: Hybrid automaton modeling a heating system

The execution of hybrid automata leads to alternatingly discrete transitions between different locations and continuous changes of the continuous variable of the system within one location. As an example, a pictural view of a *run* of the hybrid automaton for the

hybrid controller in Figure 2.1 is given in Figure 2.2.

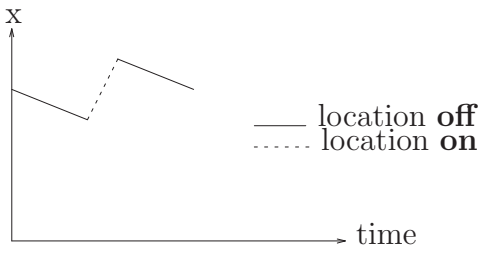


Figure 2.2: Run of the heating controller

Due to possible resets of continuous variables during discrete transitions, the global associated flows can result in *piecewise* continuous functions of the time.

The rest of this section is organized as follows: in Subsection 2.1.1, we define formally the syntax for hybrid automata and give the corresponding semantics in terms of (timed) transition systems. Subsection 2.1.2 introduces the behavioral equivalences of *bisimulation* and *simulation* of (timed) transition systems useful for both abstracting and comparing different hybrid automata.

2.1.1 Formal Syntax and Semantics

The formal syntax of hybrid automata is given in Definition 2.1.

Definition 2.1 (Hybrid Automaton)

A hybrid automaton is a tuple $H = \langle L, E, X, Init, Inv, F, G, R \rangle$ *with the following components:*

- *a finite set of locations* $L = \{l_1, ..., l_k\}$
- *a finite set of discrete transitions* $E \subseteq L \times L$
- *a finite set of continuous variables* $X = \{x_1, ..., x_n\}$
- *an initial set of conditions:* $Init \subseteq L \times \mathbb{R}^n$
- $Inv : L \to 2^{\mathbb{R}^n}$ *the invariant location labeling*
- $F : L \times \mathbb{R}^n \to \mathbb{R}^n$ *assigning to each location* $l \in L$ *a vector field* $F(l, .)$ *that defines the evolution of continuous variables within* l
- $G : E \to 2^{\mathbb{R}^n}$ *the guard edge labeling*
- $R : E \times \mathbb{R}^n \to 2^{\mathbb{R}^n}$ *the reset edge labeling*

A state in H *is a pair* $q = (l, \underline{x}) \in Q$*, where* $l \in L$ *and* $x \in Inv(l) \subseteq \mathbb{R}^n$*.* l *is called the discrete component of* q *and* $\underline{x}$ *the continuous component of* q*.*

Below, we illustrate the formal syntax of hybrid automata given in Definition 2.1 using the hybrid automaton for the heating controller already being introduced in Figure 2.1

Example 2.1 (Heating system)

The formal syntax of the automaton graphically illustrated in Figure 2.1 is given by the tuple $H = \langle L, E, X, Init, Inv, F, G, R \rangle$ *with*

- *Set of Locations:* $L = \{\boldsymbol{off}, \boldsymbol{on}\}$
- *Set of discrete transitions:* $E = \{(\boldsymbol{off}, \boldsymbol{on}), (\boldsymbol{on}, \boldsymbol{off})\}$
- *Set of continuous variables:* $X = \{x\}$
- *Initial condition:* $Init = (\boldsymbol{off}, 20)$
- *Location invariants:* $\boldsymbol{off} \mapsto (18, \infty)$ *and* $\boldsymbol{on} \mapsto (-\infty, 24)$
- *vector field:* $\boldsymbol{off} : \dot{x} = -0.1 \quad$ *and* $\quad \boldsymbol{on} : \dot{x} = 5$
- *guard edge labeling:* $(\boldsymbol{off}, \boldsymbol{on}) \mapsto (-\infty, 20) \quad$ *and* $\quad (\boldsymbol{on}, \boldsymbol{off}) \mapsto (22, \infty)$
- *reset edge labeling:* $((\boldsymbol{off}, \boldsymbol{on}), x) \mapsto x \quad$ *and* $\quad ((\boldsymbol{on}, \boldsymbol{off}), x) \mapsto x$

The semantics of the hybrid automaton H is formally given by the associated *timed transition system* defined in Definition 2.2.

Definition 2.2 (Timed Transition System)

Let $H = \langle L, E, X, Init, Inv, F, G, R \rangle$ *be a hybrid automaton with* $|X| = n$*. The* timed transition system $T_H^t = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ *is defined by the following components:*

- $Q \subseteq L \times \mathbb{R}^n$ *is the state space.* $(l, \underline{x}) \in Q$ *iff* $\underline{x} \in Inv(l)$
- $Q_0 \subseteq Q$ *is the set of initial states* $(l, \underline{x}) \in Q_0$ *iff* $(l, \underline{x}) \in Init$
- $l_{\rightarrow} = \{e\} \cup \mathbb{R}$
- $\rightarrow \subseteq Q \times l_{\rightarrow} \times Q$ *is the set of continuous and discrete transitions defined as follows:*
  - Continuous Transition*:*
    There exists a continuous transition $q = (l, \underline{x}) \xrightarrow{t} q' = (l, \underline{x}')$ *with* $t \in \mathbb{R}^+$ *iff there exists a differentiable function* $f : [0, t] \rightarrow \mathbb{R}^n$*, s.t.*
    1. $f(0) = \underline{x}$ *and* $f(t) = \underline{x}'$
    2. $\forall t' \in [0, t] : f(t') \vDash Inv(l)$ *and* $\dot{f}(t)$ *satisfies the conditions given by the differential rule* $F(l)$
  - Discrete Transition*:*
    There is a discrete transition $(l, \underline{x}) = q \xrightarrow{e} q' = (l', \underline{x}')$ *iff*

1. $(l, l') \in E$
2. *the guard conditions are fulfilled:* $\underline{x} \in G(l, l')$
3. *the reset edge labeling conditions are fulfilled:* $R((l, l'), \underline{x}) = x'$

Now, a *run* $q \rightsquigarrow q'$ of the hybrid automaton H is defined to be a sequence of transitions $q = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} ... \xrightarrow{a_n} q_n = q'$ with $a_i \in \{e\} \cup \mathbb{R}^+$ in the associated timed transition system. Note, however, that different runs can encode the same behavior. Consider therefore the two runs $(l, \underline{x}_0) \xrightarrow{t_1} (l, \underline{x}_1) \xrightarrow{t_2} (l, \underline{x}_2)$ and $(l, \underline{x}_0) \xrightarrow{t_1+t_2} (l, \underline{x}_2)$, which both encode the same continuous path starting in $(l, \underline{x}_0)$ and having a duration of $t_1 + t_2$. Thus, runs which only differ in the encoding of the continuous parts are considered equivalent for the rest of the thesis. Furthermore, in order to avoid the problem of distinguishing between finite and infinite runs during verification, in this thesis we only consider hybrid automata, where every run $q \rightsquigarrow q'$ can be extended to an infinite run, i.e. a run whose duration time of the sum of the continuous transitions is infinite.

Abstracting time from timed transition systems, we obtain the notion of *time abstract transition systems*, formally defining the time abstract semantics of hybrid automata.

Definition 2.3 (Time Abstract Transition System)

The time abstract transition system $T_H = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ *of the hybrid automaton* $H = \langle L, E, X, Init, Inv, F, G, R \rangle$ *is defined by*

- $Q \subseteq L \times \mathbb{R}^n$ *is the state space.* $(l, \underline{x}) \in Q$ *iff* $\underline{x} \in Inv(l)$
- $Q_0 \subseteq Q$ *is the initial set of the system,* $(l, \underline{x}) \in Q_0$ *iff* $(l, \underline{x}) \in Init$.
- $l_{\rightarrow} = \{e\} \cup \{\delta\}$
- $\rightarrow \subseteq Q \times l_{\rightarrow} \times Q$ *is the set of continuous and discrete transitions defined as follows:*
  - continuous transition*:*
    There exists a continuous transition $q = (l, \underline{x}) \xrightarrow{\delta} q' = (l, \underline{x}')$ *iff there exists* $t \in \mathbb{R}^+$ *with* $q \xrightarrow{t} q'$ *in the timed transition system* T_H^t *of* H
  - discrete transition*:*
    There is a discrete transition $q = (l, \underline{x}) \xrightarrow{e} q' = (l', \underline{x}')$ *iff there exists a transition* $q \xrightarrow{e} q'$ *in the timed transition system* T_H^t *of* H.

Like in the timed transition system a *run* $q \rightsquigarrow q'$ of the hybrid automaton H can be described by a sequence of transitions $q = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} ... \xrightarrow{a_n} q_n = q'$ with $a_i \in \{e, \delta\}$ in the associated time abstract transition system.

For the further analysis of hybrid automata H and the associated (time-abstract) transition system one also needs the notion of regions, predecessor- and successor-regions, that are introduced below.

- A subset $r \subseteq Q$ is called a *region* of T_H. If $r \subseteq Q_0$ then r is called *initial region.*

- The predecessor-region $Pre_a(r)$ of a region r by an abstract transition $a \in l_{\rightarrow}$ is defined by $Pre_a(r) := \{x \in Q \mid \exists y \in r : x \xrightarrow{a} y\}$.
  The successor region $Post_a(r)$ is defined by $Post_a(r) := \{x \in Q \mid \exists y \in r : y \xrightarrow{a} x\}$.

In the rest of the thesis, only time abstract transition systems will be considered since for checking safety properties it is not important how long it takes to get somewhere but only whether it is possible to get there.

Example 2.2 (Heating system continued)

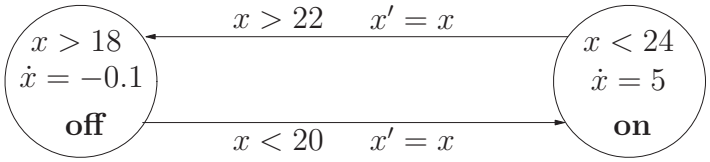


Figure 2.3: Hybrid automaton modeling a heating system

Let us consider again the hybrid automaton representing a simple heating system, which is represented in Figure 2.3. The timed transition system $T_H^t = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ *of the hybrid automaton* H *consists of the elements*

- $Q = \{\textbf{\textit{off}}, \textbf{\textit{on}}\} \times (18, 24)$
- $Q_0 = (\textbf{\textit{off}}, 20)$
- $l_{\rightarrow} := \{e\} \cup \mathbb{R}$
- *Let* $(l, x), (l', x') \in Q$ *and* $t \in \mathbb{R}^+$
  - $(l, x) \xrightarrow{t} (l', x')$, *i.e.* $((l, x), t, (l', x')) \in \rightarrow$
    if $l = l' = \textbf{\textit{off}} \quad \wedge \quad x - 0.1t = x' \quad \wedge \quad x' > 18$ *or*
    if $l = l' = \textbf{\textit{on}} \quad \wedge \quad x + 5t = x' \quad \wedge \quad x' < 24$
  - $(l, x) \xrightarrow{e} (l, x')$, *i.e.* $((l, x), e, (l', x')) \in \rightarrow$
    if $l = \textbf{\textit{off}} \quad \wedge \quad l' = \textbf{\textit{on}} \quad \wedge \quad x < 20 \wedge x' = x$ *or*
    if $l = \textbf{\textit{on}} \quad \wedge \quad l' = \textbf{\textit{off}} \quad \wedge \quad x > 22 \wedge x' = x$

A possible run of H *would be* $(\textbf{\textit{off}}, 20) \xrightarrow{0.5} (\textbf{\textit{off}}, 19.5) \xrightarrow{e} (\textbf{\textit{on}}, 19.5)$.
When considering the time abstract transition system T_H *of this hybrid automaton the only changes to the timed transition system* T_H^t *are*

- $l_{\rightarrow} = \{e\} \cup \{\delta\}$
- *Let* $(l, x), (l', x') \in Q$
  - $(l, x) \xrightarrow{\delta} (l, x')$, *i.e.* $((l, x), \delta, (l, x')) \in \rightarrow$
    iff $\exists t \in \mathbb{R}^+ : (l, x) \xrightarrow{t} (l, x')$ *in the timed transition system* T_H^t

A possible run of H would be $(\textbf{\textit{off}}, 20) \xrightarrow{\delta} (\textbf{\textit{off}}, 19.5) \xrightarrow{e} (\textbf{\textit{on}}, 19.5)$
The δ-predecessor of the region $\{\textbf{\textit{off}}\} \times (19, 20)$ *is* $Pre_\delta(r) = \{\textbf{\textit{off}}\} \times (18, 20)$

2.1.2 Behavioral Equivalence

The notions of *simulation* and *bisimulation* [GSM07, Par81] provide well established formal tools for comparing the behaviors of transition systems. Informally, when a transition system T_1 simulates a transition system T_2, T_1 subsumes all behaviors of T_2. If also T_2 simulates T_1, (i.e. T_1 and T_2 are simulation equivalent) then T_1 and T_2 are indistinguishable by any formula in the *universal fragment* of the expressive modal μ-calculus logic [BBLS93]. If T_1 and T_2 are bisimilar, then they further expose identical behaviors in terms of *any* μ-calculus specification. As an example, abstractions of hybrid automata are usually required in the literature to at least simulate the time abstract transition system of the original hybrid automaton. This requirement guarantees that the abstractions give at least an overapproximation of the underlying hybrid dynamics.

Definition 2.4 (Simulation Relation, Simulation Equivalence)
Let $T^1 = \langle Q^1, Q_0^1, l_{\rightarrow}^1, \rightarrow^1 \rangle$ *and* $T^2 = \langle Q^2, Q_0^2, l_{\rightarrow}^2, \rightarrow^2 \rangle$ *be two transition systems and let P be a partition of* $Q^1 \cup Q^2$. *A non-empty relation on* $\leq_S \subseteq Q^1 \times Q^2$ *is called a simulation from* T^1 *to* T^2 *if for all* $p \leq_S q$*:*

1. $p \in Q_0^1$ *iff* $q \in Q_0^2$,
2. $[p] \equiv_P [q]$ *where* $[p]$ *denotes the class of p in P, and*
3. *For all* $a \in l_{\rightarrow}^1$*: If there exists a* $p' \in Q^1$ *s.t.* $p \xrightarrow{a} p'$, *then there exists a* $q \in Q^2$, *s.t.* $p' \leq_S q'$ *and* $q \xrightarrow{a} q'$.

If there exists a simulation from T^1 *to* T^2 *then* T^2 *simulates* T^1, *short* $T^1 \geq_S T^2$. *If in addition also* $T^2 \geq_S T^1$ *holds, then the transition systems are called simulation equivalent, short* $T^1 \equiv_S T^2$.

The simulation relation defines a preorder on transition systems, since

- $T \leq_S T$ (reflexivity)
- $T^1 \leq_S T^2$ and $T^2 \leq_S T^3 \Rightarrow T^1 \leq_S T^3$ (transitivity).

However, the antisymmetry does not hold. The notion of simulation equivalence $\equiv_S$ is obtained by adding exactly the property of antisymmetry to $\leq_S$.

Definition 2.5 (Bisimulation Equivalence)
Let $T^1 = \langle Q^1, Q_0^1, l_{\rightarrow}^1, \rightarrow^1 \rangle$ *and* $T^2 = \langle Q^2, Q_0^2, l_{\rightarrow}^2, \rightarrow^2 \rangle$ *be two time abstract transition systems and let P be a partition on* $Q^1 \cup Q^2$. *A non-empty relation on* $\equiv_B \subseteq Q^1 \times Q^2$ *is called a bisimulation from* T^1 *to* T^2 *iff for all* $p \equiv_B q$*:*

1. $p \in Q_0^1$ *iff* $q \in Q_0^2$,

2. $[p] \equiv_P [q]$,

3. *For all* $a \in l_{\rightarrow}^1$*: If there exists a* $p' \in Q^1$ *s.t.* $p \xrightarrow{a} p'$*, then there exists a* $q \in Q^2$*, s.t.* $p' \equiv_S q'$ *and* $q \xrightarrow{a} q'$*, and*

4. *For all* $a \in l_{\rightarrow}^2$*: If there exists a* $q' \in Q^2$ *s.t.* $q \xrightarrow{a} q'$*, then there exists a* $p \in Q^1$*, s.t.* $p' \equiv_S q'$ *and* $p \xrightarrow{a} p'$.

T^1 *and* T^2 *are called bisimilar, short* $T^1 \equiv_B T^2$ *iff there exists a bisimulation.*

The bisimulation relation defines an equivalence relation on transition systems, since symmetry, reflexivity and transitivity hold.

Remark 2.1

By the definition of bisimulation, it directly follows that bisimilar transition systems $T^1 \equiv_B T^2$ *are also simulation equivalent. The other direction is not true. To this purpose consider the following example:*

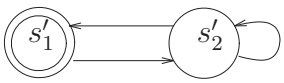


Figure 2.4: T'

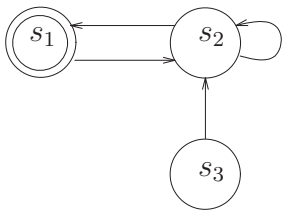


Figure 2.5: T

It is easy to verify, that the transition systems T *and* T' *are simulation equivalent, i.e.* $T \equiv_S T'$*. However, they are not bisimilar, which is seen by contradiction as follows:*
Assume, that the systems were bisimilar. Then

- $s_1 \equiv s_1'$ *and these states are equivalent to no other state, since they are the only initial states*

- $s_2 \equiv s_2'$ *because of* $s_1 \rightarrow s_2$

Now the state s_3 *can neither be equivalent to* s_1' *nor* s_2' *since* s_3 *is no initial state and* $s_3 \nrightarrow s_1$*. Thus,* T *and* T' *cannot be bisimilar.*

2.1.3 Decidability Results for Hybrid Automata

Hybrid automata are ubiquitous in many safety critical application fields. However, because of the complexity given by the mixture of continuous and discrete dynamics, in most cases even the question whether a set of states can be reached from the initial states is undecidable. Only in few simple cases, where either the continuous or the discrete dynamics are highly restricted, the reachability problem is decidable.

In this context, the notions of bisimulation and simulation equivalence play a central role. In fact, similar and bisimilar hybrid automata are not distinguishable by reachability properties. Hence, families of hybrid automata whose members can be reduced to a finite similar or bisimilar representative have a decidable reachability problem. In the rest of this subsection, we briefly survey known families of hybrid automata with a decidable reachability problem, as well as results on undecidability.

2.1.3.1 Timed Automata

In *timed automata* [AD94], the continuous variables denote *clocks*. Clocks measure the time elapsed and thus the continuous flow of these variables is described by $\dot{x}_i = 1$. Furthermore, on the location switchings, clocks either maintain their value or are reset to a constant. Formally, the definition of timed automata is given below:

Definition 2.6 (Timed Automaton)

A timed automaton *is a hybrid automaton* $H = \langle L, E, X, Init, Inv, F, G, R\rangle$ *satisfying the additional conditions*

- $\forall x \in X$ *the differential rule* $\dot{x} = 1$ *defines the flows of the* x_i *in each location* $l \in L$
- *for all discrete transitions* $(l, \underline{x}) \rightarrow (l', \underline{x}')$*:* $x_i' = x_i \vee x_i' \in [a_i, b_i] \subseteq \mathbb{R}$

Because of the simple continuous dynamics of timed automata it is possible to construct a finite bisimulation of any timed automaton although it has exponentially many states. Thus, the reachability problem for timed automata is decidable. Furthermore, the reachability problem has been proved PSPACE-complete for timed automata [AD94].

2.1.3.2 Rectangular Automata

A generalization of timed automata are rectangular automata. Here, instead of having clocks, the continuous variables x_i are allowed to flow according to the flow conditions $\dot{x}_i \in I_i$, where the I_i are arbitrary intervals in $\mathbb{R}$ with rational endpoints. Rectangles $I \subseteq \mathbb{R}^n$ are given by $I = \prod_{i=1}^{n} I_i$. Furthermore, it is required, that the invariant and guard conditions are also made in terms of rectangles. This condition ensures, that the values of two continuous variables are never compared within guard conditions and location invariants.

Definition 2.7 (Rectangular Automaton)

A rectangular automaton *is a hybrid automaton* $H = \langle L, E, X, Init, Inv, F, G, R\rangle$ *satisfying the additional conditions*

- $\dot{\underline{x}} \in I^{flow}(l)$ *where* $I^{flow}(l)$ *is a rectangle*
- *for all discrete transitions* $e = ((l, \underline{x}), (l', \underline{x}'))$*:* $x_i' = x_i \vee x_i' \in I_i^{post}(e)$ *where the* $I_i^{post}(e)$ *are intervals.*
- *for all regions* $l \in L$*:* $Inv(l) = I^{inv}(l)$ *is a rectangle in* $\mathbb{R}^n$
- *for all discrete transitions* $e \in E$*:* $G(e) = I^{guard}(e)$ *is a rectangle in* $\mathbb{R}^n$

A rectangular automaton is initialized*, if for each continuous variable* x_i *and each discrete transition* $(l, l') = e \in E$ *either the flows within* l *and* l' *coincide or the variable* x_i *is reinitialized, i.e.* $x_i' \in I_i^{post}(e)$ *for a given interval* $I_i^{post}(e) \subseteq \mathbb{R}$*.*

An example for a rectangular automaton is the hybrid automaton for the heating controller already being introduced. In each location, the continuous change of the system variable x is constant and during discrete transitions this variable will not be changed. Furthermore, the guard conditions and location invariants are made in terms of intervals. However, the heating controller is not an initialized hybrid automaton, since this would require x to be reinitialized during each discrete transition.

Initialized rectangular automata can be translated to timed automata, so that the obtained timed automaton is timed bisimilar to the rectangular automaton. Thus, initialized rectangular automata are decidable with respect to reachability. A PSPACE-complete complexity result holds also for the reachability problem of initialized rectangular automata [HKPV98].

2.1.3.3 O-minimal Hybrid Automata

Another class of hybrid automata with known decidability results for the reachability problem is the class of o-minimal hybrid automata. For the definition of o-minimal hybrid automata consider a structure $M = \langle \mathbb{R}, <, ...\rangle$ over the real numbers, whose underlying theory $Th(M)$ is the set of first order sentences that hold in M. A set $Y \subseteq \mathbb{R}^n$ is called definable in the structure M if and only if there exists a first order formula $\psi(x_1, ..., x_n)$ such that $Y = \{(a_1, ..., a_n) \mid M \vDash \psi(a_1, ..., a_n)\}$. Then the structure M is called o-minimal iff every definable subset of $\mathbb{R}$ is a finite union of points and (possibly unbounded) intervals.

Definition 2.8 (O-minimal Hybrid Automaton)

A hybrid automaton $H = \langle L, E, X, Init, Inv, F, G, R\rangle$ *is called o-minimal (order minimal), iff*

- *for each location* l*, the smooth vector field* $F(l)$ *is complete,*
- *for each* $(l, l') \in E$*, the reset function* $R : E \to \mathbb{R}^n$ *does not depend on continuous variables (constant reset), and*

- *for each $l \in L$ and $(l, l') \in E$, the sets Inv(l), R(l, l'), G(l), Init(l) and the flow F($l, .$) are definable in the same o-minimal structure.*

Every o-minimal hybrid system admits a finite bisimulation. The computability of the finite bisimulation depends on the o-minimal structure underlying its definition. In particular, positive results apply to the class of semi-algebraic o-minimal hybrid automata.

2.1.3.4 Undecidability Results

As stated in the subsections above, the families of timed automata, initialized rectangular automata and o-minimal hybrid automata are decidable with respect to reachability. However, even slight generalizations of these classes already yield undecidability results for the reachability problem:

- If timed automata are extended so that skewed clocks with $\dot{x}_i = c_i \in \mathbb{R}$ are considered, then the resulting hybrid automata are undecidable.
- Uninitialized rectangular automata are undecidable w.r.t. reachability.
- A closely related family to o-minimal hybrid automata is the class of fully o-minimal automata, where the reset functions need not be constants any more but arbitrary o-minimal functions. This relaxation of the conditions for the reset functions already yields an undecidability result for the reachability problem.

2.2 Verification Tasks

When analyzing (hybrid) automata, the verification tasks can mostly be characterized by the consideration of the following four classes of properties: safety properties, liveness properties, persistence properties and fairness properties.

- Safety properties state that throughout any feasible run of the system some safety constraint will always hold. Safety properties can be reduced to the reachability problem, i.e. if unsafe states can be reached from the initial states.
- Liveness properties state that some properties can be eventually reached, no matter what time is needed for to reach them.
- Persistance properties state that for any computation there will be a point of time after which the persistance property will always hold.
- Fairness properties state that some properties will hold infinitely often.

In order to deal with the verification tasks related to safety, liveness, persistence and fairness, several languages have been developed. One very expressive language is the propositional μ-calculus that covers many other languages. A more readable and intuitive, but less expressive language is the language CTL. Both languages will be explained in the sections below, assuring discrete and dense time frameworks.

2.2.1 Lattice Theory

Lattice theory builds the theoretical background of the propositional μ-calculus. This section gives a short overview of the definitions of lattices, monotonic and continuous functions on lattices, and finally introduces the fundamental fixpoint-theorem of Tarski-Knaster.

In order to give a formal definition of the notion of lattices, we first need to introduce partial orders. A *partial order* on a set D is a relation $\sqsubseteq$ on $D \times D$, which satisfies for $x, y, z \in D$

- reflexivity: $x \sqsubseteq x$
- antisymmetry: $x \sqsubseteq y$ and $y \sqsubseteq x \Rightarrow x = y$
- transitivity: $x \sqsubseteq y$ and $y \sqsubseteq z \Rightarrow x \sqsubseteq z$

If any two elements of D can be compared, i.e. for all $x, y \in D$ either $x \sqsubseteq y$ or $y \sqsubseteq x$ holds, the order is called a *total ordering*. For a subset $M \subseteq D$ the element $x \in D$ is called an *upper bound* of M, iff for all $m \in M$, $m \sqsubseteq x$ holds. The element x is the *least upper bound*, iff all other upper bounds x' of M fulfill $x \sqsubseteq x'$, and we then write $x = sup(M)$ (supremum of M). Lower bounds and greatest lower bound ($inf(M)$) or infimum of M) are defined analogously. If $sup(M) \in M$ or $inf(M) \in M$, they are called maximal element respectively minimal element of the set M.

We are now ready to formally define the notion of *complete lattices*.

Definition 2.9 (Complete Lattice)

Let D be a partial ordered set. D is called a complete lattice*, iff all $M \subseteq D$ have $sup(M), inf(M) \in D$. $inf(D)$ and $sup(D)$ are called the* bottom *and the* top *of D, and are denoted by $\bot$ and $\top$.*

An example of a complete lattice is $(\mathbb{N} \cup \infty, \leq)$. Moreover, any finite set D endowed of a total ordering $\sqsubseteq$ forms a complete lattice. Other examples are the set of functions $f : D \to E$ with an ordering defined by pointwise comparison of the function values, i.e. $f \sqsubseteq g :\Leftrightarrow \forall d \in D : f(d) \sqsubseteq g(d)$. This construction of a lattice will be useful in the next section.

Definition 2.10 (Monotonicity and Continuity)

Let $(D, \sqsubseteq_D)$ and $(E, \sqsubseteq_E)$ be complete lattices and let $f : D \to E$.

- *f is called* monotonic *iff $x \sqsubseteq_D y \Rightarrow f(x) \sqsubseteq_E f(y)$*
- *f is called* continuous *iff for every directed set $M \neq \emptyset$ it holds $f(sup(M)) = sup(f(M))$ and $f(inf(M)) = inf(f(M))$*

The relation between monotonic and continuous functions is characterized by the following lemma.

Lemma 2.1

Let $(D, \sqsubseteq_D)$ and $E, \sqsubseteq_E$ be complete lattices and let $f : D \to E$. Then:

- *If f is continuous, it is also monotonic. The converse is not true.*
- *If D is finite and if f is monotonic, then f is also continuous.*

Proof

see [Sch04] q.e.d.

An element $x \in D$ is called fixpoint of the function $f : D \to D$, iff $f(x) = x$, i.e. the application of the function f on x does not have an effect. If fixpoints exist, the smallest fixpoint is denoted by $\mu x.f$, and the greatest fixpoint by $\nu x.f$. It is obvious, that not every function has fixpoints. However, as shown in the fixpoint theorem of Tarski-Knaster, *any monotonic function over complete lattices has fixpoints.* Moreover, the theorem states an iteration procedure which converges to greatest and smallest fixpoint. However, only when the lattice is finite, it can be guaranteed that the fixpoint iterations for greatest and smallest fixpoints stop after a finite number of iteration steps.

Theorem 2.2 (Tarski-Knaster Fixpoint Theorem)

Let D be a complete lattice. Then every monotonic function $f : D \to D$ has fixpoints. The smallest fixpoint of f can be obtained by $\mu x.f(x) := sup\{f^{n+1}(\bot) | n \in \mathbb{N}\}$. The greatest fixpoint of f can be obtained by $\nu x.f(x) := inf\{f^{n+1}(\top) \,|\, n \in \mathbb{N}\}$.

Proof

see [Sch04, Tar55] q.e.d.

2.2.2 CTL and μ-Calculus on Discrete and Continuous Time Systems

When dealing with temporal logics for the specification, it is very important to distinguish between the analysis of discrete-time systems, e.g. processors, and the verification of systems endowed of some continuous dynamics, e.g. a hybrid system. In the discrete time framework there always exists a clearly defined next point of time. Thus it makes sense to define temporal operators as the so-called next-operator X with the interpretation, that $X\phi$ is true if and only if at the next point of time (i.e. in the next state of the transition system) ϕ holds.

In the continuous-time framework, however, this is not the case. In fact, during the continuous changes of the system, there exists no specified next point of time, since $\mathbb{R}^+$ is a dense set. For these reasons, syntax and semantics of CTL temporal logic and μ-calculus vary slightly in the cases of discrete-time and continuous-time framework, as outlined in the next subsections.

