The Journal of Logic and Algebraic Programming 78 (2009) 74–97

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

An algebra of hybrid systems☆

Peter Höfner, Bernhard Möller *

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany

ARTICLE INFO

Article history:
Available online 18 October 2008

Keywords:
Hybrid system
Semiring
Quantale
Equational reasoning

ABSTRACT

Hybrid systems are heterogeneous systems characterised by the interaction of discrete and continuous dynamics. We present a trajectory-based algebraic model for describing hybrid systems; the trajectories used are closely related to streams. The algebra is based on left quantales and left semirings and provides a new application for these algebraic structures. We show that hybrid automata, which are probably the standard tool for describing hybrid systems, can conveniently be embedded into our algebra. Moreover we point out some important advantages of the algebraic approach. In particular, we show how to handle Zeno effects, which are excluded by most other authors. The development of the theory is illustrated by a running example and a larger case study.

1. Introduction

Hybrid systems are heterogeneous systems characterised by the interaction of discrete and continuous dynamics and hence a particular kind of reactive systems. Due to their widespread applications there was a rapid growth of interest in such systems during the last decade. Hybrid systems are an effective tool for modelling, design and analysis of a large number of technical systems such as traffic controls [46,18,22], automated manufacturing [17] and much more [45]; but they are also applicable in fields like chemistry and biology [37]. The most elementary and classical kind of hybrid system usually consists of a controlling subsystem, the controller for short, made up of digital components, e.g., hardware, and a controlled subsystem. The controller has discrete behaviour and the controlled subsystem shows continuous behaviour. In general, the behaviour of the controller depends on its current state and the behaviour of the controlled system and cannot be considered in isolation. Often, more complicated hybrid systems arise by composing smaller systems.

Nearly from the beginning of their formalisation, hybrid systems have been modelled as hybrid automata [23]. These automata have, next to nodes (corresponding to states) and transition edges, variables and differential equations. These additional features reflect the behaviour of the environment in each node. In fact, hybrid automata can be seen as a generalisation of timed automata [5]. The study of hybrid systems in computer science is still largely focused on hybrid automata (e.g. [3]). There are only few other approaches (e.g. [11]; see also Section 7).

On the other hand, over the last few decades, variants of Kleene algebras have turned out to be fundamental first-order structures in computer science. They have found widespread applications ranging from program analysis and semantics (e.g. [21] and its references) to combinatorial optimisation and concurrency control [14]. They offer a concise syntax for modelling actions, programs or state transitions under non-deterministic choice, sequential composition and iteration. Since the equational theory of Kleene algebra is that of regular expressions [16] they are strongly connected to finite automata. Additionally, there exist variants to cover infinite behaviour as described, e.g., with Büchi automata [12]. Moreover it has recently been shown that Kleene algebras as well as their variants provide a reasonable base for automated deduction [30,31].

☆ Significantly extended and revised version of [28].

* Corresponding author.
E-mail addresses: hoefner@informatik.uni-augsburg.de (P. Höfner), bernhard.moeller@informatik.uni-augsburg.de (B. Möller).

1567-8326/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2008.08.005

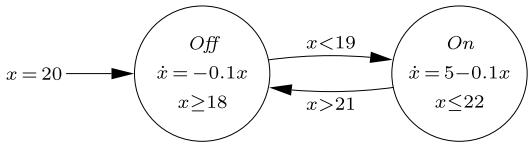

Fig. 1. Thermostat automaton.

In this paper we combine the concept of hybrid systems and the concept of Kleene algebra and propose an Algebra of Hybrid Systems. This algebra, which provides also a calculus of hybrid systems, allows the characterisation and description of hybrid systems in an abstract way. In particular, the algebra lifts results from real time analysis to equations about hybrid systems and provides equational axioms for hybrid systems that enable equational reasoning. Moreover, the proposed algebra yields a more general understanding of hybrid systems. Although the axioms and rules are derived from a model, the outcome is mostly purely algebraic and can therefore be applied to all other areas where such algebras occur.

Our concrete algebraic model for hybrid systems uses trajectories as elements, with discrete trajectories being isomorphic to streams. Each trajectory corresponds to a finite or infinite prefix of one single run of a hybrid automaton. Therefore it is straightforward to give a faithful mapping from the formalism of hybrid automata into our setting. Furthermore, unlike most other approaches, the algebra provides a simple and concise way of modelling Zeno effects.

The paper is structured as follows. In Section 2 we motivate our algebra by a concrete hybrid system that models a temperature control. This example is used as a running example through the whole paper to illustrate and motivate the theory. In Section 3 we then develop our concrete algebraic model preparing the abstraction to the setting of idempotent left semirings. We also show how Zeno effects can be integrated into the algebraic model. In Section 4 we give a constructive schema to convert hybrid automata into algebraic expressions. Furthermore we present an algebraic definition of several composition operators for hybrid automata and their algebraic counterparts. In Section 5 we discuss safety and liveness properties of hybrid systems. In more detail, we show how time restrictions and range assertions can be handled by certain algebraic versions of temporal operators related to ones defined by von Karger [50] and Sintzoff [48]. These operators enjoy many useful and new properties. To round off the paper, in Section 6, we apply our algebra of hybrid systems to a more complicated example. Section 7 presents a comparison with related work which is followed by conclusion and outlook in Section 8.

2. Introductory example and basic definitions

We motivate our formal definitions by an introductory example. Moreover, we recapitulate the standard definitions of hybrid automata, transitions, trajectories and runs.

Example 2.1 (*Temperature control*). The hybrid automaton of Fig. 1, adapted from [23], models a thermostat. The variable x represents the temperature. Initially, it is equal to 20 degrees and the heater is off (control mode *Off*). The temperature falls according to the flow condition $\dot{x} = -0.1x$. If the jump condition $x < 19$ is reached, the heater may start. The invariant condition $x \geq 18$ ensures that the heater will start at the latest when the temperature is equal to 18 degrees. In control mode *On*, the temperature rises according to the flow condition $\dot{x} = 5 - 0.1x$. If the temperature reaches the second jump condition, the heater is switched off and the procedure starts again (with the reached temperature as the new initial value).

In general, a *hybrid automaton* H [4,20,23] consists of the following components.

Variables. A finite set $X = \{x_1, \ldots, x_n\}$ of real-valued variables. The number n is called the *dimension* of H. We write $\dot{X}$ for the set $\{\dot{x}_1, \ldots, \dot{x}_n\}$ of dotted variables, which represent the timewise first derivatives of the x_i during continuous change. We write X' for the set $\{x'_1, \ldots, x'_n\}$ of primed variables, which represent the values of the x_i immediately after a discrete change.

Control graph. A finite directed multigraph (M, E). The vertices in M are called (*control*) *modes*. The edges in E are called (*control*) *switches*.

Invariant and flow conditions. The vertex labelling functions *inv* and *flow*. They assign to each control mode $v \in M$ an invariant $inv(v)$, a predicate with free variables from X, and a flow condition $flow(v)$, a predicate with free variables from $X \cup \dot{X}$.

Initial condition. The vertex labelling function *init* assigns to at least one control mode $v \in M$ an initial condition $init(v)$, a predicate with free variables from X.

Jump conditions. An edge labelling function *jump*. It assigns to each control switch $e \in E$ a predicate $jump(e)$ with free variables from $X \cup X'$.

If a control mode does not contain a differential equation for the variable x_i then we assume that this variable is constant, i.e., that the mode implicitly contains the equation $\dot{x}_i = 0$. An edge that leads from mode v to mode w is also called a *transition* $t_{v,w}$. The automaton *can perform* that transition if the end values X of mode v and the starting values X' of mode w satisfy

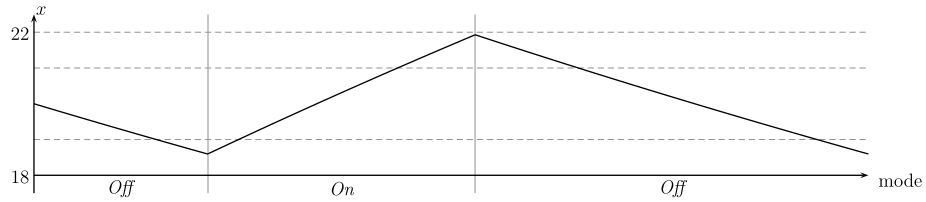

Fig. 2. A single trajectory of the temperature control.

the predicate $jump(t_{v,w})$. A transition is called a *proper jump* if it changes at least one value $x \in X$ to a new value $x' \in X'$ with $x \neq x'$. Note that Example 2.1 admits no proper jumps. In Section 3.3 we will extend this example by a proper jump.

With each hybrid automaton one can associate traces, runs and trajectories. Since we will use these concepts to define our algebra of hybrid systems, we recapitulate them.

A *transition trace* [51] of a hybrid automaton is a (finite or infinite) sequence of transitions $t_{v_k,v_{k+1}}$ which the hybrid automaton can perform as time passes. The (*mode*) *trace* of a hybrid system corresponding to a transition trace is the sequence of modes through which the transition trace passes. Last we define a *run* or *trajectory* (cf. e.g. [48]) corresponding to a trace (and a transition trace) as a function from time to n-tuples of values for all n variables. In the next section we define trajectories over a generalised time domain in more detail.

Example 2.2 (*Thermostat continued*). Formally, the hybrid automaton for the temperature control of Fig. 1 is defined by the set of variables $X = \{x\}$, the control modes $M = \{Off, On\}$, the control switches $E = \{(Off, On), (On, Off)\}$. The invariant function $inv(v)$ assigns $x \geq 18$ to mode *Off* and $x \leq 22$ to *On*. The flow condition $flow(v)$ is $\dot{x} = -0.1x$ inside mode *Off* and $\dot{x} = 5 - 0.1x$ inside *On*. An initial condition exists only for the mode *Off* and sets the value $x = 20$. Finally the jump conditions are defined by $x < 19$ for the edge (Off, On) and $x > 21$ for (On, Off).

One possible trajectory is illustrated in Fig. 2.

3. Trajectory-based model

3.1. Basic algebra of hybrid systems

As already mentioned, trajectories reflect the variation of the values of the variables over time. Let V be a set of *values* and D a set of *durations* (e.g. $\mathbb{N}$, $\mathbb{Q}_{\geq 0}$, $\mathbb{R}_{\geq 0}, \ldots$). We assume a cancellative addition $+$ on D and an element $0 \in D$ such that $(D, +, 0)$ is a commutative monoid. Furthermore, we assume that the relation $d_1 \leq d_2 \Leftrightarrow_{df} \exists d \,.\, d_1 + d = d_2$ is a linear order on D. Then 0 is the least element and $+$ is isotone w.r.t. $\leq$. Moreover, 0 is indivisible, i.e., $d_1 + d_2 = 0 \Leftrightarrow d_1 = d_2 = 0$. D may include the special value ∞. If so, ∞ is required to be an annihilator w.r.t. $+$ and hence is the greatest element of D (and cancellativity of $+$ is restricted to elements in $D - \{\infty\}$). For $d \in D$ we define the interval $\mathsf{intv}\, d$ of admissible times as

$$\mathsf{intv}\, d =_{df} \begin{cases} [0, d] \text{ if } d \neq \infty \\ [0, d[\text{ otherwise.} \end{cases}$$

A *trajectory* τ is a pair (d, g), where $d \in D$ and $g : \mathsf{intv}\, d \to V$. Then d is the *duration* of the trajectory and the image of $\mathsf{intv}\, d$ under g is its *range* $\mathsf{ran}\,(d, g)$.

A special role is played by *zero-length trajectories* of the form $\underline{x} =_{df} (0, g)$ with $x \in V$ and $g(0) =_{df} x$; they represent single values of the system.

We define composition of trajectories (d_1, g_1) and (d_2, g_2) as

$$(d_1, g_1) \cdot (d_2, g_2) =_{df} \begin{cases} (d_1 + d_2, g) & \text{if } d_1 \neq \infty \wedge g_1(d_1) = g_2(0) \\ (d_1, g_1) & \text{if } d_1 = \infty \\ \text{undefined} & \text{otherwise} \end{cases}$$

with $g(t) = g_1(t)$ for all $t \in [0, d_1]$ and $g(t + d_1) = g_2(t)$ for all $t \in \mathsf{intv}\, d_2$. This is well defined by cancellativity of $+$ on durations other than ∞.

Fig. 3 illustrates the main idea for composing trajectories. Sometimes the condition $g_1(d_1) = g_2(0)$ for composing trajectories is too restrictive. In Section 3.3 we present a possibility to relax the condition and allow jumps at the composition point for the function describing the timewise behaviour.

For a zero-length trajectory $\underline{v}$ we have $\underline{v} \cdot (d, g) = (d, g)$ if $v = g(0)$; otherwise the composition is undefined. Likewise, $(d, g) \cdot \underline{v} = (d, g)$ if $v = g(d)$ or $d = \infty$.

A *process* is a set of trajectories, consisting of possible behaviours of a hybrid system. Note that we do not put any restrictions (such as prefix-closure) on a process. The set of all processes is denoted by PRO.

The greatest process, namely the set of all trajectories, is denoted by TRA.

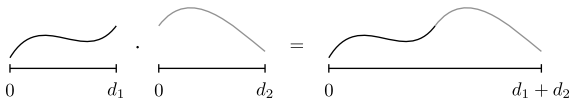

Fig. 3. Composition of two finite trajectories.

For a discrete infinite set of durations D, e.g. $D = \mathbb{N}$, trajectories are isomorphic to nonempty finite or infinite words over the value set V. Moreover if V consists of values of computations, then the elements of PRO can be viewed as sets of computation streams (e.g. [13]).

The *purely finite* and *purely infinite parts* of a process A are defined as

$$\inf A =_{df} \{(d,g) \mid (d,g) \in A,\ d = \infty\}, \qquad \mathsf{fin}\, A =_{df} A - \inf A.$$

Composition is lifted to processes A, B as follows:

$$A \cdot B =_{df} \inf A \cup \{a \cdot b \mid a \in \mathsf{fin}\, A,\ b \in B,\ a \cdot b \text{ defined}\} \tag{1}$$

The set I of all zero-length trajectories is the neutral element for this operation. A restricted form of composition, the *chop* $A^\frown B$, yields only trajectories that, after a finite trajectory of A, actually enter the second process. It is defined as $A^\frown B =_{df} (\mathsf{fin}\, A) \cdot B$, which implies $A \cdot B = (\inf A) \cup A^\frown B$.

Sets of zero-length trajectories, corresponding to sets of values, can be used to restrict processes. Let R be such a set and A be an arbitrary process. Then $R \cdot A$ consists of those trajectories of A whose initial value lies in R, while $A \cdot R$ is the set of trajectories of A whose final value, if any, is in R.

Example 3.1 (*Thermostat continued*). To use trajectories for our thermostat example, we first set $V = D = \mathbb{R}$. Now we define two processes, one for each control mode:

$$A^{\mathrm{Off}} =_{df} \{(d,x) \mid d \in D,\ \dot{x} = -0.1x\},$$
$$A^{\mathrm{On}} =_{df} \{(d,x) \mid d \in D,\ \dot{x} = 5 - 0.1x\}.$$

A^{Off} models all possible behaviours when the heater is off, whereas A^{On} describes the thermostat when the heater is on. The (singleton) set of possible initial values is given by $R_{20} =_{df} \{\underline{20}\}$. Hence, we can formalise the starting sequence of the thermostat described above as

$$R_{20} \cdot A^{\mathrm{Off}} \cdot A^{\mathrm{On}}.$$

Note that so far we have not modelled jump and invariant conditions. For this we use sets of zero-length trajectories describing sets of values and restrict the ranges of trajectories accordingly. Generally, we represent an interval of values as a set of zero-length trajectories by setting

$$R_{[l,u]} =_{df} \{\underline{x} \mid x \in [l,u]\}.$$

Then the sequence "*Off–jump–On*" equals $A^{\mathrm{Off}} \cdot R_{[18,19]} \cdot A^{\mathrm{On}}$. This eliminates from the full composition $A^{\mathrm{Off}} \cdot A^{\mathrm{On}}$ all trajectories in which the temperature at the joining point is outside the interval [18,19].

Since we want to describe the whole behaviour of the thermostat, we need the possibility for iteration. Let * and $^\omega$ be operators for finite and infinite iteration (we will show their existence in Section 3.4). Then the whole system is described by

$$R_{20} \cdot T^* \text{ or } \quad R_{20} \cdot T^\omega,$$

where $T =_{df} A^{\mathrm{Off}} \cdot R_{[18,19]} \cdot A^{\mathrm{On}} \cdot R_{[21,22]}$.

In such a way, any hybrid automaton can be replaced by a corresponding regular-like expression. This is shown in Section 4.1. Before that, we provide in Section 3.3 a method for modelling proper jumps by introducing an additional compatibility relation.

3.2. Algebraic structure

But first let us have a closer look at the algebraic structure of the basic algebra of hybrid systems.

A *left semiring* is a quintuple $(S, +, 0, \cdot, 1)$ such that $(S, +, 0)$ is a commutative monoid and $(S, \cdot, 1)$ is a monoid such that $\cdot$ is left-distributive over $+$ and *left-strict*, i.e., $0 \cdot a = 0$. The left semiring is *idempotent* if $+$ is idempotent and $\cdot$ is right-isotone, i.e., $a + a = a$ and $b \leq c \Rightarrow a \cdot b \leq a \cdot c$, where the *natural order* $\leq$ on S is given by $a \leq b \Leftrightarrow_{df} a + b = b$. Left-isotony of $\cdot$ follows from its left-distributivity. Moreover, 0 is the $\leq$-least element. A *weak semiring* is a left semiring in which $\cdot$ is also right-distributive. A *semiring* is a weak semiring in which composition is also right-strict; when we want to emphasise this, we also speak of a *full* semiring.

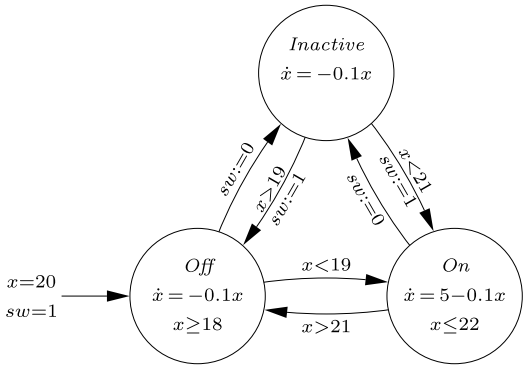

Fig. 4. Extended thermostat automaton.

The natural order induces an upper semilattice in which $a + b$ is the supremum of a and b and 0 is the least element. A left semiring is *Boolean* if this semilattice is even a Boolean algebra with complement $\overline{a}$ and infimum $a \sqcap b =_{df} \overline{\overline{a} + \overline{b}}$. In this case we have the *shunting rule*

$$a \sqcap b \leq c \Leftrightarrow a \leq \overline{b} + c. \tag{2}$$

An idempotent left/weak semiring S is called a *left/weak quantale* if S is a complete lattice under the natural order and $\cdot$ is universally disjunctive in its left argument. Following [16], one might also call a left quantale a *left standard Kleene algebra*.

An important Boolean semiring (that is even a weak quantale with universally right-disjunctive composition) is REL, the algebra of binary relations over a set under relational composition.

Checking all the axioms for the case of processes, we get

Lemma 3.2

(1) *The processes under union as addition and composition as multiplication form a Boolean weak quantale* $\mathrm{PRO} =_{df} (\mathcal{P}(\mathrm{TRA}), \cup, \emptyset, \cdot, I)$.
(2) *Additionally,* $\cdot$ *is positively disjunctive in its right argument, and chop inherits the disjunctivity properties from* $\cdot$ *and is associative, too.*

3.3. Adding proper jumps

The constraint $g_1(d_1) = g_2(0)$ for composability of trajectories (d_1, g_1) and (d_2, g_2) is very restrictive in a number of situations.

Example 3.3 (*Thermostat continued*). We extend Example 2.1 by an additional switch *sw* that activates or deactivates the whole temperature control. Therefore this example contains proper jumps (in the behaviour of the switch) as well as "non-proper" jumps (in the change of temperature). The values $sw = 1$ and $sw = 0$ represent the situations where the control is activated and deactivated, respectively. The whole system is illustrated in Fig. 4.

The system can always be deactivated by setting the switch to 0 (independent of the current temperature). When reactivating the system there is a choice between the modes *Off* and *On*. It is a genuine non-deterministic choice if the temperature is between 19 and 21 degrees. Remember that all modes implicitly contain the equation $\dot{sw} = 0$.

To relax the composition of trajectories we introduce a *compatibility relation* $\asymp \subseteq V \times V$ that describes the behaviour at the point of composition. It allows certain proper jumps at the connection point between two trajectories (d_1, g_1) and (d_2, g_2). This is meaningful, since jumps within trajectories are already allowed by our definition. Note that we do not postulate any condition for $\asymp$. But in most cases $\asymp$ will be at least reflexive to accommodate the case of equal values $g_1(d_1)$ and $g_2(0)$. If one wants to *enforce* jumps at every composition point, $\asymp$ has to be irreflexive (like in our example).

To model a more liberal form of composition that takes $\asymp$ into account, we extend a finite trajectory (d, g) at the right end, i.e., at time d, using the compatibility relation. To this end we express that, up to $\asymp$, we do not care about the exact final value $g(d)$. Therefore we inflate the original trajectory to a process that before time d agrees with the original trajectory, but shows all values admitted by $\asymp$ at time d:

$$(d, g)_{\asymp} =_{df} \{(d, \hat{g}) \mid \hat{g}(x) = g(x),\ x \in [0, d[,\ g(d) \asymp \hat{g}(d)\}.$$

Since for an infinite trajectory (d, g) a right composition partner does not matter anyway, we set $(d, g)_{\asymp} =_{df} \{(d, g)\}$ if $d = \infty$.

The composition of (d_1, g_1) and (d_2, g_2) considering the compatibility relation $\asymp$ is then the composition

$$(d_1, g_1)_{\asymp} \cdot \{(d_2, g_2)\}$$

over PRO. The extension operation is lifted pointwise to processes. We have decided not to incorporate the compatibility relation into the definition of trajectory composition, since it would be technically cumbersome to do so.

Symmetrically, we can also employ the compatibility relation at the left end or even at both ends of a trajectory.

Example 3.4 (*Thermostat continued*). In our example the set V of values is given by $\mathbb{R} \times \{0, 1\}$, where the first component of a pair represents the temperature and the second one the value of the switch. We define the compatibility relation as

$$\asymp \; =_{df} \{((x,0),(x,1)) \mid x \in \mathbb{R}\} \cup \{((x,1),(x,0)) \mid x \in \mathbb{R}\}.$$

The set of trajectories that start in the mode *Off* and then go to *Inactive* is described by

$$A^{\mathrm{Off}}_{\asymp} \cdot A^{\mathrm{In}},$$

where A^{Off} and A^{In} describe the behaviours inside the modes *Off* and *Inactive*. In particular, $A^{\mathrm{Off}} =_{df} \{(d,g) \mid d \in D,\ g(t) = (x(t),1),\ \dot{x} = -0.1x\}$ and A^{In} can be constructed in a similar way.

3.4. Finite and infinite iteration

Now we turn to an algebraic characterisation of iteration. A *left Kleene algebra* [42] is a structure $(S, {}^*)$ consisting of an idempotent semiring S and an operation * for iterating an element an arbitrary but finite number of times. Such an operation has to satisfy the left *unfold* and *induction* axioms

$$1 + a \cdot a^* \leq a^*, \qquad b + a \cdot c \leq c \Rightarrow a^* \cdot b \leq c. \tag{3}$$

To express infinite iteration we axiomatise an ω operator over a left Kleene algebra. A *left omega algebra* [15,42] is a pair $(S, {}^\omega)$ such that S is a left Kleene algebra and ${}^\omega$ satisfies the *unfold* and *coinduction* axioms

$$a^\omega = a \cdot a^\omega, \qquad c \leq a \cdot c + b \Rightarrow c \leq a^\omega + a^* \cdot b. \tag{4}$$

Two consequences of these axioms are that each omega algebra has the greatest element $\top =_{df} 1^\omega$ and that $a^\omega = a^\omega \cdot \top$ (see [42]).

Concerning the existence of these operations we can use the fact that PRO is a left quantale, hence, in particular, a complete lattice, and the Knaster/Tarski fixpoint theorem (e.g. [19]). We denote by $\mu x \,.\, f(x)$ and $\nu x \,.\, f(x)$ the least and greatest fixpoints of an isotone function f from a complete lattice to itself.

Lemma 3.5

(1) *Every left quantale can be extended to a left Kleene algebra by defining* $a^* =_{df} \mu x \,.\, a \cdot x + 1$.

(2) *If the left quantale is weak and a completely distributive lattice then it can be extended to a left* ω *algebra by setting* $a^\omega =_{df} \nu x \,.\, a \cdot x$. *In this case,*

$$\nu x \,.\, a \cdot x + b \;=\; a^\omega + a^* \cdot b.$$

For the proof see the Appendix.

Since, by Lemma 3.2, PRO forms a weak quantale, we have finite iteration * and infinite iteration ${}^\omega$ with all their laws available for processes.

3.5. Purely finite and purely infinite elements

In Section 3.1 we already introduced the purely finite and purely infinite parts of a process. A general algebraic treatment of these notions can be performed using their behaviour under composition. Definition (1) entails, for process $A \in \mathrm{PRO}$, that $A \cdot \emptyset = \inf A$. Hence a process is purely infinite, i.e., consists of infinite trajectories only, if $A = \inf A = A \cdot \emptyset$. Dually, a process B is purely finite, i.e., consists of finite trajectories only, if its purely infinite part is trivial, that is, if $\inf B = \emptyset$.

Hence, for an idempotent left semiring S, we define the purely infinite part of $a \in S$ as $\inf a =_{df} a \cdot 0$ and call *a purely infinite* if $a \cdot 0 = a$. This property is equivalent to a being a *left zero*, i.e., to $\forall b : a \cdot b = a$. Often there exists a largest purely infinite element N characterised by $a \leq \mathsf{N} \Leftrightarrow a \cdot 0 = a$. In PRO, $\mathsf{N} = \{(d,g) : d = \infty\}$ is the set of all trajectories of infinite length. The definition of N implies, for all a,

$$\mathsf{N} \cdot a \leq \mathsf{N} \quad \text{and} \quad a \cdot \mathsf{N} \leq \mathsf{N}. \tag{5}$$

Dually, we call an element *a purely finite* if $\inf a = a \cdot 0 = 0$, i.e., if its purely infinite part is trivial. In many semirings there exists a largest purely finite element F characterised by $a \leq \mathsf{F} \Leftrightarrow a \cdot 0 = 0$. In PRO, $\mathsf{F} = \{(d,g) : d < \infty\}$ contains all trajectories of finite length. The definition of F implies

$$\mathsf{F} \cdot \mathsf{F} = \mathsf{F}. \tag{6}$$

In Boolean left quantales N and F always exist and satisfy

$$\mathsf{N} = \top \cdot 0, \qquad \mathsf{F} = \overline{\mathsf{N}},$$

where $\top =_{df} \overline{0}$ denotes the greatest element. Moreover, in this case every element can be split into its purely finite and purely infinite parts: $a = \mathsf{fin}\, a + \mathsf{inf}\, a$, where $\mathsf{fin}\, a = a \sqcap \mathsf{F}$ and $\mathsf{inf}\, a = a \sqcap \mathsf{N}$. An idempotent left semiring with this property is called *separated*; for further details see [42]. The above equations imply

$$(\mathsf{inf}\, a) \;\sqcap\; b = \mathsf{inf}\,(a \sqcap b), \qquad (\mathsf{fin}\, a) \;\sqcap\; b = \mathsf{fin}\,(a \sqcap b). \tag{7}$$

The purely finite and purely infinite parts of a composition satisfy

$$a \cdot b \;=\; \mathsf{inf}\, a + \mathsf{fin}\, a \cdot b, \tag{8}$$

$$\mathsf{inf}\,(a \cdot b) \;=\; \mathsf{inf}\, a + \mathsf{fin}\, a \cdot \mathsf{inf}\, b, \tag{9}$$

$$\mathsf{fin}\,(a \cdot b) \;=\; \mathsf{fin}\,(\mathsf{fin}\, a \cdot b) \;\geq\; \mathsf{fin}\, a \cdot \mathsf{fin}\, b. \tag{10}$$

If S is weak, the latter inequation strengthens to an equality.

We now state further general laws concerning purely finite and purely infinite parts.

Lemma 3.6. *Let S be a Boolean left quantale and $a, b, c, d \in S$.*

(1) $a \leq \mathsf{F} \Leftrightarrow a = \mathsf{fin}\, a \Leftrightarrow \mathsf{inf}\, a = 0$ *and*
$a \leq \mathsf{N} \Leftrightarrow a = \mathsf{inf}\, a \Leftrightarrow \mathsf{fin}\, a = 0$.
(2) *For* $a, b \leq \mathsf{F}$ *and* $c, d \leq \mathsf{N}$ *we have* $a + c \leq b + d \Leftrightarrow a \leq b \wedge c \leq d$.
(3) $a^\omega = (\mathsf{fin}\, a)^* \cdot \mathsf{inf}\, a + (\mathsf{fin}\, a)^\omega$,
(4) $\mathsf{inf}\, a^\omega = (\mathsf{fin}\, a)^* \cdot \mathsf{inf}\, a + \mathsf{inf}\,((\mathsf{fin}\, a)^\omega)$,
(5) $\mathsf{fin}\, a^\omega = (\mathsf{fin}\, a)^\omega \sqcap \mathsf{F} \leq (\mathsf{fin}\, a)^\omega$.

The proof is given in the Appendix.

Part (1) gives equivalent characterisations of purely finite and purely infinite elements which are calculationally useful in various circumstances. Part (2) means that a sum of a purely finite and a purely infinite element can uniquely be decomposed again. If a is a process, Part (3) says that infinite iteration of trajectories from a can take two forms: it may proceed a while with finite trajectories, but then add an infinite trajectory which prohibits further iteration — or it keeps iterating finite trajectories forever.

Parts (4) and (5) follow from that using Part (2). Part (5) fits well with intuition, since in PRO it means that Zeno effects (infinite iterations that take finite duration) can only occur when some trajectories in a process a are finite. Part (4) says that infinite behaviour results from entering an infinite part after a finite iteration of finite parts of the iterated process or by iterating finite parts of that process that all have long enough durations that their infinite iteration takes infinite duration. In the next section we will look at Zeno effects in detail.

3.6. Zeno effects

Zeno of Elea's famous paradox of Achilles and the tortoise is well known. However, with few exceptions (e.g. [7,34]) authors do not treat Zeno effects within hybrid systems in detail, even if they appear in their theoretical models. In this section we present a possible way of handling Zeno effects in PRO and characterise the Zeno and Zeno-free parts of hybrid systems.

Roughly spoken, a Zeno effect occurs if an infinite iteration does not take infinite duration.

To speak about such phenomena we can use the purely finite and purely infinite parts of processes defined in Section 3.5. Furthermore, it is useful to determine A^ω for a process $A \in \mathrm{PRO}$.