2.2.2.1 CTL on Discrete Time Systems

The main components of the temporal logic CTL for discrete time systems are besides of the usual propositional operators $\neg, \vee, \wedge$ the operators X and $\underline{\mathrm{U}}$ and the path-quantifiers E and A. $X\phi$ defines the property that ϕ is true at the next point of time. $\phi \underline{\mathrm{U}} \psi$ describes the property, that until the first point of time when ψ holds, ϕ also holds. While these properties are checked for specified paths, formulas of the form $E\phi$ denote all states, where a path starts, which satisfies ϕ and $A\phi$ denotes exactly the states $q \in Q$, where all paths starting in q satisfy the condition ϕ.

Definition 2.11 (CTL)

Let AP be a finite set of propositional letters and let $p \in AP$. Then the language CTL is defined by

$$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid EX\phi \mid AX\phi \mid E(\phi_1 \underline{U} \phi_2) \mid A(\phi_1 \underline{U} \phi_2)$$

The semantics of CTL is recursively defined by the following rules on labeled transition systems.

Definition 2.12 (Semantics of CTL)

Let AP be a set of propositional letters, let $\phi, \psi \in CTL$, let $T = \langle Q, Q_0, l_{\rightarrow} \rightarrow \rangle$ be a labeled transition system, let s be a state of T and let $l_{AP} : Q \rightarrow 2^{AP}$. Then the semantics $T \vDash \phi$ of a formula $\phi \in CTL$ is recursively defined by:

- $\phi \in AP$: $s \vDash \phi$ *iff* $\phi \in l_{AP}(s)$
- $s \vDash \neg\phi$ *iff* $s \nvDash \phi$
  $s \vDash \phi \vee \psi$ *iff* $s \vDash \phi$ *or* $s \vDash \psi$
  $s \vDash \phi \wedge \psi$ *iff* $s \vDash \phi$ *and* $s \vDash \psi$
- $s \vDash EX\phi$ *iff there exists a transition* $s \rightarrow s'$ *in* T *with* $s' \vDash \phi$
  $s \vDash AX\phi$ *iff for all transitions* $s \rightarrow s'$ *in* T *it holds* $s' \vDash \phi$
- $s \vDash E(\phi \underline{U} \psi)$ *iff there exists a run* ρ *starting in* $s = \rho(0)$ *and* $n \in \mathbb{N}$ *with corresponding points of time* t_n *satisfying*

  $$\rho(t_n) \vDash \psi \quad \text{and} \quad \forall 0 \leq k < n : \rho(t_k) \vDash \phi$$

  $s \vDash A(\phi \underline{U} \psi)$ *iff for all runs* ρ *starting in* $s = \rho(0)$ *there exists an* $n \in \mathbb{N}$ *with corresponding points of time* t_n *satisfying*

  $$\rho(t_n) \vDash \psi \quad \text{and} \quad \forall 0 \leq k < n : \rho(t_k) \vDash \phi$$

T is a model for ϕ, iff all initial states Q_0 model ϕ.
Short: $T \vDash \phi$ *iff* $\forall s \in Q_0 : s \vDash \phi$

2.2.2.2 μ-Calculus on Discrete Time Systems

The propositional μ-calculus was first developed in 1983 [Koz83], after it had been shown, that important properties like termination and totality of programs cannot be expressed in first order logics used so far. Because of this reason, fixpoint operators were introduced. The main components of the μ-calculus are - besides of the propositional operators $\neg, \vee$ and $\wedge$ - the modal operators $\Box$ and $\Diamond$, which directly correspond to AX and EX in the temporal logic CTL (except for finite paths). Given a discrete time system the formula $\Box\phi$ holds true for a state, iff all possible successors of the state fulfill ϕ. Analogously, $\Diamond\phi$ holds true, if at least one successor fulfills the property ϕ. The fixpoint operators denote smallest and greatest fixpoint of states satisfying ϕ. Formally, the language L_μ is defined as follows:

Definition 2.13 (The Language L_μ)

Let AP be a finite set of propositional letters,let $p \in AP$ and let $T = \langle Q, Q_0, l_\rightarrow, \rightarrow\rangle$ be a labeled transition system. Then the set of μ-calculus pre-formulas is defined by the following syntax:

$$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \Diamond\phi \mid \Box\phi \mid \mu Z.\phi \mid \nu Z.\phi$$

The set L_μ of μ-calculus formulas is defined as the subset of pre-formulas, where each subformula of the type $\mu Z.\phi$ and $\nu Z.\phi$ satisfies that all occurrences of Z in ϕ occur under an even number of negation symbols.

In μ-calculus, the semantics assigns for a discrete time system $T = \langle Q, Q_0, l_\rightarrow, \rightarrow\rangle$ to each formula $\phi \in L_\mu$ exactly the set $[\![\phi]\!] \subseteq Q$ of states satisfying the formula. This assignment is done recursively on the structure of the μ-calculus formulas:

Definition 2.14 (Semantics of formulas in L_μ)

Let AP be a set of propositional letters, let $T = \langle Q, Q_0, l_\rightarrow, \rightarrow\rangle$ be a labeled transition system and let $l_{AP} : Q \rightarrow 2^{AP}$. Then the semantics $[\![\phi]\!] : Q \rightarrow \{0, 1\}$ of a formula $\phi \in L_\mu$ is recursively defined by:

- $\phi \in AP$: $[\![\phi]\!](q) = 1$ *iff* $\phi \in l_{AP}(q)$
- $[\![\neg\phi]\!] := \neg[\![\phi]\!]$
  $[\![\phi \vee \psi]\!] := [\![\phi]\!] \vee [\![\psi]\!]$
  $[\![\phi \wedge \psi]\!] := [\![\phi]\!] \wedge [\![\psi]\!]$
- $[\![\Diamond\phi]\!](q) = 1$ *iff* $\exists a \in l_\rightarrow \exists q' \in Q : q \xrightarrow{a} q' \wedge [\![\phi]\!](q') = 1$
  $[\![\Box\phi]\!](q) = 1$ *iff* $\forall a \in l_\rightarrow \forall q' \in Q : q \xrightarrow{a} q' \Rightarrow [\![\phi]\!](q') = 1$
- *The fixpoint operators are defined in the following way:*
  Let $[\phi]_Z^\psi$ be the formula obtained by replacing all occurrences of Z with ψ. Given a fixpoint formula $\sigma Z.\phi$ with $\sigma \in \{\mu, \nu\}$ its k-th approximation $apx_k(\sigma Z.\phi)$ is recursively defined as follows:

$$apx_0(\mu Z.\phi) := 0 \quad \textit{and} \quad apx_{k+1}(\mu Z.\phi) := [\phi]_Z^{apx_k(\mu Z.\phi)}$$
$$apx_0(\nu Z.\phi) := 1 \quad \textit{and} \quad apx_{k+1}(\nu Z.\phi) := [\phi]_Z^{apx_k(\nu Z.\phi)}$$

Then smallest and greatest fixpoints $[\![\sigma Z.\phi]\!]$ are defined by

- *smallest fixpoint:* $[\![\mu Z.\phi]\!] := \bigvee_{k\in\mathbb{N}} [\![apx_k(\mu Z.\phi)]\!]$
- *greatest fixpoint:* $[\![\nu Z.\phi]\!] := \bigwedge_{k\in\mathbb{N}} [\![apx_k(\nu Z.\phi)]\!]$

T is a model for ϕ, iff for all initial states $q_0 \in Q_0$ it holds $[\![\phi]\!](q_0) = 1$.
Short: $T \vDash \phi \quad :\Leftrightarrow [\![\phi]\!](Q_0) = 1$

The fixpoint operations defined in Definition 2.14 are well-defined since the fixpoint theorem of Tarski-Knaster (Theorem 2.2) can be applied. To this end, define the total order $0 \sqsubseteq 1$ on $\{0,1\}$ for functions $f, g : Q \to \{0,1\}$ the partial order $f \sqsubseteq g$ by pointwise comparison: $f \sqsubseteq g$ iff for all $q \in Q$ it holds $f(q) \sqsubseteq g(q)$. Then the functions $f : Q \to \{0,1\}$ together with the partial order $\sqsubseteq$ form a complete lattice with $f_{min} := 0$ (denoting the emptyset) and $f_{max} := 1$ (denoting the complete state space Q) as minimal and maximal element. By the fixpoint theorem of Tarski-Knaster every continuous function $\gamma : (Q \to \{0,1\}) \to (Q \to \{0,1\})$ has fixpoints, and the least and greatest fixpoint can be achieved by the iteration starting in f_{min} respectively f_{max}. However, the negation is not monotonic and thus not a continuous function. This is the reason why in L_μ all occurrences of a bound variable in a fixpoint formula have to be positive, i.e. have to occur after an even number of negations. This condition suffices to guarantee, that the functions are continuous. See therefore also [Sch04]. If the state space of T is finite, then the fixpoint operations stop after a finite number of iteration steps and $\sigma Z.\phi$ with $\sigma \in \{\mu, \nu\}$ can also be characterized by $[\![\sigma Z.\phi]\!] := [\![apx_{\hat{k}}(\mu Z.\phi)]\!]$ for some suitable $\hat{k} \in \mathbb{N}$.

In Definition 2.14 one can easily see, that some of the operators are redundant:

- $\phi \vee \psi = \neg(\phi \wedge \psi)$
- $\Box\phi = \neg(\Diamond\neg\phi)$

However, this syntactic sugar makes the formulas more readable.

In discrete time systems the propositional μ-calculus is stronger than CTL, since every CTL-formula can be translated into an equivalent μ-calculus formula. The equivalence of the propositional operators $\{\neg, \vee, \wedge\}$ and the next-operators EX and AX to $\Diamond$ and $\Box$ is obvious and for infinite paths the translation of the $\underline{\mathrm{U}}$ operators is given by:

- $T \vDash E(\phi \underline{\mathrm{U}} \psi)$ iff $T \vDash \mu Z.\psi \vee \phi \wedge \Diamond Z$
- $T \vDash A(\phi \underline{\mathrm{U}} \psi)$ iff $T \vDash \mu Z.\psi \vee \phi \wedge \Box Z$

Furthermore, it can be shown that the language L_μ is strictly more expressive than CTL. With CTL for example it is not possible to describe the properties of the form that some conditions hold alternatingly true, when the alternation depth is not defined beforehand. A more detailed example will be given in section 2.2.2.3 below.

2.2.2.3 CTL and μ-Calculus on Hybrid Automata

The definitions for $CTL$and L_μ for discrete time systems described in the previous section are not completely compatible with the description of properties on systems endowed also of continuous-time dynamics as hybrid automata. While the propositional operators $\{\neg, \vee, \wedge\}$ and the fixpoint operators $\mu Z.\phi$ and $\nu Z.\phi$ can be translated in a one to one correspondence, this is not the case for the modal operators $\Box, \Diamond$ of L_μ and the temporal operators $\{X, \underline{\mathrm{U}}\}$ of CTL . While the semantics for the temporal operator $\underline{\mathrm{U}}$ can be redefined without too many problems in the presence of continuous dynamics, this is not the case with the next-operators. In fact, in the continuous-time framework there does not exist a clearly defined next point of time and the natural interpretation of the next-operators is not possible any more. In [GSM07] it has been proposed to simply omit the X-operator in the syntax of $CTL$for hybrid automata. Other sources propose for L_μ to redefine $\Box$ and $\Diamond$, such that it does not mean 'at the next point of time' any more but 'after one transition', not regarding if this is a continuous transition of arbitrary length or a discrete one. Furthermore, in order to distinguish between continuous and discrete transitions these operators are divided (in a modal splitting) into their subparts $[a]\phi$ and $\langle a\rangle\phi$ with $a \in \{e, \delta\}$, where a state q fulfills $[a]\phi$ if and only if all $\xrightarrow{a}$ successors fulfill ϕ and it fulfills $\langle a\rangle\phi$ iff at least one $\xrightarrow{a}$successor fulfills ϕ.

However, although this interpretation of the modal operators is a quite natural one, it has the main disadvantage, that the until-operator (interpreted on a continuous-time framework) cannot be encoded by fixpoint operators and modal operators (see example 2.3).

One way to overcome this problem is simply to add the $\underline{\mathrm{U}}$-operator to the basic operators of L_μ for hybrid automata. This naturally guarantees that L_μ is more expressive than CTL also in presence of continuous-time framework (see example 2.4).

Definition 2.15 (The Language L_μ for Hybrid Automata)

The set of μ-calculus preformulas for a set of labels $a \in \{e, \delta\}$ and propositions $p \in AP$ is defined by the following syntax:

$$\phi := p \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \langle a\rangle\phi \mid [a]\phi \mid E(\phi_1 \underline{U} \phi_2) \mid A(\phi_1 \underline{U} \phi_2) \mid \mu Z.\phi \mid \nu Z.\phi$$

The set L_μ of μ-calculus formulas is defined as the subset of pre-formulas, where each subformula of the type $\mu Z.\phi$ and $\nu Z.\phi$ satisfies that all occurrences of Z in ϕ occur under an even number of negation symbols.
Consider a hybrid automaton H and its timed and time abstract transition systems T_H^t and T_H. Let $l_{AP} : Q \to 2^{AP}$, where AP is a set of propositional letters, and let $p \in AP$.
The semantics of μ-calculus formulas $[\![\phi]\!] : Q \to \{0, 1\}$ on H is given by the following rules:

- $\phi \in AP$: $[\![\phi]\!](q) = 1$ *iff* $\phi \in l_{AP}(q)$
- $[\![\neg\phi]\!] := \neg\, [\![\phi]\!]$ *and* $[\![\phi \vee \psi]\!] := [\![\phi]\!] \vee [\![\psi]\!]$ *and* $[\![\phi \wedge \psi]\!] := [\![\phi]\!] \wedge [\![\psi]\!]$

- $[\![E(\phi \underline{U} \psi)]\!](q) = 1$ *iff there exists a run* $q = q_1 \xrightarrow{a_1} ... \xrightarrow{a_n} q_n$ *in* T_H^t *with* $a_i \in \{e\} \cup \mathbb{R}^+$ *satisfying the following properties:*

  - $[\![\psi]\!](q_n) = 1$
  - *for* $1 \leq i < n$: $[\![\phi]\!](q_i) = 1$
  - *for* $a_i \in \mathbb{R}^+$: *all states* q' *being traversed according to the continuous flow* $q_{i-1} \rightsquigarrow q_i$ *given by* $f_i : [0, a_i] \to \mathbb{R}^n$ *(Definition 2.2) satisfy* $[\![\phi]\!](q') = 1$

  $[\![A(\phi \underline{U} \psi)]\!](q) = 1$ *iff for all runs starting in* q *there exists a representation* $q = q_1 \xrightarrow{a_1} ... \xrightarrow{a_n} q_n \xrightarrow{a_{n+1}} ...$ *in* T_H^t *with* $a_i \in \{e\} \cup \mathbb{R}^+$ *which satisfies:*

  - $[\![\psi]\!](q_n) = 1$
  - *for* $1 \leq i < n$: $[\![\phi]\!](q_i) = 1$
  - *for* $a_i \in \mathbb{R}^+$: *all states* q' *being traversed according to the continuous flow* $q_{i-1} \rightsquigarrow q_i$ *given by* $f_i : [0, a_i] \to \mathbb{R}^n$ *(Definition 2.2) satisfy* $[\![\phi]\!](q') = 1$

- $[\![\langle a \rangle \phi]\!](q) = 1$ *iff* $\exists q \xrightarrow{a} q' : [\![\phi]\!](q') = 1$
  $[\![[a]\phi]\!](q) = 1$ *iff* $\forall q \xrightarrow{a} q' : [\![\phi]\!](q') = 1$

- *The fixpoint operators are defined in the following way:*
  Let $[\phi]_Z^\psi$ *be the formula obtained by replacing all occurrences of* Z *with* ψ.
  Given a fixpoint formula $\sigma Z.\phi$ *with* $\sigma \in \{\mu, \nu\}$ *its k-th approximation* $apx_k(\sigma Z.\phi)$ *is recursively defined as follows:*

$$apx_0(\mu Z.\phi) := 0 \quad \text{and} \quad apx_{k+1}(\mu Z.\phi) := [\phi]_Z^{apx_k(\mu Z.\phi)}$$
$$apx_0(\nu Z.\phi) := 1 \quad \text{and} \quad apx_{k+1}(\nu Z.\phi) := [\phi]_Z^{apx_k(\nu Z.\phi)}$$

  Then smallest and greatest fixpoints $[\![\sigma Z.\phi]\!]$ *are defined by*

  - *smallest fixpoint:* $[\![\mu Z.\phi]\!] := \bigvee_{k \in \mathbb{N}} [\![apx_k(\mu Z.\phi)]\!]$
  - *greatest fixpoint:* $[\![\nu Z.\phi]\!] := \bigwedge_{k \in \mathbb{N}} [\![apx_k(\mu Z.\phi)]\!]$

H is a model for ϕ, *iff for all initial states* $[\![\phi]\!](q_0) = 1$.
Short: $H \vDash \phi :\Leftrightarrow [\![\phi]\!](Q_0) = 1$

Example 2.3 illustrates, that unlike in the discrete-time framework the until-operators $A(\phi \underline{\mathrm{U}} \psi)$ and $E(\phi \underline{\mathrm{U}} \psi)$ cannot be encoded by fixpoints and modal operators when dealing with hybrid automata and in Lemma 2.3 it will be shown, that in the context of hybrid automata it is not possible to encode the temporal operator $\underline{\mathrm{U}}$ by fixpoints and modal operators.

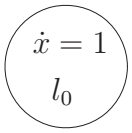


Figure 2.6: Counterexample for $H \models \mu Z.\psi \vee \phi \wedge \Diamond Z \not\Rightarrow H \models E(\phi \underline{\mathrm{U}} \psi)$

Example 2.3

Take the simple hybrid automaton of Figure 2.6 consisting only of one location where the variable x increases linearly in time. Let the initial state be 0.
Let $\phi := (x < 2)$ and $\psi := x = 3$. Then

- *$H \not\models E(\phi \underline{U} \psi)$: obvious*
- *$H \vDash \mu Z.\psi \vee \phi \wedge (\langle e \rangle Z \vee \langle \delta \rangle Z)$:*
  $[\![apx_0(\mu Z.\psi \vee \phi \wedge (\langle e \rangle Z \vee \langle \delta \rangle Z))]\!](r) = 1$ for $r \in \{(l_0, 3)\}$
  Thus $[\![apx_1(\mu Z.\psi \vee \phi \wedge (\langle e \rangle Z \vee \langle \delta \rangle Z))]\!](r) = 1$ for $r \in \{l_0\} \times ([0, 2) \cup \{3\})$ since in the time abstract transition system T_H for all $0 \leq x < 2$ there exists a transition $x \xrightarrow{\delta} 3$.

This shows, that the CTL -formula $E(\phi \underline{U} \psi)$ cannot be translated in a μ-calculus formula like in the discrete-time framework.

Lemma 2.3

In the setting of hybrid automata the language L_μ with the temporal operators $E(\phi \underline{U} \psi)$ and $A(\phi \underline{U} \psi)$ is strictly more expressive than L_μ without these operators.

Proof

To prove the claim it suffices to give one example, where it is not possible for any μ-calculus formula without the temporal operator $\underline{\mathrm{U}}$ to model the formula $E(\phi \underline{\mathrm{U}} \psi)$ or $A(\phi \underline{\mathrm{U}} \psi)$. Consider therefore again the hybrid automaton of the previous example given in Figure 2.6. Define the two propositional letters ϕ and ψ by

- $\phi := x \in [2, 3)$
- $\psi := x \in \mathbb{N}$

Then, the formulas $E(\phi \underline{\mathrm{U}} \psi)$ and $A(\phi \underline{\mathrm{U}} \psi)$ both are true for the states $\{(l, x) \mid x \in [2, 3] \cup \mathbb{N}\}$.
It is possible to transform any μ-calculus formula without temporal operators to an equivalent formula without greatest fixpoints and $\wedge$, where the modal operators $[\delta]$ and $\langle \delta \rangle$ only occur directly before ϕ and ψ. Translating propositional operators and fixpoints to set-operations yield:

$$\neg \cong \mathbb{R}^+\setminus \quad \text{and} \quad \vee \cong \cup \quad \text{and} \quad \mu \cong \bigcup_{n\in\mathbb{N}}$$

Together with the fact that $\emptyset$ and $\mathbb{R}^+$ represent the propositional letters 0 and 1, this yields that the subsets of $\mathbb{R}^+$, which can be encoded via μ-calculus formulas over the atomic propositions ϕ and ψ, are sets of the σ-algebra[1] generated by $\{\phi,\ \psi,\ a_n...a_1\phi,\ a_n...a_1\psi \mid n \in \mathbb{N},\ a_i \in \{\langle\delta\rangle, [\delta]\}\}$. Using the fact that in this example it holds

1. $\mathbb{R}^+ = \langle\delta\rangle\psi = \langle\delta\rangle\langle\delta\rangle\psi = [\delta]\langle\delta\rangle\psi$
2. $\emptyset = [\delta]\psi = \langle\delta\rangle[\delta]\psi = [\delta][\delta]\psi$
3. $[0,3) = \langle\delta\rangle\phi = \langle\delta\rangle\langle\delta\rangle\phi$
4. $\emptyset = [\delta]\phi = \langle\delta\rangle[\delta]\phi = [\delta][\delta]\phi = [\delta]\langle\delta\rangle\phi$

this yields that the subsets of $\mathbb{R}^+$, which can be encoded by μ-calculus formulas without the temporal operator $\underline{\mathrm{U}}$, are in the σ-algebra

$$\sigma(\{\phi,\ \psi,\ a_n...a_1\phi,\ a_n...a_1\psi \mid n \in \mathbb{N},\ a_i \in \{\langle\delta\rangle, [\delta]\}\}) =$$
$$\sigma(\{\phi, \psi, \langle\delta\rangle\phi, [\delta]\phi, \langle\delta\rangle\psi, [\delta]\psi\}) = \sigma(\{\emptyset, [0,3), \mathbb{R}^+\}) = \{\emptyset, [0,3), [3,\infty), \mathbb{R}^+\}$$

Since $[2,3] \cup \mathbb{N} \notin \{\emptyset, [0,3), [3,\infty), \mathbb{R}^+\}$ for this example it is not possible to encode the formulas $E(\phi\underline{\mathrm{U}}\psi)$ and $A(\phi\underline{\mathrm{U}}\psi)$ by any μ-calculus formula without the temporal operator $\underline{\mathrm{U}}$.
This yields the claim. q.e.d.

The next example shows, that L_μ defined for hybrid automata as given in Definition 2.15 really is more expressive than the corresponding definition of CTL for hybrid automata.

Example 2.4

Consider the following formula describing the set of states having paths satisfying alternatingly with arbitrary alternation depth that either ϕ_1 is true or $\phi_2\underline{U}\phi_3$ holds:

$$\varphi := \mu Z.\phi_3 \vee (E(\phi_1\underline{U}(E(\phi_2\underline{U}Z))))$$

[1] σ-algebra:
Let Ω be a non-empty set. Then $\mathfrak{A} \subseteq 2^\Omega$ is called a σ-algebra iff

- $\Omega, \emptyset \in \mathfrak{A}$,
- $A \in \mathfrak{A} \Rightarrow \Omega \setminus A \in \mathfrak{A}$, and
- $A_1, A_2, ... \in \mathfrak{A} \Rightarrow \bigcup_{n\in\mathbb{N}} A_n \in \mathfrak{A}$

Let $\mathfrak{C} \subseteq 2^\Omega$. Then, the smallest σ-algebra containing $\mathfrak{C}$ is called the σ-algebra generated by $\mathfrak{C}$ and is denoted by $\sigma(\mathfrak{C})$.

Because of the built-in alternation of arbitrary length, φ cannot be described by pure CTL formulas since by unwinding the formula, it is only possible to produce formulas for finite fixed alternation depths. The exact translation would yield the infinite union

$$\bigvee_{k\in\mathbb{N}} apx_k(\mu Z.\phi_3 \vee (E(\phi_1 \underline{U}(E(\phi_2 \underline{U} Z)))))$$

which is not allowed in CTL and cannot be transformed in a finite formula since for all $k \in \mathbb{N}$ there exist examples, where $H \vDash apx_k(\mu Z.\phi_3 \vee (E\phi_1 \underline{U}(E(\phi_2 \underline{U} Z))))$ but not $H \vDash apx_{k-1}(\mu Z.\phi_3 \vee (E\phi_1 \underline{U}(E(\phi_2 \underline{U} Z))))$. Thus, L_μ is more expressive than CTL .

2.3 Three-valued Logic

Three-valued logic has been applied in many areas of computer science like [SS07, RSW04]. Usually, three-valued logics are applied, if the underlying problems cannot be solved by using the normal boolean logic, e.g. if the problems are proven to be undecidable or too complex.

While the original two-valued boolean logic is based on the truth values 1 and 0 the three-valued logic provides an additional truth value, denoted by $\perp$. The truth-value $\perp$ allows one to express that there exists not enough information to assign one of the original truth values 1 and 0.

Another meaning of the truth value $\perp$ occurs when dealing with abstractions of hybrid automata. Here, usually infinitely many states of the underlying time abstract transition system are considered are combined to one abstract state. So, in this setting it is possible, that in one abstract state there exist both points satisfying some given condition ϕ and not falsifying the same condition. In this case, even if one has complete knowledge of the behavior of this abstract state, it is not possible to assign either the truth value 1 or 0 to this abstract state, so $\perp$ will be assigned.

$$\phi(s) = \begin{cases} 1 & \text{s fulfills } \phi \\ 0 & \text{s does not fulfill } \phi \\ \perp & \text{it can neither be proven nor disproven that s fulfills } \phi \end{cases}$$

In analogy to the two-valued boolean logic, negation, disjunction and conjunction can be performed in the three-valued setting as shown in the following tables. In the rest of the thesis, the three-valued operations will be denoted by $\neg_3$, $\vee_3$ and $\wedge_3$.

ϕ	$\neg_3\phi$
0	1
$\perp$	$\perp$
1	0

$\wedge_3$	0	$\perp$	1
0	0	0	0
$\perp$	0	$\perp$	$\perp$
1	0	$\perp$	1

$\vee_3$	0	$\perp$	1
0	0	$\perp$	1
$\perp$	$\perp$	$\perp$	1
1	1	1	1

2.4 Basics of Computer Algebra

In this section, we introduce some basic notions of *computer algebra* like the concept of Gröbner bases. Exhaustive introduction to these topics can be found in textbooks like [PG02]. The standard notions required in this section will be extensively used in Chapter 3 of the thesis, exploring connections between computer algebra and automated reasoning on hybrid automata.

Definition 2.16 (Polynomial Ring over $\mathbb{R}$)

The polynomial ring over $\mathbb{R}$ is defined as

$$\mathbb{R}[\underline{x}] := \mathbb{R}[x_1, ..., x_n] = \{ \sum_{\alpha \in A} c_\alpha \cdot x_1^{\alpha_1} \cdot ... \cdot x_n^{\alpha_n} \mid c_\alpha \in \mathbb{R},\ A \subset \mathbb{N}^n \text{ finite} \}$$

where:

- $\underline{x}^\alpha := x_1^{\alpha_1} \cdot ... \cdot x_n^{\alpha_n}$ *is called a monomial in x_1, ..., x_n with exponent vector $\alpha \in \mathbb{N}^n$,*
- $|\alpha| := \alpha_1 + ... + \alpha_n$ *is the degree of the monomial $\underline{x}^\alpha$,*
- $Mon(\underline{x}) := \{\underline{x}^\alpha | \alpha \in \mathbb{N}^n\}$,
- *given a polynomial $f = \sum_{\underline{\alpha} \in A,\, c_\alpha \neq 0} c_{\underline{\alpha}} \cdot x_1^{\alpha_1} \cdot ... \cdot x_n^{\alpha_n}$, then $c_\alpha \underline{x}^\alpha$ is called a term of f and c_α the coefficient of $\underline{x}^\alpha$. The degree of f is defined by $deg(f) := max\{|\alpha| \mid \alpha \in A, c_\alpha \neq 0\}$.*

Definition 2.17 (Ideal)

Let $\mathbb{R}[\underline{x}]$ be the polynomial ring over $\mathbb{R}$ and $I \subseteq \mathbb{R}[\underline{x}]$. Then I is called an ideal iff

- $\forall a, b \in I : a + b \in I$, *and*
- $\forall r \in \mathbb{R}[\underline{x}], a \in I : r \cdot a \in I$ *holds.*

Notation: $I \trianglelefteq \mathbb{R}[\underline{x}]$

$\langle p_1, ..., p_k \rangle = \{ \sum_{i=1}^{k} q_i \cdot p_i \mid q_i \in \mathbb{R}[\underline{x}] \}$ *is the ideal generated by $p_1, ..., p_k$, i.e. $\langle p_1, ..., p_k \rangle$ is the smallest ideal containing $\{p_1, ..., p_k\}$.*

Theorem 2.4 (Hilbert's Basis Theorem)

All ideals $I \trianglelefteq \mathbb{R}[\underline{x}]$ are finitely generated, i.e. for all ideals I there exist generators $p_1, ..., p_k$ in $\mathbb{R}[\underline{x}]$ s.t. $I = \langle p_1, ..., p_k \rangle$.

A fundamental problem in computer algebra is given by the so-called *ideal membership problem*, asking, whether a given polynomial lies within an ideal $I \trianglelefteq \mathbb{R}[\underline{x}]$. The idea to solve this problem is to define a kind of normal form of polynomials w.r.t. the given ideal I. Ideally, this normal form should be unique and tell, whether a polynomial lies within I or not. Such a normal form can be achieved by Gröbner bases techniques. In order to formally introduce the concept of normal forms, first monomial orderings need to be defined:

Definition 2.18 (Monomial Ordering)

A monomial ordering on $\mathbb{R}[\underline{x}]$ *is a relation* $>$ *on* $Mon(\underline{x})$ *which satisfies the following*

- $>$ *is a total ordering on* $Mon(\underline{x})$.
- $>$ *is multiplicative , i.e.* $\underline{x}^\alpha > \underline{x}^\beta$ *implies* $\underline{x}^\alpha \cdot \underline{x}^\gamma > \underline{x}^\beta \cdot \underline{x}^\gamma$.
- $>$ *is a well-ordering, i.e.* $\forall \alpha \in \mathbb{N}^n : \underline{x}^\alpha > 1$.

Let $>$ *be a monomial ordering and* $f = \sum_{\alpha \in A} c_\alpha \cdot \underline{x}^\alpha \in \mathbb{R}[\underline{x}]$ *and let* $\underline{x}^{\alpha_0} = max\{\underline{x}^\alpha | \alpha \in A,\, c_\alpha \neq 0\}$. *Then, the leading monomial, the leading term and the leading coefficient of* f *are defined by*

- $lt(f) := c_{\alpha_0}\underline{x}^{\alpha_0}$ *is the leading term of* f.
- $lm(f) := \underline{x}^{\alpha_0}$ *is the leading monomial of* f.
- $lc(f) := c_{\alpha_0}$ *is the leading coefficient of* f.

Moreover, if $lt(f) = 1$ *then the polynomial* f *is called monic.*
Finally, $>$ *is an elimination ordering on* $Mon(x_1, ..., x_n, y_1, ..., y_n)$ *w.r.t.* $\underline{x}$ *iff* $lm(f) \in \mathbb{R}[\underline{y}] \Rightarrow f \in \mathbb{R}[\underline{y}]$.

Remark 2.2

The following are the most important examples for monomial orderings:

- *lexicographic order (w.r.t.* $x_1 > x_2 > ... > x_n$*)*
  $\underline{x}^\alpha >_{lp} \underline{x}^\beta :\Leftrightarrow \exists s\ \ \alpha_1 = \beta_1, ..., \alpha_{s-1} = \beta_{s-1}, \alpha_s > \beta_s$
- *graded lexicographic order*
  $\underline{x}^\alpha >_{Dp} \underline{x}^\beta :\Leftrightarrow$
  $|\alpha| > |\beta|$ *or* $(|\alpha| = |\beta|$ *and* $\exists s$ *such that* $\alpha_1 = \beta_1, ..., \alpha_{s-1} = \beta_{s-1}, \alpha_s > \beta_s)$
- *graded reverse lexicographic order*
  $\underline{x}^\alpha >_{dp} \underline{x}^\beta :\Leftrightarrow$
  $|\alpha| > |\beta|$ *or* $(|\alpha| = |\beta|$ *and* $\exists s$ *such that* $\alpha_n = \beta_n, ..., \alpha_{s+1} = \beta_{s+1}, \alpha_s > \beta_s)$

The lexicographic order with $x_1 > ... > x_n$ *is moreover an elimination ordering w.r.t.* $\{x_1, .., x_k \mid k < n\}$.

The next step is to define a normal form of a polynomial q w.r.t. to a given ideal $I \trianglelefteq \mathbb{R}[\underline{x}]$. Ideally this normal form should be zero iff $q \in I$. So we first of all need to define a reduction relation. A canonical way to defining it is via multivariate division with remainder.

Definition 2.19 (Normal Form & Reduction Relation)

Let $>$ be a monomial ordering on $\mathbb{R}[\underline{x}]$, let $q = \sum c_\alpha \cdot \underline{x}^\alpha$ be a polynomial in $\mathbb{R}[\underline{x}]$ and $I = \{p_1, ..., p_k\} \subset \mathbb{R}[\underline{x}]$. The polynomial q can be reduced by a polynomial $p \in I$ iff $lm(p)$ divides a term $c \cdot t$ of f. The reduction $f \xrightarrow{p} f'$ has the form: $f' = f - \frac{c \cdot t}{lt(p)} p$. When no reduction with polynomials in I can be done, f has a normal form*. $NF_I(f))$ denotes a normal form of f, i.e. f is fully reduced by I.*

Remark 2.3

The normal form of a polynomial w.r.t. to a given ideal I is not uniquely determined. Example:
Let $I = \langle x^2 + x, x^2 - x \rangle$. Then:

- $x^3 \xrightarrow{x^2+x} -x^2 \xrightarrow{x^2+x} -x$ *is in normal form.*
- $x^3 \xrightarrow{x^2+x} -x^2 \xrightarrow{x^2-x} +x$ *is in normal form.*

Changing now the generators yields: $I = \langle x^2 + x, x^2 - x \rangle = \langle 2x \rangle = \langle x \rangle$, so the normal form of x^3 w.r.t. $I = \langle x \rangle$ would obviously be zero.
Taking special generators of the ideal, the so-called Gröbner basis one can assure that the normal form will be unique.

Definition 2.20 (Leading Ideal)

Let $I \trianglelefteq \mathbb{R}[\underline{x}]$ and let $>$ be a monomial ordering. Then:
The leading ideal of I w.r.t. to $>$ is defined by

$$L(I) := L_>(I) := \langle lt(p) \mid p \in I \rangle$$

i.e. $L(I)$ is the ideal generated by the leading terms of I.

As can be seen in Remark 2.3, the normal forms of a polynomial q w.r.t. a given ideal $I \trianglelefteq \mathbb{R}[\underline{x}]$ are not unique. This problem can be dealt with by introducing Gröbner bases.

Definition 2.21 (Gröbner Basis)

A finite subset $G = \{p_1, ..., p_k\}$ of $I \trianglelefteq \mathbb{R}[\underline{x}]$ is called a Gröbner basis for I w.r.t. $>$ iff $L(I) = \langle lt(p_1), ..., lt(p_k) \rangle$.
Equivalently: for each $f \in I \backslash \{0\}$ there is an element $p \in G$ s.t. $lt(p) \mid lt(f)$
A Gröbner basis G is reduced iff

- $0 \notin G$,
- *for $f, g \in G$, $f \neq g$: $lm(f) \nmid lm(g)$,*
- *all elements in G are monic,and*

- *for all $f \in G$ the tail $f - lt(f)$ of f is reduced wrt G.*

Unlike arbitrary bases of ideals $I \trianglelefteq \mathbb{R}[\underline{x}]$, Gröbner bases ensure the property that the normal form of any polynomial $q \in \mathbb{R}[\underline{x}]$ w.r.t. I is uniquely determined. Thus, the ideal membership problem reduces to the question, whether the normal form of the given polynomial is equal to zero. These important facts are summarized in the following theorem:

Theorem 2.5

Let $I \trianglelefteq \mathbb{R}[\underline{x}]$ and let $>$ be a monomial ordering. Then:

- *There exists a reduced Gröbner Basis G of I.*
- *G is uniquely determined.*
- *$\langle G \rangle = I$.*
- *The normal form w.r.t. G is uniquely defined.*
- *$f \in I \Leftrightarrow NF_G(f) = 0$.*
- *Let f and g be polynomials. Then:*
  $NF_G(f + g) = NF_G(NF_G(f) + NF_G(g)) = NF_G(f) + NF_G(g)$.

Proof

see e.g. [PG02]

Remark 2.4

Gröbner bases depend on the chosen ordering. The ordering also influences the speed of computations, since computing Gröbner bases w.r.t. some orderings is faster than w.r.t. to other orderings. One way to compute the Gröbner basis of an ideal is Buchberger's algorithm, shown below.

Based on the results stated in Theorem 2.5, Buchberger developed an algorithm [Buc76], which determines the Gröbner basis of a given ideal. The Buchberger algorithm (Algorithm 1) is a generalization of the Gaussian elimination and the Euclidean algorithm. Today, often variants of the algorithm are used, which usually work faster than the original algorithm [Fau99].

Algorithm 1: Buchberger's Algorithm

input : $I = \langle p_1, ..., p_k \rangle \trianglelefteq \mathbb{R}[\underline{x}]$, a monomial ordering $>$
output: A Gröbner basis G of I w.r.t. $>$

1 **begin**
2 $G := \langle p_1, ..., p_k \rangle$
3 $P := \{(p_i, p_j) \mid 1 \leq i < j \leq k\}$
4 **while** $P \neq \emptyset$ **do**
5 Choose $(f, g) \in P$
6 $P := P \setminus (f, g)$
7 $\underline{x}^{\alpha} := \mathtt{lm}(f)$; $\underline{x}^{\beta} := \mathtt{lm}(g)$
8 $\underline{x}^{\gamma} := \mathtt{lcm}(\underline{x}^{\alpha}, \underline{x}^{\gamma})$ `/* lcm is the least common multiple */`
9 $h(f, g) := \mathtt{lc}(g) \cdot \underline{x}^{\gamma-\alpha} \cdot f - \mathtt{lc}(f) \cdot \underline{x}^{\gamma-\beta} \cdot g$
10 $h := \mathtt{NF}_G(h(f, g))$
11 **if** $h \neq 0$ **then**
12 $P := P \cup \{(h, g) \mid g \in G\}$
13 $G := G \cup \{h\}$;
14 **return** G
15 **end**

Chapter 3

Algebraic Methods for Automated Reasoning about Hybrid Automata

Recently, many authors proposed to use the machinery of computer algebra and in particular Gröbner bases techniques in order to analyze hybrid automata. In [RCT05, SSM04, San05, TK04] the authors propose different methods for generating polynomial invariants of hybrid automata and overapproximating the reachable set of states. In this setting, in particular the class of algebraic hybrid automata is considered. Formally, an *algebraic hybrid automaton* is a hybrid automaton, where all location invariants and guards are algebraic and the reset function and continuous dynamics are given in terms of polynomials $\dot{\underline{x}}, \underline{x}' \in \mathbb{R}[\underline{x}]$. Although algebraic hybrid automata allow only algebraic invariants and polynomial transition relations, algebraic hybrid automata are still undecidable w.r.t. the reachability problem. This result directly follows from the fact, that timed automata with skewed clocks, which have been shown to be undecidable [ACHH93] (cmp. Section 2.1.3.4) are a subclass of algebraic hybrid automata.

However, the special form of the invariants and transition relations of algebraic hybrid automata give the possibility to translate some questions to ideal membership problems, which can then be solved by Gröbner bases methods. Assume e.g. that every reachable state of a location l satisfies the location conditions $I(l) = \{p_i(\underline{x}) = 0,\ i = 1, .., k\}$. In other words, every reachable state of l lies in the zero-set of the p_i, $i = 1, ..k$, which is formally denoted as $V(p_1, ..., p_k) := \{\underline{x} \in \mathbb{R}^n :\ p_1(\underline{x}) = ... = p_k(\underline{x}) = 0\}$. The latter set can also be described as $V(I) := \{\underline{x} \in \mathbb{R}^n \,|\, \forall p \in I : p(\underline{x}) = 0\}$ for the ideal $I = \langle p_1, ..., p_k \rangle$, since all elements of an ideal vanish at a point $\underline{x} \in \mathbb{R}$ iff all the generators vanish at this point. Hence, $\{(l, \underline{x}) \mid \underline{x} \in V(q_1, ..., q_j) \wedge l \in L\} \subseteq \mathit{Inv}(l)$ can be reduced to the ideal membership problem, i.e. checking, whether the polynomials $q_1, ..., q_j$ lie in the ideal $\mathit{Inv}(l)$.

This chapter is structured into two main sections. Section 3.1 gives an overview on recent literature concerning the usage of Gröbner bases for the automated reasoning on hybrid automata. Section 3.2 improves a previous result in [SSM04, San05].

3.1 Literature Reviews

Based on [RCT05, SSM04, San05] in Section 3.1.1, and in Section 3.1.2 two different approaches using Gröbner basis techniques are presented.

3.1.1 Generating Polynomial Invariants for Algebraic Hybrid Systems with Linear Continuous Dynamics

In [RCT05] a method for automatically generating polynomial invariants of algebraic hybrid automata with *linear continuous dynamics* is proposed. Formally, for each location $l \in L$ the continuous dynamics of the hybrid automaton considered in [RCT05] is given by the differential rule $\dot{\underline{x}} = A_l \underline{x} + b_l$ with $A_l \in \mathbb{Q}^{n \times n}$ and $b_l \in \mathbb{Q}^n$.

Let $Reach_l$ denote the set of reachable states in the location l and let $\Phi_l(x^*, t)$ be the state being reachable from (l, x^*) performing a continuous transition having duration t, $t \in \mathbb{R}^+$. Then, the set $Reach_l$ of H can be described by the following equations

$$Reach_l = \bigcup_{t \in \mathbb{R}^+} \Phi_l(Start_l) \tag{3.1}$$

$$Start_l = Init_l \cup \bigcup_{e=(l',l) \in E} R(e, Reach_{l'}) \cap G(e) \tag{3.2}$$

In particular, $Start_l = Init_l \cup \bigcup_{e=(l',l) \in E} R(e, Reach_{l'}) \cap G(e))$ represents the set of states, where l can be entered initially or via a discrete transition whereas $\bigcup_{t \in \mathbb{R}^+} \Phi_l(Start_l, t)$ is the set of states which can be reached from an element of $Start_l$ by a continuous transition. The least fixpoint of the system of equations $\{Reach_l \mid l \in L\}$ w.r.t. to the inclusion ordering $\subseteq$ is the exact set of reachable states. However, any fixpoint yields an overapproximation of the reachable states.

The aim of the authors of [RCT05] is exactly that of computing some fixpoint (in general not the least one) of the above system of equations, exploiting the algebraic definition of relevant sets in algebraic hybrid automata, as well as the linear shape of the involved differential equations. In particular, in [RCT05], Tiwari and Rodríguez-Carbonell rely on the possibility of expressing the continuous flow (i.e. solutions of the involved linear differential equations) in each location by means of an algebraic equation, modulo the introduction of new variables and corresponding algebraic relations (Subsection 3.1.1.1). On this ground, the authors of [RCT05] first derive an algorithmic technique involving Gröbner bases to compute the *best* algebraic overapproximation of the set of states reachable by evolving continuously in a given location from an initial algebraic set. In other words, they provide algebraic techniques for solving Equation 3.2, once the parameter $Start_l = Init_l \cup \bigcup_{e=(l',l) \in E} R(e, Reach_{l'}) \cap G(e)$ is given as an algebraic set. Then, they cope with the termination in the fixpoint computation of the system of equations $\{Reach_l, Start_l\}_{l \in L}$ by using abstract interpretation techniques. In particular, termination is guaranteed by introducing a widening operator.

These two steps sketched above are described more detailed in the next two subsections.

3.1.1.1 Step 1: Algebraic Invariant Generation for Linear Continuous Dynamic Systems

A key point in the technique described in [RCT05] is the algebraic characterization of the set of points that can be reached following a linear continuous evolution starting from an algebraic set and subject to algebraic invariants. In particular, consider the set of points reachable in the state l of the algebraic hybrid automaton H, following a continuous evolution from the algebraic set $Start_l$:

$$\bigcup_{t\in\mathbb{R}^+} \Phi_l(Start_l, t) = \{\underline{x} \in \mathbb{R}^n \mid \exists \underline{x}^* \in Start_l \exists t \in \mathbb{R}^+ : \underline{x} = \Phi_l(\underline{x}^*, t)\}$$

where

$$\Phi_l(\underline{x}^*, t) = e^{A_l t}\underline{x}^* + e^{A_l t}(\int_0^t e^{-A_l \tau} d\tau) b_l$$

is the solution of the system of differential equations $\dot{\underline{x}} = A_l\underline{x} + b_l$ starting in $\underline{x}^*$.

Let A have only eigenvalues $a_1 + a_2 i \in \mathbb{C}$ with $a_1, a_2 \in \mathbb{Q}$. Using now the fact that first $\int_0^t e^{-A\tau} d\tau$ can be written as a sum of terms of the form $ct^k e^{\pm at} cos(bt)$ and $ct^k e^{\pm at} sin(bt)$ and second, the eigenvalues of A have rational coefficients, it is possible to express the solution $\Phi(\underline{x}^*, t)$ in terms of polynomials by introducing auxiliary variables and further constraints:

$$\begin{aligned}\{\underline{x} \in \mathbb{R}^n \mid \exists \underline{x} \in Start_l \exists t \in \mathbb{R}^+ \exists u, v, w, z \in \mathbb{R} : \\ \underline{x} = \Phi(\underline{x}^*, t, u, v, w, z) \wedge uv = 1 \wedge w^2 + z^2 = 1\}\end{aligned}$$

The two equations $uv = 1$ (corresponding to $e^x \cdot e^{-x} = 1$) and $w^2 + z^2 = 1$ (corresponding to $sin^2(x) + cos^2(x) = 1$) suffice to capture all algebraic invariants of the system.

Theorem 3.1

Let H be a hybrid automaton with linear continuous dynamics. Let $p_1, ..., p_n \in \mathbb{Q}[\underline{x}^*, t, u, v, w, z]$ be the polynomials approximating the flow Φ and let $Start_l$ be an algebraic set. Then:

$$I(Reach_l(Start_l)) = \langle I(Start_l), -x_1 + p_1, ..., -x_n + p_n, uv - 1, w^2 + z^2 - 1\rangle \cap \mathbb{R}[\underline{x}]$$

Proof (Idea)

1.: $\supseteq$ is obvious

2.: $\subseteq$

Take an arbitrary polynomial $q \in I(Reach_l(Start_l)) = I(\bigcup_{t\in\mathbb{R}^+} \Phi_l(Start_l, t))$ and normalize it w.r.t. $\langle -x_1 + p_1, ..., -x_n + p_n, uv - 1, w^2 + z^2 - 1\rangle$. It then can be shown that the normalized polynomial is an element of the ideal $I(Start_l) \trianglelefteq \mathbb{R}[X, X^*, t, u, v, w, z]$ which yields the inclusion. q.e.d.

This theorem can be generalized to arbitrary eigenvalues. Using the fact that $R = \{\pm Re(\lambda) \,|\, \lambda \text{ eigenvalue of } A\}$ is finite and thus that the $\mathbb{Q}$-vectorspace generated by these elements has a finite basis $\{p_1, ..., p_k\}$ implies that for all $a \in R$, there exist coefficients $c_1^a, ..., c_k^a \in \mathbb{Q}$ such that $a = \sum_{i=1}^{k} c_i^a p_i$. By adding correction factors to the basis $\{p_1, ..., p_k\}$ it can also be assumed w.l.o.g. that the c_i^a are integers. Thus, for all $a \in R$ the factors in $e^{at} = e^{\sum c_i^a p_i}$ again can be substituted by $u_i = e^{p_i t}$ and $v_i = e^{-p_i t}$. Then, adding the new equations $u_i v_i = 1$ ensures that Theorem 3.1 also holds for matrices with arbitrary eigenvalues.

In general, Theorem 3.1 states that it is possible to obtain by Gröbner bases techniques the *best* algebraic overapproximation of reachable sets in *purely* continuous systems with linear continuous dynamics, e.g. algebraic hybrid automata with linear continuous dynamics and no discrete transitions.

3.1.1.2 Step 2: A Widening Operator To Extend the Generation of Polynomial Invariants to Hybrid (Linear) Dynamics

In the previous section, the algorithmic technique to compute one iteration of the fixpoint problem given by the equations $\{Reach_l, Start_l\}_{l \in L}$ (Equations 3.2) is described. However, in general the computation of the fixpoint iteration for the set of equations $\{Reach_l \,|\, l \in L\}$ will not terminate in finite time. Thus the authors of [RCT05] propose to use the general framework of abstract interpretation by defining a widening operator. This widening operator forces the termination of the fixpoint iteration, but it results in further overapproximation. The widening operator $\bigtriangledown$ has to satisfy the following conditions:

- For $n, m \in \mathbb{N}$: $(Reach_l^m)_{l \in L}, (Reach_l^n)_{l \in L} \subseteq (Reach_l^n)_{l \in L} \bigtriangledown (Reach_l^m)_{l \in L}$
- For any increasing chain $(Reach_l^0)_{l \in L} \subseteq (Reach_l^1)_{l \in L} \subseteq ...$ the new increasing chain recursively defined by $(Reach_l^0)'_{l \in L} = (Reach_l^0)_{l \in L}$ and $(Reach_l^{n+1})'_{l \in L} = (Reach_l^n)'_{l \in L} \bigtriangledown (Reach_l^{n+1})_{l \in L}$ is not strictly increasing, i.e. stays constant after a finite number of steps.

The authors in [RCT05] suggest the usage of a widening operator respecting the above properties and defined by

$$I \bigtriangledown_d J := I(V(\{p \in GB(I \cap J, \prec) \,|\, deg(p) \leq d\}))$$

where $\prec$ is the graded term ordering. The above widening operator is derived from the idea of considering only polynomials of degree d or less instead of polynomials of arbitrary degree.

3.1.2 Generating Polynomial Invariants for Algebraic Hybrid Automata

In [SSM04, San05], a method for generating algebraic invariants for general algebraic hybrid automata is proposed. Unlike the invariant generation in [SSM04, San05] the automata are not restricted to have linear continuous dynamics. However, the method

is never guaranteed to find all algebraic invariants in contrast to what happens for the application of the technique in [RCT05] to purely linear continuous dynamic systems[1].

The key idea of the technique in [SSM04, San05] is to derive for a given *template assertion*, i.e. a parametric polynomial with unknown coefficients and bounded degree, constraints for the coefficients so that the resulting polynomials are ensured to be *inductive invariants*, i.e. invariants, which hold initially and are preserved by all continuous and discrete transitions of the automaton.

Using the notations in [SSM04, San05] inductive invariants can be defined on the base of the notion of *inductive assertion maps*: An *inductive assertion map* η for a hybrid automaton H is a function that maps each location $l \in L$ to an invariant $\eta(l)$ and satisfies the following three conditions:

- **Initiation**: $\eta(l_0)$ has to hold initially, i.e. $Init \models \eta(l_0)$
- **Discrete Transition**: If η holds before the discrete transition, it also has to hold afterwards, i.e. for each $e = (l_i, l_j) \in E$: $\eta(l_i) \wedge G(e) \models \eta(l_j)'$
- **Continuous Transition**: If η holds true before the continuous transition $(l, \underline{x}) \xrightarrow{\delta} (l, \underline{x}')$ it also has to hold true after, i.e. $\underline{x} \models \eta(l) \Rightarrow \underline{x}' \models \eta(l)$. If η is of the form $\eta(l)(x) = 0$, then this condition can be rewritten as $Inv(l) \wedge \eta(l)(\underline{x}) = 0 \models \dot{\eta}(l)(\underline{x}) = 0$

The notion of *template assertions* is further defined in [SSM04, San05] as parametric polynomial invariants $\sum_{c_\alpha} c_\alpha \cdot \underline{x}^\alpha = 0$, where the c_α are $\mathbb{R}$-linear combinations of the template variables $A = \{a_1, ..., a_k\}$

Given a hybrid automaton H, let η be a map associating to each location $l \in L$ of H a template assertion (over the template variables $A(l) = \{a_{l,1}, ..., a_{l,n}\}$). Then, in [SSM04, San05] a method for instantiating these template variables is presented such that the resulting assertion map is an inductive assertion map. This is done by mapping each condition of initiation, discrete transition and continuous transition to ideal membership problems, whose solutions yield constraints for the template variables in η. To give an example, the initiation condition $Init \models \eta(l_0)$ is mapped to the ideal membership problem $\eta(l_0) \in \langle Init \rangle$ and is solved via Gröbner basis techniques. In particular, the standard reduction techniques explained in Section 2.4 need to be extended to the notion of template assertions, as illustrated below.

Definition 3.1 (Template Normal Form NF)

Let $f = \sum c_\alpha \cdot \underline{x}^\alpha$ be a template and $P = \{p_1, ..., p_k\} \subset \mathbb{R}[\underline{x}]$. The template f can be reduced by a polynomial $p \in P$ iff $lm(p)$ divides a term $c \cdot t$ of f. The reduction $f \xrightarrow{p} f'$ has the form: $f' = f - \frac{c \cdot t}{lt(p)} p$. If no reduction with polynomials in P can be done, then f has a normal form.

$NF_P(f)$ denotes the normal form of f, i.e. f is fully reduced by P.

[1] This also holds for linear hybrid automata for which termination is not forced by any widening operator

Remark 3.1

The normal form of a template f wrt P is not unique. Uniqueness can be achieved, if a reduced Gröbner basis $G(P)$ of P is computed. In particular: $f \in \langle P \rangle \Leftrightarrow NF_{G(P)} f = 0$. Thus, the ideal membership problem reduces to computing the normal form w.r.t. the Gröbner basis of the given ideal.

Given the above definitions, the algorithm in [SSM04, San05] proceeds in three steps to derive an algebraic inductive assertion map η for a hybrid automaton H.

1. A template assertion of fixed degree is associated to each location l in H.
2. A set of constraints on the template variables is generated, which ensures to instantiate η to an inductive assertion map. Deriving the system of constraints boils down to formulate initialization, discrete and continuous transition as ideal membership problems, which can be solved by means of computer algebra.
3. The system of constraint equations from step two is solved.

3.1.2.1 Step 1: Fixing the Templates

For each location $l \in L$ of the hybrid automaton H, a degree d_l is chosen and the corresponding set of template variables $A_l = \{c_{l,\alpha} \mid |\alpha| \leq d_l\}$ is derived. For maximum of generality, each location should have its own template variables. Given $l \in L$, the template associated to l will then have the form $\eta(l) = \sum_{|\alpha| \leq d_l} c_{l,\alpha} \cdot \underline{x}^{\alpha}$.

3.1.2.2 Step 2: Deriving Constraints for the Template Variables

The following rules are used to map each condition of inductive assertion maps to ideal membership problems, whose solutions lead to a system of constraints over the template variables.

Initiation:
The initiation condition is encoded by $\eta(l) \in \langle Init \rangle$ being equivalent to $NF_{Init}(\eta(l)) = 0$.

Discrete Transition:
Let $(l_i, \underline{x}) \xrightarrow{e} (l_j, \underline{x}')$ be a discrete transition. Then η has to satisfy $\eta(l_i) \wedge G(e) \models \eta(l_j)'$. An exact encoding of this condition would mean to compute the Gröbner basis of the ideal generated by $\eta(l_i) = 0 \wedge G(e)$. The problem here is that $\eta(l_i)$ is a template, so that the construction of a Gröbner basis would be too difficult [SSM04, San05]. Hence the authors in [SSM04, San05] propose different ways to encode *stronger conditions* which can be handled more easily. These conditions are enumerated in the Table 3.1 below:

- *Local Transition Condition* (LC):
  $G(e)$ has to hold during the transition and η has to hold at the postlocation, ignoring any further information given by location invariants of pre- and postlocation and $\eta(l_i)$.

Name	Condition	Encoding
LC	$G(e) \models \eta(l_j)' = 0$	$NF_{G(e)}(\eta(l_j)') = 0$
CS	$G(e) \models \eta(l_i) = \eta(l_j)'$	$NF_{G(e)}(\eta(l_i) - \eta(l_j)') = 0$
CV	$\exists \lambda \in \mathbb{R} : G(e) \models \eta(l_j)' = \lambda \cdot \eta(l_i)$	$NF_{G(e)}(\lambda \cdot \eta(l_i) - \eta(l_j)') = 0$
PS	$\exists q \in \mathbb{R}[\underline{x}] : G(e) \models \eta(l_j)' = q \cdot \eta(l_i)$	$NF_{G(e)}(q \cdot \eta(l_i) - \eta(l_j)') = 0$

Table 3.1: Conditions for Discrete Transitions

Name	Condition	Encoding
CV	$Inv(l) \models \dot{\eta}(l) = 0$	$NF_{Inv(l)}(\dot{\eta}(l)) = 0$
CS	$\exists \lambda \in \mathbb{R} : Inv(l) \models \lambda \cdot \eta(l) - \dot{\eta}(l) = 0$	$NF_{Inv(l)}(\lambda \cdot \eta(l) - \dot{\eta}(l)) = 0$
PS	$\exists q \in \mathbb{R}[\underline{x}] : Inv(l) \models p \cdot \eta(l) - \dot{\eta}(l) = 0$	$NF_{Inv(l)}(p \cdot \eta(l) - \dot{\eta}(l)) = 0$

Table 3.2: Conditions for Continuous Transitions

- *Constant Value Condition* (CV):
  The values of η at pre- and postlocations have to coincide. Together with the fact that $\eta(l_i)(\underline{x}) = 0$, this yields $\eta(l_i)(\underline{x}') = 0$

- *Constant Scale Condition* (CS):
  The transition may change the value $\eta(l_1)(\underline{x})$ of the prelocation by a constant factor $\lambda \in \mathbb{R}$, i.e. $\lambda \cdot \eta(l_i)(\underline{x}) = \eta(l_j)(\underline{x}')$. Together with the fact $\eta(l_i)(\underline{x}) = 0$, this again yields $\eta(l_j)(\underline{x}) = 0$.

- *Polynomial Scale Condition* (PS):
  Here, the transition may change the value of η by a polynomial factor p.

Continuous Transition:
By definition, η has to satisfy $Inv(l)(\underline{x}) \wedge \eta(l)(\underline{x}) = 0 \models \dot{\eta}(l)(\underline{x}) = 0$. Similar to the case of discrete transitions, an exact encoding of the condition is impracticable [SSM04, San05], so again the authors introduce stronger conditions summarized in Tab. 3.2.

- *Constant Value Condition* (CV):
  The invariant $\eta(l)$ remains constant during the continuous path. This means that the first derivative w.r.t. time gained by the product rule of multivariate analysis $\dot{\eta}(l) := \sum_i \frac{\partial \eta(l)}{\partial x_i} \cdot \dot{x}_i$ has to be the zero-polynomial. Together with the fact $\eta(l)(\underline{x}) = 0$ at the beginning of the continuous transition, it holds for the whole path.

- *Constant Scale Condition*(CS):
  The first derivative of $\eta(l)$ has to be a multiple of $\eta(l)$. Together with $\eta(l)(\underline{x}) = 0$, it then holds for the whole path.