For a purely infinite process A it is easy to see that $A^\omega = A$. For an arbitrary process infinite iteration can be determined by the general decomposition law $a^\omega = (\mathsf{fin}\, a)^* \cdot \mathsf{inf}\, a + (\mathsf{fin}\, a)^\omega$ (see Lemma 3.6 (3)). Therefore it suffices to determine A^ω for purely finite processes A.

We define the *prefix relation* $\sqsubseteq$ between trajectories $\tau_1 = (d_1, g_1)$ and $\tau_2 = (d_2, g_2)$ by

$$\tau_1 \sqsubseteq \tau_2 \;\Leftrightarrow_{df}\; d_1 \leq d_2 \wedge g_2|_{\mathsf{intv}\, d_1} = g_1.$$

The first conjunct on the right hand side is equivalent to $\mathsf{intv}\, d_1 \subseteq \mathsf{intv}\, d_2$; the stroke $|_X$ means function restriction to subset X. It is easy to see that $\sqsubseteq$ is a partial order with $\tau_1 \sqsubseteq \tau_2 \;\Leftrightarrow\; \exists \tau_3 : \tau_1 \cdot \tau_3 = \tau_2$. Moreover, if $\tau_1 \sqsubseteq \tau_2$ then $\tau_3 \cdot \tau_1 \sqsubseteq \tau_3 \cdot \tau_2$. Infinite trajectories are maximal w.r.t. this order.

To describe infinite concatenations of trajectories from a purely finite process A, let $\mathrm{ISEQ}(A)$ be the set of infinite sequences $\mathrm{T} = (\tau_n)_{n \in \mathbb{N}}$ of trajectories $\tau_n \in A$ such that all iterated compositions $\pi_n =_{df} \prod_{m<n} \tau_m (n \in \mathbb{N})$ are defined. By the above remarks these satisfy $\pi_n \sqsubseteq \pi_{n+1}$ and hence form an ascending chain w.r.t. $\sqsubseteq$, a set of longer and longer trajectories that agree in their initial parts. Infinite iteration then results by passing to some sort of limit for such a chain. We reflect this idea by writing

$\prod \mathrm{T} \sqsubseteq \tau$, for arbitrary trajectory $\tau \in \mathrm{TRA}$ and sequence $\mathrm{T} \in \mathrm{ISEQ}(A)$, if for all $n \in \mathbb{N}$ we have $\pi_n \sqsubseteq \tau$ with π_n defined as above. This means that τ has the "infinite composition" of the τ_n as a prefix.

Theorem 3.7. *Let A be a purely finite process and define the function $H : \mathrm{PRO} \to \mathrm{PRO}$ by $H(X) =_{df} A \cdot X$.*
(1) *Let X be expanded by H, i.e., assume $X \subseteq H(X)$. Then for every $\xi \in X$ there is sequence $\mathrm{T} \in \mathrm{ISEQ}(A)$ with $\prod \mathrm{T} \sqsubseteq \xi$.*
(2) $A^\omega = \{\tau \in \mathrm{TRA} \mid \exists \mathrm{T} \in \mathrm{ISEQ}(A) : \prod \mathrm{T} \sqsubseteq \tau\}$.

The proof can be found in the Appendix.

The fact that A^ω contains arbitrary extensions of infinite A-iterations also explains why the property $A^\omega = A^\omega \cdot \top$ (see Section 3.4) is not completely unnatural: for arbitrary $B \in \mathrm{PRO}$ the process $B \cdot \top$ is the extension closure of B. Hence $A^\omega = A^\omega \cdot \top$ reflects the fact that, operationally, after a Zeno gap the behaviour doesn't matter, since the gap cannot be "crossed" anyway.

Now, generalising from PRO to a weak omega algebra S, we call an element $a \in S$ *divergent* or *Zeno-free*, if $a^\omega \leq \mathsf{N}$. An element a is called *Zeno* if it is not Zeno-free and it is called *convergent* if $a^\omega \leq \mathsf{F}$. The least element 0 is the only element which is convergent, divergent and Zeno-free, since $0^\omega = 0$. It is straightforward to see that in full semirings (where 0 is also a right annihilator) every element is convergent.

However, A^ω is not completely adequate for reasoning about and exclusion of Zeno effects. For many purposes its extension-closedness gets in the way, since it yields a too loose description of infinite iteration. For that reason we introduce another iteration operator † which narrows down the set of possible behaviours. However, in contrast to ω, its definition works only for special time domains.

Let again A be purely finite and assume that the time domain D is complete, i.e., contains suprema for all its subsets. For a sequence $\mathrm{T} \in \mathrm{ISEQ}(A)$ with iterated compositions $\pi_n = (d_n, g_n)$ as above set $d_\mathrm{T} =_{df} \sup\{d_n \mid n \in \mathbb{N}\}$.

By a construction similar to the one used in Section 3.3 for the treatment of proper jumps, we define the process P_T by

$$\begin{aligned}(d_\mathrm{T}, g) \in P_\mathrm{T} \quad \Leftrightarrow_{df} \quad & g : \mathsf{intv}\, d_\mathrm{T} \to V, \\ & g(t) = g_n(t) \text{ if } t \leq d_n, \\ & g(d_\mathrm{T}) \in V \text{ arbitrary if } d_\mathrm{T} > d_n \text{ for all } n \in \mathbb{N} \text{ and } d_\mathrm{T} \neq \infty.\end{aligned}$$

For $d_\mathrm{T} = \infty$ the iteration does not show a Zeno effect and P_T is a singleton process consisting just of one infinite trajectory. For $d_\mathrm{T} \neq \infty$, two cases arise. First, we may have $d_\mathrm{T} = \max\{d_n \mid n \in \mathbb{N}\}$. This can only happen when the sequence T becomes stationary with infinitely many trajectories of length zero and identical value v at the end (if there were differing ones not all iterated compositions π_n would be defined). This means a special kind of Zeno behaviour, viz. "stepping on the spot" forever. Therefore in this case the value $g(d_\mathrm{T})$ agrees with v and P_T is again a singleton process. This entails the property

$$I^\dagger = I \tag{11}$$

for the multiplicative identity I of PRO, whereas $I^\omega = \top = \mathrm{TRA}$, so that the operator † indeed omits trailing behaviour. The second case, where $d_\tau \neq \max\{d_n \mid n \in \mathbb{N}\}$, i.e., $d_\mathrm{T} > d_n$ for all $n \in \mathbb{N}$, means "proper" Zeno behaviour where the trajectories in T become shorter and shorter while their iterated compositions become longer and longer without ever reaching the "limit time" d_T. To form proper trajectories out of the iterated compositions we add arbitrary values at d_T but nothing at times properly later than d_T. Now we set

$$A^\dagger =_{df} \bigcup_{\mathrm{T} \in \mathrm{ISEQ}(A)} P_\mathrm{T}.$$

With this construct, Zeno effects can be excluded by considering only the properly infinite trajectories in $\mathsf{inf}\, A^\dagger = A^\dagger \sqcap \mathsf{N}$. This could not be achieved reasonably with A^ω, since that includes trajectories which are infinite because they add an arbitrary infinite behaviour to a Zeno initial part. This is made precise by Part (1) of the following result.

Theorem 3.8. *Let H be as in Theorem 3.7.*
(1) *$A^\dagger$ is a fixpoint of H.*
(2) *Let X be expanded by H, i.e., assume $X \subseteq H(X)$. Then every $\xi \in X$ has a prefix in $A^\dagger$.*
(3) $A^\omega = A^\dagger \cdot \top$.

Again, the proof can be found in the Appendix.

An immediate consequence of Part (3) and Eq. (10) is that $A^\dagger$ and A^ω coincide iff A is Zeno-free.

Example 3.9 (*Thermostat continued*). We can now describe all non-Zeno behaviours as

$$R_{20} \cdot T^\dagger \sqcap \mathsf{N},$$

where T equals again $A^{\mathrm{Off}} \cdot R_{[18,19]} \cdot A^{\mathrm{On}} \cdot R_{[21,22]}$.

Example 3.10. To give another example, we define a scaling function $sc_n : \mathrm{TRA} \to \mathrm{TRA}$ with $sc_n(d, f) =_{df} (\frac{d}{n}, g)$, where $n \in \mathbb{N}$ and $g(x) = f(x \cdot n)$. Then, given a trajectory $T_1 = (d_1, f_1)$ with $f_1(0) = f_1(d)$, we define a process

$$P = \{sc_n(T_1) \mid n \in \mathbb{N}\}.$$

It is easy to see, that P^ω and $P^\dagger$ contain an infinite number of finite trajectories as well as an infinite number of infinite trajectories. Therefore, P is neither convergent nor divergent, but Zeno. $P^\dagger \sqcap \mathsf{F}$ (the Zeno-part of $P^\dagger$) is closely related to the paradox of Achilles and the tortoise, because P contains trajectories with arbitrarily short durations.

4. Embedding hybrid automata

4.1. The basic construction

We now show, in a generic way, how to model hybrid automata (see Section 2) using these concepts. Consider a hybrid automaton of dimension n with control graph (M, E). Then as value set we choose $V =_{df} M \times \mathbb{R}^n$.

In a given mode $v \in M$ the behaviour of the automaton in the interval $[0, d]$ coincides with a trajectory (d, g) such that $g(t) = (v, f(t))$ for some function $f : [0, d] \to \mathbb{R}^n$ that satisfies the invariant and flow conditions of v. This corresponds to Henzinger's relation $(v, f(0)) \xrightarrow{d} (v, f(d))$ [23].

The compatibility relation is given by

$$(v, x) \asymp (w, y) \Leftrightarrow (v = w \wedge x = y) \vee ((v, w) \in E \wedge jump(v, w)(x, y)),$$

where the first part $(v = w \wedge x = y)$ deals with compositions that do not leave a control mode and the second part models the event belonging to the edge (v, w) (if the edge is present).

The generic construction of an algebraic expression from a given automaton now proceeds by the following steps:

- For each control mode v of the automaton we define a process

$$P^v =_{df} \{(d, g) \mid d \in D, \forall t \in \mathsf{intv}\, d : g(t) = (v, f(t)),\ f : \mathsf{intv}\, d \to \mathbb{R}^n, \forall t \in \mathsf{intv}\, d : flow(v)(f(t), \dot{f}(t)),\ \forall x \in \mathsf{ran} f : inv(v)(x)\}.$$

- For each P^v determine $P^v_{\asymp}$ with $\asymp$ as above.
- In [35] Kleene has shown how to construct a regular expression from a given automaton. Similarly, this construction can be carried out with hybrid automata using the above processes $P^v_{\asymp}$. While in the original construction the star for finite iteration is used, here one has to decide, whenever iteration occurs, whether it should be finite or infinite iteration (* or $^\omega$).
- Note, that hybrid automata can include Zeno effects. Therefore such effects might also occur in the corresponding algebraic expressions. To avoid such behaviour one can replace $^\omega$ by † and apply a meet operation with the set of all infinite trajectories at the outermost level as in Example 3.9.

Often, it is not necessary to store the control mode in the value set, i.e., V can be chosen as $\mathbb{R}^n$ instead of $M \times \mathbb{R}^n$. Examples of this are given in Example 3.4 and Section 6.

4.2. Composition of hybrid automata

More complicated hybrid systems arise often by composing smaller systems. The product of two finite automata as well as the parallel composition are well known. Similar to these constructions Henzinger defines a product and a parallel composition for hybrid automata [23]. In this section we discuss their algebraic counterparts.

Product. Following Section 4.1 and the definition of product of hybrid automata [23] we define for two hybrid automata H_1, H_2 (with disjoint sets of modes M_1 and M_2) the following edge labelling functions (for jumps and events)

$$(v_1, v_2) \xrightarrow{a} (w_1, w_2) \Leftrightarrow_{df} \exists a_1 \in H_1, a_2 \in H_2 : v_1 \xrightarrow{a_1} w_1,\ v_2 \xrightarrow{a_2} w_2.$$

To translate this behaviour to our algebraic model we just look at the product semiring.

For two (left) semirings $(A, +_A, 0_A, \cdot_A, 1_A)$ and $(B, +_B, 0_B, \cdot_B, 1_B)$ the *product (semiring)* is defined as

$$(A \times B, +_\times, 0_A \times 0_B, \cdot_\times, 1_A \times 1_B),$$

where $+_\times$ and $\cdot_\times$ are componentwise operators. By standard results from universal algebra the product structure indeed forms a (left) semiring again. Furthermore the construction is equivalent to the product construction for hybrid automata.

Parallel composition. Parallel composition of hybrid systems can also be used for specifying larger systems. An algebraic expression or a hybrid automaton is given for each part of the system. Communication between the components may occur via shared variables and synchronisation labels. At first glance, the parallel composition seems to be more complicated than the product. But as we will see, it is easily handled in the algebraic model. It again uses a Cartesian product, like the product semiring (only in a different place). We consider again Henzinger's definition that only looks at synchronisation at transition points.

Two hybrid automata H_1 and H_2 with the same set D of durations are assumed to interact via common events.

First we look at "unsynchronised" parallel runs of hybrid systems. For trajectories $\tau_1 = (d_1, g_1)$ and $\tau_2 = (d_2, g_2)$ with $d_1, d_2 \in D$ we first define $\tau_1 \| \tau_2$ for some special cases:

$$\tau_1 \| \tau_2 = \begin{cases} (d_1, g_1 \triangledown g_2) & \text{if } d_1 = d_2 \\ (d_2, \mathtt{const}_{d_2}(g_1) \triangledown g_2) & \text{if } d_1 = 0 \\ (d_1, g_1 \triangledown \mathtt{const}_{d_1}(g_2)) & \text{if } d_2 = 0, \end{cases}$$

where $(f \triangledown g)(x) = (f(x), g(x))$ and $\mathtt{const}_d(f)(x) = f(0)$ is the constant function on $[0, d]$. We see that in the case of two semirings of processes with the same set of durations the parallel-composed trajectories form again trajectories. Viz., if the first process contains only trajectories τ_1 with functions $g_1 : \mathsf{intv}\, D \to V$ and for all trajectories τ_2 of the second process we have $g_2 : \mathsf{intv}\, D \to V'$, then the parallelised process semiring contains trajectories with functions of type $\mathsf{intv}\, D \to V \times V'$. The cross product avoids the problem of shared variables by duplicating them. Below we show how to synchronise two systems at transition points.

Often the above definition is sufficient. But, sometimes one also has to consider the cases $0 < d_1 < d_2$ or $0 < d_2 < d_1$. For those cases there are some choices and decisions to be made. For example: should the trajectories start at a common time point? Should they end after the same duration?

If $d_1 < d_2$, then, by definition of the order on D, there exists $d_3 \in D$ with $d_1 + d_3 = d_3 + d_1 = d_2$. Therefore the trajectory $\tau_1 = (d_1, g_1)$ can be lengthened to duration d_2 using constant trajectories as

$$\tau_1 \cdot (d_3, g_3) \quad \text{or} \quad (d_3, g_3') \cdot \tau_1,$$

where $g_3(x) = g_1(d_1)$ and $g_3'(x) = g_1(0)$. Using the first of these products in the parallel composition $(\tau_1 \cdot (d_3, g_3)) \| \tau_2$ means that the trajectories τ_1 and τ_2 start at a common time point, whereas $((d_3, g_3') \cdot \tau_1) \| \tau_2$ enforces that τ_1 and τ_2 end together. Again this operation can be lifted to processes.

Next, we want to synchronise H_1 and H_2 via reachable events, i.e., events that have to occur after a finite duration. If a is a common event of H_1 and H_2, then H_1 and H_2 must synchronise on a-transitions after a finite duration; if a is an event of H_1 but not of H_2 then during the transition of H_1 the state of H_2 has to be kept constant and vice versa.