- *Polynomial Scale Condition* (PS):
  The first derivative of $\eta(l)$ has to be a polynomial multiple of $\eta(l)$. Again, together with $\eta(l)(\underline{x}) = 0$, it then holds for the whole path.

Condition	Restriction	Constraint types
Initiation	-	linear equalities
Transition	Local (LC)	linear equalities
	Constant Value (CV)	linear equalities
	Constant Scale (CS)	eigenvalue problems
	Polynomial Scale (PS)	non-linear algebraic

Table 3.3: Complexity of the equations for different transition types

Using Theorem 3.2, the above mapping of the conditions initiation, discrete and continuous transition to ideal membership problems induce a system of constraints over the template variables.

Theorem 3.2

Let $p = \sum_{c_\alpha} c_\alpha \cdot \underline{x}^\alpha \in \mathbb{R}[\underline{x}]$. *Then:* $\forall \underline{x} \in \mathbb{R}^n : p(\underline{x}) = 0 \Leftrightarrow \forall \alpha : c_\alpha = 0$

By Theorem 3.2, it is possible to extract the following equations from the conditions $NF_\cdot(.) = 0$: Let $0 = NF_\cdot(.) = \sum_\alpha a_\alpha \cdot \underline{x}^\alpha$. Then $\forall \alpha : a_\alpha = 0$. The above system of equations depends only on the template variables, unknown constants (for the CS case) and unknown polynomials (for the PS case). The complexity of the generated equations depends on the kind of transition conditions adopted, as illustrated in Tab. 3.3.

3.1.2.3 Step 3: Solving the Constraints

Solving the system of linear / nonlinear equations obtained in step 2 will either lead to no solutions (when the system is infeasible) or to a solution depending on 0 or more parameters λ_i, which are undetermined and can be freely chosen in $\mathbb{R}$. In order to get rid of the parameters λ_i, [SSM04, San05] proposed to set these free parameters equal to 1. This specialization still gives a solution, since the λ_i can be freely chosen and the resulting solution is parameter free. However, it is possible to extract more information from these parameters, which will be discussed in Section 3.2.1.4.

3.1.2.4 An Application Example

Consider the hybrid automaton in Figure 3.1 representing a bouncing ball. Assume further the initial condition $Init = \{y = \delta = 0,\ v = 16\}$.

Step 1: Fixing the Templates
Let us take templates of degree 2, i.e.

$$\eta = a + a_y y + a_v v + a_\delta \delta + a_{yy} y^2 + a_{yv} yv + a_{y\delta} y\delta + a_{vv} v^2 + a_{v\delta} v\delta + a_{\delta\delta} \delta^2$$

Step 2: Deriving Constraints for Template Variables via Ideal Membership

- Initiation: $NF_{Init}(\eta) = a + 16a_v + 256a_{vv}$
- Discrete Transition (LC): $NF_{\langle y \rangle}(\eta') = a + \frac{a_v}{2} v + \frac{a_{vv}}{4} v^2$

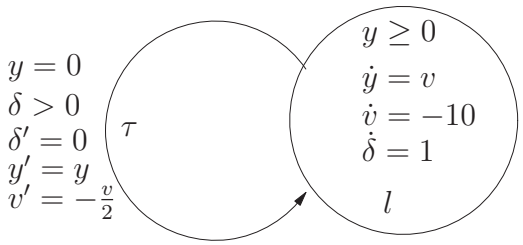


Figure 3.1: Bouncing Ball (see [SSM04, San05])

- Continuous Transition (CV):

$$\begin{aligned} NF_{I(l)}(\dot{\eta}) &= a_y v + a_{yy} yv + a_{y\delta} v\delta - 16a_v - 16a_{vv} v - 16a_{yv} y - 16a_{v\delta}\delta \\ &\quad + a_\delta + a_{\delta\delta}\delta + a_{y\delta} y + a_{v\delta} v \\ &= (-16a_v + a_\delta) + (a_{y\delta} - 16a_{yv})y + (a_y - 16a_{vv} + a_{v\delta})v \\ &\quad + (a_{\delta\delta} - 16a_{v\delta})\delta + a_{yy} yv + a_{y\delta} v\delta \end{aligned}$$

Step 3: Solving the Constraints
Initiation, discrete and continuous transition lead to the following linear constraints:

$$\begin{aligned} 0 &= a + 16a_v + 256a_{vv} \\ &= a = \frac{a_v}{2} = \frac{a_{vv}}{4} \\ &= -16a_v + a_\delta = a_{y\delta} - 16a_{yv} = a_y - 16a_{vv} + a_{v\delta} = a_{yy} = a_{y\delta} \end{aligned}$$

Solving this system yields the solution space for η: $\forall\lambda \in \mathbb{R} : \lambda \cdot (-y + 5\delta^2 + v\delta) = 0$
Hence the assertion map $\eta = -y + 5\delta^2 + v\delta$ is an inductive assertion map.

3.2 Polynomial Invariants via Algebraic Techniques

In this Section we will improve the algorithm for generating inductive assertion maps [SSM04] and make it incremental. Furthermore it will be shown, that Gröbner bases methods are not first choice when considering combined over- and underapproximation of the reachable set of states, which e.g. can be used for bounded model checking.

3.2.1 Incremental Generation of Polynomial Invariants

In this section, we attempt to improve on previous works using Gröbner bases techniques for the automated reasoning on hybrid automata. In particular, we consider the issue of generating inductive polynomial invariants on algebraic hybrid automata and we build up on the techniques developed in [SSM04, San05].

A major drawback of the method proposed in [SSM04, San05] for the generation of inductive algebraic invariants is the need of fixing in advance a bound for the target polynomials. In particular, the user does not know which dimensions of the templates will lead to useful invariants. Templates of too small degree will only lead to the trivial

invariant *true*. On the other hand, templates of too high degree won't yield more invariants than templates of smaller degrees. Another problem is that in case where there are no solutions (when the system of equations is infeasible), any search is a waste of time. While the problem of manipulating polynomials of fixed degrees is inherent in the technique in [SSM04, San05], we can at least make the invariant generation process *incremental* feature, in the sense that the effort done to check for polynomials of degree n is partly reused when it is necessary/desirable to deal with higher degrees.

Throughout this section, we propose a variant of the method in [SSM04, San05] that has the characteristics of proceeding incrementally in the generation of polynomial invariants. On the way, further improvements to [SSM04, San05] are suggested. In detail, Section 3.2.1.1 explores the possibility of strengthening the conditions for initiation, discrete and continuous transition. In Section 3.2.1.2 we show how to reduce the number of template variables in the incremental process of generating inductive assertions. In Section 3.2.1.3 an incrementality feature is added to the constraint generation step of the technique in [SSM04, San05]. Finally, in Section 3.2.1.4 the issue of interpreting the resulting systems of coefficient equations more effectively than proposed in [SSM04, San05].

3.2.1.1 Strengthening of the Initial Conditions

When defining the conditions for initiation, discrete and continuous transition, the authors of [SSM04, San05] make only very limited use of the fact that an inductive assertion map only has to hold on the reachable states. As an example, consider a discrete transition $(l_1, \underline{x}) \xrightarrow{e} (l_2, \underline{x}')$ and the corresponding discrete transition rule $G(e) \models \eta(l_2)' = 0$ described in Section 3.1.2. Since the transition has to be feasible, the location invariants both of pre- and postlocations have to hold during the discrete transition. This yields the strengthened condition $G(e) \wedge Inv(l_1) \wedge Inv(l_2)' \models \eta(l_2)' = 0$. With the above intuition, we proceed by providing a detailed description of possible strengthenings for the rules involving initiation, discrete and continuous transitions in [SSM04, San05].

Initiation Rule:
When initiating the hybrid automaton with starting values at the initial location l_0 then obviously the location invariant of this location also has to hold. This leads to the new initiation rule IN*: $\eta(l_0) \in \langle Init, Inv(l_0)\rangle :\Leftrightarrow NF_{Init \cup Inv(l_0)}(\eta(l)) = 0$

The new initiation condition is especially important for the cases where not all of the continuous variables are initiated with special values.

Discrete Transition Rule:
A discrete transition $l_1 \xrightarrow{e} l_2$ can only be triggered if the conditions $Inv(l_1)$ of the prelocation beforehand and $Inv(l_2)$ of the postlocation afterwards are satisfied. Assume $Inv(l_1) = (p_1(\underline{x}) = 0 \wedge ... \wedge p_{k_1}(\underline{x}) = 0)$ and denote the invariant conditions for the postlocation l_2 by $Inv(l_2)' = (q_1(\underline{x}') = 0 \wedge ... \wedge q_{k_2}(\underline{x}') = 0)$. Then, the following new discrete transition rules can be obtained:

- Local Transition (LC*): $Inv(l_1) \wedge Inv(l_2)' \wedge G(e) \models \eta(l_2)' = 0$.
  This is equivalent to $NF_{G(e) \cup Inv(l_1) \cup Inv(l_2)'}(\eta(l_2)') = 0$.
- Constant Value (CV*): $Inv(l_1) \wedge Inv(l_2)' \wedge G(e) \models \eta(l_2)' = \eta(l_1)$.

This is equivalent to $NF_{G(e)\cup Inv(l_1)\cup Inv(l_2)'}(\eta(l_2)' - \eta(l_1)) = 0$.

- Constant Scale (CS*):$\exists\lambda \in \mathbb{R} : Inv(l_1) \wedge Inv(l_2)' \wedge G(e) \models \eta(l_2)' = \lambda \cdot \eta(l_1)$.
  This is equivalent to $\exists\lambda \in \mathbb{R} : NF_{G(e)\cup Inv(l_1)\cup Inv(l_2)'}(\eta(l_2)' - \lambda \cdot \eta(l_1)) = 0$.

- Polynomial Scale (PS*): $\exists p \in \mathbb{R}[\underline{x}] : Inv(l_1) \wedge Inv(l_2)' \wedge G(e) \models \eta(l_2)' = p \cdot \eta(l_1)$.
  This is equivalent to $\exists p \in \mathbb{R}[\underline{x}] : NF_{G(e)\cup Inv(l_1)\cup Inv(l_2)'}(\eta(l_2)' - p \cdot \eta(l_1)) = 0$.

Continuous Transition Rule:
As in the discrete case a continuous transition can only be triggered, if certain circumstances are satisfied. Consider a path $(l, \underline{x}_1) \xrightarrow{\delta} (l, \underline{x}_2)$. Then on the whole path the location conditions $Inv(l)$ has to be satisfied, i.e. the value of the polynomials in $Inv(l)$ is constantly zero and thus the Lie-Derivative must also be zero. In other words:

Lemma 3.3

Let l be a location with location conditions $Inv(l) = \{p_1(\underline{x}) = 0, ...p_k(\underline{x}) = 0\}$. Let $(l, \underline{x}_0)$ be reachable. Then for any continuous transition $(l, \underline{x}_0) \xrightarrow{\delta} (l, \underline{x}_1)$ the condition $\dot{Inv}(l)(\underline{x}) := \{\dot{p}_1(\underline{x}) = 0, ..., \dot{p}_1(\underline{x}) = 0\}$ is satisfied on the whole path (necessary condition).

Proof

Let $(l, \underline{x}_0) \xrightarrow{\delta} (l, \underline{x}_1)$ be a feasible continuous path, i.e. on the whole path the location conditions $Inv(l)$ are satisfied. Since the path is feasible, the p_i are constantly zero during the continuous evolution. Thus, on the whole path the Lie-derivative of the polynomials is equal to zero, i.e.

$$\dot{Inv}(l) := \{\dot{p}_1(\underline{x}) = 0, ..., \dot{p}_1(\underline{x}) = 0\} \qquad \text{q.e.d.}$$

Lemma 3.3 especially ensures that at any point where continuous transitions are possible, $\dot{Inv}(l)$ also is satisfied. Thus, we obtain the following new continuous transition rules:

- Constant Value (CV*): $Inv(l) \wedge \dot{Inv}(l) \models \dot{\eta}(l) = 0$.
  This is equivalent to $NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{\eta}(l)) = 0$.

- Constant Scale (CS*): $\exists\lambda \in \mathbb{R} : Inv(l) \wedge \dot{Inv}(l) \models \dot{\eta}(l) - \lambda \cdot \eta(l) = 0$.
  This is equivalent to $\exists\lambda \in \mathbb{R} : NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{\eta}(l) - \lambda \cdot \eta(l)) = 0$.

- Polynomial Scale (PS*): $\exists p \in \mathbb{R}[x] : Inv(l) \wedge \dot{Inv}(l) \models \dot{\eta}(l) - p \cdot \eta(l) = 0$.
  This is equivalent to $\exists p \in \mathbb{R}[x] : NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{\eta}(l) - p \cdot \eta(l)) = 0$.

Correctness of the new Conditions

Whenever an inductive assertion map satisfies one of the original conditions for initiation, discrete and continuous transition, this also holds for the corresponding conditions described above. Furthermore, the new conditions are really stronger than the original ones, i.e. there exist inductive assertion maps satisfying the '*'-conditions, but not the original ones. This result is summarized in the following Theorem and a counterexample for the backward direction is given in Example 3.1.

Theorem 3.4

Let η be an inductive assertion map of the hybrid automaton H. Then, for initiation, discrete and continuous transition the following holds:

- *Initiation: IN implies IN*.*
- *Discrete Transition: {LC, CS, CV, PS} imply {LC*, CS*, CV*, PS*}.*
- *Continuous Transition: {CS, CV, PS} imply {CS*, CV*, PS*}.*

In each case the converse is not true.

Proof (Sketch)

Let e.g. $l_1 \xrightarrow{e} l_2$ be a discrete transition and let η satisfy LC for discrete transition, i.e. $NF_{G(e)}(\eta(l_2)') = 0$. Then also $NF_{Inv(l_1)\cup G(e)\cup Inv(l_2)'}(\eta(l_2)') = 0$. Thus for discrete transition LC implies LC*. The other cases are analogous. q.e.d.

Example 3.1

Consider the hybrid automaton in Figure 3.2 with initial condition $Init = \{y = 1\}$ and term ordering length-lex with $x > y$. Let η be of degree one, i.e. $\eta(l_0) = a + a_x x + a_y y$

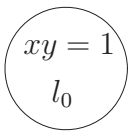


Figure 3.2: Hybrid Automaton with Different Results for Conditions IN and IN*

- *Initiation condition IN:* $\langle Init\rangle = \langle y - 1\rangle$
  $\Rightarrow$ $NF_{Init}(\eta(l_0)) = a + a_y + a_x x$
  $\Rightarrow$ *The corresponding assertion map is* $\forall\lambda \in \mathbb{R} : \eta(l_0) = \lambda \cdot (y - 1)$.
  $\Rightarrow$ *By Theorem 3.11, the invariant* $y = 1$ *is found.*
  $\Rightarrow$ *Every reachable state has to satisfy* $y = 1$.
- *Initiation condition IN*:* $\langle Init \cup Inv(l_0)\rangle = \langle x - 1, y - 1\rangle$
  $\Rightarrow$ $NF_{Init\cup Inv(l_0}(\eta(l_0)) = a + a_y + a_x$
  $\Rightarrow$ *The corresponding assertion map is* $\forall\lambda, \delta \in \mathbb{R} : \eta(l_0) = \lambda\cdot(y-1)+\delta\cdot(x-1)$.
  $\Rightarrow$ *By Theorem 3.11, the invariants* $y = 1$ *and* $x = 1$ *are found.*
  $\Rightarrow$ *Every reachable state has to satisfy* $y = 1$ *and* $x = 1$.

$\Rightarrow$ *The overapproximation of reachable states corresponding to the condition IN* is more precise than the one corresponding to IN.*

3.2.1.2 Incremental Reduction of the Number of Templates Variables

When deriving invariants via inductive assertion maps, we are only interested in getting new information. This means that the invariants should have normal forms w.r.t. already known invariants, since adding polynomial combinations of the known invariants does not change the amount of information. Using the strengthened conditions for initiation, discrete and continuous transition introduced in Section 3.2.1.1 together with this fact, it is possible to decrease the number of template variables before the beginning of any computation. This observation is formalized in Theorem 3.7 which states that it is possible to preliminarily set all those template variables to zero, which are reducible by the already known invariants. This reduction of the number of template variables is especially interesting when an incremental approach of the algorithm is used. In this case it is possible to use the knowledge gained in each step directly in order to reduce the number of variables for further steps.

Lemma 3.5 gives a motivation for the reduction of the number of template variables while Theorem 3.7 shows that no information gets lost by doing so. Lemma 3.6 is just a prerequisite of this theorem.

Lemma 3.5

Let $G = \{p_1, ..., p_k\}$ be a Gröbner basis for the invariants already holding in a location l and $LT(G) := \{lt(p_1), ..., lt(p_k)\}$. Then for any further invariant $q(x) := \sum_{\alpha} c_\alpha \cdot x^\alpha$ of this location having normal form w.r.t. G, the following is satisfied:
$\forall \alpha \in \mathbb{N}^n$: *If any monomial of $LT(G)$ divides x^α then $c_\alpha = 0$.*

Proof

Let $q(x) := \sum_{\alpha} c_\alpha \cdot x^\alpha$ be a reduced invariant. Assume $\exists \alpha \in \mathbb{N}^n : \exists i \in 1, ..., k : lt(p_i) | x^\alpha$ and $c_\alpha \neq 0$. Then q can be reduced via: $q \rightarrow q - \frac{lc(p_i)}{c_\alpha} \cdot \frac{x^\alpha}{lt(p_i)} \cdot p_i$. This is a contradiction to that q already has normal form w.r.t. G. q.e.d.

Lemma 3.6

Let η be an inductive assertion map. Then the following holds:

- *Initiation:*
  $NF_{Init \cup Inv(l_0)}(\eta(l_0)) = NF_{Init \cup Inv(l_0)}(NF_{Inv(l_0)}\eta(l_0))$
- *Discrete Transition:*
  Let $q(l) \in \{\eta(l_2)', \eta(l_2)' - \eta(l_1), \eta(l_2)' - \lambda\eta(l_1), \eta(l_2)' - p \cdot \eta l_1\}$ depending on the actual discrete transition condition. Then

  $$NF_{Inv(l') \cup \rho \cup Inv'(l)}(q(l)) = NF_{Inv(l') \cup \rho \cup Inv(l)'}(NF_{Inv(l)'}q(l))$$

- *Continuous Transition: Let $\lambda \in \mathbb{R}$ and $p \in \mathbb{R}[\underline{x}]$*
  - $NF_{Inv(l) \cup \dot{Inv}(l)}(\dot{\eta}(l)) = NF_{Inv(l) \cup \dot{Inv}(l)}(\dot{NF_{Inv(l)}}(\eta(l))$

– $NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{\eta}(l)-\lambda\eta(l)) = NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{NF_{Inv(l)}}(\eta(l))-NF_{Inv(l)}(\lambda\eta(l)))$

– $NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{\eta}(l)-p\eta(l)) = NF_{Inv(l)\cup \dot{Inv}(l)}(\dot{NF_{Inv(l)}}(\eta(l))-NF_{Inv(l)}(p\eta(l)))$

Proof

Initiation:

Let $NF_{Inv(l_0)}(\eta(l_0)) = \eta(l_0) - p$ with $p \in \langle Inv(l_0)\rangle \trianglelefteq \langle Init, Inv(l_0)\rangle \trianglelefteq \mathbb{R}^n$. Then

$$\begin{aligned} NF_{Init\cup Inv(l_0)}(\eta(l_0)) &= NF_{Init\cup Inv(l_0)}((\eta(l_0)-p)+p) \\ &= NF_{Init\cup Inv(l_0)}(NF_{Inv(l_0)}\eta(l_0)+p) \\ &= NF_{Init\cup Inv(l_0)}(NF_{Inv(l_0)}\eta(l_0)) + NF_{Init\cup Inv(l_0)}(p) \\ &= NF_{Init\cup Inv(l_0)}(NF_{Inv(l_0)}\eta(l_0)) + 0 \\ &= NF_{Init\cup Inv(l_0)}(NF_{Inv(l_0)}\eta(l_0)) \end{aligned}$$

Discrete Transition:

Let $q \in \{\eta(l_2)', \eta(l_2)'-\eta(l_1), \eta(l_2)'-\lambda\eta(l_1), \eta(l_2)'-p\cdot\eta l_1\}$ depending on the discrete transition condition. Let $NF_{Inv(l)'}(q) = q-p$ with $p \in \langle Inv(l)\rangle \trianglelefteq \langle Inv(l')\cup\rho\cup Inv'(l)\rangle$. Then

$$\begin{aligned} NF_{Inv(l')\cup\rho\cup Inv'(l)}(q) &= NF_{Inv(l')\cup\rho\cup Inv'(l)}(q-p+p) \\ &= NF_{Inv(l')\cup\rho\cup Inv'(l)}(NF_{Inv(l)'}(q)+p) \\ &= NF_{Inv(l')\cup\rho\cup Inv'(l)}(NF_{Inv(l)'}(q)) + NF_{Inv(l')\cup\rho\cup Inv'(l)}(p) \\ &= NF_{Inv(l')\cup\rho\cup Inv'(l)}(NF_{Inv(l)'}(q)) + 0 \\ &= NF_{Inv(l')\cup\rho\cup Inv'(l)}(NF_{Inv(l)'}(q)) \end{aligned}$$

Continuous Transition:

Let the transition condition be CV* and let $I(l) = \langle p_1, ..., p_k\rangle$. Let

$$\eta(l) = NF_{I(l)}(\eta(l)) + \sum_{i=1}^{k} g_i \cdot p_i$$

Then

$$\dot{\eta}(l) = \dot{NF_{I(l)}}(\eta(l)) + \dot{\sum_{i=1}^{k}} g_i \cdot p_i$$

By product rule

$$\dot{(\sum_{i=1}^{k}} g_i \cdot p_i) = \sum_{i=1}^{k} g_i \cdot \dot{p}_i + \dot{g}_i \cdot p_i.$$

Together with $\dot{p}_i \in \dot{I}(l)$ this yields

$$\begin{aligned} NF_{I(l)\cup\dot{I}(l)}(\dot{\eta}(l)) &= NF_{I(l)\cup\dot{I}(l)}(\dot{NF_{I(l)}}(\eta(l))) + NF_{I(l)\cup\dot{I}(l)}(\sum_{i=1}^{k} g_i \cdot \dot{p}_i + \dot{g}_i \cdot p_i) \\ &= NF_{I(l)\cup\dot{I}(l)}(\dot{NF_{I(l)}}(\eta(l))) + 0 \\ &= NF_{I(l)\cup\dot{I}(l)}(\dot{NF_{I(l)}}(\eta(l))) \end{aligned}$$

The other two cases follow analogously. q.e.d.

Theorem 3.7

Let $G = \{p_1, ..., p_k\}$ be a Gröbner basis for the invariants already holding in a location l and $LT(G) := \{lt(p_1), ..., lt(p_k)\}$. Consider $A = \{\alpha \mid \exists x^\beta \in LT(G) : x^\beta | x^\alpha\}$. The template assertion maps $T = \sum_\alpha c_\alpha \underline{x}^\alpha$ and $T' = \sum_{\alpha \notin A} c_\alpha \underline{x}^\alpha$ are equivalent for all combinations of the strengthened transition conditions, i.e. they lead to the same set of new invariants for l.

Proof

Let $\eta(l) = \sum_\alpha c_\alpha \cdot \underline{x}^\alpha$, and let $x^{\hat{\alpha}}$ be a term of $\eta(l)$ being divided by the leading term of $p = \sum_\gamma p_\gamma \cdot \underline{x}^\gamma$, $p \in G$. Since the polynomials in G are polynomials over $\mathbb{R}$, we can assume w.l.o.g. that the leading coefficient of p is equal to 1. G is a Gröbner basis, so $NF_{I(l)}(\eta(l)) = NF_{I(l)}(\eta(l) - c_{\hat{\alpha}} \cdot \frac{\underline{x}^{\hat{\alpha}}}{lt(p)} p)$ (*). Let now $\frac{\underline{x}^{\hat{\alpha}}}{lt(p)} = \underline{x}^\beta$. Then

$$\eta(l) - c_{\hat{\alpha}} \cdot \frac{\underline{x}^{\hat{\alpha}}}{lt(p)} p = \sum_\alpha c_\alpha \cdot \underline{x}^\alpha - c_{\hat{\alpha}} \cdot \underline{x}^\beta \cdot \sum_\gamma p_\gamma \cdot \underline{x}^\gamma = \sum_\alpha c_\alpha \cdot \underline{x}^\alpha - \sum_\alpha p'_\alpha \cdot \underline{x}^\alpha$$

where in the polynomial p the terms $p_\gamma \cdot \underline{x}^{\gamma+\beta}$ are renamed to $p'_\alpha \cdot \underline{x}^\alpha$ according to $p'_\alpha = c_{\hat{\alpha}} \cdot p_\beta$ and the coefficients to terms of $\eta(l)$ not occurring in $\underline{x}^\beta \cdot p$ are set to zero. Thus,

$$\eta(l) - c_{\hat{\alpha}} \cdot \frac{\underline{x}^{\hat{\alpha}}}{lt(p)} p = \sum_\alpha (c_\alpha - p'_\alpha) \cdot \underline{x}^\alpha = \sum_{\alpha \neq \hat{\alpha}} (c_\alpha - p'_\alpha) \cdot \underline{x}^\alpha$$

because of the reduction with p. Renaming the coefficients of $\eta_{new}(l) := \eta(l) - c_{\hat{\alpha}} \frac{\underline{x}^{\hat{\alpha}}}{lt(p)} p$ yields:

$$\eta_{new}(l) = \eta(l) - c_{\hat{\alpha}} \cdot \frac{\underline{x}^{\hat{\alpha}}}{lt(p)} p = \sum_{\alpha \neq \hat{\alpha}} c'_\alpha \underline{x}^\alpha$$

Now take a look at the conditions of initiation, discrete and continuous transition. By (*) and Lemma 3.6 the results do not change by replacing η by η_{new}.
Hence, one can replace in each step of the algorithm the polynomial $\eta(l)$ by $\eta_{new}(l)$. Thus, any calculations can be done with $\eta_{new}(l)$ and afterwards adding the equations $c'_\alpha = c_\alpha - p'_\alpha$. Solving the system of coefficient equations with the coefficients

c'_α of $\eta_{new}(l)$ leads to the solution space for $\eta_{new}(l)$. Considering the original coefficients of $\eta(l)$ by $c'_\alpha = c_\alpha - p'_\alpha$ yields just the addition of $c_{\hat{\alpha}} \cdot \underline{x}^\beta \cdot p$ with arbitrary $c_{\hat{\alpha}}$, since in the calculations with $\eta_{new}(l)$, no $c_{\hat{\alpha}}$ occurs and only the equations belonging to the reduction with p have any $c_{\hat{\alpha}}$ inside.

Thus, the solution space of the original assertion $\eta(l)$ has - modulo $I(l)$ - no more solutions than $\eta_{new}(l)$.

Doing this inductively for all reducable terms of $\eta(l)$ yields the claim. q.e.d.

3.2.1.3 Incremental Generation of Constraints

In the previous subsection it has been shown how precomputed invariants can be used in order to reduce the number of template variables of higher degree. In this subsection, we now develop a way to 'recycle' the computations of assertion coefficients along the series of (increasing degree) invariants generated by the algorithm in [SSM04, San05].

In order to achieve this goal, we first establish a number of conditions for these hybrid automata, which ensure that the coefficient equations associated to the template variables of lower degree will not be changed any more when the degree of the inductive assertion map is increased. Then, a preprocessing step (see Algorithm 2) is introduced which transforms an arbitrary algebraic hybrid automaton in an algebraic hybrid automaton respecting the above conditions. Finally, the algorithm for the incremental generation of constraints is introduced (see Algorithm 3).

Lemma 3.8, Lemma 3.9 and Lemma 3.10 analyze the behavior of the coefficient equations of an inductive assertion map under discrete consecution, continuous consecution and normalization w.r.t. the Gröbner basis of existing invariants.

Lemma 3.8

Let l be a location over the location variables $\underline{x} = (x_1, ..., x_n)$. Let the first derivatives of the x_i be polynomials in $\mathbb{R}[\underline{x}]$ and let $p \in \mathbb{R}[\underline{x}]_{>k}$. Then, $\dot{p}[\underline{x}] \in \mathbb{R}[\underline{x}]_{>k-1}$.

Proof

By definition: $\dot{p}[\underline{x}] = \sum_{i=1}^{k} \frac{\partial p}{\partial x_i} \cdot \dot{x}_i$.

Let $p = \sum_{k<|\alpha|<deg(p)} c_\alpha \cdot \underline{x}^\alpha$ with $c_\alpha \cdot \underline{x}^\alpha = c_\alpha \cdot x_1^{\alpha_1} ... x_n^{\alpha_n}$. Then

$$\frac{\partial c_\alpha \cdot \underline{x}^\alpha}{\partial x_i} = \begin{cases} \alpha_i \cdot c_\alpha \cdot x_1^{\alpha_1} ... x_i^{\alpha_i - 1} ... x_n^{\alpha_n} & \alpha_i \neq 0 \\ 0 & \alpha_i = 0 \end{cases}$$

i.e.

$$deg(\frac{\partial c_\alpha \cdot \underline{x}^\alpha}{\partial x_i}) = \begin{cases} deg(c_\alpha \cdot \underline{x}^\alpha) - 1 & \alpha_i \neq 0 \\ 0 & \alpha_i = 0 \end{cases}$$

Thus, either $\frac{\partial c_\alpha \cdot \underline{x}^\alpha}{\partial x_i} \dot{x}_i = 0$ or $\frac{\partial c_\alpha \cdot \underline{x}^\alpha}{\partial x_i} \dot{x}_i \in \mathbb{R}[\underline{x}]_{>k-1}$ q.e.d.

Lemma 3.9

Let τ be a transition with $x_i' \in \mathbb{R}[\underline{x}]_{\geq k_i}$ or $x_i' = 0$. Let $p \in \mathbb{R}[\underline{x}]_{\geq k}$. Then $p(\underline{x}') \in \mathbb{R}[\underline{x}]_{\geq k \cdot min\{k_i\}}$ or $p(\underline{x}') = 0$.

Proof

Let $p = \sum_{\alpha} c_\alpha \cdot \underline{x}^\alpha$. Let $c_\alpha \cdot \underline{x}^\alpha = c_\alpha \cdot x_1^{\alpha_1} ... x_n^{\alpha_n}$ be a term of p and let all α_i be equal to 0 where $x_i' = 0$. Otherwise $c_\alpha \cdot (\underline{x}^\alpha) = 0$ is not of interest. Then,

$$c_\alpha \cdot (\underline{x}^\alpha)' = c_\alpha \cdot (x_1^{\alpha_1})' ... (x_n^{\alpha_n})'$$

and thus

$$\begin{aligned} deg(c_\alpha \cdot (\underline{x}^\alpha)') &= \sum_{i=1}^{n} \alpha_i \cdot k_i \\ &\geq \sum_{i=1}^{n} \alpha_i \cdot min\{k_i\} \\ &= min\{k_i\} \cdot \sum_{i=1}^{n} \alpha_i \\ &= min\{k_i\} \cdot |\alpha| = min\{k_i\} \cdot deg(c_\alpha \cdot \underline{x}^\alpha) \end{aligned}$$

So, $p(\underline{x}') = 0$ or $p \in \mathbb{R}[\underline{x}]_{k \cdot min\{k_i\}}$ since all terms of $p(\underline{x}')$ have a degree of $\geq k \cdot min\{k_i\}$. q.e.d.

Lemma 3.10

Let f and p be homogeneous polynomials and let f be reducable by p. Then, the reduced polynomial f' is still homogeneous and is either the zero-polynomial or has the same degree as f.

Proof

Let $f = \sum_{\alpha} c_\alpha \cdot \underline{x}^\alpha$ be homogeneous and let $c_{\hat{\alpha}} \cdot \underline{x}^{\hat{\alpha}}$ be a term of f being reducable by the homogeneous polynomial p. Then, $f \xrightarrow{p} f' = f - \frac{c_{\hat{\alpha}} \cdot \underline{x}^{\hat{\alpha}}}{lt(p)} p$. Since $\frac{c_{\hat{\alpha}} \cdot \underline{x}^{\hat{\alpha}}}{lt(p)}$ consists only of one term and p is homogeneous, $\frac{c_{\hat{\alpha}} \cdot \underline{x}^{\hat{\alpha}}}{lt(p)} \cdot p$ is also homogeneous and has degree

$$deg(\tfrac{c_{\hat{\alpha}} \cdot \underline{x}^{\hat{\alpha}}}{lt(p)}) + deg(p) = (deg(f) - deg(p)) + deg(p) = deg(f)$$

Adding up two homogeneous polynomials of the same degree either yields the zero-polynomial or a polynomial of the same degree. This yields the claim. q.e.d.

Given the results in the above Lemmas, we are now ready to provide a general idea of an incremental variant of the algorithm in [SSM04, San05], where 'incremental' refers to the fact, that

1. for the determination of the template variables referring to invariants having degree n, we make use of computations relative to lower degree templates, and

2. we make use of already obtained invariants of (lower) degree obtained by performing Gröbner basis reductions.

Assume that all the computations for templates of degree n are available, and templates of degree $n+1$ are to be determined. The new templates will have the form

$$\eta_{n+1}(l) = \sum_{|\alpha|\leq n+1} c_{l,\alpha} \cdot \underline{x}^{\alpha} = \sum_{|\alpha|\leq n} c_{l,\alpha} \cdot \underline{x}^{\alpha} + \sum_{|\alpha|=n+1} c_{l,\alpha} \cdot \underline{x}^{\alpha} =: \eta_n(l) + \eta_{n+1,diff}(l) \ (*)$$

As can be seen from (*), the new template can be divided into two smaller templates, the template $\eta(l)$ of degree $\leq n$ and a homogeneous template $\eta_{n+1,diff}(l)$ of degree $n+1$. For $\eta_n(l)$, the polynomials $\eta_n'(l)$ and $\dot{\eta}_n(l)$ have been already computed when dealing with templates of lower degrees. Furthermore, the operations $\eta(l)'$, $\dot{\eta}(l)$ and $NF_G(\eta(l))$ are additive w.r.t. addition, i.e. $\eta' + \mu' = (\eta+\mu)'$,$\dot{\eta} + \dot{\mu} = (\eta + \mu)$ and $NF_G(\eta(l) + \mu(l)) = NF_G(\eta) + NF_G(\mu)$. Thus, it suffices to apply each one of the above operations to the polynomial $\eta_{n+1,diff}(l)$ and in a second step to combine this with the already available results for $\eta_n(l)$.

Now have a look at $\eta_{n+1,diff}'$ and $\dot{\eta}_{n+1,diff}$. By definition, the template $\eta_{n+1,diff}$ is homogeneous of degree $n+1$. Hence, by Lemma 3.8 $\dot{\eta}_{n+1,diff} \in \mathbb{R}[\underline{x}]_{>n-1}$, i.e. no coefficient equation of terms of degree $\leq n-1$ will be changed by $\dot{\eta}_{n+1,diff}$. The discrete transition is a bit more complicated. If at least one of the polynomials x_i' has a constant term, all coefficient equations will be changed. In order to illustrate the problem, we will give two simple examples:

Example 3.2

Consider the following two automata, that differ only in the reset function of the discrete transition. Let furthermore the discrete transition condition LC and $\eta(l_1) = \sum_i c_i x^i$.

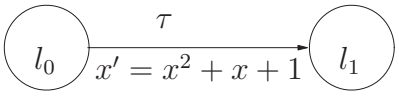


Figure 3.3: Changes induced by Discrete Transition (1)

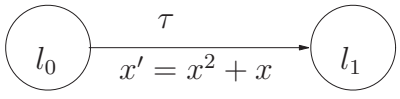


Figure 3.4: Changes induced by Discrete Transition (2)

- *Consider the automaton of Figure 3.3.*

  $\forall n \in \mathbb{N}: \ \eta_{n+1,diff}' = c_{n+1} \cdot (x')^{n+1} = c_{n+1} \cdot (x^2 + x + 1)^{n+1}$ *has terms of every degree* $\leq 2n+2$.

  Thus, in this example the transition conditions of the discrete transition e will always change every coefficient equation of $\eta(l_2)'$.

- *Consider the slightly modified automaton of Figure 3.4:*

  $\forall n \in \mathbb{N}: \ \eta'_{n+1,diff} = c_{n+1} \cdot (x')^{n+1} = c_{n+1} \cdot (x^2 + x)^{n+1}$ *has only terms of degree* $\geq n+1$.
  Thus, when increasing the degree of the template from n *to* $n+1$*, no coefficient equation for terms of degree* $\leq n$ *will be changed for the discrete transition* τ

A similar problem occurs with the normalization of η, since location invariants and guards on discrete transitions are seldom homogeneous. Because of these two problems, the hybrid automaton has to satisfy the following assumption for applying an incremental approach:

Assumption 1

- *For all discrete transitions, the polynomials* x'_i *are either the zero-polynomials or do not have constant terms.*

- *All location invariants and transition invariants are homogeneous.*

Unfortunately, hybrid automata seldom have the form claimed in Assumption 1. Therefore, before applying an incremental algorithm for the generation of inductive assertion maps, the automaton has to be modified in such a way, that (1) the original behavior of the automaton is preserved, and (2) the discrete transitions and given invariants satisfy Assumption 1. This is achieved by a 'partial homogenization' preprocessing, which transforms the given algebraic hybrid automaton into an equivalent automaton for which the following conditions hold:

1. For all discrete transitions $l \xrightarrow{e} l'$: $x'_i \in \mathbb{R}[\underline{x}]_{\geq 1}$, i.e. the x'_i do not contain a constant term.

2. All location conditions in $Inv(l)$, $\dot{Inv}(l)$ and guards on the discrete transitions are homogeneous polynomials.

The preprocessing is done in three steps: In the first step, a slack variable s is introduced, which is initialized with the value 1 and is never be changed throughout every discrete or continuous change. In other words, $\dot{s} = 0$ and the reset function for each discrete transition is set to $s' = s$. In the second step, all discrete transitions not satisfying the first item in Assumption 1 are changed in the following way:

$$\sum_{|\alpha|\geq 1} c_\alpha \underline{x}^\alpha + c_0 = x'_i \quad \mapsto \quad x'_i = \sum_{|\alpha|\geq 1} c_\alpha \underline{x}^\alpha + c_0 \cdot s \in \mathbb{R}[\underline{x}, s]_{>0}$$

In the last step, all location invariants and guards of discrete transitions are homogenized. For an invariant $p \in \mathbb{R}[\underline{x}]$ of degree k, this homogenization has the following form:

$$p(\underline{x}) = \sum_{|\alpha|\leq k} c_\alpha \underline{x}^\alpha \quad \mapsto \quad p^h(\underline{x}) = \sum_{|\alpha|\leq k} c_\alpha \underline{x}^\alpha s^{k-|\alpha|}$$

The formal description of the preprocessing algorithm sketched above is given in Algorithm 2.

Algorithm 2: Homogenize

input : $H = \langle L, E, X, Init, Inv, F, G, R\rangle$

output: $H' = \langle L', E', X', Init', Inv', F', G', R'\rangle$ satisfying Assumption 1

1 **begin**

2 $H' := H$

/* Step 1: Introducing the slack variable s */

3 $X' := X' \cup \{s\}$

4 **foreach** $l \in L'$ **do**

5 $F'(l) := F'(l) \cup \{\dot{(}s) = 0\}$

6 **foreach** $e \in E'$ **do**

7 $R'(e) := R'(e) \cup \{s' = s\}$

8 $Init' := Init' \cup \{s = 1\}$

/* Step 2: Changing the discrete transition condition */

9 **foreach** $e \in E' \wedge x_i \in X'$ **do**

10 $x_i' := \sum_{\alpha>0} c_\alpha \underline{x}^\alpha + c_{(0,\ldots,0)} \cdot s$

/* Step 3: Homogenizing location invariants and guards */

11 **foreach** $l \in L' \wedge 0 = p(\underline{x}) \in Inv'(l)$ **do**

12 $p := p^h$

13 **return** H'

14 **end**

Remark 3.2

The inductive assertion $y - 1 = 0$ for all locations l of the partly homogenized automaton will be found with any combination of CV', CS' and PS' for discrete transitions and CV', CS', PS' for continuous transitions.

The preprocessing procedure in Algorithm 2 provides the input for our incremental polynomial invariant generation algorithm, whose pseudocode is given in Algorithm 3. Such an algorithm exploits the results of Theorem 3.7 in order to incrementally reduce the number of template variables used, as well as the results of the Lemmas 3.9, 3.8, and 3.10, to incrementally generate the constraints. In Algorithm 3, the following notations and variables are used:

- S is the set of coefficient equations, which won't be changed any more,
- S' is a helper set of still changeable coefficient equations,
- $q(l), q(e)$ are polynomials for keeping the terms for continuous transitions and discrete transitions, which can still be changed,
- for $q \in \mathbb{R}[\underline{x}]$, $q_{>i}$ denotes the terms of q of degree $> i$ and $q_{=i}$ the terms of degree i,
- $\dot{\eta}^i(l)$ is defined in the following way (depending on the transition condition):
  - CV': $\dot{\eta}^i(l) := \dot{\eta}_{=i}(l)$
  - CS': $\dot{\eta}^i(l) := \dot{\eta}_{=i}(l) - \lambda_l \cdot \eta_{=i}$
  - PS': $\dot{\eta}^i(l) := \dot{\eta}_{=i}(l) - p_l \cdot \eta_{=i}$

  Analogously, $\eta^i(l)'$ is definied.
- Let $e \in E$ and $x_i' \in \mathbb{R}[\underline{x}]_{\geq k_i}$. Then, as in Lemma 3.6: $k_e := min(k_i)$.

3.2.1.4 Analysis of the Solutions

In each iteration step, Algorithm 3 produces a set of coefficient equations needed to be solved. Usually, if the set of constraints is solvable, this set gives an underestimated system of linear and nonlinear equalities, depending on the chosen transition type for discrete and continuous transition (compare also Table 3.3). Assume that the solution space for the coefficients is depending on parameters $\underline{\lambda} \in \mathbb{R}^k$, where k is the dimension of the solution space. By construction, we know that $\forall \underline{\lambda} \in \mathbb{R}^k \forall\, locations\, l : \eta(l)(\underline{\lambda}) = 0$ is an invariant. Here, $\eta(l)(\underline{\lambda})$ denotes that the coefficients of $\eta(l)$ depend on the vector $\underline{\lambda}$. In [SSM04, San05], it is proposed to instantiate the parameters $\underline{\lambda}$ by $(1, ..., 1)$. Since η is an inductive assertion map for all $\underline{\lambda} \in \mathbb{R}^k$, this obviously leads to a feasible inductive assertion map. However, with this approach, a lot of information is lost. Since without loosing the property that η is an inductive assertion map, the value of the parameters $\underline{\lambda}$ can be given any value. This information helps to determine - perhaps non-inductive - assertions from the original parameterized one.

Algorithm 3: InductiveAssertionMap

input : $H = \langle L, E, X, Init, Inv, F, G, R\rangle$, $cond \in \{LC, CV, CS, PS\}$, m
output: inductive assertions satisfying *cond*

begin
  $H := \texttt{Homogenize}(H)$
  $S := \emptyset$
  foreach $l \in L, e \in E$ **do**
    $p(l) := 0$; $p(e) := 0$;
  for $d = 0$ *to* m **do**
    $S' := \emptyset$
    foreach $l \in L$ **do**
      Set coefficients of η of degree d satisfying conditions of Theorem 3.7 equal to zero
    $S' := S' \cup \{$*coefficient equations of* $\texttt{NF}_{Init \cup Inv(l_0)}(\eta(l_0))\}$
    foreach $l \in L$ **do**
      $p(l) := p(l) + \texttt{NF}_{Inv(l) \cup \dot{Inv}(l)}(\dot{\eta}^d(l))$
      $S := S \cup \{$coefficient equations of p(l) of degree $\leq d-1\}$
      $p(l) := p(l)_{\geq d}$
      $S' := S' \cup \{$coefficient equations of$p(l)\}$
    foreach $(l_1, l_2) = e \in E$ **do**
      $p(e) := p(e) + \texttt{NF}_{Inv(l_1) \cup Inv(l_2) \cup G(e)}(\eta^d(l_2)')$
      $S := S \cup \{$coefficient equations of $p(\tau)$of degree $< (d+1) \cdot k_e\}$
    Solve S as far as possible
    if S *unsolvable* **then**
      return *No Solution for degree> d possible*
    Solve $S \cup S'$
    if $S \cup S'$ *solvable* **then**
      update *Inv* and $\dot{(}Inv)$
    return *solution of* $S \cup S'$
end

Theorem 3.11 (Interpretation of Solutions)

Let η be an inductive assertion map and let l be a location. Let the coefficients $c_{l,\alpha}$ of $\eta(l)$, short c_α, be depending on the parameters $\underline{\lambda} \in \mathbb{R}^k$, i.e. $\forall \underline{\lambda} \in \mathbb{R}^k$: $\eta(l)(\underline{\lambda}) = 0$. Let $\eta(l)(\underline{\lambda}) = \sum_\alpha c_\alpha(\underline{\lambda}) \cdot \underline{x}^\alpha = \lambda_1 \cdot p_1 + p_2$, where $p_1 \in \mathbb{R}[\lambda_1, ..., \lambda_k][\underline{x}]$ and $p_2 \in \mathbb{R}[\lambda_2, ..., \lambda_k][\underline{x}]$, i.e. divide $\eta(l)$ in the part depending on λ_1 and the part not depending on λ_1. Then:

$$\forall \underline{\lambda} \in \mathbb{R}^k\text{: } p_1 = 0 \text{ and } p_2 = 0 \text{ are algebraic assertions of the location } l$$

Proof

For all $\lambda \in \mathbb{R}^k$, $\eta(l)(\underline{\lambda}) = 0$ is an algebraic assertion of l. Thus,

$$\forall (0, \lambda_2, ..., \lambda_k) \in \mathbb{R}^k : \eta(l)((0, \lambda_2, ..., \lambda_k)) = 0$$

is an algebraic assertion of l. So, also

$$0 = \eta(l)((0, \lambda_2, ..., \lambda_k)) = 0 \cdot p_1 + p_2 = p_2$$

is an algebraic assertion of l for all $\underline{\lambda} \in \mathbb{R}^k$. Thus,

$$0 = \eta(l)(\underline{\lambda}) = \lambda_1 \cdot p_1 + p_2 = \lambda_1 \cdot p_1 + 0 = \lambda_1 \cdot p_1$$

is an algebraic assertion of l for all $\underline{\lambda} \in \mathbb{R}^k$.

This yields the claim. q.e.d.

Obviously, Theorem 3.11 can be applied inductively on the smaller polynomials p_1 and p_2. The procedure can also be applied with any other parameter $\lambda_2, ..., \lambda_k$.

3.2.2 Algebraic Computation and Canonical Representation of Reachable States

A natural step would now be to model the images and preimages of arbitrary discrete and continuous transitions by Gröbner basis methods in order to apply model checking methods to these results. In this section, we will show that methods based on algebraic geometry are not usable to compute exact images / preimages of algebraic sets of states. Furthermore, it is also not possible to achieve both an under- and an overapproximation of the set of reachable states by discrete and continuous transitions. This motivates our exploration of methods based on abstractions (and three-valued logics) in the next chapters, for automated reasoning about hybrid automata

Let H be an algebraic hybrid automaton and assume throughout this section that the flow of the continuous variables in each location $l \in L$ (i.e. the solution of the ODEs involved) can be algebraically expressed. Our purpose in the Subsections 3.2.2.1 and 3.2.2.2, respectively, is that of studying the possibilities of using Gröbner bases as a canonical representation for the set of states which result in the execution of continuous and discrete transitions. Our results finally suggest that Gröbner bases are better suited to manipulate overapproximations rather than exact computations or underapproximations, when performing successively continuous and discrete transitions.

3.2.2.1 Continuous Transitions

When considering continuous transitions, the first task is that of determining the conditions ensuring that a continuous transition can be performed. We start showing that Gröbner bases can be used to give a canonical representation of the set of states admitting such continuous transitions.

Let $l \in L$ be a location with invariants $Inv(l)$ and let $\underline{\hat{x}}$ be a starting point for a feasible trajectory w.r.t. $Inv(l)$. $\underline{\hat{x}}$ is a starting point for a feasible trajectory, iff there exists a time $t_1 \in \mathbb{R}^+$ with $(l, \underline{\hat{x}}) \xrightarrow{\delta} (l, \underline{x}(\underline{\hat{x}}, t))$ in the associated time abstract transition system T_H. Since the trajectory is feasible, all points reachable along the flow $\varphi(\underline{\hat{x}}, t)$, $t \in [0, t_1]$ have to be feasible, i.e. satisfy the location invariants $Inv(l)$. In other words:

$$\forall t \in [0, t_1] \forall p_i \in Inv(l) : (\mathbb{R}[\underline{\hat{x}}])[t] \ni p_i(\varphi(\underline{\hat{x}}, t)) = 0 \tag{3.3}$$

Expression 3.3 above can be rewritten by Equation 3.4 below, since any polynomial in $\mathbb{R}$ having infinitely many zeros is the zero-polynomial.

$$\forall t \in \mathbb{R} \forall p_i \in Inv(l) : p_i(\varphi(\underline{\hat{x}}, t)) = 0 \tag{3.4}$$

By Theorem 3.2 a polynomial over $\mathbb{R}$ is the zero-polynomial if and only if all of its coefficients are zero. This means that for all $p_i \in Inv(l)$ every coefficient of $p(\varphi(\underline{\hat{x}}, t)) := \sum_{j=1}^{k_i} c_{i,j}(\underline{\hat{x}}) t^j$ has to be zero. Summarizing these facts yields the following Lemma, giving an algebraic characterization of the set of states admitting a continuous transition:

Lemma 3.12

Let $l \in L$ be a location with location invariants $Inv(l) = \{p_1 = 0, ..., p_{k_l} = 0\}$ with $p_i(\underline{x}, t) = \sum_j c_{i,j} t^j \in \mathbb{R}[\underline{x}][t]$ with $c_{i,j}$ depending on $\underline{x}$. Then, the following are equivalent:

1. *$\underline{x} \in \mathbb{R}^n$ is the starting point of a feasible continuous transition*
2. *$\underline{x} \in V_l = \{\underline{x} \in \mathbb{R}^n \forall i, j : \; c_{i,j}(\underline{x}) = 0\}$*

Proof

1. $\Rightarrow$ 2.: shown above

2. $\Rightarrow$ 1.:

Assume $\underline{\hat{x}} \in V_l = \{\underline{x} \in \mathbb{R}^n : c_{i,j}(\underline{x}) = 0 \; for \; all \; i, j\}$. By Theorem 3.2 the trajectory $\varphi(\underline{\hat{x}}, t)$ fulfills the condition

$$\forall t \in \mathbb{R} \forall p_i \in Inv(l) : p_i(\varphi(\underline{\hat{x}}, t)) = 0$$

which yields the claim. q.e.d.

Let $l \in L$ be a location. The next step now is that of studying the possibility of using algebraic methods to determine the continuous images of an algebraic set. We show, that only overapproximations are achievable in this context.

Consider an algebraic set given by $V = \{q_1(\underline{x}) = 0, ..., q_k(\underline{x}) = 0\}$. By Lemma 3.12 $V \cap V_l$ describes exactly the subset of V, being starting points of a continuous trajectory. Then the continuous images of V can be described by

$$ContIm(V) = \{\hat{\underline{x}} \in \mathbb{R}^n \mid \exists t \in \mathbb{R}^- : \varphi(\hat{\underline{x}}, t) \in V \cap V_l\}$$

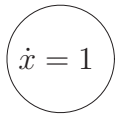
which can be calculated by projection. However, computing projections yield only the algebraic closure of the resulting set. Let $V \cap V_l = p_1(\underline{x}) = p_m(\underline{x}) = 0$. Then,

$$\overline{ContIm(V)} = \langle p_1(\underline{x}, t), ..., p_m(\underline{x}, t)\rangle \cap \mathbb{R}[\underline{x}]$$

not only characterizes the continuous images of V, but at the same time the preimages of that set! The following example illustrates this problem.

Example 3.3

Consider the automaton shown in Fig. 3.5. Here, the starting point $\hat{x} = 0$ leads to the continuous images $\mathbb{R}^+$. However, this is not an algebraic set, so the resulting ideal would be $\overline{\mathbb{R}^+} = \mathbb{R} = V(\langle 0\rangle)$.



$$\{0\} \mapsto \mathbb{R}^+$$
$$I = \langle x\rangle \mapsto I' = \langle 0\rangle$$

Figure 3.5: Images of a continuous transition starting in $\hat{\underline{x}} = 0$

3.2.2.2 Images and Preimages of Discrete Transitions

Let $(l, l') = e \in E$ be a discrete transition and let $(l, \underline{x}) \xrightarrow{e} (l', \underline{x}')$ be feasible w.r.t. the location invariants and the guard of e. Then, $\underline{x}$ satisfies $Inv(l)$ and $G(e)$ while $\underline{x}'$ has to satisfy $Inv(l')$.

Let us first consider the *preimages* of an algebraic set V. The set of discrete preimages of this set is determined by

$$DiscPreIm(V) = \{\underline{x} \in \mathbb{R}^n \mid \exists \underline{x}' \in \mathbb{R}^n : \underline{x}' = r(\underline{x}) \wedge \underline{x} \in Inv(l) \wedge \underline{x}' \in Inv(l') \wedge \underline{x} \in G(e)\}$$

which can be computed by projection:

$$\overline{DiscPreIm(V)} = V(\langle V', Inv(l), Inv'(l'), G(e), x'_1 - r_1(\underline{x}), ..., x'_n - r_n(\underline{x})\rangle \cap \mathbb{R}[\underline{x}])$$

However, because of the special form of the reset functions given by $x'_i = r_i(\underline{x})$ corresponding to $x'_i - r_i(\underline{x})$ in this case $DiscPreIm(V)$ coincides with its algebraic closure, i.e. $DiscPreIm(V) = \overline{DiscPreIm(V)}$. Thus,

$$DiscPreIm(V) = V(\langle V', Inv(l), Inv'(l'), G(e), x'_1 - r_1(\underline{x}), ..., x'_n - r_n(\underline{x})\rangle \cap \mathbb{R}[\underline{x}])$$

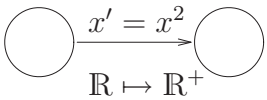
Let us now consider the *images* of the algebraic set V by the discrete transition. Analogously to the preimages of an algebraic set this set is derived by

$$\overline{DiscIm(V)} = V(\langle V, Inv(l), Inv'(l'), G(e), x'_1 - r_1(\underline{x}), ..., x'_n - r_n(\underline{x})\rangle \cap \mathbb{R}[\underline{x}'])$$

However, in this case the projection onto $\mathbb{R}[\underline{x}']$ really gives an overapproximation of $DiscIm(V)$. This is due to the fact that in general the form of the equations $x'_i = r(\underline{x})$ cannot be changed such that the variables x_i are represented only in the term of the variables x'_i, i.e. in general there exists no $r'(\underline{x}')$ with $\underline{x} = r'(\underline{x}')$. The following example will illustrate that problem.

Example 3.4

Consider the automaton depicted in Fig. 3.6. Here the image of $\mathbb{R}$ *is given by* $\mathbb{R}^+$*. Again, the algebraic closure of* $\mathbb{R}^+$ *is given by* $\overline{\mathbb{R}^+} = V(\langle 0 \rangle)$



$$\mathbb{R} \mapsto \mathbb{R}^+$$
$$I = \langle 0 \rangle \mapsto I' = \langle 0 \rangle$$

Figure 3.6: Images of a Discrete Transition starting in $\mathbb{R}$

3.2.2.3 Summary

To summarize, we have the following results for the images and preimages of discrete and continuous transitions in an algebraic framework:

- Continuous Transitions: Images and preimages of continuous transition are always encoded in the same ideal.
- Discrete Transitions: There exists an exact computation of preimages of an algebraic set, whereas only an overapproximation of the images of discrete transitions can be computed.

Thus, methods from algebraic geometry neither yield exact computations of images / preimages of discrete and continuous transitions nor do they determine both reasonable over- and underapproximations of these sets. Combining the analysis of over- and underapproximated reachable regions is particularly important for reasonably expressive and yet undecidable hybrid automata. For these reasons, in the next chapter of this thesis, we explore abstraction based methods (and three-valued logics) in order to achieve frameworks for the automated reasoning about (undecidable) hybrid systems in which both over- and underapproximated (reachability) analysis is possible.

Chapter 4

Abstractions for three-valued model-checking

Abstracting the dynamics of a hybrid automaton to a finite discrete transition system is a natural and widely used tool in the context of hybrid automata. To date, most abstraction frameworks in the literature are related to the issue of recovering overapproximated reachable sets (due to the undecidability of the reachability problem for most hybrid automata). The analysis of over-approximated dynamics for a hybrid automaton is sufficient to *certify* a given hybrid system as safe, i.e. non-capable of reaching some bad condition. If some underapproximation for the hybrid dynamics is also recognizable in the abstraction, then the task of creating counterexamples to safety can adjointly be accomplished.

The issue of designing abstractions for mixed over-/ underapproximated reachability is quite new. In [GSM07], the notion of discrete bounded bisimulation was designed for this purpose. As shown in Subsection 4.2.2, the same target can be achieved by so-called may/must-abstractions for hybrid automata over/underapproximated analysis of reachability that we define building up on some ideas in the context of discrete systems.

The abstractions reviewed/defined in this chapter with additional information for covering over- and underapproximated reachability analysis will be reused in Chapter 5 to specialize the semantics for three-valued μ-calculus model checking.

The chapter is organized as follows. In Section 4.1, we introduce the notion of abstractions for hybrid automata as given in the literature based on the simulation preorder of Definition 2.4. In Section 4.2, we present a set of abstraction frameworks augmented with suitable information to accomplish underapproximated and overapproximated analysis.

4.1 Abstractions

In the literature, the definitions of abstractions vary slightly, but the most important component that abstractions simulate the underlying hybrid automaton and can therefore model any behavior of the hybrid automaton, is an invariant of all definitions. Here, abstractions will be defined as the follows:

Definition 4.1 (Abstraction)

Let H be a hybrid automaton. An abstraction of H is a finite transition system $A = \langle R, R_0 \xrightarrow{\delta}, \xrightarrow{e} \rangle$ in which

1. *R is a finite partition of the state space of H, $R_0 \subseteq R$ is a partition of the initial states, $\xrightarrow{\delta} \subseteq R \times R$ and $\xrightarrow{e} \subseteq R \times R$*
2. *$A^* := \langle R, R_0 \xrightarrow{\delta^*}, \xrightarrow{e} \rangle$ simulates the time abstract transition system T_H associated to H, where $\xrightarrow{\delta^*}$ denotes the transitive closure of the continuous transitions $\xrightarrow{\delta}$*

In abstractions, the states $r \in R$ are also called regions, since they always resemble a (possibly infinite) set of states of the underlying time abstract transition system T_H and thus of the hybrid automaton H. Since every state $x \in Q$ is associated to exactly one region $r \in R$, this region is defined by $r(x)$.

The simulation relation between A^* and the underlying hybrid automaton H ensures that the behavior of the hybrid automaton is also present in the abstraction A. In particular, if it is possible to prove that no unsafe region can be reached in the abstraction, then the hybrid automaton is also guaranteed to be safe. The above property is formalized by the following lemma.