Following Section 3.3, transitions (and thus events) can be modelled by a compatibility relation $\asymp$ and zero-length trajectories. Let X be the set of shared variables to be synchronised. Then post-multiplying with the process

$$\{(0, g_1 \triangledown g_2) \mid g_1|_X = g_2|_X\},$$

where $|_X$ restricts the domains to X, enforces synchronisation.

Synchronisation of infinite trajectories can only be done after a finite initial duration. In the case of hybrid automata the set of durations is $\mathbb{R}$. Hence each infinite trajectory τ contains prefixes of arbitrary length, i.e., for all trajectories τ and for all $d \in \mathbb{R}$ it holds that

$$\exists \tau_1, \tau_2 : \tau = \tau_1 \cdot \tau_2,$$

where the duration of τ_1 is d. Therefore, one can use the synchronisation for finite trajectories also for infinite ones. Synchronisation after an infinite amount of time does not make sense.

An example for composing hybrid systems is given in Section 6.

5. Safety and liveness

5.1. Modularity and progress in time

In Section 3.6 we restricted processes to their Zeno-free parts. Now, we want to deal with the general case that a process is restricted by an additional condition. Abstractly, let a stand for the process and c for the condition; then we want to form the meet $a \sqcap c$. If a is a composite process we want to distribute the condition to its components if possible. If a is a sum this is easy. However, if a is a product, we need special conditions for c to do this.

We call an element c *submodular* if $\forall a, b \in S : c \sqcap (a \cdot b) \leq (c \sqcap a) \cdot (c \sqcap b)$ and *modular* if in that formula always $=$ holds instead of just $\leq$. We obtain useful characterisations of these properties. Note that by Eq. (10) in a weak semiring the element F is modular.

The following lemma summarises elementary properties for submodular elements. They will be used in the remainder to prove useful statements concerning processes.

Lemma 5.1. *Assume a Boolean weak quantale S.*

(1) *The following properties are equivalent.*
   (a) *Element $c \in S$ is submodular.*
   (b) $(c \sqcap \mathsf{F}) \cdot \overline{c} + \overline{c} \cdot \top \leq \overline{c}$.
   (c) $\mathsf{F} \cdot \overline{c} \cdot \top \leq \overline{c}$

   In particular, 1 is submodular iff $\overline{1} \cdot \overline{1} \leq \overline{1}$.

(2) *Element $c \in S$ is modular iff it is submodular and transitive, i.e., satisfies $c \cdot c \leq c$. In particular, 1 is modular iff $\overline{1} \cdot \overline{1} \leq \overline{1}$.*

(3) *If c is modular then for all a we have $c \sqcap a^+ = (c \sqcap a)^+$ and $c \sqcap a^* = (c \sqcap a)^+ + (c \sqcap 1)$, with $b^+ =_{df} b \cdot b^*$.*
(4) *If c is modular then $d \le c$ is submodular iff $(c \sqcap \mathsf{F}) \cdot (c \sqcap \overline{d}) \cdot c \le c \sqcap \overline{d}$.*

The proof is given in the Appendix.

By the shunting Rule (2) the property $\overline{1} \cdot \overline{1} \le \overline{1}$ is equivalent to $1 \le \overline{\overline{1} \cdot \overline{1}}$. The element $\overline{\overline{1} \cdot \overline{1}}$ has been called step in von Karger's work [50]; it represents the elements that cannot be decomposed into non-subidentities. Since we can think of the identity element 1 as a process that does not proceed in time, this property says that progress in time cannot be undone by composition. Therefore we call a Boolean semiring with the property $\overline{1} \cdot \overline{1} \le \overline{1}$ *progressive*.

5.2. Time requirements

Often, it is useful to restrict the duration; for example to guarantee that an event happens after a certain time.

One way of asserting this is already given by the chop operator. Every trajectory in $A^\frown B$ guarantees that, unless Zeno effects occur, a suffix in process B is actually reached. To guarantee that B is reached after a certain time d one has to restrict A in a different way.

Example 5.2 (*Thermostat continued*). Returning to Example 3.3 we now want to guarantee that the heater is inactive for at most 30 time steps. Therefore we have to restrict A^{In} by the process $A =_{df} \{(d,g) \mid d \le 30, (d,g) \in \mathrm{TRA}\}$, i.e., we have to calculate $A^{\mathrm{In}} \sqcap A$. This process is the same as

$$\{(d,g) \mid d \le 30, (d,g) \in A^{\mathrm{In}}\}.$$

Note that A is not submodular.

This gives a straightforward way to model time assertions.

5.3. Range assertions and tests

Next to that, it may also be necessary to restrict the range of a process A. Here, the range $\mathsf{ran}\, A$ is defined as $\mathsf{ran}\, A =_{df} \bigcup_{t \in A} \mathsf{ran}\, t$.

Example 5.3 (*thermostat continued*). Extending Example 2.1 we want to define a process containing all trajectories that never leave the range [18,22].

We do this by observing that every subset W of the value set V is isomorphic to the process $P_W =_{df} \{\underline{x} \mid x \in W\}$.

With $\top = \mathrm{TRA}$ and $\mathsf{F} = \mathsf{fin}(\mathrm{TRA})$ we define

$$\Diamond P_W =_{df} \mathsf{F} \cdot P_W \cdot \top, \qquad \Box P_W =_{df} \overline{\Diamond \neg P_W}.$$

Hence, $\Diamond P_W$ is the set of all trajectories that at some (finite) point in their time interval have a value in W, while $\Box P_W$ describes a safety aspect, viz. the set of all trajectories whose range satisfies the "invariant" W, i.e., $\Box P_W = \{\tau \mid \tau \in \mathrm{TRA}, \mathsf{ran}\, \tau \subseteq W\}$. Thus, the requested safety condition for the thermostat can be modelled as $\Box R_{[18,22]}$. Dually, $\Diamond P_W$ can be used to describe certain liveness aspects.

Looking again at the safety requirement of the thermostat we see that by the condition $A^{\mathrm{Off}} \cdot A^{\mathrm{On}} \le \Box R_{[18,22]}$ we indeed restrict the range of $A^{\mathrm{Off}} \cdot A^{\mathrm{On}}$ as claimed in the beginning of this section. Using the meet

$$A^{\mathrm{Off}} \cdot A^{\mathrm{On}} \sqcap \Box R_{[18,22]} \tag{th-rest}$$

is another way to enforce the restriction.

For an algebraic characterisation of processes like P_W we use the idea of tests as introduced into semirings by [39] and into Kleene algebras by Kozen [36]. One defines a *test* in an idempotent left semiring (quantale) to be an element $p \le 1$ that has a complement q relative to 1, i.e., $p + q = 1$ and $p \cdot q = 0 = q \cdot p$. The set of all tests of S is denoted by $\mathsf{test}(S)$. It is not hard to show that the complement $\neg p$ of a test p is uniquely determined by the definition and that in a weak semiring $\mathsf{test}(S)$ is closed under $+$ and $\cdot$ and forms a Boolean algebra with 0 and 1 as its least and greatest elements. (To establish this in general left semirings one has to add the assumption $p \cdot (q + r) = p \cdot q + p \cdot r$ of right-distributivity of tests among each other.) In particular, we have the *shunting rule* for tests p, q, r:

$$p \cdot q \le r \Leftrightarrow p \le \neg q + r. \tag{12}$$

Moreover, all tests are purely finite. If S itself is Boolean, then $\mathsf{test}(S)$ coincides with the set of all elements below 1.

With the above definition of tests we deviate slightly from [36], where an arbitrary Boolean algebra of subidentities is allowed as $\mathsf{test}(S)$. The reason is that, as shown in Theorem 4.15 of [21], the axiomatisation of domain to be presented below forces every complemented subidentity to be in $\mathsf{test}(S)$ anyway.

We will consistently write $a, b, \ldots$ for arbitrary semiring elements and $p, q, \ldots$ for tests.

An important property of left semirings is distribution of test multiplication over meet [42]: if the meet $a \sqcap b$ exists then so do the meets $p \cdot a \sqcap b$ and $p \cdot a \sqcap p \cdot b$ and satisfy

$$p \cdot (a \sqcap b) \;=\; p \cdot a \sqcap b \;=\; p \cdot a \sqcap p \cdot b. \tag{13}$$

If S is right-distributive, also the symmetric properties hold.

Furthermore, if S is Boolean, we have the relationships

$$\overline{p \cdot a} = \neg p \cdot a + \overline{a}, \quad \text{in particular,} \quad \overline{p} = \neg p + \overline{1}, \tag{14}$$

and the symmetric ones if S is right-distributive.

Lemma 5.4. *Consider the Boolean weak quantale* PRO.

(1) $\mathsf{test}(\mathrm{PRO}) = \mathcal{P}(\{\underline{x} \mid x \in V\})$.

(2) *For* $P \in \mathsf{test}(\mathrm{PRO})$ *we have* $P^{\dagger} = P$ *and consequently* $P^{\omega} = P \cdot \top$.

(3) *Since* 0 *is indivisible in* PRO, *the meet with a test distributes over composition, i.e., all tests in* PRO *are modular*:

$$P \in \mathsf{test}(\mathrm{PRO}) \Rightarrow P \sqcap A \cdot B = (P \sqcap A) \cdot (P \sqcap B).$$

We have already used the tests of PRO for modelling restrictions and jump conditions in Section 3.1. Part (2) generalises to the law $p^{\omega} = p \cdot \top$ for arbitrary tests p in an omega algebra. Finally, it turns out that, even for arbitrary semirings, Part (3) is equivalent to the progressivity condition introduced at the end of Section 5.1:

Lemma 5.5. *All tests of a semiring S are modular iff S is progressive.*

Proof. ($\Rightarrow$) follows by Lemma 5.1(1), since 1 is a test.
($\Leftarrow$) Given test $p \leq 1$, by Lemma 5.1(2) the elements $c = 1$ and $d = p$ satisfy the assumptions in Lemma 5.1(4). Moreover, all tests are transitive. □

Using the concept of tests we now generalise the operators $\diamondsuit$ and $\Box$ to an arbitrary Boolean left semiring S. Following Section 3.6 the greatest element $\top$, the greatest purely finite element F and the greatest purely infinite element N exist.

Let now, for $p \in \mathsf{test}(S)$,

$$\diamondsuit\, p =_{df} \mathsf{F} \cdot p \cdot \top, \qquad \Box p =_{df} \overline{\diamondsuit\, \neg p}.$$

Thus, $\Box p$ corresponds to the "always p" operator of von Karger [50], whence the notation. Since $\diamondsuit$ and $\Box$ do not yield tests as their results, they cannot be nested. This does no harm, since nested safety requirements do not seem to be useful anyway. All other algebraic operations, like addition and multiplication, are available for box and diamond. Our goal is now to derive a number of useful algebraic laws for these operators. First,

$$\diamondsuit\, 0 = 0 = \Box 0, \qquad \diamondsuit\, 1 = \top = \Box 1. \tag{15}$$

Another immediate consequence of the definitions is

Lemma 5.6. *For Boolean left semiring S and* $p \in \mathsf{test}(S)$ *the element* $\Box p$ *is submodular.*

Proof. By the definition of box we have $\mathsf{F} \cdot \overline{\Box p} \cdot \top = \mathsf{F} \cdot \mathsf{F} \cdot \neg p \cdot \top \cdot \top = \mathsf{F} \cdot \neg p \cdot \top = \overline{\Box p}$ and the claim follows from Lemma 5.1(1). □

The box operator shows useful and natural behaviour in the case of progressiveness.

Lemma 5.7. *Let* $p, q \in \mathsf{test}(S)$ *in a progressive Boolean weak semiring S.*

(1) $p \leq \Box q \Leftrightarrow p \leq q$.

(2) $p \leq \Box p$.

By Lemmas 5.4(3) and 3.2(1) PRO is progressive and Properties (1) and (2) hold. In REL, however, subidentities can be decomposed into non-subidentities (unless the underlying base set is a singleton); so these properties do not hold there.

For the following proofs and properties we introduce shorthands for the purely finite and purely infinite parts of boxes:

$$\boxed{\mathsf{F}} p =_{df} \mathsf{fin}\,(\Box p) = \mathsf{F} \sqcap \Box p, \qquad \boxed{\mathsf{N}} p =_{df} \mathsf{inf}\,(\Box p) = \mathsf{N} \sqcap \Box p. \tag{16}$$

Now we can show

Lemma 5.8. *Assume a Boolean left semiring S and* $p \in \mathsf{test}(S)$. $\Box p = p \cdot (\Box p)$. *If S is weak then also* $\boxed{\mathsf{F}} p = p \cdot \boxed{\mathsf{F}} p$ *as well as* $\Box p = (\Box p) \cdot p$ *and* $\boxed{\mathsf{F}} p = \boxed{\mathsf{F}} p \cdot p$.

Some of the following properties are satisfied only in a special kind of left semirings. Since elements of the form $\Box p$ correspond to safety properties, we call a left semiring (quantale) S *safety-closed* if $(\Box p) \cdot (\Box p) \leq \Box p$.

Since in a safety-closed left Kleene algebra $\Box p$ is transitive, it coincides with its own transitive closure, i.e., $(\Box p)^+ = \Box p$. Hence

$$a \leq \Box p \Leftrightarrow a^+ \leq \Box p. \tag{17}$$

Safety-closedness implies, next to other useful properties, that a composition satisfies a safety assertion if that is satisfied in its first component or in the second component after some finite run of the first component.

Lemma 5.9. *Assume a Boolean weak semiring S that is safety-closed.*
(1) *All boxes are modular.*
(2) *All boxes are multiplicatively idempotent, i.e.,* $(\Box p) \cdot (\Box p) = \Box p$.
(3) $\Box p \sqcap a^+ = (\Box p \sqcap a)^+$ *and* $\Box p \sqcap a^* = (\Box p \sqcap a)^+ + (\Box p \sqcap 1)$.
(4) $\Diamond\, p \sqcap a \cdot b = (\Diamond\, p \sqcap a) \cdot b + a^\frown(\Diamond\, p \sqcap b)$.

The dual of Part 4, namely that a composition satisfies a safety assertion iff its two components satisfy it ($\Box p \sqcap a \cdot b = (\Box p \sqcap a) \cdot (\Box p \sqcap b)$) follows immediately since boxes are modular (Part 1).

Example 5.10. Returning to requirement (th-rest), we can transform the safety requirement $R_{20} \cdot (A^{\mathrm{Off}} \cdot A^{\mathrm{On}})^+ \sqcap \Box R_{[18,22]}$ into $R_{20} \cdot ((A^{\mathrm{Off}} \sqcap \Box R_{[18,22]}) \cdot (A^{\mathrm{On}} \sqcap \Box R_{[18,22]}))^+$ by (17) and Lemma 5.9(1). Hence, it suffices to guarantee the safety requirement for the two component processes A^{Off} and A^{On}.

Using general theory, we can now also give an algebraic definition of the range operator introduced for PRO in Section 3.1. As a preparation we state the following.

Lemma 5.11. *Assume a left quantale in which $\cdot$ is also positively right-distributive. Then $\Diamond$ is universally disjunctive and $\Box$ is universally conjunctive. In particular, both operators are isotone.*

Proof. The property for $\Box$ follows by de Morgan's laws from the one for $\Diamond$, so we only show that. For nonempty set $L \subseteq P$ we get

$$\Diamond \left(\bigsqcup L\right) = \mathsf{F} \cdot \left(\bigsqcup L\right) \cdot \top = \bigsqcup (\mathsf{F} \cdot L) \cdot \top = \bigsqcup (\mathsf{F} \cdot L \cdot \top)$$

by positive right-disjunctivity and left-disjunctivity of $\cdot$. Moreover, we have

$$\Diamond \left(\bigsqcup \emptyset\right) = \mathsf{F} \cdot 0 \cdot \top = \mathsf{F} \cdot 0 = 0 = \bigsqcup \Diamond\, \emptyset$$

by left-strictness of $\cdot$ and $\mathsf{F} \cdot 0 = 0$. □

Therefore we can define a general operator $\mathsf{ran} : S \to \mathsf{test}(S)$ by the Galois connection

$$\mathsf{ran}\, a \leq p \Leftrightarrow_{df} a \leq \Box p. \tag{18}$$

By (18), ran is universally disjunctive. Moreover, we obtain

$$a \leq \Box(\mathsf{ran}\, a), \qquad \mathsf{ran}\,(\Box p) \leq p, \qquad p \leq \Box p \Rightarrow \mathsf{ran}\, p \leq p. \tag{19}$$

The range operator relates to the others as follows.