Lemma 4.1

Let $H = \langle L, E, X, Init, Inv, F, G, R \rangle$ be a hybrid automaton and $A = \langle R, R_0 \xrightarrow{\delta}, \xrightarrow{e} \rangle$ be an abstraction of H. Let B be a set of unsafe states and let R be consistent with respect to B, i.e. for all $r \in R : r \cap B = r \vee r \cap B = \emptyset$. Then: If an unsafe region r of the abstraction is not reachable by a path in A, then all $b \in r$ are not reachable by any run of the hybrid automaton H.

Proof

Assume that there exists a run $x_0 \rightsquigarrow b$ in H leading from an initial state to an unsafe state $b \in B$. Then, by definition of the time abstract transition system T_H there exists a run

$$x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} ... \xrightarrow{a_n} x_n = b$$

with $a_i \in \{\delta, e\}$ and $x_0 \in Q_0$. It suffices to show, that there exists an associated run

$$r(x_0) \overset{a_1}{\rightsquigarrow} r(x_1) \overset{a_2}{\rightsquigarrow} ... \overset{a_n}{\rightsquigarrow} r(x_n) = r(b)$$

with $\overset{a_i}{\rightsquigarrow} \in \{\xrightarrow{e}, \xrightarrow{\delta^*}\}$ in the abstraction A. Take therefore $x_i \xrightarrow{a_{i+1}} x_{i+1}$ with an arbitrary $i \in \{0, ..., n-1\}$.

<u>case 1:</u> $a = e$

By Definition 4.1, A^* simulates T_H, i.e. by part 3 of Definition 2.4 we have $r(x_i) \xrightarrow{e} r(x_{i+1})$

case 2: $a = \delta$

Since A^* simulates T_H, there exists a run

$$r(x_i) = r_0 \xrightarrow{\delta} ... \xrightarrow{\delta} r_{k_i} = r(x_{i+1}) = r(x_i) \xrightarrow{\delta^*} r(x_{i+1})$$

Thus, if an unsafe region r is not reachable in the abstraction, no state within r is reachable by a run in H. q.e.d.

4.2 Abstractions with Additional Information

In Lemma 4.1, it has been shown that non-reachability results obtained by model checking methods on the abstraction are preserved on the underlying hybrid automaton. However, for the truth value false this is not the case, since the simulation relation between the abstraction and the underlying hybrid automaton only guarantees an overapproximation of the behavior of the hybrid automaton. A way of dealing with that problem is to encode some additional information for underapproximated reachability. Combining the knowledge of over- and underapproximation of the hybrid automaton allows us to develop a three-valued verification tool for reachability, where both true and false answers are trustworthy. Naturally, due to the general undecidabililty of hybrid automata, one needs to admit the possibility that a third undefinite value is given in output for such a procedure.

In this section, we review / define two kinds of abstractions for over-/underapproximated reachability analysis on hybrid systems.

4.2.1 Discrete Bounded Bisimulation

Discrete bounded bisimulations (short DBBs) are introduced in [GSM07]. The main idea behind DBBs is the classical n-bounded bisimulation. In the discrete setting, two transition systems are n-bounded bisimulation related, if for each transition system paths with $\leq n$ transitions always have equivalent representatives in the other transition system. In the setting of hybrid automata and their associated time abstract transition systems however there is again the problem of continuous transitions $\xrightarrow{\delta}$, where there does not exist a clearly defined next point of time. Because of this reason in DBBs abstractions are constructed, where paths of n discrete transitions can be trusted in some way.

Definition 4.2 (Discrete Bounded Bisimulation)

Let H be a hybrid automaton and $T_H = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ be the time abstract transition system of H. Let P be a partition of Q and let $p, q \in Q$. Then the n-bounded bisimulations of T are recursively defined by:

1. *$\equiv_0 \in Q \times Q$ is the maximum relation on Q such that for all $p \equiv_0 q$*

   (a) *$[p]_P = [q]_P$ and $p \in Q_0$ iff $q \in Q_0$*

   (b) *$\forall p \xrightarrow{\delta} p' \exists q' : \; p' \equiv_0 q' \; \wedge \; q \xrightarrow{\delta} q'$*

   (c) *$\forall q \xrightarrow{\delta} p' \exists q' : \; p' \equiv_0 q' \; \wedge \; p \xrightarrow{\delta} p'$*

2. $\equiv_n \in Q \times Q$ *is the maximum relation on* Q *such that for all* $p \equiv_n q$

(a) $p \equiv_{n-1}$

(b) $\forall p \xrightarrow{\delta} p' \exists q' : p' \equiv_0 q' \wedge q \xrightarrow{\delta} q'$

(c) $\forall q \xrightarrow{\delta} p' \exists p' : p' \equiv_0 q' \wedge p \xrightarrow{\delta} p'$

(d) $\forall p \xrightarrow{e} p' \exists q' : p' \equiv_{n-1} q' \wedge q \xrightarrow{e} q'$

(e) $\forall q \xrightarrow{e} q' \exists p' : p' \equiv_{n-1} q' \wedge p \xrightarrow{e} p'$

For $n \in \mathbb{N}$ *the relation* $\equiv_n$ *will be called n-DBB equivalence.*

Definition 4.3 (Series of DBB Abstractions)

Let H *be a hybrid automaton and* $T_H = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ *the associated time abstract transition system. Let* P *be a partition of* Q *and consider the n-DBB equivalence* $\equiv_n$*. Then the n-DBB abstraction* $H_{\equiv_n} = \langle Q', Q'_0, l_{\rightarrow} \rightarrow' \rangle$ *is defined by*

- $Q' = Q_{/\equiv_n}$, $Q'_0 = Q_{0/\equiv_n}$
- $\forall \alpha, \beta \in Q' :$
  - $\alpha \xrightarrow{e} \beta$ *iff* $\exists a \in \alpha \exists b \in \beta : a \xrightarrow{e} b$
  - $\alpha \xrightarrow{\delta} \beta$ *iff* $\exists a \in \alpha \exists b \in \beta : a \xrightarrow{\delta} b$ *and the path* $a \rightsquigarrow b$ *only traverses* α *and* β

Lemma 4.2

Let H *be a hybrid automaton and let* $H_{\equiv_n}$ *be an n-DBB abstraction of* H*. Then, with considering the transitive closure of the continuous transition* $\xrightarrow{\delta^*}$ *the transition system* $H_{\equiv_n}$ *is a simulation of* H*.*

Proof

obvious q.e.d.

The above lemma ensures that DBBs encode an overapproximation of the behaviors of the original automaton. The fact that an under-approximated reachability analysis is also possible is guaranteed by the following lemma.

Lemma 4.3

Let p *and* q *be two states in the hybrid automaton* H *and let* $\equiv_n$ *be the n-DBB equivalence on* T_H *with respect to a partition* P*. If* $p \equiv_n q$*, then for all* $m \leq n$ *it holds:*

- *For all* p' *such that* $p \overset{m}{\rightsquigarrow} p'$ *there exists a* q' *such that* $p' \equiv_{n-m} q'$ *and* $q \xrightarrow{m} q'$*.*
- *For all* q' *such that* $q \overset{m}{\rightsquigarrow} q'$ *there exists a* p' *such that* $p' \equiv_{n-m} q'$ *and* $p \xrightarrow{m} p'$*.*

Proof

see [GSM07] q.e.d.

For the family of fully o-minimal hybrid automata, which is undecidable with respect to the reachability problem it has been shown, that the n-DBB abstractions have a finite state space and thus are because of Lemma 4.2 abstractions according to Definition 4.1. The algorithm constructs the n-th discrete bounded bisimulation recursively by the $(n-1)$th DBB.

Algorithm 4: nDBB

input : $H = \langle L, E, X, Init, Inv, F, G, R\rangle$, starting partition P_0
$n \in \mathbb{N}$

output: Partition of the state space to determine $H_{\equiv n}$

```
1  begin
2      P:= coarsest partition refining P_0 compatible with Q_0
3      while ∃B, B' ∈ P : ∅ ≠ B ∩ Pre_δ(B') ≠ B do
4          n := n − 1, P_old := P
5          forall (l, l') = e ∈ E do
6              forall B' ∈ P_old, B ∈ P : ∅ ≠ B ∩ Pre_e(B') ≠ B do
7                  B_1 := B ∩ Pre_e(B'), B_2 := B \ B_1
8                  P := (P \ {B}) ∪ {B_1, B_2}
9          while ∃B, B' ∈ P : ∅ ≠ B ∩ Pre_δ(B') ≠ B do
10             B_1 := B ∩ Pre_δ(B'), B_2 := B \ B_1
11             P := (P \ {B}) ∪ {B_1, B_2}
12     return P
13 end
```

Example 4.1

Consider the slightly modified heating system introduced in Fig. 4.1.

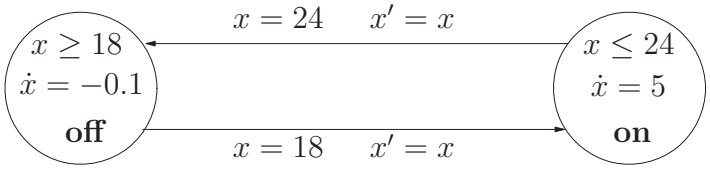


Figure 4.1: Fully o-minimal Hybrid Automaton modeling a Heating System

Let the initial partitioning be $P = \{(\textbf{\textit{off}}, (20, 24]), (\textbf{\textit{off}}, 20), (\textbf{\textit{off}}, [18, 20)), (\textbf{\textit{on}}, [18, 24])\}$. *Applying Alg. 4 this yields:*

- $n = 0$*:*
  $P_0 = \{(\textbf{\textit{off}}, (20, 24]), (\textbf{\textit{off}}, 20), (\textbf{\textit{off}}, (18, 20)), (\textbf{\textit{off}}, 18), (\textbf{\textit{on}}, [18, 24)), (\textbf{\textit{on}}, 24)\}$

- $n > 0$*:*
  $P_n = P_0$*, i.e. the abstraction shown in Fig. 4.2 is a bisimulation.*

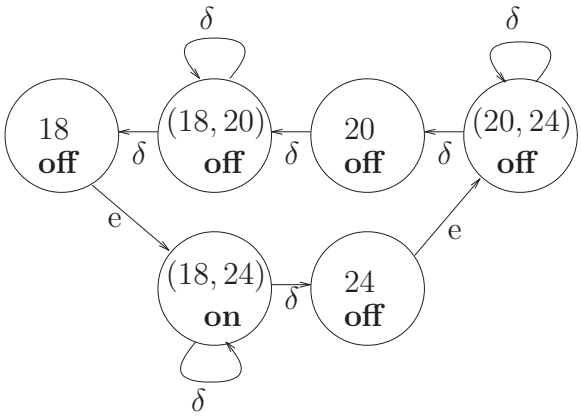


Figure 4.2: 0-DBB Abstraction of the Heating Controller of Fig. 4.1

4.2.2 Abstractions with May- and Must-Relations

In the context of abstractions for discrete systems [SG04], the notions of *may-* and *must-*transitions refers to the possibility that given two abstract classes A and B either some state in A may have a transition to a state in B or all states in A are the sources of an edge targeting an element of B. Naturally, may-edges (must-edges) refer to overapproximated (underapproximated) transitions among classes of an abstract system. The above ideas can be extended to the context of hybrid automata as formalized in Definition 4.4.

Definition 4.4 (May/Must Abstractions)
Let $A = \langle R, R_0, \xrightarrow{\delta}, \xrightarrow{e} \rangle$ be an abstraction of the hybrid automaton H. Then A is an abstraction with may/must relations, iff the following properties hold:

- $\xrightarrow{\delta} \supseteq \xrightarrow{\delta}_{must}$*, where* $\xrightarrow{\delta}_{must}$ *is defined as follows:*
  $r \xrightarrow{\delta}_{must} r'$ *iff for all* $x \in r$ *there exists an* $x' \in r'$ *such that there exists a continuous path* $x \rightsquigarrow x'$ *in the hybrid automaton* H *only traversing the regions* r *and* r'*.*

- $\xrightarrow{e} \supseteq \xrightarrow{e}_{must}$ *where* $\xrightarrow{e}_{must}$ *is defined as follows:*
  $r \xrightarrow{e}_{must} r'$ *iff for all* $x \in r$ *there exists an* $x' \in r'$ *s.t.* $x \xrightarrow{e} x'$ *in* H*.*

The subautomaton $\langle R, R_0, \xrightarrow{\delta}_{must}, \xrightarrow{e}_{must} \rangle$ of A consisting only must-transitions is called A_{must}.

.

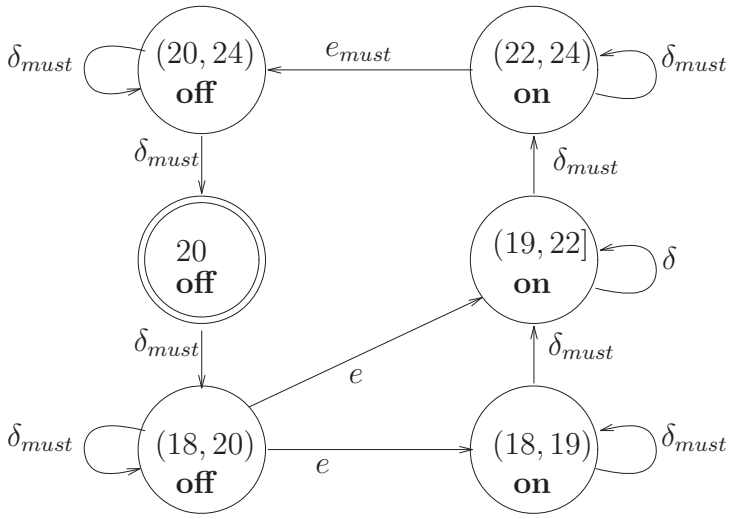


Figure 4.3: Abstraction A_1 with may/must of the heating controller

Example 4.2 (Heating Controller Continued)

For an example with may and must-transitions consider again the heating system introduced in Chapter 2 and its abstraction shown in Figure 5.1.

The definition of must-transitions ensures, that A_{must} is simulated by T_H, i.e. that every behavior in A_{must} can be modeled by T_H. The fact that may-/must abstractions for hybrid automata can be used not only to over-approximate but also to underapproximate the reachable states of H follows by Lemma 4.4 and Corollary 4.5 below.

Lemma 4.4

Let H be a hybrid automaton and let A be an abstraction of H with may/must relations. Then the subautomaton A_{must} of A is simulated by T_H, i.e. $A_{must} \leq_S T_H \leq_S A^*$.

Proof

Define the relation $\leq_S$ in the following way: $r \leq_S x :\Leftrightarrow r = r(x)$.
Then by definition of $\xrightarrow{a}_{must}$ with $a \in \{\delta, e\}$:

$$\begin{aligned} r \xrightarrow{a}_{must} r' &\Rightarrow \forall x \in r \exists x' \in r' : x \xrightarrow{a} x' \\ &\Rightarrow \forall r \geq_S x \exists x' \geq_S r' : x \xrightarrow{a} x' \end{aligned}$$

This yields the claim. q.e.d.

Corollary 4.5

Let $H = \langle L, E, X, Init, Inv, F, G, R\rangle$ be a hybrid automaton and $A = \langle R, R_0 \xrightarrow{\delta}, \xrightarrow{e}\rangle$ be an abstraction of H. Let B be a set of unsafe states and let R be consistent with respect to B, i.e. for all $r \in R : r \cap B = r \vee r \cap B = \emptyset$. Then: If an unsafe region $r \subseteq B$ is reachable in the abstraction A_{must}, then an unsafe state $b \in B$ is reachable by a run of the hybrid automaton H.

Proof

Let $r_0 \rightsquigarrow_{must} r$ be a path in A_{must} leading from an initial region to the unsafe region $r \subseteq B$. Then

$$r_0 \xrightarrow{a_1}_{must} r_1 \xrightarrow{a_2}_{must} ... \xrightarrow{a_n}_{must} r_n = r$$

with $a_i \in \{\delta, e\}$ and $r_0 \in R_0$. It suffices to show, that there exists an associated run

$$x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} ... \xrightarrow{a_n} x_n = b$$

with $a_i \in \{e, \delta\}$ and $x_i \in r_i$ in the time abstract transition system T_H. By Lemma 4.4 A_{must} is simulated by T_H. Thus for any $i \in \{1, .., n\}$ and for all $x \in r_i$ there exists an $x' \in r_{i+1}$ with $x \xrightarrow{a_{i+1}} x'$. Thus for any starting point $x_0 \in r_0$ there exists a run

$$x_0 \xrightarrow{a_1} x_1 \xrightarrow{a_2} ... \xrightarrow{a_n} x_n = b$$

This yields the claim. q.e.d.

Chapter 5

Three-valued μ-Calculus Model-Checking on Hybrid Automata

Our next aim is to extend the theory of μ-calculus for hybrid automata to abstractions of hybrid automata to obtain efficient algorithms for model checking purposes. We cannot hope to obtain general preservation results from the abstraction to the hybrid automaton in a two-valued framework, since the simple reachability problem is for even for simple hybrid automata undecidable.

Furthermore, since for abstractions each region usually contains infinitely many states of the original hybrid automaton, it cannot be guaranteed that all states within an region have the same truth value with respect to some μ-calculus formula ϕ. Thus, such a region can neither be endowed with the truth value true nor with the truth value false.

One way to avoid this problem is to extend the original set of truth-values $\{0, 1\}$ to a set of three truth-values $\{0, \bot, 1\}$, where $\bot$ has both the meaning of 'not enough information for a decision' and 'the region contains points of both original truth values'. This extension is theoretically possible since the Tarski-Knaster fixpoint theorem and thus the foundations of μ-calculus is applicable: Define therefore a total order on the truth values $\{0, \bot, 1\}$ by $0 \sqsubseteq \bot \sqsubseteq 1$ and extend it to arbitrary functions $f : R \to \{0, \bot, 1\}$ by pointwise comparison: $f \sqsubseteq g :\Leftrightarrow \forall\, r \in R :\ f(r) \leq g(r)$ Together with the minimal element $f_{min}(r) := 0$ for every $r \in R$ and $f_{max}(r) := 1$ for all $r \in R$ the functions $f : R \to \{0, \bot, 1\}$ form a complete lattice. Thus as in the two-valued case the Tarski-Knaster fixpoint theorem applies. The application of the Tarski-Knaster fixpoint theorem also analogously applies with any other total ordering on the truth values $\{0, \bot, 1\}$.

In Section 4.2 we introduced different abstraction frameworks, mainly basic simulation-based abstractions and suitable enhancements to accomplish not only overapproximated but also underapproximated reachability analysis. Beyond reachability, basic simulation based abstractions are known to preserve only universally quantified μ-calculus formulas.

In principle, abstractions with additional information encoding also fragments of exact trajectories of the original hybrid automaton could be employed to recover also existentially quantified μ-calculus formulas. However, suitable three-valued semantics for this purpose should be defined on such enriched abstractions.

In this chapter, we will define preservative three-valued μ-calculus semantics for DBB and may/must abstractions proceeding in two steps: First, in Section 5.1, we will introduce a parametrized semantic framework, where preservation results for general μ-calculus formulas depend only on the concrete instantiation of the modal operators. Section 5.2 and Section 5.3 specialize the previous semantics on DBB abstractions and abstractions with may/must.

5.1 General Framework and Preservation Results

Let H be a hybrid automaton with associated time abstract transition system $T_H = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$. Let $l_{AP} : Q \rightarrow 2^{AP}$ be a labeling function that assigns each state $q \in Q$ a set of true atomic propositions of AP. Assume from now on that given abstractions for hybrid automata are consistent w.r.t. the labeling function l_{AP}:

- $\forall r \in R \forall x_1, x_2 \in R : \; l_{AP}(x_1) = l_{AP}(x_2)$

i.e. in any region r the points of r always satisfy the same atomic propositions. In this case it is easy to extend the labeling function in a natural way to $l_{AP} : R \rightarrow 2^{AP}$. Given the above assumptions, Definition 5.1 introduces a general three-valued scheme for μ-calculus on abstractions of hybrid automata, where all operators but modal ones are interpreted independently from the properties of the underlying abstractions.

Definition 5.1 (μ-Calculus Semantics of Abstractions)

Let H be a hybrid automaton, $A = \langle R, R_0, l_{\rightarrow}, \rightarrow \rangle$ be an abstraction of H and let ϕ and ψ be formulaes of μ-calculus. Then we define for every μ-calculus formula ϕ a function $[\![\phi]\!] : R \rightarrow \{0, \perp, 1\}$:

1. *Let ϕ be an atomic proposition. Then:*
   $[\![\phi]\!](r) = 1$ *iff* $\phi \in l_{AP}(r)$
   $[\![\phi]\!](r) = 0$ *iff* $\phi \notin l_{AP}(r)$
2. $[\![\neg\phi]\!] = \neg_3 [\![\phi]\!]$
3. $[\![\phi \vee \psi]\!] = [\![\phi]\!] \vee_3 [\![\psi]\!]$
4. $[\![\phi \wedge \psi]\!] = [\![\phi]\!] \wedge_3 [\![\psi]\!]$
5. $[\![\bigstar]\!]$ *with* $\bigstar \in \{\langle\delta\rangle\phi, \langle e\rangle\phi, [\delta]\phi, [e]\phi, E(\phi \underline{U} \psi), A(\phi \underline{U} \psi)\}$
   are defined according to the properties of the underlying abstraction. The definitions are required to fulfill the following conditions for $\bigstar$:
   $[\![\bigstar]\!](r) = 1 \Rightarrow \; \forall\, x \in r : \; [\![\bigstar]\!]_H(x) = 1$
   $[\![\bigstar]\!](r) = 0 \Rightarrow \; \forall\, x \in r : \; [\![\bigstar]\!]_H(x) = 0$
   where $[\![\bigstar]\!]_H$ refers to the semantics of μ-calculus defined on hybrid automata.

6. *The fixpoint operators are defined in the following way:*
   Let $[\phi]_Z^\psi$ *be the formula obtained by replacing all occurrences of* Z *with* ψ.
   Given a fixpoint formula $\sigma Z.\phi$ *with* $\sigma \in \{\mu, \nu\}$ *its k-th approximation* $apx_k(\sigma Z.\phi)$ *is recursively defined as follows:*
   $apx_0(\mu Z.\phi) := 0 \quad and \quad apx_{k+1}(\mu Z.\phi) := [\phi]_Z^{apx_k(\mu Z.\phi)}$
   $apx_0(\nu Z.\phi) := 1 \quad and \quad apx_{k+1}(\nu Z.\phi) := [\phi]_Z^{apx_k(\nu Z.\phi)}$
   Then least and greatest fixpoints $[\sigma Z.\phi]$ *are defined by* $[\![apx_{\hat{k}}\sigma Z.\phi]\!]$ *where* $\hat{k}$ *is the smallest index for which it holds* $[\![apx_{\hat{k}}(\sigma Z.\phi)]\!] = [\![apx_{\hat{k}+1}(\sigma Z.\phi)]\!]$.
   $\mu Z.\phi$ *is called smallest fixpoint.*
   $\nu Z.\phi$ *is called greatest fixpoint.*

A is a model for ϕ *iff* $\forall r \in R_0 : [\![\phi]\!](r) = 1$. *A is no model for* ϕ *iff* $\exists r \in R_0 : [\![\phi]\!](r) = 0$. *Short:*

- $A \vDash \phi :\Leftrightarrow [\![\phi]\!](R_0) = 1$
- $A \nvDash \phi :\Leftrightarrow \exists\, r \in R_0 : [\![\phi]\!](r) = 0$

The fixpoint operations are well-defined, because on the one hand the Tarski-Knaster theorem applies (see the introduction to this section) and on the other hand the fixpoint iteration stops after a finite number of iteration steps, since the state space R of A is required to be finite and thus unlike to the infinite state systems, a fixpoint has to be reached after a finite number of iterations. Thus, it is possible to replace the infinite union / intersection in the way it has been done in Definition 5.1.

The next aim now is to prove, that any result of the form $A \vDash \phi$ or $A \nvDash \phi$ is preserved when looking at the original hybrid automaton. This will be done in two steps. In the first step (Lemma 5.1) this claim will be shown for any fixpoint-free formula. In a second step (Theorem 5.3), this result will be extended to arbitrary μ-calculus formulas. The proofs will be done by structural induction over the structure of μ-calculus formulas and in case of the fixpoint operators in addition via induction over the natural numbers.

Lemma 5.1

Let H *be a hybrid automaton and* A *be an abstraction of* H. *If the semantics of three-valued* μ*-calculus on* A *is defined according to Definition 5.1, then for any fixpoint-free formula* ϕ, *it holds:*

- $[\![\phi]\!](r) = 1 \Rightarrow \forall\, x \in r : [\![\phi]\!]_H(x) = 1$
- $[\![\phi]\!](r) = 0 \Rightarrow \forall\, x \in r : [\![\phi]\!]_H(x) = 0$

Proof (By structural induction)

Let $v \in \{0, 1\}$

1. Let ϕ be an atomic proposition. Then by the assumption:
   $[\![\phi]\!](r) = 1 \Rightarrow \forall\, x \in r : [\![\phi]\!]_H(x) = 1$
   $[\![\phi]\!](r) = 0 \Rightarrow \forall\, x \in r : [\![\phi]\!]_H(x) = 0$

2. $[\![\neg\phi]\!](r) = v \Rightarrow \forall\, x \in r : [\![\neg\phi]\!]_H(x) = v$:

$$\begin{aligned}
[\![\neg\phi]\!](r) = 1 &\Rightarrow [\![\phi]\!](r) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\neg\phi]\!]_H(x) = 1 \\
[\![\neg\phi]\!](r) = 0 &\Rightarrow [\![\phi]\!](r) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\neg\phi]\!]_H(x) = 0
\end{aligned}$$

3. $[\![\phi \vee \psi]\!](r) = v \Rightarrow \forall\, x \in r :\ [\![\phi \vee \psi]\!]_H(x) = v$:

$$\begin{aligned}
[\![\phi \vee \psi]\!](r) = 1 &\Rightarrow [\![\phi]\!](r) = 1 \ \vee\ [\![\psi]\!](r) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 1 \ \vee\ [\![\psi]\!]_H(x) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi \vee \psi]\!]_H(x) = 1 \\
[\![\phi \vee \psi]\!](r) = 0 &\Rightarrow [\![\phi]\!](r) = 0 \ \wedge\ [\![\psi]\!](r) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 0 \ \wedge\ [\![\psi]\!]_H(x) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi \vee \psi]\!]_H(x) = 0
\end{aligned}$$

4. $[\![\phi \wedge \psi]\!](r) = v \Rightarrow \forall\, x \in r :\ [\![\phi \wedge \psi]\!]_H(x) = v$:

$$\begin{aligned}
[\![\phi \wedge \psi]\!](r) = 1 &\Rightarrow [\![\phi]\!](r) = 1 \ \wedge\ [\![\psi]\!](r) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 1 \ \wedge\ [\![\psi]\!]_H(x) = 1 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi \wedge \psi]\!]_H(x) = 1 \\
[\![\phi \wedge \psi]\!](r) = 0 &\Rightarrow [\![\phi]\!](r) = 0 \ \vee\ [\![\psi]\!](r) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 0 \ \vee\ [\![\psi]\!]_H(x) = 0 \\
&\Rightarrow \forall\, x \in r :\ [\![\phi \wedge \psi]\!]_H(x) = 0
\end{aligned}$$

5. By definition of $\bigstar$ in Definition 5.1 it holds:
   $[\![\bigstar]\!](r) = 1 \Rightarrow \forall\, x \in r :\ [\![\bigstar]\!]_H(x) = 1$
   $[\![\bigstar]\!](r) = 0 \Rightarrow \forall\, x \in r :\ [\![\bigstar]\!]_H(x) = 0$ q.e.d.

Lemma 5.2 is just an equivalent statement to the statement in Lemma 5.1 but it will be needed in this particular form to show the properties also for fixpoint formulas in the following Theorem 5.3.

Lemma 5.2

Let H be a hybrid automaton and A be an abstraction H. Let the semantics of the three-valued μ-calculus be as in Definition 5.1. Then for any fixpoint-free formula ϕ it holds:

- $[\![\phi]\!]_H(x) = 1 \Rightarrow [\![\phi]\!](r(x)) \in \{\bot, 1\}$
- $[\![\phi]\!]_H(x) = 1 \Rightarrow [\![\phi]\!](r(x)) \in \{0, \bot\}$

Proof (By contradiction)

Assume: $[\![\phi]\!](\hat{x}) = 1$ but $[\![\phi]\!](r(\hat{x})) = 0$
By Lemma 5.1, it is shown that

$$[\![\phi]\!](r) = 0 \Rightarrow \forall\, x \in r :\ [\![\phi]\!](x) = 0$$

Thus especially $[\![\phi]\!](\hat{x}) = 0$, which is a contradiction to the assumption.
The other case is analogous. q.e.d.

We are now ready to prove the main theorem leading to the preservation results of Corollary 5.4.

Theorem 5.3

Let H be a hybrid automaton, A be an abstraction of H and let the semantics of three-valued μ-calculus be as in definition 5.1. Then for any μ-calculus formula ϕ:

- $[\![\phi]\!](r) = 1 \Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 1$
- $[\![\phi]\!](r) = 0 \Rightarrow \forall\, x \in r :\ [\![\phi]\!]_H(x) = 0$

Proof (By structural induction)

The main idea of this proof is the fact that for an arbitrary but fixed abstraction, a fixpoint formula $\sigma Z.\phi$ with $\sigma \in \{\mu, \nu\}$ is equivalent to a fixpoint-free formula, namely to the formula $apx_{\hat{k}}(\sigma Z.\phi)$ with $\hat{k} \in \mathbb{N}$ being smallest such that $[\![apx_{\hat{k}}(\sigma Z.\phi)]\!] = [\![apx_{\hat{k}+1}(\sigma Z.\phi)]\!]$. Thus it can be represented by a fixpoint-free formula for which the validity already has been shown in Lemma 5.1.