Lemma 5.12. *If S is positively right-disjunctive then* $\mathsf{ran}\, p = p$.

Proof. By the third property of (19) it remains to show $p \leq \mathsf{ran}\, p$. Using the Galois connection (18) and Lemma 5.7 (1), for arbitrary test q, we have

$$\mathsf{ran}\, p \leq q \Leftrightarrow p \leq \Box q \Leftrightarrow p \leq q.$$

Now setting $q = \mathsf{ran}\, p$ yields the claim. □

5.4. A sufficient criterion for safety-closedness

For the technical developments of this section we need additional operators. In any left quantale, the *left residual* a/b exists and is characterised by the Galois connection

$$x \leq a/b \Leftrightarrow_{df} x \cdot b \leq a.$$

In PRO, this operation is characterised pointwise by $\tau \in V/U \Leftrightarrow \forall \sigma \in U : \tau \cdot \sigma \in V$ (provided $\tau \cdot \sigma$ is defined). Based on the left residual, in a left Boolean quantale the *right detachment* $a \lfloor b$ can be defined as

$$a \lfloor b =_{df} \overline{\overline{a}/b}.$$

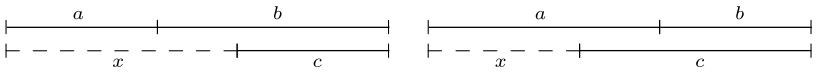

Fig. 5. Local linearity.

The pointwise characterisation in PRO reads $\tau \in V \lfloor U \Leftrightarrow \exists \sigma \in U : \tau \cdot \sigma \in V$. Informally, this means that $V \lfloor U$ consists of trajectories which result from detaching a U-trajectory at the right from some V-trajectory. By de Morgan's laws, the Galois connection for / transforms into the exchange law $a \lfloor b \leq x \Leftrightarrow \overline{x} \cdot b \leq \overline{a}$ for $\lfloor$ that generalises the Schröder rule of relational calculus.

Two straightforward consequences are

$$(\Box p) \lfloor a \leq \Box p \quad \text{and} \quad \boxed{\mathsf{F}} p \lfloor a \leq \boxed{\mathsf{F}} p. \qquad \text{(box detachment)}$$

Intuitively, this means that in PRO any prefix of a trajectory that satisfies a safety assertion again satisfies the assertion. Moreover, $\lfloor$ is isotone in both arguments and satisfies $a \lfloor 1 = a$.

A left Boolean quantale is said to be *locally linear* [50] if it satisfies

$$(a \cdot b) \lfloor c = a \cdot (b \lfloor c) + a \lfloor (c \lfloor b).$$

The law describes the case analysis that appears when c is cut off $a \cdot b$ from the right. We distinguish two cases — c is a postfix of b or b is a postfix of c. We illustrate this behaviour in Fig. 5, where the elements a, b, c are singleton processes of which only the time intervals are shown.

Local linearity of PRO can be proved as in the case of the semiring of formal languages, as done in [26]. Hence, by the following lemma, PRO is safety-closed.

Lemma 5.13. *If S is a Boolean weak and locally linear quantale then S is safety-closed.*

The proof is given in the Appendix.

Sometimes one has safety properties of the form that first a predicate p has to be satisfied and afterwards another predicate q has to hold. The following laws are useful for checking whether a composition of processes satisfies such a condition.

Lemma 5.14. *Assume a Boolean weak and locally linear quantale S. Then for all $a, b \in S$ and $p, q \in \mathsf{test}(S)$ the following properties hold.*

(1) $a \cdot b \sqcap \boxed{\mathsf{F}} p \cdot \Box q = (a \sqcap \boxed{\mathsf{F}} p) \cdot (b \sqcap \Box q) \; + \; (a \sqcap \boxed{\mathsf{F}} p) \cdot (b \sqcap \boxed{\mathsf{F}} p \cdot \Box q) \; + (a \sqcap \boxed{\mathsf{F}} p \cdot \Box q) \cdot (b \sqcap \Box q)$.
(2) $a \cdot b \sqcap \boxed{\mathsf{N}} p = (a \sqcap \boxed{\mathsf{N}} p) + (a \sqcap \boxed{\mathsf{F}} p) \cdot (b \sqcap \boxed{\mathsf{N}} p) = (a \sqcap \Box p) \cdot (b \sqcap \boxed{\mathsf{N}} p)$.
(3) $a \cdot b \sqcap \Box p \cdot \Box q = (a \sqcap \Box p) \cdot (b \sqcap \Box q) \; + \; (a \sqcap \Box p) \cdot (b \sqcap \Box p \cdot \Box q) \; + (a \sqcap \Box p \cdot \Box q) \cdot (b \sqcap \Box q)$.
(4) *If additionally $p \leq \Box p$ holds, the summand $(a \sqcap \boxed{\mathsf{F}} p) \cdot (b \sqcap \Box q)$ can be omitted from the right hand sides of Parts (1) and (3).*

The lengthy proof can be found in the Appendix of [29]. For single, finite trajectories Part (1) is illustrated in Fig. 6. Here, the change between properties p and q can occur either exactly at the composition point of a and b, inside a or inside b. That is why the formula on the right hand side of Part (1) consists of three summands.

An application of Lemma 5.14(1) is to combine safety requirements of the shape $R_{[l,u]}$. Since $\boxed{\mathsf{F}} p \cdot \Box q = \Box p \frown \Box q$, a safety requirement of this form guarantees that the process $\Box q$ is actually entered.

We conclude by yet another equivalent characterisation of time progress.

Lemma 5.15. *Assume a Boolean weak quantale S. Then S is progressive iff*

$$\forall a \in S : \forall p \in \mathsf{test}(S) : p \lfloor a = a \sqcap p.$$

See again the Appendix for a proof.

5.5. Temporal operators

Specifications are particular processes that express desired patterns. Following Sintzoff [48], we define quantifier-like operators relating a specification W to a purported implementing process B. If one considers the values in V as states then the set $\{t(0) : t \in B \cap W\}$ gives all starting values of the trajectories in B admitted by W as well. However, it is more convenient to represent this set as a test in the left semiring of processes, viz. as

$$\{\underline{t(0)} \mid t \in B \cap W\}. \qquad (20)$$

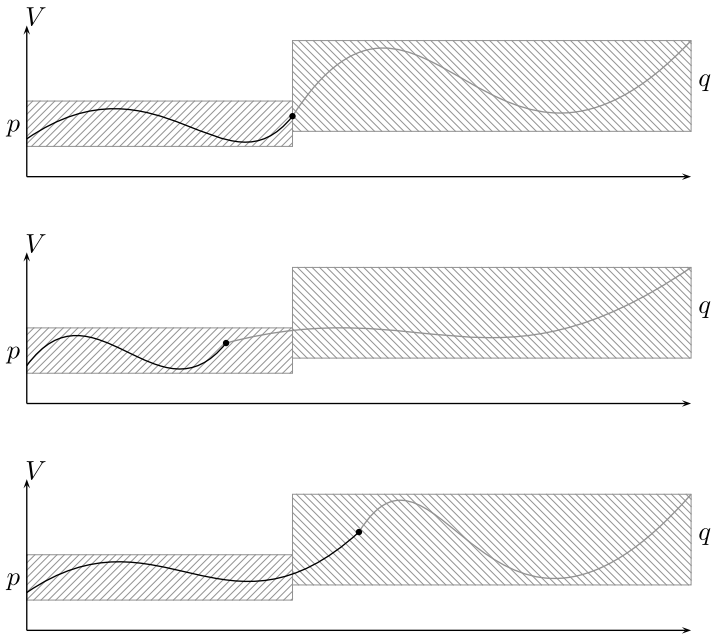

Fig. 6. Composed trajectories satisfying $\mathbb{E}p \cdot \Box q$.

To model this, we introduce into our algebra an abstract domain operator that assigns to a set of computations the test that describes precisely its initial values. In combination with restriction, domain yields an abstract preimage operation and codomain an abstract image operation.

A *left domain semiring (quantale)* is a pair $(S, \ulcorner)$, where S is a left semiring (quantale) and the *domain* operation $\ulcorner : S \to \mathsf{test}(S)$ satisfies

$$a \leq \ulcorner a \cdot a, \quad \text{(d1)} \qquad \ulcorner(p \cdot a) \leq p, \quad \text{(d2)} \qquad \ulcorner(a \cdot \ulcorner b) \leq \ulcorner(a \cdot b). \quad \text{(d3)}$$

The axioms are the same as in [21]; their relevant consequences can still be proved over left semirings (quantales) [42]. In particular, $\ulcorner$ is universally disjunctive and hence $\ulcorner 0 = 0$. Moreover, the conjunction of (d1) and (d2) is equivalent to each of

$$\ulcorner a \leq p \Leftrightarrow a \leq p \cdot a, \quad \text{(llp)} \qquad \ulcorner a \leq p \Leftrightarrow \neg p \cdot a \leq 0. \quad \text{(gla)}$$

Property (llp) says that $\ulcorner a$ is the least left preserver of a; (gla) that $\neg\ulcorner a$ is the greatest left annihilator of a.

Lemma 5.16. *The tuple* $\mathrm{PRO} =_{df} (\mathcal{P}(\mathrm{TRA}), \cup, \emptyset, \cdot, I, \ulcorner)$ *forms a Boolean positively right-disjunctive domain quantale with* $\ulcorner A = \{\underline{g(0)} : (d, g) \in A\}$.

Contrarily to the case of arbitrary semirings [41] with complete sublattice of tests, the domain operation is guaranteed to exist in left quantales [21].

A useful property is the following.

Lemma 5.17. *If the underlying semiring satisfies* $p \leq \Box p$ *then* $\ulcorner(\Box p) = p$.

Proof. Axiom (d2) and Lemma 5.8 imply $\ulcorner(\Box p) \leq p$. The reverse inequation follows from the assumption $p \leq \Box p$, isotony of domain and $\ulcorner p = p$. □

Using the domain operation, Eq. (20) compacts into $\ulcorner(B \cap W)$. Therefore, a first algebraic definition of Sintzoff's quantifiers reads as follows (the primes indicate that we will use a different definition later on):

$$\mathsf{E}'B \,.\, W =_{df} \ulcorner(B \cap W), \tag{21}$$

$$\mathsf{A}'B \,.\, W =_{df} \neg\mathsf{E}'B \,.\, \overline{W} = \neg\ulcorner(B \cap \overline{W}), \tag{22}$$

$$\mathsf{AE}'B \,.\, W =_{df} \mathsf{A}'B \,.\, W \cap \mathsf{E}'B \,.\, W. \tag{23}$$

This definition works in general Boolean left domain semirings. However, as the resulting quantifiers are operators of type $\mathrm{PRO} \to (\mathrm{PRO} \to \mathsf{test}(\mathrm{PRO}))$, they cannot easily be composed. Therefore, Sintzoff gives a different semantics to combinations

of these quantifiers. We want to avoid this by introducing new quantifiers that omit the final projection into $\mathsf{test}(\mathrm{PRO})$. Doing this, we also allow a look into the "future" of trajectories and not only at the starting states. In other words, our new quantifiers in PRO should model formulas like

$$t \in \mathsf{E}B \,.\, W \;\Leftrightarrow_{df}\; \exists u \in B : t \cdot u \in W,$$
$$t \in \mathsf{A}B \,.\, W \;\Leftrightarrow_{df}\; \forall u \in B : t \cdot u \in W.$$

Hence, the process $\mathsf{E}B \,.\, W$ consists of all trajectories that can be completed by a B-trajectory to yield a trajectory in W. Thus, $\mathsf{E}B \,.\, W$ is the inverse image of W under the operation $\cdot B$, while $\mathsf{A}B \,.\, W$ is the largest process whose image under $\cdot B$ is contained in W.

These quantifiers are operators of type $\mathrm{PRO} \to \mathrm{PRO}$ and their sequential composition simply is function composition. If, as with E' and A', a projection into $\mathsf{test}(\mathrm{PRO})$ is desired it can be added at the outermost level by finally applying one of the three quantifiers above. For their algebraic characterisation we basically want to use Eqs. (21) and (22), but express them with the help of detachment. Therefore we establish a connection between that and the domain operator.

Lemma 5.18 *In a Boolean quantale, one has*

$$\ulcorner(b \sqcap w) = w \lfloor b \sqcap 1 = b \lfloor w \sqcap 1.$$

In the detachment formulas of this lemma, forming the meet with 1 performs the projection into the test algebra, and we obtain our revised operators by omitting this meet. There is a choice in which of these two formulas to use. We take the first one, since it results in a more direct translation of the universal quantifier A'. Assume a Boolean quantale S and $a, b \in S$. Then

$$\mathsf{E}b \,.\, w =_{df} w \lfloor b, \quad \mathsf{A}b \,.\, w =_{df} \overline{\mathsf{E}b \,.\, \overline{w}} = w/b, \quad \mathsf{AE}b \,.\, w =_{df} (\mathsf{A}b \,.\, w) \sqcap (\mathsf{E}b \,.\, w).$$

These quantifiers allow the following modal view: E is a kind of diamond, whereas A is a box operator. Correspondingly, we have the following properties that are typical of modal operators.

Lemma 5.19

(1) $\mathsf{E}a.w$ *is universally disjunctive and* $\mathsf{A}a.w$ *is universally conjunctive in* w.
(2) $\mathsf{E}(a \cdot b) \,.\, c = \mathsf{E}a \,.\, (\mathsf{E}b \,.\, c)$ *and* $\mathsf{A}(a \cdot b) \,.\, c = \mathsf{A}a \,.\, (\mathsf{A}b \,.\, c)$.
(3) *If* $\cdot$ *is positively disjunctive in its right argument then* $\mathsf{E}a$ *is positively disjunctive and* $\mathsf{A}a$ *is positively antidisjunctive in* a.

Sintzoff has used these operators to determine strategies in discrete-decision games [48]. He has also shown that game theory helps in understanding hybrid and reactive systems, since it deals with interaction between dynamics. For example, a hybrid system can be presented as a game where the *controlling* and the *controlled* components are, respectively, the proponent and the opponent [32]. As the controller has to counteract all possible failures induced by "moves" of the controlled system, it has to force the opponent into a "losing" position where nothing can go wrong anymore. In PRO, moves correspond to process transformers of the shapes $\mathsf{E}B$ and $\mathsf{A}B$. They describe the possible and guaranteed reachabilities from a game position using B-trajectories.

6. A case study

To round off the paper, we give a longer case study. It concerns a railroad gate control and was introduced in [23]. For that, we assume a circular track that is between 2000 and 5000 m long and a railway crossing with a gate. A sketch of the architecture is given in Fig. 7.

A moving train on the track is modelled by the hybrid automaton of Fig. 8. The variable x represents the distance of the train from the gate. Initially, the speed of the train is between 40 and 50 m/s. At the distance of 1000 m from the gate, the train issues an *approach* event and may slow down to 30 m/s. At the distance of 100 m behind the gate, the train issues an *exit* event.