For fixpointfree formulas the theorem has already been shown in Lemma 5.1. So it suffices to show the claim for the fixpoint operators $\mu Z.\phi$ and $\nu Z.\phi$.
Let $T_H = \langle Q, Q_0, l_{\rightarrow}, \rightarrow \rangle$ be the time abstract transition system of H.
Let now $\phi = \sigma Z.\psi$ with $\sigma \in \{\mu, \nu\}$. Let k be the smallest index such that for the abstraction

$$[\![\sigma Z.\phi]\!] = [\![apx_k(\sigma Z.\phi)]\!]$$

Case 1: $\sigma = \mu$
Assume that there exists an $x \in Q$ such that

$$1 = [\![\mu Z.\phi]\!]_H(x) = \bigvee_{n \in \mathbb{N}} [\![apx_n(\mu Z.\phi)]\!]_H(x)$$

but $[\![\mu Z.\phi]\!](r) \notin \{1, \perp\}$. Since $[\![\mu Z.\phi]\!]_H(x) = 1$ there exists a smallest index $n \in \mathbb{N}$ s.t. $[\![apx_n(\mu Z.\phi)]\!]_H(x) = 1$. Thus by Lemma 5.2

$$[\![apx_n(\mu Z.\phi)]\!](r(x)) \in \{1, \perp\}.$$

For $n \geq k$ by definition of k

$$[\![apx_n(\mu Z.\phi)]\!](r(x)) = [\![apx_k(\mu Z.\phi)]\!](r(x)) = [\![\mu Z.\phi]\!](r(x)) \in \{1, \perp\}$$

and for $n < k$ by monotonicity of the construction of the fixpoint

$$\{1, \perp\} \ni [\![apx_n(\mu Z.\phi)]\!](r(x)) \leq [\![apx_k(\mu Z.\phi)]\!](r(x)) = [\![\mu Z.\phi]\!](r(x))$$

with the ordering $0 <\perp< 1$.
Assume now that there exists an x in the state space of H such that $[\![\mu Z.\phi]\!](r(x)) = 0$ but $[\![\mu Z.\phi]\!]_H(x) = 1$. Let $\hat{n}$ be the smallest index s.t.
$[\![apx_{\hat{n}}(\mu Z.\phi)]\!]_H(x) = 1$ and let $n = max\{k, \hat{n}\}$. Then by Lemma 5.2:

$$[\![apx_n(\mu Z.\phi)]\!]_H(x) = 1 \Rightarrow [\![apx_n(\mu Z.\phi)]\!](r(x)) \in \{1, \perp\}$$

This is a contradiction to the assumption. Thus the theorem holds for the smallest fixpoint.

Case 2: $\sigma = \nu$
analogously q.e.d.

Corollary 5.4 (Preservation)

Let H be a hybrid automaton and A be an abstraction of H. Then for any μ-calculus formula ϕ:

- $A \models \phi \quad \Rightarrow \quad H \models \phi$
- $A \not\models \phi \quad \Rightarrow \quad H \not\models \phi$

Proof

Direct consequence of Theorem 5.3. q.e.d.

The next aim now is to determine the cases in which the three-valued semantics of the U-operator on the abstractions can be given in terms of the other modal operators maintaining the preservation results. To this purpose recall that in the continuous setting the U-operator cannot be translated to an equivalent μ-calculus formula without the usage of the temporal operator U.

The main problem has already been shown in example 2.3, where a $\xrightarrow{\delta}$-successor can fulfill some desired property but some points during the continuous evolution towards this point do not fulfill this property. This problem can be dealt with by the notion of

direct successors for the continuous evolution. In the setting of an abstraction one has a direct successor, when the continuous evolution $x \rightsquigarrow x'$ only traverses the regions $r(x)$ and $r(x')$. In this case the counterexamples do not work any more and it can be shown that the $\underline{\mathrm{U}}$-operator can be modeled in the same way by fixpoint operations as in the setting of discrete transition systems.

In the following, let therefore A be an abstraction that satisfies the following conditions for the next operators $\langle\delta\rangle$ and $[\delta]$:

- $[\![\langle\delta\rangle\phi]\!](r) = 1$ iff for all $x \in r$ there exists an x' s.t. the continuous path $x \rightsquigarrow x'$ only traverses the regions of x and x' and $[\![\phi]\!](r') = 1$, i.e. all $x \in r$ have a direct successor satisfying the condition ϕ.
- $[\![[\delta]\phi]\!](r) = 0$ iff $[\![\neg(\langle\delta\rangle\neg\phi)]\!](r) = 1$, i.e. iff all $x \in r$ have a direct successor not satisfying the condition ϕ.

If these conditions are fulfilled, define analogously to the μ-calculus in discrete frameworks the next-operators $\Box$ and $\Diamond$ by

- $\Diamond\phi := \langle e\rangle\phi \vee \langle\delta\rangle\phi$
  By the conditions above a region r fulfills the condition $\Diamond\phi$ if and only if for all $x \in r$ there exists a direct successor x' satisfying ϕ
- $\Box\phi := [e]\phi \wedge [\delta]\phi$

Then, the $\underline{\mathrm{U}}$-operator is redundant:

Theorem 5.5

Let H be a hybrid automaton and A be an abstraction of H satisfying the conditions described above. Then for any μ-calculus formulas ϕ and ψ:

1. $A \vDash \mu Z.\psi \vee \phi \wedge \Diamond Z \Leftrightarrow H \vDash E(\phi \underline{U} \psi)$
   $A \nvDash \mu Z.\psi \vee \phi \wedge \Diamond Z \Leftrightarrow H \nvDash E(\phi \underline{U} \psi)$
2. $A \vDash \mu Z.\psi \vee \phi \wedge \Box Z \Leftrightarrow H \vDash A(\phi \underline{U} \psi)$
   $A \nvDash \mu Z.\psi \vee \phi \wedge \Box Z \Leftrightarrow H \nvDash A(\phi \underline{U} \psi)$

Proof

to 1.:

It suffices to show that:

- $[\![\mu Z.\psi \vee \phi \wedge \Diamond Z]\!](r) = 1 \Rightarrow \forall x \in r : [\![E(\phi \underline{\mathrm{U}} \psi)]\!]_H(x) = 1$
- $[\![\mu Z.\psi \vee \phi \wedge \Diamond Z]\!](r) = 0 \Rightarrow \forall x \in r : [\![E(\phi \underline{\mathrm{U}} \psi)]\!]_H(x) = 0$

Let $[\![apx_k(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r) = 1$. If $k = 0$, then by definition

$$[\![apx_0(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r) = 1 \Leftrightarrow [\![\psi]\!](r) = 1$$

and thus the claim holds.
If $k > 0$, then

$$[\![apx_k(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r) = 1 \Leftrightarrow [\![\psi \vee \phi \wedge \Diamond(apx_{k-1}(\mu Z.\psi \vee \phi \wedge \Diamond Z))]\!](r) = 1$$

If $[\![\psi]\!](r) = 1$, the claim holds obviously. Otherwise

$$[\![\phi]\!](r) = 1 \text{ and } [\![\Diamond(apx_{k-1}(\mu Z.\psi \vee \phi \wedge \Diamond Z))]\!](r) = 1.$$

By induction each point x in every region $r' \in R$ satisfying $[\![apx_{k-1}(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r') = 1$ satisfies $x \vDash E(\phi \underline{\mathrm{U}} \psi)$. Thus by definition of $\Diamond$ and $[\![\phi]\!](r) = 1$ every point $x \in r$ has a discrete or continuous successor x' satisfying $[\![\Diamond(apx_{k-1}(\mu Z.\psi \vee \phi \wedge \Diamond Z))]\!](r(x')) = 1$ and on the whole path $x \rightsquigarrow x'$ either the condition ϕ or the condition ψ holds (depends on whether $r(x')$ already satisfies ψ or only ϕ). This yields

$$[\![\mu Z.\psi \vee \phi \wedge \Diamond Z]\!](r) = 1 \Rightarrow \forall x \in r : [\![E(\phi \underline{\mathrm{U}} \psi)]\!]_H(x) = 1$$

Let now $[\![E(\phi \underline{\mathrm{U}} \psi)]\!]_H(x) = 1$, i.e. there exists a path

$$x = x_n \rightarrow ... \rightarrow x_0 = x'$$

in H with corresponding path

$$r_n \rightarrow ... \rightarrow r_0$$

in A being a witness for that. Let without loss of generality the region r_0 be the first region in the path satisfying $[\![\psi]\!](r_0) \in \{1, \bot\}$. It will now be shown by induction, that for all $0 \leq i \leq n$ it holds:

$$[\![apx_i(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r_i) \in \{1, \bot\}$$

$i = 0$:
By definition of the fixpoint for any $r \in R$

$$[\![apx_0(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r) = [\![\psi]\!](r) \in \{1, \bot\}$$

and thus

$$[\![apx_0(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r_0) \in \{1, \bot\}$$

$i > 0$:
By induction hypothesis

$$[\![apx_{i-1}(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r_{i-1}) \in \{1, \bot\}$$

Thus by the definition of $\langle a \rangle$ with $a \in \{\delta, e\}$ and therefore also $\Diamond$

$$[\![apx_i(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!](r_i) \neq 0$$

Together with the fact that for a suitable $\hat{k} \in \mathbb{N}$

$$[\![\mu Z.\psi \vee \phi \wedge \Diamond Z]\!] = [\![apx_{\hat{k}}(\mu Z.\psi \vee \phi \wedge \Diamond Z)]\!]$$

this yields the claim.

to 2.:
Analogously q.e.d.

5.2 Abstractions with May and Must Transitions

In this section, we instantiate the semantics scheme given in Definition 5.1 to the case in which given abstractions are may/must abstractions.

As defined in Chapter 4 before, abstractions with may and must relations consist in parallel of an overapproximation (the may-relation) and an underapproximation (the must-relation) of the underlying hybrid automaton. Both components will be used, when defining the truth-values for the modal operators. For example, for $a \in \{e, \delta\}$ the formula $[\![\langle a \rangle \phi]\!](r) = 1$ is obviously fulfilled, if there exists a must-transition $r \xrightarrow{a}_{must} r'$ with $[\![\phi]\!](r') = 1$, since in this case, each point in r has a $\xrightarrow{a}$ successor satisfying the condition ϕ. On the other hand, $[\![\langle a \rangle \phi]\!](r) = 1$ holds only, if every $\xrightarrow{a}$ successor does not fulfill the condition ϕ. The other modal operators use the may- and must-relations in an analogous way. This leads to the following definition:

Definition 5.2 (μ-Calculus Semantics Completion)

Let H be a hybrid automaton, $A = \langle R, R_0 \xrightarrow{\delta}, \xrightarrow{e} \rangle$ be an abstraction with may/must of H and let ϕ and ψ be μ-calculus formulas. Then the semantics of the three-valued μ-calculus of Definition 5.1 for $a, a_i \in \{\delta, e\}$ is completed by:

- $[\![\langle a \rangle \phi]\!](r) = 1$ *iff* $\exists r \xrightarrow{a}_{must} r' : [\![\phi]\!](r') = 1$
  $[\![\langle a \rangle \phi]\!](r) = 0$ *iff* $\forall r \xrightarrow{a} r' : [\![\phi]\!](r') = 0$
  $[\![\langle a \rangle \phi]\!](r) = \perp$ *iff neither* $[\![\langle a \rangle \phi]\!](r) = 1$ *nor* $[\![\langle a \rangle \phi]\!](r) = 0$

- $[\![[a] \phi]\!](r) = 1$ *iff* $\forall r \xrightarrow{a} r' : [\![\phi]\!](r') = 1$
  $[\![[a] \phi]\!](r) = 0$ *iff* $\exists r \xrightarrow{a}_{must} r' : [\![\phi]\!](r') = 0\}$
  $[\![[a] \phi]\!](r) = \perp$ *iff neither* $[\![[a] \phi]\!](r) = 1$ *nor* $[\![[a] \phi]\!](r) = 0$

- $[\![E(\phi \underline{U} \psi)]\!](r) = 1$ *iff*
  $\exists r = r_n \xrightarrow{a_{n-1}}_{must} \ldots \xrightarrow{a_0}_{must} r_0 = r' : [\![\psi]\!](r_0) = 1 \wedge [\![\phi]\!](r_i) = 1$ *for* $i > 0$
  $[\![E(\phi \underline{U} \psi)]\!](r) = 0$ *iff* $\forall n \in \mathbb{N} \nexists r = r_n \xrightarrow{a_{n-1}} \ldots \xrightarrow{a_0} r_0 = r' :$
  $[\![\psi]\!](r_0) \in \{1, \perp\} \wedge [\![\phi]\!](r_i) \in \{1, \perp\}$ *for* $i > 0$
  $[\![E(\phi \underline{U} \psi]\!])(r) = \perp$ *iff neither* $[\![E(\phi \underline{U} \psi]\!])(r) = 1$ *nor* $[\![E(\phi \underline{U} \psi]\!])(r) = 0$

- *Let* $\{r_n\}_{n\in\mathbb{N}}$ *be a infinitely long path in A. Then*
  $[\![(A\phi\underline{U}\psi)]\!](r) = 1$ *iff* $\forall\{r_n\}_{n\in\mathbb{N}}$ *with* $r_0 = r \exists k \in \mathbb{N}:$
  $[\![\phi]\!](r_i) = 1$ *for* $i < k \ \wedge\ [\![\psi]\!](r_k) = 1$
  $[\![A(\phi\underline{U}\psi)]\!](r) = 0$ *iff* $\exists r = r_n \overset{a_{n-1}}{\rightarrow}_{must} \ldots \overset{a_0}{\rightarrow}_{must} r_0 = r':$
  $[\![\phi]\!](r_i) \in \{1, \perp\}$ *for* $i > 0 \wedge [\![\phi]\!](r_0) = 0 \wedge [\![\psi]\!](r_i) = 0$ *for* $i \geq 0$
  $[\![A(\phi\underline{U}\psi)]\!](r) = \perp$ *iff neither* $[\![(A\phi\underline{U}\psi)]\!](r) = 1$ *nor* $[\![(A\phi\underline{U}\psi)]\!](r) = 0$

By Corollary 5.4 preservation results for general formulas according to the μ-calculus semantics completion in Definition 5.1 depend only on the fact that the modal operators behave as presented on a may/must abstraction.

Theorem 5.6

Let H be a hybrid automaton, A be an abstraction of H and let the interpretation of μ-calculus be as in Definition 5.2. Then, for any μ-calculus formula ϕ we have:

- $A \models \phi \Rightarrow H \models \phi$
- $A \not\models \phi \Rightarrow H \not\models \phi$

Proof

By Corollary 5.4 it suffices to show, that the modal operators $\bigstar \in \{\langle\delta\rangle\phi, \langle e\rangle\phi, [\delta]\phi, [e]\phi, E\phi\underline{\mathrm{U}}\psi, A\phi\underline{\mathrm{U}}\psi\}$ fulfill the property

$$[\![\bigstar]\!](r) = 1 \Rightarrow \forall\, x \in r:\ [\![\bigstar]\!]_H(x) = 1$$
$$[\![\bigstar]\!](r) = 0 \Rightarrow \forall\, x \in r:\ [\![\bigstar]\!]_H(x) = 0$$

Case 1: $\bigstar = \langle a\rangle\phi$ with $a \in \{e, \delta\}$

$$\begin{aligned}
[\![\langle a\rangle\phi]\!](r) = 1 &\Rightarrow \exists r \overset{a}{\rightarrow}_{must} r':\ [\![\phi]\!](r') = 1\\
&\Rightarrow \forall x \in r \exists x' \in r':\ x \overset{a}{\rightarrow} x' \wedge [\![\phi]\!]_H(x') = 1\\
&\Rightarrow \forall\, x \in r:\ [\![\langle a\rangle\phi]\!]_H(x) = 1\\
[\![\langle a\rangle\phi]\!](r) = 0 &\Rightarrow \forall r \overset{a}{\rightarrow} r':\ [\![\phi]\!](r') = 0\\
&\Rightarrow \forall x \in r \forall x \overset{a}{\rightarrow} x':\ [\![\phi]\!](x') = 0\\
&\Rightarrow \forall\, x \in r:\ [\![\langle a\rangle\phi]\!](x) = 0
\end{aligned}$$

Case 2: $\bigstar = [a]\phi$ with $a \in \{e, \delta\}$

$$\begin{aligned}
[\![[a]\phi]\!](r) = 1 &\Rightarrow \forall r \overset{a}{\rightarrow} r':\ [\![\phi]\!](r') = 1\\
&\Rightarrow \forall x \in r \forall x \overset{a}{\rightarrow} x':\ [\![\phi]\!](x') = 1\\
&\Rightarrow \forall\, x \in r:\ [\![[a]\phi]\!](x) = 1\\
[\![[a]\phi]\!](r) = 0 &\Rightarrow \exists r \overset{a}{\rightarrow}_{must} r':\ [\![\phi]\!](r') = 0\\
&\Rightarrow \forall x \in r \exists x' \in r':\ x \overset{a}{\rightarrow} x' \wedge [\![\phi]\!](x') = 0\\
&\Rightarrow \forall\, x \in r:\ [\![[a]\phi]\!](x) = 0
\end{aligned}$$

Case 3: $\bigstar = E(\phi \underline{\mathrm{U}} \psi)$
Let $[\![E(\phi \underline{\mathrm{U}} \psi)]\!](r) = 1$. By definition there exists a path

$$r = r_n \overset{a_{n-1}}{\rightarrow}_{must} \ldots \overset{a_0}{\rightarrow}_{must} r_0 = r'$$

with $[\![\psi]\!](r_0) = 1 \wedge [\![\phi]\!](r_i) = 1$ for $i > 0$. Since all transitions are must-transitions for all $x \in r$ there exists a path

$$x = x_n \overset{a_{n-1}}{\rightarrow} \ldots \overset{a_0}{\rightarrow} x_0 = x'$$

with $[\![\psi]\!]_H(x_0) = 1$ and on the whole path $x = x_n \rightsquigarrow x_0 = x'$ the condition ϕ is satisfied.
Let now $[\![E(\phi \underline{\mathrm{U}} \psi)]\!](r) = 0$ but for at least one $x \in r$ there exists a path

$$x = x_n \overset{a_{n-1}}{\rightarrow} \ldots \overset{a_0}{\rightarrow} x_0 = x'$$

with $a_i \in \{e, \delta\}$. Then the corresponding path in A has the form

$$r = r_n \overset{a_{n-1}}{\rightarrow} \ldots \overset{a_0}{\rightarrow} r_0 = r'$$

with $a_i \in \{e, \delta^*\}$ Since on the whole path $x \rightsquigarrow x'$ the condition ϕ holds and $[\![\psi]\!](x') = 1$ this yields for the corresponding regions of the path $r(x) \rightsquigarrow r(x')$ in A that $[\![\phi]\!](r, r') = 1$ (note, that because of the transitive closure of the $\overset{\delta}{\rightarrow}$ transitions the path $r \rightsquigarrow r'$ can consist of more regions than $r_n, \ldots, r_0$). Furthermore $[\![\psi]\!](r_0) \in \{1, \perp\}$. Thus by definition

$$[\![E(\phi \underline{\mathrm{U}} \psi)]\!](r) \neq 0$$

which is a contradiction to the assumption.

Case 4: $\bigstar = A(\phi \underline{\mathrm{U}} \psi)$
analogously q.e.d.

Furthermore, it is possible to show that in the setting of abstractions with may and must-transitions the operator $\underline{\mathrm{U}}$ is redundant, since the must-transitions only allow continuous transitions to direct successors and thus the conditions of Theorem 5.5 are satisfied.

Theorem 5.7

Let H be a hybrid automaton and let A be an abstraction with may and must relation. Let the semantics of μ-calculus be given by Definition 5.1 and 5.2. Then the operator $\underline{\mathrm{U}}$ is redundant.

Proof

By Theorem 5.5, it suffices to show that

- $[\![\langle\delta\rangle\phi]\!](r) = 1$ iff for all $x \in r$ there exists an x' s.t. the continuous path $x \rightsquigarrow x'$ only traverses the regions of x and x' and $[\![\phi]\!](r(x')) = 1$, i.e. all $x \in r$ have a direct successor satisfying the condition ϕ.
- $[\![[\delta]\phi]\!](r) = 0$ iff $[\![\neg(\langle\delta\rangle\neg\phi)]\!](r) = 1$, i.e. iff all $x \in r$ have a direct successor not satisfying the condition ϕ.

This is obviously the case, since the $\xrightarrow{\delta}_{must}$ transitions per definition only allow direct successors. q.e.d.

As an example consider again the heating controller introduced in Fig. 2.1. Figure 5.1 and Fig. 5.2 show two different abstractions with may / must of this heating controller.

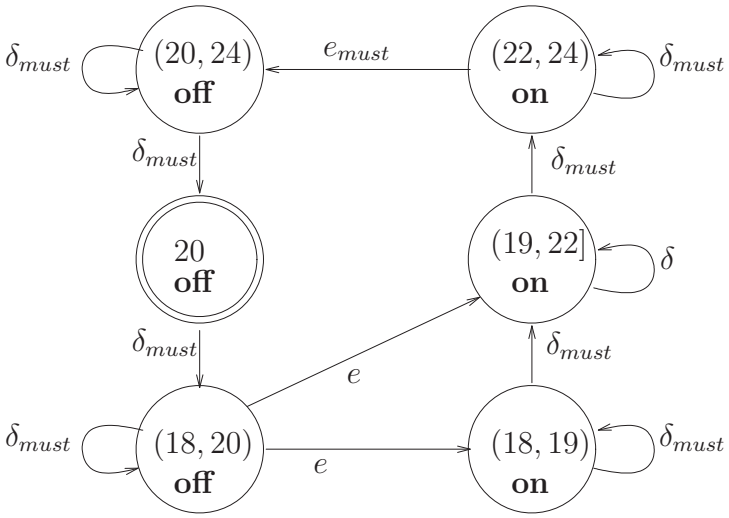


Figure 5.1: Abstraction A_1 with may/must of the heating controller

Consider now the formula $EF\phi = E(1\underline{\mathrm{U}}\phi)$, where ϕ denotes a propositional letter being true only for the abstract state $(\mathbf{off}, (20, 24))$. This formula holds in those states that can reach a state, where the temperature is between 20 and 24 degree and the heating is off.

- Abstraction A_1 of Fig. 5.1:
  $[\![EF\phi]\!](r) = 1$ for $r \in \{(\mathbf{off}, (20, 24)),\ (\mathbf{on}, (22, 24)),\ (\mathbf{on}, [19, 22]),\ (\mathbf{on}, (18, 19))\}$
  $[\![EF\phi]\!](r) = \perp$ for $r \in \{(\mathbf{off}, (18, 20)),\ (\mathbf{off}, 20)\}$
  $\Rightarrow$ Neither $A_1 \models EF\phi$ nor $A_1 \not\models EF\phi$ can be proved.
- Abstraction A_2 of Fig. 5.2:
  $[\![EF\phi]\!](r) = 1$ for $r \in \{(\mathbf{off}, (20, 24)),\ (\mathbf{on}, (22, 24)),\ (\mathbf{on}, [19, 22]),\ (\mathbf{on}, (18, 19)),\ (\mathbf{off}, [19.5, 20)),\ (\mathbf{off}, 20)\}$
  $[\![EF\phi]\!](r) = \perp$ for $r \in \{(\mathbf{off}, (18, 19.5))\}$
  $\Rightarrow$ Here, $A_2 \models EF\phi$ can be proved.

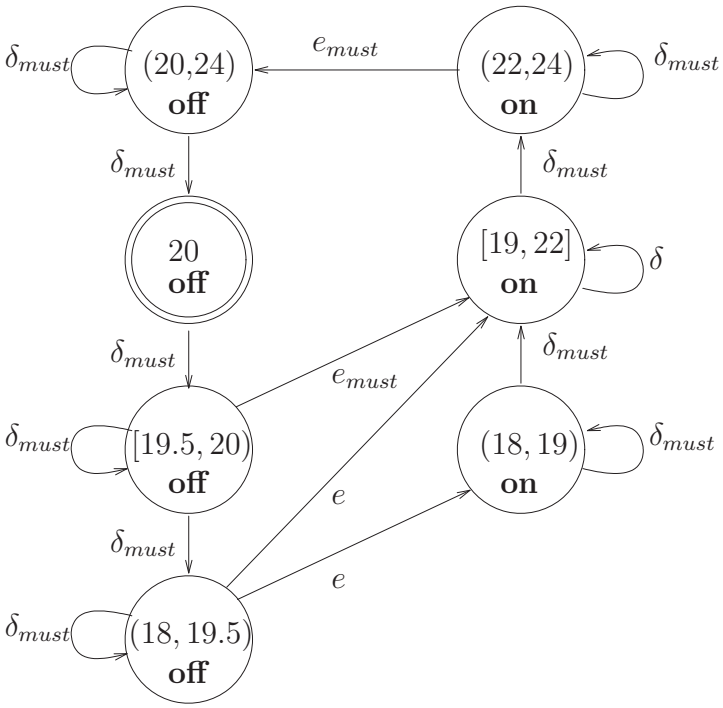


Figure 5.2: Abstraction A_2 with may/must of the heating controller

The same results are obtained by for the translation of $EF\phi = E(1\underline{\mathrm{U}}\phi)$ to the fixpoint-formula $\mu Z.\phi \vee \Diamond Z$.

5.3 Discrete Bounded Bisimulation Abstractions

In this section, we instantiate the semantics-scheme for three-valued μ-calculus given in Definition 5.1 to the case in which the considered abstractions are DBB-Abstractions.

Definition 5.3 (μ-Calculus Semantics Completion)

Let H be a hybrid automaton and $H_{\equiv_n} = \langle Q_{/\equiv n}, Q_{0/\equiv n}, l_{\rightarrow} \rightarrow_{/\equiv n} \rangle$ be its n-DBB abstraction. Then the semantics of the definition of the three-valued μ-calculus is completed by:

- $[\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *iff* $\exists [x']_{\equiv n} \in Q_{/\equiv n} : [x]_{\equiv n} \xrightarrow{\delta} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) = 1$
  $[\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 0$ *iff* $\nexists [x']_{\equiv n} \in Q_{/\equiv n} : [x]_{\equiv n} \xrightarrow{\delta^*} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) = 1$
  $[\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = \perp$ *iff neither* $[\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *nor* $[\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 0$

- $[\![[\delta]\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *iff* $\forall [x]_{\equiv n} \xrightarrow{\delta^*} [x']_{\equiv n} : [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) = 1 \wedge [\![\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$
  $[\![[\delta]\phi]\!]_{\equiv n}([x]_{\equiv n}) = 0$ *iff* $\exists [x']_{\equiv n} \in Q_{/\equiv n} : [x]_{\equiv n} \xrightarrow{\delta} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) = 0$
  $[\![[\delta]\phi]\!]_{\equiv n}([x]_{\equiv n}) = \perp$ *iff neither* $[\![[\delta]\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *nor* $[\![[\delta]\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$

- $[\![\langle e\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *iff* $\exists [x']_{\equiv n} \in Q_{/\equiv n} : [x]_{\equiv n} \xrightarrow{\delta} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n-1}([x']_{\equiv n}) = 1$

$[\![\langle e\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 0$ *iff* $\nexists [x']_{\equiv n} \in Q_{/\equiv n} : [x]_{\equiv n} \xrightarrow{e} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) \neq 0$
$[\![\langle e\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = \perp$ *iff neither* $[\![\langle e\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *nor* $[\![\langle e\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) = 0$.