We now want to derive the corresponding algebraic expression for this automaton. For this we follow the schema of Section 4.1. To simplify matters we skip the control modes, since all control modes have the same structure. Furthermore, we do not define processes for each mode. Instead we define the following general processes:

$$T^{[a,b]} =_{df} \{(d,x) \mid d \in \mathbb{R}_{\geq 0},\, a \leq \dot{x} \leq b\},$$
$$P_{dist} =_{df} \{\underline{dist}\} = \{(0,x) \mid x = dist\},$$
$$P_{\leq dist} =_{df} \{\underline{dist}\} = \{(0,x) \mid x \leq dist\}.$$

Process $T^{[a,b]}$ restricts the speed of the train to a velocity between a and b; the duration of the trajectories is not restricted at all. The zero-length process P_{dist} is used to test whether the train is at a certain distance of *dist* from the gate or not. For example P_0 tests if the train passes the gate at the moment.

To model the jump condition given in Fig. 8, we use the compatibility relation $\asymp =_{df} \{(-100,x) \mid x \in [1900, 4900]\}$. Depending on the length of the track it sets the distance after the train has passed the gate.

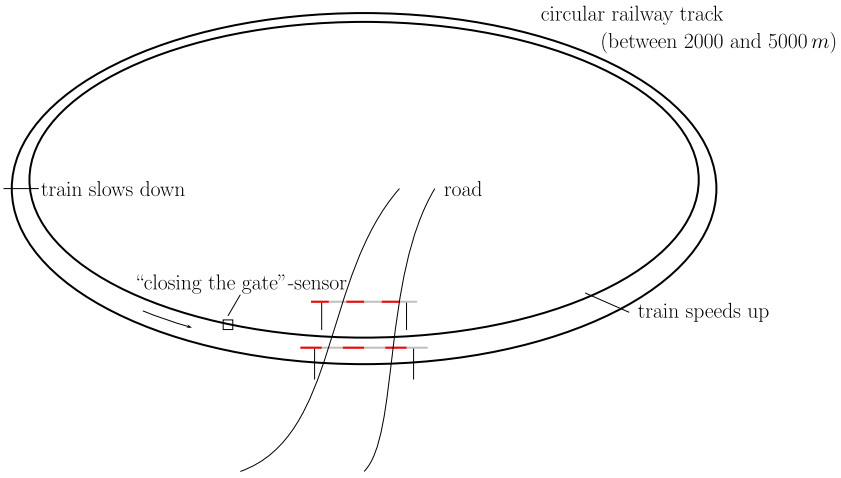

Fig. 7. Architecture of the railroad gate controller.

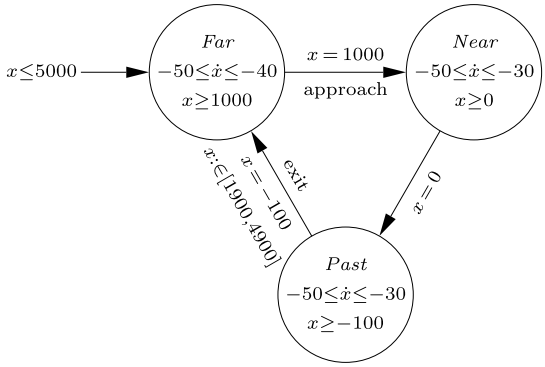

Fig. 8. Train automaton.

Using these elements the following algebraic expression for the train automaton results from our schema:

$$\mathrm{TR} =_{df} P_{\leq 5000} \cdot \left((T^{[-50,-40]} \cdot P_{1000} \cdot T^{[-50,-30]} \cdot P_0 \cdot T^{[-50,-30]} \cdot P_{-100})_{\asymp} \right)^{\omega}.$$

The initial test $P_{\leq 5000}$ sets the starting point of the train: the distance between the gate and the train has to be smaller than 5000 m. As described in Section 4.1 the compatibility relation is employed at the right end of the repeated process. It is only needed at the point where we want to *enforce* a jump in the function describing the distance between the train and the gate. The other multiplications require the identity relation as compatibility relation, since we want to avoid jumps. Hence we do not need an explicit compatibility relation for the other products.

Note that in the algebraic expression we can replace ω by †, since the tests P_{dist} together with the given velocities of the train enforces that there are no Zeno-effects.

As the second component of the railroad gate control we have a gate automaton (Fig. 9).

The variable y of the gate automaton represents the position of the gate in degrees. Initially, the gate is open ($y = 90$). When a *lower* event is received, the gate starts closing at a rate of 9 degrees per second and when a *raise* event happens, the gate starts opening at the same rate. The given schema to convert hybrid automata to algebraic expressions yields

$$\mathrm{GA} =_{df} O \cdot ((M_l \cdot M_r)^* \cdot (C + O))^{\omega},$$

where

$$\begin{array}{lll} O & =_{df} & \{(d, \mathtt{const}(90)) \mid d \in \mathbb{R}_{\geq 0}\} \quad \text{models control mode } \textit{Opened}, \\ C & =_{df} & \{(d, \mathtt{const}(0)) \mid d \in \mathbb{R}_{\geq 0}\} \quad \text{models control mode } \textit{Closed}, \\ M_l & =_{df} & \{(d, y) \mid \dot{y} = -9,\ d \in \mathbb{R}_{\geq 0}\} \quad \text{models control mode } \textit{Down}, \\ M_r & =_{df} & \{(d, y) \mid \dot{y} = 9,\ d \in \mathbb{R}_{\geq 0}\} \quad \text{models control mode } \textit{Up} \end{array}$$

and `const` is again the constant function. $M_l \cdot M_r$ is iterated because the gate can start opening even if it is not totally closed ($y = 0$) and it can start closing even if the gate is not absolutely opened ($y = 90$).

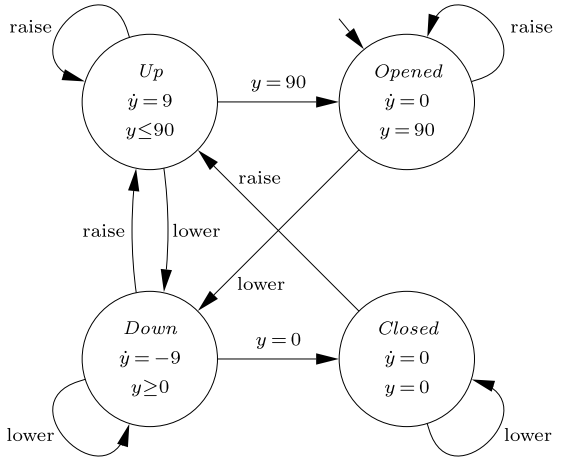

Fig. 9. Gate automaton.

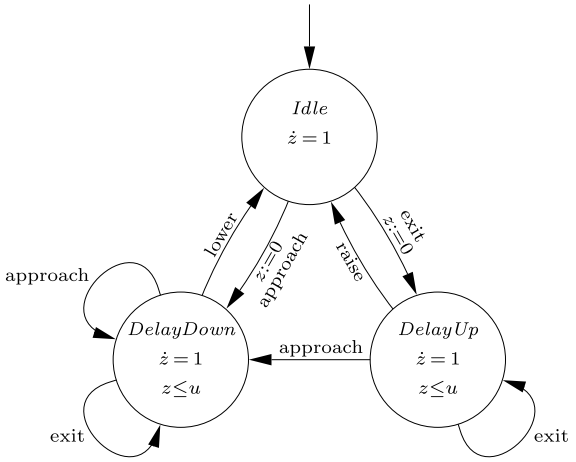

Fig. 10. Controller automaton.

The simplest way to combine both expressions is

$$\mathrm{TR} \parallel \mathrm{GA}$$

where $\parallel$ is the pointwise lifted parallel composition of Section 4.2. But this algebraic expression contains all combinations of the train trajectories and the gate trajectories, e.g., the gate can be opened when the train passes. Hence a simple combination is not useful.

To combine these two automata and to guarantee safety, one can use a third automaton – a controller automaton – as done in [23] (cf. Fig. 10).

This controller has a reaction delay of up to u seconds. For example if the train issues an *approach* event, the automaton switches to the mode *DelayDown*. The elapsed time is measured by the variable z. At some point before z reaches the reaction upper bound u the automaton starts the *lower* event and the gate begins to close (the gate automaton is now in mode *Down*).

To simplify matters, we assume a reaction time of 0 s. (Different delay times are also possible, but the algebraic expressions become more complicated, although the structure would be the same.) When an *approach* event is received, the controller immediately issues a *lower* event and when an *exit* event is received, the controller starts immediately a *raise* event. In sum we have

$$\begin{aligned}\mathrm{TG} = {} & (O \| (P_{\leq 5000} \cdot T^{[-50,-40]} \cdot P_{1000})) \\ & \cdot (((M_l \cdot C) \| (T^{[-50,-30]} \cdot P_0)) \cdot (C \| (T^{[-50,-30]} \cdot P_{-100})_{\asymp}) \\ & \cdot ((M_r \cdot O) \| (T^{[-50,-40]} \cdot P_{1000})))^{\dagger}\end{aligned} \tag{24}$$

Let us have a look at the single components. The first part $(O \| (P_{\leq 5000} \cdot T^{[-50,-40]} \cdot P_{1000}))$ models the initial behaviour; the gate has to be open, the train starts somewhere before the gate (not farther than 5000 metres), and moves until it reaches the point $x = 1000$. Each of the components in the infinite iteration loop has as right operand of the parallel composition one control mode of the train automaton together with the attached event and as left operand the corresponding behaviour of the gate. Since the gate components end up in the modes C and O where the gate is opened or closed, respectively, processes like $M_l \cdot C$ can be lengthened to any duration longer than the shortest duration of M_l. Therefore we do not need a constant function for the parallel composition (as discussed in Section 4.2). Note that the nested iteration of GA has been removed, because that behaviour cannot occur. Furthermore, this example might, in contrast to the algebraic expression of the train automaton, contain Zeno effects; therefore ω and † might behave differently.

Aspects of safety. The algebra of processes not only compacts the description by a parallelised hybrid automaton (which was not given by Henzinger), but also contains many aspects of safety. E.g., the expression $M_l \cdot C$ itself guarantees that the gate is closed at the time when the train passes the gate. This guarantee is not given in the original paper. Furthermore, it is easy to see that if the initial distance between the gate and the train is smaller than 1000, we have for the first factor of (24)

$$(P_{<1000} \cdot T^{[-50,-40]} \cdot P_{1000}) = 0.$$

Thus we know that such an initial distance is not safe, since it is not possible that the gate gets closed in time. This problem is not discussed in [23]. In general, if an algebraic expression or a part of it at a strict position (after a finite run) is equal to zero, the corresponding system is not safe. Another aspect of safety is the Zeno problem. In our example, Zeno effects can occur in the hybrid automaton as well as in our algebraic expressions. But those effects can be excluded by taking

$$\mathrm{TG} \sqcap \mathrm{N},$$

as discussed in Section 3.6. Sometimes it is desirable and necessary to introduce range assertions. For instance, we may, besides the normal conditions of operation, want to guarantee that no train is faster than 40 metres per seconds (e.g. if there is construction work on the track). Then we have to modify Expression (24). Using the range assertions of Section 5.3 the algebraic expression can be modified to

$$\mathrm{TG} \sqcap \Box\, T^{[0,-40]}.$$

With this, we have a characterisation of the modified system and can now check safety, etc.

7. Related work

As mentioned before, the research concerning hybrid systems is mostly focused on hybrid automata [23]. Within that area there are different approaches to safety and liveness properties. But, most of the research covers only a certain class of hybrid automata, the linear hybrid automata. For example, [3] discusses reachability and verification problems for linear hybrid systems. In contrast to these papers our approach does not restrict hybrid systems at all.

An algebraic framework dealing with hybrid systems is the process algebra of [11]. It is obtained by extending a combination of two extensions of ACP [10], namely the process algebra with continuous relative timing from [9] and the process algebra with propositional signals from [8]. It has, in addition to equational axioms, some rules to derive further equations with the help of real analysis. However, it does not contain transformation rules for larger systems in our style; moreover, it does not define operators for the analysis of the purely finite and purely infinite parts of behaviours.

An algebraic theory of general networks is presented in [49].

Besides the theories of hybrid automata and algebras there is further related work. For example in [33] a variant of timed CSP [47] is introduced that allows limited dealing with continuous behaviour. In [44] the π-calculus [40] is modified such that it can deal with continuous behaviour.

Further approaches to hybrid systems are Hybrid I/O automata [38], the work on tools like CHARON [6,1] and HyTech [24] as well as the logics for hybrid systems [20]. But these approaches have not yet been put into algebraic form.

8. Conclusion and outlook

This paper provides a comprehensive algebraic theory of hybrid systems based on left semirings and iteration algebras. Although one has to take some care, since the basic laws are weaker than those for standard semirings, things work out reasonably well and many results come for free. We have presented a model of trajectories and processes which then has been abstracted to admit a general semiring view. We have shown how to embed hybrid automata into that setting. Based

on an analysis of the purely finite and purely infinite parts of behaviours we have demonstrated how Zeno effects can conveniently be handled. We have given algebraic definitions of several composition operators for hybrid systems. We have discussed safety and liveness properties as well as time restrictions and range assertions and certain temporal operators. It should be noted that nevertheless the whole development is based on few and well-known algebraic concepts.

The aim of further work is to use this framework to give a fully algebraic treatment of the duration calculus based on the approaches of [50,25]. Another aim is to form a connection with game theory and game algebra to obtain improved controllers for hybrid systems. Finally, it has to be checked in how far I/O automata can be treated in this style to make the theory even more useful. It seems that the semantic models used in [20,38] can be made into left quantales, too, so that our results would carry over to these frameworks. It will also be interesting to apply the approach in further case studies. On the more theoretical side, an algebraic treatment of time abstraction as well as further analysis of safety via range assertions and of liveness issues is necessary. The structures of Kleene and omega algebras should allow a convenient algebraic treatment of reachability questions [21]. The algebraic semantics for CTL^* given in [43] prepares the connection to various logics for hybrid systems [20]. Finally, since the theory of semirings is completely first-order and Horn, it lends itself to mechanisation using off-the-shelf theorem provers, as has recently been shown in [30,31]. Therefore the field of hybrid systems should be tractable. In [27] we have already automatically proved some liveness and safety properties for two small hybrid systems.

Acknowledgements

We are grateful to Kim Solin and Georg Struth as well as to the anonymous referees for helpful comments and discussions.

Appendix A. Deferred proofs

Proof of 3.5. We give this proof to pinpoint the use of our assumptions; a similar proof for the more restrictive setting of full quantales appears, e.g., in [2]. It uses the principles of least and greatest fixpoint fusion (see e.g. [19]): Let $f, g, h : L \to L$ be isotone functions on a complete lattice $(L, \leq)$ with least element 0 and greatest element $\top$ such that $g \circ h = f \circ g$.

- If g is continuous, i.e., preserves suprema of nonempty chains, and strict, i.e., satisfies $g(0) = 0$, then $g(\mu h) = \mu f$.
- If g is cocontinuous, i.e., preserves infima of nonempty chains, and costrict, i.e., satisfies $g(\top) = \top$, then $g(\nu h) = \nu f$.

In both parts of the proof we use $f(x) =_{df} a \cdot x + b$, whereas g and h will change.

(1) The star axioms (specialized to the case $b = 1$) are equivalent to the statement that a^* is the least contracted element of the function $h(x) =_{df} a \cdot x + 1$; hence by the Knaster/Tarski fixpoint theorem it coincides with the least fixpoint of that function. Therefore the star unfold axiom holds by construction.
Now we use least fixpoint fusion with $g(x) =_{df} x \cdot b$ to show that $a^* \cdot b$ is the least fixpoint and hence the least contracted element of f, which is the contents of the star induction axiom.
By the definition of a left quantale, g is continuous and strict. Furthermore,

$$g(h(x)) = (a \cdot x + 1) \cdot b = a \cdot x \cdot b + b = f(g(x)),$$

and $a^* \cdot b = \mu f$, as required.

(2) The omega unfold axiom holds by construction.
We set $c =_{df} a^* \cdot b$ and $e =_{df} a^\omega + c$ and show that e is the greatest fixpoint and hence the greatest element expanded by f, which is precisely the contents of the ω coinduction axiom.
This time we use $g(x) =_{df} x + c$ and $h(x) =_{df} a \cdot x$. Function g is obviously costrict. It is also cocontinuous, since we assume the underlying left quantale to be completely distributive. For the commutativity condition we calculate using, first, that c is a fixpoint of f by the proof of Part (1) and, second, weakness of the underlying quantale,

$$\begin{aligned} g(h(x)) &= a \cdot x + c = a \cdot x + f(c) = a \cdot x + a \cdot c + b \\ &= a \cdot (x + c) + b = a \cdot g(x) + b = f(g(x)). \end{aligned}$$

This establishes $\nu f = a^\omega + a^* \cdot b$ as required. □

Proof of 3.6

(1) This follows by elementary Boolean algebra.

(2) Since ($\Leftarrow$) is just isotony, it suffices to prove ($\Rightarrow$). We show the first conjunct, the second being symmetric. Using Part (1) we calculate

$$a = (a \sqcap \mathsf{F}) + (c \sqcap \mathsf{F}) = (a + c) \sqcap \mathsf{F} \leq (b + d) \sqcap \mathsf{F} = (b \sqcap \mathsf{F}) + (d \sqcap \mathsf{F}) = b.$$

(3) This is Lemma 6.8(d) of [42].

(4) By Part (3), distributivity of inf and $\mathsf{inf}\,\mathsf{inf}\,x = \mathsf{inf}\,x$,

$$\begin{aligned} &\mathsf{inf}\,a^\omega \\ &= \mathsf{inf}\,((\mathsf{fin}\,a)^* \cdot \mathsf{inf}\,a + (\mathsf{fin}\,a)^\omega) \\ &= \mathsf{inf}\,((\mathsf{fin}\,a)^* \cdot \mathsf{inf}\,a) + \mathsf{inf}\,((\mathsf{fin}\,a)^\omega) \\ &= (\mathsf{fin}\,a)^* \cdot \mathsf{inf}\,a + \mathsf{inf}\,((\mathsf{fin}\,a)^\omega). \end{aligned}$$

(5) By Part (3), distributivity of fin, $\mathsf{fin}\,(b \cdot \mathsf{N}) = 0$ and $\mathsf{fin}\,b \leq b$ we have

$$
\begin{aligned}
& \mathsf{fin}\, a^\omega \\
= \;& \mathsf{fin}\,((\mathsf{fin}\, a)^{*} \cdot \mathsf{inf}\, a + (\mathsf{fin}\, a)^\omega) \\
= \;& \mathsf{fin}\,((\mathsf{fin}\, a)^{*} \cdot \mathsf{inf}\, a) + \mathsf{fin}\,(\mathsf{fin}\, a)^\omega \\
\leq \;& 0 + (\mathsf{fin}\, a)^\omega \\
= \;& \mathsf{fin}\,(\mathsf{fin}\, a)^\omega \\
\leq \;& (\mathsf{fin}\, a)^\omega. \quad \square
\end{aligned}
$$

Proof of 3.7. Let $\mathrm{OM}(A) =_{df} \{\tau \in \mathrm{TRA} \mid \exists \mathrm{T} \in \mathrm{ISEQ}(A) : \prod \mathrm{T} \sqsubseteq \tau\}$.

(1) The claim is equivalent to $X \subseteq \mathrm{OM}(A)$. Consider $\sigma \in X$. We construct a sequence $\mathrm{T} = (\tau_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$ inductively as follows. Set $\sigma_{-1} =_{df} \sigma$. Since $X \subseteq A \cdot X$, there are $\tau_0 \in A$ and $\sigma_0 \in X$ with $\sigma_{-1} = \tau_0 \cdot \sigma_0$. Now assume that τ_i and σ_i have been constructed. By the same argument as above σ_i can be decomposed into τ_{i+1} and σ_{i+1}. Now, by construction $\prod \mathrm{T} \sqsubseteq \sigma$. Hence $\sigma \in \mathrm{OM}(A)$ and we are done.

(2) As a prerequisite, we observe that finite trajectories τ are left cancellative w.r.t. composition, i.e., satisfy

$$\tau \cdot \rho = \tau \cdot \sigma \Rightarrow \rho = \sigma$$

provided $\tau \cdot \rho$ and $\tau \cdot \sigma$ are defined.
Now we show that $\mathrm{OM}(A)$ is expanded by H. Consider an arbitrary $\sigma \in \mathrm{OM}(A)$. By definition there is a $\mathrm{T} = (\tau_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$ with $\prod \mathrm{T} \sqsubseteq \sigma$. Then also $\tau_0 \sqsubseteq \sigma$ and hence, by finiteness of τ_0 and the above cancellation property, there is a unique ρ with $\sigma = \tau_0 \cdot \rho$. Define $\Xi = (\xi_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$ by $\xi_n = \tau_{n+1}$ for all $n \in \mathbb{N}$. Then $\prod \Xi \sqsubseteq \rho$, i.e., $\rho \in \mathrm{OM}(A)$. Therefore $\sigma = \tau_0 \cdot \rho \in A \cdot Y$. Hence $\mathrm{OM}(A) \subseteq A \cdot \mathrm{OM}(A)$.
Together with Part (1) this means that $\mathrm{OM}(A)$ is the greatest expanded element of H and hence its greatest fixpoint. Now the claim follows by Lemma 3.5(2). $\square$

Proof of 3.8

(1) That $A^\dagger$ is expanded by H can be shown as for $\mathrm{OM}(A)$ in Part (2) of Theorem 3.7. It remains to show that $A^\dagger$ is also contracted by H, i.e., $A \cdot A^\dagger \subseteq A^\dagger$. Assume $\sigma \in A$ and $\tau \in A^\dagger$, say $\tau \in P_{\mathrm{T}}$ for some $\mathrm{T} = (\tau_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$. Define $\Xi = (\xi_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$ by $\xi_0 =_{df} \sigma$ and $\xi_{n+1} = \tau_n$ for all $n > 0$. Then $\sigma \cdot \tau \in P_\Xi \subseteq A^\dagger$.

(2) Consider $\sigma = (e, f) \in X$. By Part (1) of Theorem 3.7 there is a sequence $\mathrm{T} = (d_n, g_n)_{n\in\mathbb{N}} \in \mathrm{ISEQ}(A)$ with $\prod \mathrm{T} \sqsubseteq \sigma$. Let $d =_{df} \sup\{d_n \mid n \in \mathbb{N}\}$ and define $\tau =_{df} (d, g) \in A^\dagger$ by $g(t) = g_n(t)$ if $t \leq d_n$ and $g(d) =_{df} f(d)$ if $d \neq \infty$. Then by construction $\tau \sqsubseteq \sigma$.

(3) We observe that the set of elements expanded by H is closed under extension, i.e., if $X \subseteq A \cdot X$ and Y is arbitrary then also $X \cdot Y \subseteq A \cdot X \cdot Y$. Therefore $A^\dagger \cdot \top$ is expanded by H and hence $A^\dagger \cdot \top \subseteq A^\omega$. For the reverse inclusion consider $\tau \in A^\omega$. By Part (2) there is a $\sigma \in A^\dagger$ with $\sigma \sqsubseteq \tau$. But then $\tau \in A^\dagger \cdot \top$. $\square$

Proof of 5.1

(1) ((a) $\Rightarrow$ (b)) The claim is equivalent to $c \sqcap ((c \sqcap \mathsf{F}) \cdot \overline{c} + \overline{c} \cdot \top) \leq 0$ by shunting (2). Then by Boolean algebra, submodularity applied twice, Boolean algebra again, left annihilation and $c \sqcap \mathsf{F} \leq \mathsf{F}$,

$$
\begin{aligned}
& c \sqcap ((c \sqcap \mathsf{F}) \cdot \overline{c} + \overline{c} \cdot \top) \\
= \;& (c \sqcap (c \sqcap \mathsf{F}) \cdot \overline{c}) + (c \sqcap \overline{c} \cdot \top) \\
\leq \;& (c \sqcap c \sqcap \mathsf{F}) \cdot (c \sqcap \overline{c}) + (c \sqcap \overline{c}) \cdot (c \sqcap \top) \\
= \;& (c \sqcap \mathsf{F}) \cdot 0 + 0 \cdot (c \sqcap \top) = 0
\end{aligned}
$$

((b)$\Rightarrow$(c)) By Boolean algebra, distributivity, (b) and isotony,
$\mathsf{F} \cdot \overline{c} \cdot \top = (c \sqcap \mathsf{F} + \overline{c} \sqcap \mathsf{F}) \cdot \overline{c} \cdot \top = (c \sqcap \mathsf{F}) \cdot \overline{c} \cdot \top + (\overline{c} \sqcap \mathsf{F}) \cdot \overline{c} \cdot \top \leq \overline{c} \cdot \top + \overline{c} \cdot \top \leq \overline{c}$.

((c)$\Rightarrow$(a)) Consider first a product $a \cdot b$ with purely finite a, i.e., with $a \leq \mathsf{F}$. By Boolean algebra and distributivity,

$$a \cdot b = (c \sqcap a) \cdot (c \sqcap b) + (c \sqcap a) \cdot (\overline{c} \sqcap b) + (\overline{c} \sqcap a) \cdot b$$

By $a \leq \mathsf{F}$ and the assumption about c, we have $\mathsf{F} \cdot \overline{c} \leq \overline{c}$ and $\overline{c} \cdot \top \leq \overline{c}$, so that the last two summands are $\leq \overline{c}$ by isotony. Hence,

$$c \sqcap a \cdot b = c \sqcap (c \sqcap a) \cdot (c \sqcap b) \leq (c \sqcap a) \cdot (c \sqcap b).$$

For arbitrary a we calculate, using fin/inf decomposition, Boolean algebra and the claim for $\mathsf{fin}\, a \leq \mathsf{F}$,

$$
\begin{aligned}
& c \sqcap a \cdot b \\
= \;& c \sqcap (\mathsf{inf}\, a + \mathsf{fin}\, a \cdot b) \\
= \;& (c \sqcap \mathsf{inf}\, a) + (c \sqcap (\mathsf{fin}\, a \cdot b)) \\
\leq \;& (c \sqcap \mathsf{inf}\, a) + (c \sqcap \mathsf{fin}\, a) \cdot (c \sqcap b) \\
= \;& \mathsf{inf}\,(c \sqcap a) + \mathsf{fin}\,(c \sqcap a) \cdot (c \sqcap b) \\
= \;& (c \sqcap a) \cdot (c \sqcap b).
\end{aligned}
$$

Finally, for $c = 1$ the left hand side of Formula (b) spells out to $(1 \sqcap \mathsf{F}) \cdot \overline{1} + \overline{1} \cdot \top = 1 \cdot \overline{1} + \overline{1} \cdot \overline{1} + \overline{1} \cdot 1 = \overline{1} + \overline{1} \cdot \overline{1}$, which shows the claim.

(2) ($\Rightarrow$) We only need to show transitivity of c, which holds by

$$c \sqcap c \cdot c = (c \sqcap c) \cdot (c \sqcap c) = c \cdot c.$$

($\Leftarrow$) By isotony, $(c \sqcap a) \cdot (c \sqcap b) \leq a \cdot b$ and $(c \sqcap a) \cdot (c \sqcap b) \leq c \cdot c \leq c$, which shows $(c \sqcap a) \cdot (c \sqcap b) \leq c \sqcap a \cdot b$. The reverse inequation holds by submodularity of c.
The assertion about 1 follows, since 1 is transitive.

(3) ($\geq$) Isotony and $c \cdot c \leq c$ show $(c \sqcap a)^+ \leq a^+$ and $(c \sqcap a)^+ \leq c^+ = c$.
($\leq$) By shunting (2), star induction (3), distributivity and join splitting we have

$$\begin{aligned}
& c \sqcap a^+ \leq (c \sqcap a)^+ \\
\Leftrightarrow\ & a^+ \leq \overline{c} + (c \sqcap a)^+ \\
\Leftarrow\ & a + a \cdot (\overline{c} + (c \sqcap a)^+) \leq \overline{c} + (c \sqcap a)^+ \\
\Leftrightarrow\ & c \sqcap (a + a \cdot (\overline{c} + (c \sqcap a)^+)) \leq (c \sqcap a)^+ \\
\Leftrightarrow\ & c \sqcap a \leq (c \sqcap a)^+ \wedge c \sqcap a \cdot \overline{c} \leq (c \sqcap a)^+ \wedge c \sqcap a \cdot (c \sqcap a)^+ \leq (c \sqcap a)^+
\end{aligned}$$

The first conjunct holds by neutrality, isotony and $1 \leq (c \sqcap a)^*$. For the second one we have, by modularity of c,

$$c \sqcap a \cdot \overline{c} = (c \sqcap a) \cdot (c \sqcap \overline{c}) = (c \sqcap a) \cdot 0 \leq c \sqcap a \leq (c \sqcap a)^+.$$

The third conjunct is shown, using again modularity, by

$$c \sqcap a \cdot (c \sqcap a)^+ = (c \sqcap a) \cdot (c \sqcap (c \sqcap a)^+) \leq (c \sqcap a) \cdot (c \sqcap a)^+ \leq (c \sqcap a)^+.$$

The equation for * is immediate from $a^* = a^+ + 1$, the equation for $^+$ and distributivity of $\sqcap$.

(4) Assume $d \leq c$. Then by Boolean algebra $\overline{d} = \overline{c} + c \sqcap \overline{d}$. By this, shunting (2), modularity (twice) and Boolean algebra, we have

$$\begin{aligned}
& \mathsf{F} \cdot \overline{d} \cdot \top \leq \overline{d} \\
\Leftrightarrow\ & \mathsf{F} \cdot \overline{d} \cdot \top \leq \overline{c} + c \sqcap \overline{d} \\
\Leftrightarrow\ & c \sqcap \mathsf{F} \cdot \overline{d} \cdot \top \leq c \sqcap \overline{d} \\
\Leftrightarrow\ & (c \sqcap \mathsf{F}) \cdot (c \sqcap \overline{d}) \cdot (c \sqcap \top) \leq c \sqcap \overline{d} \\
\Leftrightarrow\ & (c \sqcap \mathsf{F}) \cdot (c \sqcap \overline{d}) \cdot c \leq c \sqcap \overline{d}. \quad \square
\end{aligned}$$

Proof of 5.7

(1) By Lemma 5.5 all tests of S are modular. Hence by definition of $\Box q$, shunting (2), modularity (thrice), meet on tests and Boolean test algebra (twice)

$$\begin{aligned}
& p \leq \Box q \\
\Leftrightarrow\ & p \sqcap \mathsf{F} \cdot \neg q \cdot \top \leq 0 \\
\Leftrightarrow\ & (p \sqcap \mathsf{F}) \cdot (p \sqcap \neg q) \cdot (p \sqcap \top) \leq 0 \\
\Leftrightarrow\ & p \cdot p \cdot \neg q \cdot p \leq 0 \\
\Leftrightarrow\ & p \cdot \neg q \leq 0 \\
\Leftrightarrow\ & p \leq q.
\end{aligned}$$

(2) Set $q = p$ in Part (1). $\square$

Proof of 5.8. We first show $\Box p = p \cdot (\Box p)$.
For that we start with $\Box p = p \cdot (\Box p) + \neg p \cdot (\Box p)$ and show that $\neg p \cdot (\Box p) \leq 0$. By Eq. (13), shunting (2) and the definition of box we have $\neg p \cdot \Box p = \Box p \sqcap \neg p \cdot \top \leq 0 \Leftrightarrow \overline{p \cdot \top} \leq \mathsf{F} \cdot \neg p \cdot \top$. By Eq. (14) this is equivalent to $\neg p \cdot \top \leq \mathsf{F} \cdot \neg p \cdot \top$, which holds by $1 \leq \mathsf{F}$ (Eq. (6)).
Assume now that S is weak. Then $\boxed{\mathsf{F}} p = p \cdot \boxed{\mathsf{F}} p$ is immediate from (10) and (6).
Next, we show $\Box p = (\Box p) \cdot p$.
Splitting $\Box p$ into its purely finite and purely infinite parts and using distributivity, we get the equivalent claim $\boxed{\mathsf{F}} p + \boxed{\mathbb{N}} p \leq \boxed{\mathsf{F}} p \cdot p + \boxed{\mathbb{N}} p \cdot p = \boxed{\mathsf{F}} p \cdot p + \boxed{\mathbb{N}} p$. By Lemma 3.6 (2) this reduces to $\boxed{\mathsf{F}} p \leq \boxed{\mathsf{F}} p \cdot p$. Similar arguments as above yield $\boxed{\mathsf{F}} p \cdot \neg p \leq 0$ and hence $\boxed{\mathsf{F}} p = \boxed{\mathsf{F}} p \cdot p + \boxed{\mathsf{F}} p \cdot \neg p = \boxed{\mathsf{F}} p \cdot p$. $\square$

Proof of 5.9

(1) Immediate from Lemma 5.6, Lemma 5.1 2 and safety-closedness, i.e., transitivity of boxes.