- $[\![[e]\phi]\!]_{\equiv n} := [\![\neg(\langle e\rangle\neg\phi)]\!]_{\equiv n}$
- *For* $[\![E(\phi \underline{U} \psi)]\!]_{\equiv n}$*:*
  - $[\![E(\phi \underline{U} \psi)]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *iff there exists a path* $\{[x_i]_{\equiv n}\}_{0\leq i\leq k}$ *in* $H/_{\equiv n}$ *with*
    1. $\forall i < k : [x_i]_{\equiv n} \xrightarrow{\delta} [x_{i+1}]_{\equiv n} \wedge [\![\phi \vee \psi]\!]_{\equiv n}([x_i]_{\equiv n}) = 1$
    2. $[\![\psi]\!]_{\equiv n}([x_k]_{\equiv n}) = 1$ *or*
       $[x_k]_{\equiv n} \xrightarrow{e} [x_{k+1}]_{\equiv n} \wedge [\![E(\phi \underline{U} \psi)]\!]_{\equiv n-1}([x_{k+1}]_{\equiv n-1}) = 1$
  - $[\![E\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 0$ *iff for all paths* $\{[x_i]_{\equiv n}\}_{0\leq i\leq k}$ *in* $H/_{\equiv n}$ *and* $\forall i \in \mathbb{N}$*:*
    $[\![\psi]\!]_{\equiv n}([x_i]_{\equiv n}) = 1 \Rightarrow \exists j < i : [\![\phi \vee \psi]\!]_{\equiv n}([x_j]_{\equiv n}) = 0$
  - $[\![E\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = \perp$ *iff neither* $[\![E\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *nor*
    $[\![E\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 0$
- *For* $[\![A(\phi \underline{U} \psi)]\!]_{\equiv n}$*:*
  - $[\![A\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *iff for all paths* $\{[x_i]_{\equiv n}\}_{0\leq i\leq k}$ *in* $H/_{\equiv n}$ *there exists* $k \in \mathbb{N}$*:*
    1. $\forall i < k : [\![\phi \vee \psi]\!]_{\equiv n}([x_i]_{\equiv n}) = 1$
    2. $[\![\psi]\!]_{\equiv n}([x_k]_{\equiv n}) = 1$
  - $[\![A(\phi \underline{U} \psi)]\!]_{\equiv n}([x]_{\equiv n}) = 0$ *iff there exists a path* $\{[x_i]_{\equiv n}\}_{0\leq i\leq k}$ *in* $H/_{\equiv n}$ *with:*
    1. $\forall i < k : [x_i]_{\equiv n} \xrightarrow{\delta} [x_{i+1}]_{\equiv n} \wedge [\![\phi \wedge \neg\psi]\!]([x_i]_{\equiv n}) = 1$
    2. $[\![\phi \vee \psi]\!]_{\equiv n}([x_k]_{\equiv n}) = 0 \quad \vee$
       $[x_k]_{\equiv n} \xrightarrow{e} [x_{k+1}]_{\equiv n} \wedge [\![A\phi \underline{U} \psi]\!]_{\equiv n-1}([x_{k+1}]_{\equiv n-1}) = 0$
  - $[\![A\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = \perp$ *iff neither* $[\![A\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 1$ *nor*
    $[\![A\phi \underline{U} \psi]\!]_{\equiv n}([x]_{\equiv n}) = 0$

Like in the last section it again has to be shown, that the preservation conditions are fulfilled by this definition of the modal operators. This is done in the following

Theorem 5.8

Let H be a hybrid automaton, H_n be an n-DBB abstraction of H and let the interpretation of μ-calculus be as in Definition 5.1 and Definition 5.3. Then for any formula $\phi \in L_\mu$:

- $H_{\equiv_n} \models \phi \Rightarrow H \models \phi$
- $H_{\equiv_n} \not\models \phi \Rightarrow H \not\models \phi$

Proof

By Corollary 5.4 it suffices to show, that the modal operators

$$\bigstar \in \{[\delta]\phi, [e]\phi, \langle\delta\rangle\phi, \langle e\rangle\phi, E(\phi \underline{\mathrm{U}} \psi), A(\phi \underline{\mathrm{U}} \psi)\}$$

fulfill the property:

$$\begin{aligned} [\![\bigstar]\!]_{\equiv_n}(q) = 1 &\Rightarrow \forall\, x \in q :\ [\![\bigstar]\!]_H(x) = 1 \\ [\![\bigstar]\!]_{\equiv_n}(q) = 0 &\Rightarrow \forall\, x \in q :\ [\![\bigstar]\!]_H(x) = 0 \end{aligned}$$

Case 1: $\bigstar = \langle a\rangle\phi$ with $a \in \{\delta, e\}$
It suffices to show that

- $[\![\langle\delta\rangle\phi]\!]_{\equiv_n}(q) = 1 \Rightarrow \forall\, x \in q :\ [\![\langle\delta\rangle\phi]\!]_H(x) = 1$
  $[\![\langle\delta\rangle\phi]\!]_{\equiv_n}(q) = 1 \Rightarrow \forall\, x \in q :\ [\![\langle\delta\rangle\phi]\!]_H(x) = 1$
- $[\![\langle e\rangle\phi]\!]_{\equiv_n}(q) = 0 \Rightarrow \forall\, x \in q :\ [\langle e\rangle\phi]\!]_H(x) = 1$
  $[\![\langle e\rangle\phi]\!]_{\equiv_n}(q) = 0 \Rightarrow \forall\, x \in q :\ [\![\langle e\rangle\phi]\!]_H(x) = 0$

By part 2b)+ 2c) (or 1b)+1c) for $n = 1$) of Definition 4.2:

$$\begin{aligned} [\![\langle\delta\rangle\phi]\!]_{\equiv n}(q) = 1 &\Rightarrow \exists q \xrightarrow{\delta} q' :\ [\![\phi]\!]_{\equiv n}(q') = 1 \\ &\Rightarrow \forall x \in q \exists x' \in q' :\ x \xrightarrow{\delta} x' \wedge [\![\phi]\!]_H(x) = 1 \\ &\Rightarrow \forall x \in q :\ [\![\langle\delta\rangle\phi]\!]_H(x) = 1 \\ [\![\langle\delta\rangle\phi]\!]_H(x) = 1 &\Rightarrow \exists x \xrightarrow{\delta} x' :\ [\![\phi]\!]_H(x') = 1 \\ &\Rightarrow [x]_{\equiv n} \xrightarrow{\delta^*} [x']_{\equiv n} \wedge [\![\phi]\!]_{\equiv n}([x']_{\equiv n}) \in \{1, \bot\} \\ &\Rightarrow [\![\langle\delta\rangle\phi]\!]_{\equiv n}([x]_{\equiv n}) \neq 0 \end{aligned}$$

Let us now have a look at $\langle e\rangle$. By definition of $[\![\langle e\rangle\phi]\!]_n(q) = 1$ there exists a transition $q \xrightarrow{e} q'$ with $[\![\phi]\!]_{n-1}(q') = 1$. Then, by Definition 4.2 part 2d) and 2e) this yields that for all $x \in q$ there exists an x' with $[x']_{\equiv_{n-1}} = [q]_{\equiv_{n-1}}$. Together with $[\![\phi]\!]_{\equiv n-1}(q') = 1$ this yields the claim.
Assume now $[\![\langle e\rangle\phi]\!]_{\equiv_n}(q) = 0$. If there would exist an $x \in q$ with $[\![\langle e\rangle\phi]\!]_H(x) = 1$, then $x \xrightarrow{e} x'$ with $[\![\phi]\!]_(x') = 1$ and thus $[\![\phi]\!]_{\equiv_n}(q(x') \in \{1, \bot\}$. But then by definition of $[e]$ it holds $[\![\langle e\rangle\phi]\!]_{\equiv_n}(q) \neq 0$, which is a contradiction to the assumption.

Case 2: $\bigstar = [a]\phi$ with $a \in \{\delta, e\}$
By definition of the $[a]$-operator via the $\langle a\rangle$-operator and since the claim has already been shown for $\langle a\rangle$ this follows immediately.

Case 3: $\bigstar \in \{E(\phi \underline{\mathrm{U}} \psi), A(\phi \underline{\mathrm{U}} \psi)\}$
see [GSM07] q.e.d.

The next theorem shows some monotonicity result, i.e. that any $\{1,0\}$-result achieved with a lower abstraction depth is preserved when going to a higher abstraction depth. For practical purposes this especially means that if some fixpoint-calculations have already been done for lower abstraction depths, these calculations can be reused in such a way, that only the regions, which had formerly the result $\perp$ have to be considered again, since the truth-values for the other regions cannot be changed any more.

Theorem 5.9 (Monotonicity)

Let H_n and H_k with $n > k$ be n-DBB abstractions of the hybrid automaton H and let ϕ be a fixpoint-free formula of μ-calculus. Then it holds for any x in the state space of H:

- $[x]_{\equiv_k} \in [\phi]_{\equiv_k}(1) \Rightarrow [x]_{\equiv_n} \in [\phi]_{\equiv_n}(1)$
- $[x]_{\equiv_k} \in [\phi]_{\equiv_k}(0) \Rightarrow [x]_{\equiv_n} \in [\phi]_{\equiv_n}(0)$

Proof (by structural induction and induction over $\mathbb{N}$)

By induction over the natural numbers it suffices to show, that

- $[\![\phi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \Rightarrow [\![\phi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1$
- $[\![\phi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \Rightarrow [\![\phi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1$

This will be done by structural induction.

- $\phi \in l_{AP}$:
  Since $[x_{\equiv_{k+1}}] \subseteq [x]_{\equiv k}$ it holds
  - $\phi \in l_{AP}([x]_{\equiv k}) \Rightarrow \phi \in l_{AP}([x]_{\equiv k+1})$
  - $\phi \notin l_{AP}([x]_{\equiv k}) \Rightarrow \phi \notin l_{AP}([x]_{\equiv k+1})$
- $\phi = \neg\psi$:
  - By definition (def) of $\neg$ and by structural induction (ind):
    $$\begin{aligned}[\![\neg\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 &\overset{def}{\Rightarrow} [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 \\ &\overset{ind}{\Rightarrow} [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0 \\ &\overset{def}{\Rightarrow} [\![\neg\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1\end{aligned}$$
  - By definition of $\neg$ and by structural induction :
    $$\begin{aligned}[\![\neg\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 &\overset{def}{\Rightarrow} [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \\ &\overset{ind}{\Rightarrow} [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1 \\ &\overset{def}{\Rightarrow} [\![\neg\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0\end{aligned}$$

- $\phi = \varphi \vee \psi$:
  - By definition of $\vee$ and by structural induction:
$$\begin{aligned}[\![\varphi \vee \psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 &\overset{def}{\Rightarrow} [\![\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \vee [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \\ &\overset{ind}{\Rightarrow} [\![\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1 \vee [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1 \\ &\overset{def}{\Rightarrow} [\![\varphi \vee \psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1\end{aligned}$$
  - By definition of $\vee$ and by structural induction:
$$\begin{aligned}[\![\varphi \vee \psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 &\overset{def}{\Rightarrow} [\![\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 \wedge [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 \\ &\overset{ind}{\Rightarrow} [\![\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0 \wedge [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}} = 0 \\ &\overset{def}{\Rightarrow} [\![\varphi \vee \psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0\end{aligned}$$
- $\phi = \varphi \wedge \psi$:
  - By definition of $\wedge$ and by structural induction:
$$\begin{aligned}[\![\varphi \wedge \psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 &\overset{def}{\Rightarrow} [\![\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \wedge [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 \\ &\overset{ind}{\Rightarrow} [\![\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1 \wedge [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1 \\ &\overset{def}{\Rightarrow} [\![\varphi \wedge \psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1\end{aligned}$$
  - By definition of $\wedge$ and by structural induction:
$$\begin{aligned}[\![\varphi \wedge \psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 &\overset{def}{\Rightarrow} [\![\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 \vee [\![\psi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 \\ &\overset{ind}{\Rightarrow} [\![\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0 \vee [\![\psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0 \\ &\overset{def}{\Rightarrow} [\![\varphi \wedge \psi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0\end{aligned}$$
- $\phi = \langle\delta\rangle\varphi$:
  Let $[x]_{\equiv_k} \xrightarrow{\delta} [y]_{\equiv_k}$ in H_k. Then in the next further abstraction H_{k+1} for every $\hat{x} \in [x]_{\equiv_k}$ there exists a $\hat{y} \in [y]_{\equiv_k}$, s.t. $[\hat{x}]_{\equiv_{k+1}} \xrightarrow{\delta} [\hat{y}]_{\equiv_{k+1}}$. Thus with $[\hat{y}]_{\equiv_k} = [y]_{\equiv_k}$:
  - By definition of $\langle\delta\rangle$ and by structural induction:
$$\begin{aligned}[\![\langle\delta\rangle\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 &\overset{def}{\Rightarrow} \exists [x]_{\equiv_k} \xrightarrow{\delta} [y]_{\equiv_k} : [\![\varphi]\!]_{\equiv_k}([y]_{\equiv_k}) = 1 \\ &\overset{ind}{\Rightarrow} \exists [x]_{\equiv_{k+1}} \xrightarrow{\delta} [\hat{y}]\!]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_{k+1}}([\hat{y}]_{\equiv_{k+1}}) = 1 \\ &\overset{def}{\Rightarrow} [\![\langle\delta\rangle\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1\end{aligned}$$
  - By definition of $\langle\delta\rangle$ and by structural induction:
$$\begin{aligned}[\![\langle\delta\rangle\varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 &\overset{def}{\Rightarrow} \forall [x]_{\equiv_k} \xrightarrow{\delta} [y]_{\equiv_k} : [\![\varphi]\!]_{\equiv_k}([y]_{\equiv_k}) = 0 \\ &\overset{ind}{\Rightarrow} \forall [x]_{\equiv_{k+1}} \xrightarrow{\delta} [\hat{y}]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_{k+1}}([\hat{y}]_{\equiv_{k+1}}) = 0 \\ &\overset{def}{\Rightarrow} [\![\langle\delta\rangle\varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0\end{aligned}$$

- $\phi = \langle e \rangle \varphi$:
  Let $[x]_{\equiv_k} \xrightarrow{e} [y]_{\equiv_k}$ in H_k. Then in the next further abstraction H_{k+1} for every $\hat{x} \in [x]_{\equiv_k}$ there exists a $\hat{y} \in [y]_{\equiv_k}$, s.t. $[\hat{x}]_{\equiv_{k+1}} \xrightarrow{e} [\hat{y}]_{\equiv_{k+1}}$. Thus with $[\hat{y}]_{\equiv_k} = [y]_{\equiv_k}$:

  – By definition of $\langle e \rangle$ and by structural induction:

$$\begin{aligned}
[\![\langle e \rangle \varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 1 &\overset{def}{\Rightarrow} \exists [x]_{\equiv_k} \xrightarrow{e} [y]_{\equiv_k} : [\![\varphi]\!]_{\equiv_{k-1}}([y]_{\equiv_{k-1}}) = 1 \\
&\Rightarrow \exists [x]_{\equiv_{k+1}} \xrightarrow{e} [\hat{y}]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_{k-1}}([\hat{y}]_{\equiv_{k-1}}) = 1 \\
&\overset{ind}{\Rightarrow} \exists [x]_{\equiv_{k+1}} \xrightarrow{e} [\hat{y}]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_k}([\hat{y}]_{\equiv_k}) = 1 \\
&\overset{def}{\Rightarrow} [\![\langle e \rangle \varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 1
\end{aligned}$$

  – By definition of $\langle e \rangle$ and by structural induction:

$$\begin{aligned}
[\![\langle e \rangle \varphi]\!]_{\equiv_k}([x]_{\equiv_k}) = 0 &\overset{def}{\Rightarrow} \forall [x]_{\equiv_k} \xrightarrow{e} [y]_{\equiv_k} : [\![\varphi]\!]_{\equiv_{k-1}}([y]_{\equiv_{k-1}}) = 0 \\
&\Rightarrow \forall [x]_{\equiv_{k+1}} \xrightarrow{e} [\hat{y}]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_{k-1}}([\hat{y}]_{\equiv_{k-1}}) = 0 \\
&\overset{ind}{\Rightarrow} \forall [x]_{\equiv_{k+1}} \xrightarrow{e} [\hat{y}]_{\equiv_{k+1}} : [\![\varphi]\!]_{\equiv_k}([\hat{y}]_{\equiv_k}) = 0 \\
&\overset{def}{\Rightarrow} [\![\langle e \rangle \varphi]\!]_{\equiv_{k+1}}([x]_{\equiv_{k+1}}) = 0
\end{aligned}$$

- $\phi = [\delta]\varphi$ or $\phi = [e]\varphi$:
  follows directly, since $[.]$ is defined via $\langle . \rangle$ and $\neg$ and it has already been shown for these operators.

- $\phi = E(\varphi \underline{\mathrm{U}} \psi)$ or $\phi = A(\varphi \underline{\mathrm{U}} \psi)$:
  see [GSM07] q.e.d.

Theorem 5.10

Let H be a hybrid automaton and let $H_{\equiv n}$ be a n-DBB abstraction. Let the semantics of μ-calculus be given by Definition 5.1 and 5.3. Then the operator $\underline{U}$ is redundant.

Proof

By Theorem 5.5 it suffices to show, that

- $[\![\langle \delta \rangle \phi]\!](r) = 1$ iff for all $x \in r$ there exists an x' s.t. the continuous path $x \rightsquigarrow x'$ only traverses the regions of x and x' and $[\![\phi]\!](r(x')) = 1$, i.e. all $x \in r$ have a direct successor satisfying the condition ϕ.

- $[\![[\delta]\phi]\!](r) = 0$ iff $[\![\neg(\langle \delta \rangle \neg \phi)]\!](r)(1)$, i.e. iff all $x \in r$ have a direct successor not satisfying the condition ϕ.

By Definition 4.2 and Definition 5.3 this is obviously the case. q.e.d.

Chapter 6

Summary and Outlook

This diploma thesis gives an overview of algebraic methods for the analysis of algebraic hybrid automata. Based on the work of [SSM04, San05] the algorithm for computing inductive assertion maps has been made incremental. Furthermore, it has been shown, that methods of algebraic geometry are not a first choice for the computation of images and preimages of discrete and continuous transitions, since only overapproximations of these sets can be achieved.

The second part of this diploma thesis provides the theoretical foundation of abstractions of hybrid automata for model checking properties of the μ-calculus. In order to achieve this goal, a general parametrized framework has been developed, which only depends on the concrete interpretation of the modal operators. Since different abstraction classes offer different types of information and thus always need an adapted interpretation of this information, this general framework offers the opportunity to be specialized concerning the definition of the modal operators which still maintain the preservation results for the general framework. Applications of the three-valued μ-calculus to two classes of abstractions, namely abstractions with may and must relations and n-bounded bisimulations have been provided in this thesis. In princip, any class of abstractions providing an over- and an underapproximation of the behavior of the underlying hybrid automaton can be applied to this framework.

This thesis only provides the theoretical background of three-valued μ-calculus. The effectiveness, strengths and weaknesses of this framework still have to be evaluated in future projects.

An interesting approach for future work is to use three-valued μ-calculus in order to make directed abstraction refinements only in parts not already providing enough information for a final decision, whether a given condition is fulfilled or not.

Bibliography

[ACHH93] Alur, R., Courcoubetis, C., Henzinger, T., and Ho, P.-H., *Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems*, Hybrid Systems (R. Grossmann, A. Nerode, A. Ravn, and H. Rischel, eds.), LNCS, vol. 736, Springer, 1993, pp. 209–229.

[ACM84] Arnon, D. S., Collins, G. E., and McCallum, S., *Cylindrical algebraic decomposition I: the basic algorithm*, SIAM J. Comput. **13** (1984), no. 4, 865–877.

[AD94] Alur, R., and Dill, D. L., *A theory of timed automata*, Theoretical Computer Science **126** (1994), no. 2, 183–235.

[AHH93] Alur, R., Henzinger, T. A., and Ho, P.-H., *Automatic Symbolic Verification of Embedded Systems*, IEEE Real-Time Systems Symposium, 1993, pp. 2–11.

[AHLP00] Alur, R., Henzinger, T., Lafferriere, G., and Pappas, G., *Discrete abstractions of hybrid systems*, Proceedings of the IEEE **88** (2000), 971–984.

[ATP01] Alur, R., Torre, S. L., and Pappas, G. J., *Optimal Paths in Weighted Timed Automata*, Proceedings of the 4th International Workshop on Hybrid Systems, Springer-Verlag, 2001, pp. 49–62.

[BBLS93] Bensalem, S., Bouajjani, A., Loiseaux, C., and Sifakis, J., *Property Preserving Simulations*, Computer Aided Verification (CAV) (Montreal, Canada) (G. von Bochmann and D. Probst, eds.), LNCS, vol. 663, Springer, 1993, pp. 260–273.

[BCCZ99] Biere, A., Cimatti, A., Clarke, E. M., and Zhu, Y., *Symbolic Model Checking without BDDs.*, TACAS, 1999, pp. 193–207.

[BFH+01] Behrmann, G., Fehnker, A., Hune, T., Larsen, K. G., Pettersson, P., Romijn, J., and Vaandrager, F., *Minimum-Cost Reachability for Priced Timed Automata*, Proceedings of the 4th International Workshop on Hybrid Systems, vol. 2034, Springer-Verlag, 2001, pp. 147–161.

[BG99] Bruns, G., and Godefroid, P., *Model Checking Partial State Spaces with 3-Valued Temporal Logics*, Computer Aided Verification (CAV) (Trento, Italy) (N. Halbwachs and D. Peled, eds.), LNCS, vol. 1633, Springer, 1999, pp. 274–287.

[BMRT04] BRIHAYE, T., MICHAUX, C., RIVIÈRE, C., AND TROESTLER, C., *On O-Minimal Hybrid Systems*, Hybrid Systems: Computation and Control (HSCC) (Philadelphia, PA, USA) (R. Alur and G. Pappas, eds.), LNCS, vol. 2993, Springer, 2004, pp. 219–233.

[Buc76] BUCHBERGER, B., *A theoretical basis for the reduction of polynomials to canonical forms*, SIGSAM Bull. **10** (1976), no. 3, 19–29.

[CGP99] CLARKE, E., GRUMBERG, O., AND PELED, D., *Model Checking*, MIT Press, 1999.

[CMPM05] CASAGRANDE, A., MYSORE, V., PIAZZA, C., AND MISHRA, B., *Independent Dynamics Hybrid Automata in Systems Biology*, International Conference on Algebraic Biology (AB'05) (Tokyo, Japan), Universal Academy Press, Inc., November 2005, pp. 61–73.

[CPM05] CASAGRANDE, A., PIAZZA, C., AND MISHRA, B., *Semi-Algebraic Constant Reset Hybrid Automata - SACoRe*, Conference on Decision and Control and European Control Conference (CDC-ECC'05) (Seville, Spain), IEEE Computer Society, 2005, pp. 678–683.

[Dav99a] DAVOREN, J. M., *On Hybrid Systems and the Modal μ-calculus*, Lecture Notes in Computer Science **1567** (1999), 38–69.

[Dav99b] DAVOREN, J., *Topologies, Continuity and Bisimulations*, Theoretical Informatics and Applications **33** (1999), no. 4/5, 357–381.

[DGG97] DAMS, D., GERTH, R., AND GRUMBERG, O., *Abstract Interpretation of Reactive Systems*, ACM Transactions on Programming Languages and Systems (TOPLAS) **19** (1997), no. 2, 253–291.

[Fau99] FAUGÈRE, J., *A new efficient algorithm for computing Gröbner bases (F4).*, Journal of Pure and Applied Algebra **139(1–3)** (1999), 61–88.

[Fit94] FITTING, M., *Kleene's three valued logics and their children*, Fundam. Inf. **20** (1994), no. 1-3, 113–131.

[Gen05] GENTILINI, R., *Reachability Problems on Extended O-Minimal Hybrid Automata*, RR 07-05, Dep. of Computer Science, University of Udine, Italy, 2005.

[GHJ01] GODEFROID, P., HUTH, M., AND JAGADEESAN, R., *Abstraction-Based Model Checking Using Modal Transition Systems*, Conference on Concurrency Theory (CONCUR) (Aalborg, Denmark) (K. Larsen and M. Nielsen, eds.), LNCS, vol. 2154, Springer, 2001, pp. 426–440.

[GSM07] GENTILINI, R., SCHNEIDER, K., AND MISHRA, B., *Series of Abstractions of Hybrid Automata for Monotonic CTL Model Checking*, Symposium on Logical Foundations of Computer Science (New York City, USA), Springer, 2007.

[GT01] GHOSH, R., AND TOMLIN, C. J., *Lateral Inhibition through Delta-Notch Signaling: A Piecewise Affine Hybrid Model*, LNCS **2034** (2001), 232–245.

[GTT03] GHOSH, R., TIWARI, A., AND TOMLIN, C., *Automated Symbolic Reachability Analysis with Application to Delta-Notch Signaling Automata*, Hybrid Systems: Computation and Control HSCC, LNCS, vol. 2623, Springer, 2003, pp. 233–248.

[Hen96] HENZINGER, T., *The Theory of Hybrid Automata*, Symposium on Logic in Computer Science (LICS) (New Brunswick, New Jersey), 1996, pp. 278–292.

[HHK95] HENZINGER, M. R., HENZINGER, T. A., AND KOPKE, P. W., *Computing simulations on finite and infinite graphs*, Symposium on Foundations of Computer Science, IEEE Computer Society, 1995, p. 453.

[HKPV98] HENZINGER, T., KOPKE, P., PURI, A., AND VARAIYA, P., *What's decidable about hybrid automata?*, Journal of Computer and System Sciences **57** (1998), no. 1, 94–124.

[Kle71] KLEENE, S. C., *Introduction to Metamathematics*, Wolters-Noordhoff, Groningen, 1971.

[Kop96] KOPKE, P., *The Theory of Rectangular Hybrid Automata*, Ph.D. thesis, Cornell University, 1996.

[Koz83] KOZEN, D., *Results on the Propositional μ-Calculus*, Theoretical Computer Science **27** (1983), no. 3, 333–354.

[KS90] KANNELLAKIS, P. C., AND SMOLKA, S. A., *CCS Expressions, Finite State Processes, and Three Problems of Equivalence*, Information and Computation **86** (1990), no. 1, 43–68.

[KV05] KOROVINA, M., AND VOROBJOV, N., *Upper and lower bounds on sizes of finite bisimulations of Pfaffian dynamical systems*, Symposium on Foundations of Computer Science, IEEE, 2005.

[LPS00] LAFFERRIERE, G., PAPPAS, G., AND SASTRY, S., *O-Minimal Hybrid Systems*, Mathematics of Control, Signals, and Systems **13** (2000), no. 1, 1–21.

[LPY99] LAFFERRIERE, G., PAPPAS, J., AND YOVINE, S., *A New Class of Decidable Hybrid Systems*, Int. Workshop on Hybrid Systems, Springer, 1999, pp. 137–151.

[Mil80] MILNER, R., *A Calculus of Communicating Systems*, LNCS, vol. 92, Springer, 1980.

[Mil00] MILLER, J., *Decidability and Complexity Results for Timed Automata and Semi-linear Hybrid Automata.*, Int. Workshop on Hybrid Systems, 2000, pp. 296–309.

[Mis97] MISHRA, *Computational Real Algebraic Geometry*, Handbook of Discrete and Computational Geometry, CRC Press (J. E. Goodman and J. O'Rourke, eds.), 1997.

[PAM+05] PIAZZA, C., ANTONIOTTI, M., MYSORE, V., POLICRITI, A., WINKLER, F., AND MISHRA, B., *Algorithmic Algebraic Model Checking I: Challenges from Systems Biology.*, Int. Conf. on Computer Aided Verification, LNCS, vol. 3576, Springer, 2005, pp. 5–19.

[Par81] PARK, D., *Concurrency and Automata on Infinite Sequences*, GI-Conference on Theoretical Computer Science (Karlsruhe, Germany), LNCS, vol. 104, Springer, 1981, pp. 167–183.

[PG02] PFISTER, G., AND GREUEL, G. M., *A Singular Introduction to Commutative Algebra*, Springer, 2002.

[PJ04] PRAJNA, S., AND JADBABAIE, A., *Safety Verification of Hybrid Systems Using Barrier Certificates*, International Workshop on Hybrid Systems: Computation and Control (HSCC) (Philadelphia, PA, USA) (R. Alur and G. Pappas, eds.), LNCS, vol. 2993, Springer, 2004, pp. 477–492.

[PT87] PAIGE, R., AND TARJAN, R. E., *Three partition refinement algorithms*, SIAM Journal On Computing **16** (1987), no. 6, 973–989.

[RAH97] R. ALUR, C. C., AND HENZINGER, T. A., *Computing Accumulated Delays in Real-Time Systems*, Formal Methods in System Design **11** (1997), 137–156.

[RCT05] RODRÍGUEZ-CARBONELL, E., AND TIWARI, A., *Generating Polynomial Invariants for Hybrid Systems*, Hybrid Systems: Computation and Control, HSCC, 2005, pp. 590–605.

[RGM07] R. GENTILINI, K. S., AND MISHRA, B., *Successive Abstractions of Hybrid automata for Monotonic* CTL *Model Checking*, International Symposium on Logical Foundations of Computaer Science (LFCS'07), LNCS, Springer-Verlag, 2007, To Appear.

[RS05] RATSCHAN, S., AND SHE, Z., *Safety Verification of Hybrid Systems by Constraint Propagation Based Abstraction Refinement*, Hybrid Systems: Computation and Control (HSCC), vol. LNCS 3414, 2005, pp. 573–589.

[RSW04] REPS, T., SAGIV, M., AND WILHELM, R., *Static Program Analysis via 3-Valued Logic*, Computer Aided Verification (CAV) (Boston, MA, USA) (R. Alur and D. Peled, eds.), LNCS, vol. 3114, Springer, 2004, pp. 15–30.

[San05] SANKARANARAYANAN, S., *Mathematical Analysis of Programs*, Ph.D. thesis, Stanford University, 2005.

[Sch04] SCHNEIDER, K., *Verification of Reactive Systems*, Springer Verlag, 2004.

[SG04] Shoham, S., and Grumberg, O., *Monotonic Abstraction-Refinement for CTL*, Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (Barcelona, Spain) (K. Jensen and A. Podelski, eds.), LNCS, vol. 2988, Springer, 2004, pp. 546–560.

[SS07] Schuele, T., and Schneider, K., *Bounded Model Checking of Infinite State Systems*, Formal Methods in System Design (FMSD) **30** (2007), no. 1, 51–81.

[SSM04] Sankaranarayanan, S., Sipma, H., and Manna, Z., *Constructing Invariants for Hybrid Systems*, Hybrid Systems: Computation and Control (HSCC) (Philadelphia, PA, USA) (R. Alur and G. Pappas, eds.), LNCS, vol. 2993, Springer, 2004, pp. 539–554.

[Tar51] Tarski, A., *A Decision Method for Elementary Algebra and Geometry*, 2nd ed. Berkeley, CA: University of California Press (1951).

[Tar55] Tarski, A., *A Lattice-Theoretical Fixpoint Theorem and its Applications*, Pacific Journal of Mathematics **5** (1955), no. 2, 285–309.

[TK02] Tiwari, A., and Khanna, G., *Series of Abstraction for Hybrid Automata*, Hybrid Systems: Computation and Control, vol. LNCS 2289, 2002, pp. 465–478.

[TK04] Tiwari, A., and Khanna, G., *Nonlinear systems: Approximating reach sets*, 2004.

[van96] van den Dries, L., *O-minimal structures*, Logic: From Foundations to Applications (W. Hodges, ed.), Clarendon Press, 1996, pp. 99–108.

[van98] van den Dries, L., *Tame topology and o-minimal structures*, London Math. Soc. Lecture Note Ser., vol. 248, Cambridge University Press, 1998.

[vM94] van den Dries, L., and Miller, C., *On the real exponential field with restricted analytic functions*, Israel J. Math. **85** (1994), no. 1-3, 19–56.

[Wil97] Wilkie, A. J., *Schanuel conjecture and the decidability of the real exponential field*, Algebraic Model Theory (1997), 223–230.

Vorliegende Diplomarbeit wurde von mir selbstständig verfasst. Es wurden keine anderen als die angegebenen Quellen und Hilfsmittel benutzt.

Kaiserslautern, January 6, 2008

Kerstin Bauer