(2) This is a consequence of Part (1), since

$$\Box p = \Box p \sqcap \top = \Box p \sqcap \top \cdot \top = (\Box p \sqcap \top) \cdot (\Box p \sqcap \top) = \Box p \cdot \Box p.$$

(3) Immediate from Part (1) and Lemma 5.1(3).

(4) We show the claim for purely finite a. For purely infinite a the proof is straightforward since $a \cdot b = a$. For general a the proof proceeds by splitting a into its purely finite and purely infinite part. Set $d =_{df} \Diamond p$ and $s =_{df} \overline{d} = \Box \neg p$. By Boolean algebra and distributivity,

$$d \sqcap a \cdot b = d \sqcap (d \sqcap a) \cdot b + d \sqcap (s \sqcap a) \cdot (d \sqcap b) + d \sqcap (s \sqcap a) \cdot (s \sqcap b).$$

The first of these summands is below $(d \sqcap a) \cdot b$, the second one is below $a \cdot (d \sqcap b)$ and the third one is 0 by Part (1) and $d \sqcap s = 0$. Hence, the sum is below $(d \sqcap a) \cdot b + a \cdot (d \sqcap b)$.
The converse inequation holds by $d \cdot b \leq d$, $a \leq \mathsf{F}$, $\mathsf{F} \cdot d \leq d$ and isotony. $\square$

Proof of 5.18. We show only the first equation, $\ulcorner(b \sqcap w) = b \lfloor w \sqcap 1$ can be shown in a similar way. Using (gla), Eq. (13), shunting (2), the exchange rule, Eq. (14) and shunting again, we get

$$
\begin{aligned}
& \ulcorner(b \sqcap w) \le p \\
\Leftrightarrow\ & \neg p \cdot (b \sqcap w) \le 0 \\
\Leftrightarrow\ & \neg p \cdot b \sqcap w \le 0 \\
\Leftrightarrow\ & \neg p \cdot b \le \overline{w} \\
\Leftrightarrow\ & w \llcorner b \le \overline{\neg p} \\
\Leftrightarrow\ & w \llcorner b \le p + \overline{1} \\
\Leftrightarrow\ & w \llcorner b \sqcap 1 \le p + \overline{1} \quad \square
\end{aligned}
$$

Proof of 5.19. We only show the properties for A. The properties for E follow immediately by the relationship $\mathsf{E}a.w = \overline{\mathsf{A}a.\overline{w}}$.

(1) By the principle of indirect inequality, for a set $W \subseteq S$ we have

$$
\begin{aligned}
& u \le \sqcap(\mathsf{A}a\,.\,W) \\
\Leftrightarrow\ & u \cdot a \le \sqcap(W/a) \\
\Leftrightarrow\ & \forall w \in W :\ u \le w/a \\
\Leftrightarrow\ & \forall w \in W :\ u \cdot a \le w \\
\Leftrightarrow\ & u \cdot a \le \sqcap W \\
\Leftrightarrow\ & u \le (\sqcap W)/a \\
\Leftrightarrow\ & u \le \mathsf{A}a\,.\,(\sqcap W)
\end{aligned}
$$

(2) By definition of A and of residuals we directly get

$$
\begin{aligned}
& u \le \mathsf{A}a\,.\,(\mathsf{A}b\,.\,c) \\
\Leftrightarrow\ & u \le (c/b)/a \\
\Leftrightarrow\ & u \cdot a \le c/b \\
\Leftrightarrow\ & u \cdot a \cdot b \le c \\
\Leftrightarrow\ & u \le c/(a \cdot b) \\
\Leftrightarrow\ & u \le \mathsf{A}(a \cdot b)\,.\,c
\end{aligned}
$$

(3) Similar to Part (1). $\square$

References

[1] Eric Aaron, Harold Sun, Franjo Ivancic, Dimitris Metaxas, A hybrid dynamical systems approach to intelligent low-level navigation, CA '02: Proceedings of the Computer Animation, IEEE Press, 2002, pp. 154–163.
[2] Chritiene Aarts, Roland Carl Backhouse, Eerke A. Boiten, Henk Doornbos, Netty van Gasteren, Rik van Geldrop, Paul F. Hoogendijk, Ed Voermans, Jaap van der Woude, Fixed-point calculus, Inf. Process. Lett. 53 (3) (1995) 131–136.
[3] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, Sergio Yovine, The algorithmic analysis of hybrid systems, Theoretical Computer Science 138 (1) (1995) 3–34.
[4] Rajeev Alur, Costas Courcoubetis, Thomas Henzinger, Pei-Hsin Ho, Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems, Hybrid Systems, Springer, 1993, pp. 209–229.
[5] Rajeev Alur, David Dill, A theory of timed automata, Theoretical Computer Science 126 (2) (1994) 183–235.
[6] Rajeev Alur, Radu Grosu, Yerang Hur, Vijay Kumar, Insup Lee, Modular specification of hybrid systems in charon, Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 1790, Springer, 2000, pp. 6–19.
[7] Aaron Ames, Alessandro Abate, Shankar Sastry, Sufficient conditions for the existence of Zeno behavior, in: Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference, 2005, pp. 696–701.
[8] Jos Baeten, Jan Bergstra, Process algebra with propositional signals, ACP '95: Algebra of Communicating Processes, Elsevier, 1997, pp. 381–405.
[9] Jos Baeten, Corneils Middelburg, Process Algebra with Timing, Monographs in Theoretical Computer Science, Springer, 2002
[10] Jos Baeten, Pieter Weijland, Process algebra, Cambridge University Press, 1990
[11] Jan Bergstra, Cornelis Middelburg, Process algebra for hybrid systems, Theoretical Computer Science 335 (2–3) (2005) 215–280.
[12] Claude Bolduc, Jules Desharnais, Static analysis of programs using omega algebra with tests, in: Wendy MacCaull, Michael Winter, Ivo Düntsch (Eds.), Relational Methods in Computer Science, vol. 3929, Lecture Notes in Computer Science, 2006, pp. 60–72.
[13] Manfred Broy, Ketil Stølen, Specification and development of interactive systems: Focus on streams, interfaces, and refinement, Springer, 2001
[14] Ernie Cohen, Using Kleene algebra to reason about concurrency control. Technical report, Telcordia, Morristown, N.J., 1994.
[15] Ernie Cohen, Separation and reduction, in: Roland Backhouse, José N. Oliveira (Eds.), Mathematics of Program Construction (MPC 2000), Lecture Notes in Computer Science, vol. 1837, Springer, 2000, pp. 45–59.
[16] John Conway, Regular Algebra and Finite Machines, Chapman & Hall, 1971
[17] Martin Corbett, Designing hybrid automated manufacturing systems: A european perspective, Proceedings of the First International Conference on Ergonomics of Hybrid Automated Systems I, Elsevier, 1988, pp. 167–172.
[18] Werner Damm, Hardi Hungar, Ernst-Rüdiger Olderog, On the verification of cooperating traffic agents, in: Frank de Boer, Marcello Bonsangue, Graf Susanne, Willem de Roever (Eds.), Formal Methods for Components and Objects, Lecture Notes in Computer Science, vol. 3188, Springer, 2004, pp. 77–110.
[19] Brian Davey, Hilary Priestley, Introduction to Lattices and Order, second ed., Cambridge University Press, 2002
[20] Jen Davoren, Anil Nerode, Logics for hybrid systems, Proceedings of the IEEE 88 (7) (2000) 985–1010.
[21] Jules Desharnais, Bernhard Möller, Georg Struth, Kleene algebra with domain, ACM Transactions on Computational Logic 7 (4) (2006) 798–833.
[22] Johannes Faber, Roland Meyer, Model checking data-dependent real-time properties of the european train control system, FMCAD '06: Proceedings of the Formal Methods in Computer Aided Design, IEEE Press, 2006, pp. 76–77.
[23] Thomas Henzinger, The theory of hybrid automata, in: IEEE Symposium on Logic in Computer Science (LICS '96), IEEE Press, pp. 278–292, 1996. Extended Version: in: Kemal Inan and Robert Kurshan, (Eds.), Verification of Digital and Hybrid Systems, vol. 170, NATO ASI Series F: Computer and Systems Sciences, Springer, pp. 265–292, 2000.
[24] Thomas Henzinger, Pei-Hsin Ho, Wong-Toi Howard, HYTECH: A model checker for hybrid systems, CAV '97: Conference on Computer Aided Verification, Lecture Notes in Computer Science, vol. 1254, Springer, 1997, pp. 460–463.
[25] Peter Höfner, An algebraic semantics for duration calculus, in: Judit Gervain (Ed.), Proc. 10th ESSLLI Student Session, Heriot-Watt University Edinburgh, Scotland, 2005, pp. 99–111.
[26] Peter Höfner, From sequential algebra to Kleene algebra: Interval modalities and duration calculus. Technical Report 2005-5, Institut für Informatik, Universität Augsburg, 2005.

[27] Peter Höfner, Automated reasoning for hybrid systems — two case studies, in: Rudolf Berghammer, Bernhard Möller, Georg Struth (Eds.), Relations and Kleene Algebra in Computer Science, Lecture Notes in Computer Science, vol. 4988, Springer, 2008, pp. 191–205.
[28] Peter Höfner, Bernhard Möller, Towards an algebra of hybrid systems, in: Wendy MacCaull, Michael Winter, Ivo Düntsch (Eds.), Relational Methods in Computer Science, Lecture Notes in Computer Science, vol. 3929, Springer, 2006, pp. 121–133.
[29] Peter Höfner, Bernhard Möller, An algebra for of hybrid systems. Technical Report 2007-08, Institut für Informatik, Universität Augsburg, 2007.
[30] Peter Höfner, Georg Struth, Automated reasoning in Kleene algebra, in: Frank Pfenning (Ed.), CADE 2007, Lecture Notes in Artificial Intelligence, Subseries of Lecture Notes in Computer Science, vol. 4603, Springer, 2007, pp. 279–294.
[31] Peter Höfner, Georg Struth, Can refinement be automated? in: Eerke Boiten, John Derrick, Graeme Smith, (Eds.), International Refinement Workshop – Refine 2007, Electronic Notes in Computer Science, Elsevier, pp. 53–73, 2007, (to appear).
[32] Rufus Isaacs, Differential Games. Wiley, 1965. Republished: Dover, 1999.
[33] Jifeng He, From CSP to Hybrid Systems, in: William Roscoe (Ed.), A Classical Mind: Essays in Honour of C.A.R. Hoare, Prentice Hall, 1994, pp. 171–189.
[34] Karl Henrik Johansson, Magnus Egerstedt, John Lygeros, Sastry Shankar, On the regularization of Zeno hybrid automata, Systems & Control Letters 38 (1999) 141–150.
[35] Stephen Kleene, Representation of events in nerve nets and finite automata, in: Claude Shannon, John McCarthy (Eds.), Automata Studies, Annals of Mathematics Studies, vol. 34, Princeton University Press, 1956, pp. 3–41.
[36] Dexter Kozen, Kleene algebra with tests, Trans. Programming Languages and Systems 19 (3) (1997) 427–443.
[37] Patrick Lincoln, Ashish Tiwari, Symbolic systems biology: Hybrid modeling and analysis of biological networks, in: Rajeev Alur, George Pappas (Eds.), Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 2993, Springer, 2004, pp. 660–672.
[38] Nancy A. Lynch, Roberto Segala, Frits W. Vaandrager, Hybrid I/O automata, Information and Computation 185 (1) (2003) 105–157.
[39] Ernest Manes, David Benson, The inverse semigroup of a sum-ordered semiring, Semigroup Forum 31 (1985) 129–152.
[40] Robin Milner, Communicating and Mobile Systems: The π-calculus, Cambridge University Press, 1999
[41] Bernhard Möller, Complete tests do not guarantee domain. Technical Report 2005-6, Institut für Informatik, Universität Augsburg, 2005.
[42] Bernhard Möller, Kleene getting lazy, Science of Computer Programming 65 (2007) 195–214.
[43] Bernhard Möller, Peter Höfner, Georg Struth, Quantales and temporal logics, in: Michael Johnson, Varmo Vene (Eds.), Algebraic Methodology and Software Technology, AMAST 2006, Lecture Notes in Computer Science, vol. 4019, Springer, 2006, pp. 263–277.
[44] William Rounds, Hosung Song, The ϕ-calculus: A language for distributed control of reconfigurable embedded systems, in: Oded Maler, Amir Pnueli (Eds.), Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 2623, Springer, 2003, pp. 435–449.
[45] Alberto Sangiovanni-Vincentelli, Thomas Henzinger, Bruce Krogh, Oded Maler, Manfred Morari, Costas Pantelides, Goerge Pappas, Tunc Simsec, Janos Sztipanovits, Stavros Tripakis, Hybrid systems applications: An oxymoron?, Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 2034, Springer, 2001, pp. 5–6.
[46] Shankar Sastry, George Meyer, Claire Tomlin, John Lygeros, Datta Godbole, George Pappas, Hybrid control in air traffic management systems, Proceedings of the Thirty-Fourth IEEE Conference on Decision and Control, IEEE Press, 1995, pp. 1478–1483.
[47] Steve Schneider, Jim Davies, Daniel Jackson, George Reed, Joy Reed, William Roscoe, Timed CSP: Theory and practice, Proceedings of the Real-Time: Theory in Practice, REX Workshop, Springer, 1992, pp. 640–675.
[48] Michel Sintzoff, Iterative synthesis of control guards ensuring invariance and inevitability in discrete-decision games, in: Olaf Owe, Stein Krogdahl, Tom Lyche (Eds.), From Object-Orientation to Formal Methods, Essays in Memory of Ole-Johan Dahl, Lecture Notes in Computer Science, vol. 2635, Springer, 2004, pp. 272–301.
[49] Gheorghe Ştefănescu, Network Algebra, Springer, 2000
[50] Burghard von Karger, Temporal algebra, Mathematical Structures in Computer Science 8 (3) (1998) 277–320.
[51] Martin von Mohrenschildt, Closed form solutions of hybrid systems. CRL Report 371, Faculty of Engineering, McMaster University, 1999.