Faculty of Mathematical Sciences

University of Twente

University for Technical and Social Sciences

P.O. Box 217
7500 AE Enschede
The Netherlands
Phone: +31-53-4893400
Fax: +31-53-4893114
Email: memo@math.utwente.nl

Memorandum No. 1549

Compositionality issues in discrete, continuous and hybrid systems

A.J. van der Schaft and J.M. Schumacher[1]

October 2000

ISSN 0169-2690

[1]Department of Econometrics and Operations Research, Tilburg University, P.O. Box 90153, 5000 LE Tilburg

Compositionality Issues in Discrete, Continuous, and Hybrid Systems

A.J. van der Schaft
Faculty of Mathematical Sciences, University of Twente
P.O. Box 217, 7500 AE Enschede, the Netherlands
Phone 31-53-4893449, Fax 31-53-4340733
a.j.vanderschaft@math.utwente.nl

J.M. Schumacher
Department of Econometrics and Operations Research
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, the Netherlands
jms@kub.nl

Abstract

Models of complex dynamical systems are often built by connecting submodels of smaller parts. The key to this method is the operation of "interconnection" or "composition" which serves to define the whole in terms of its parts. In the setting of smooth differential equations the composition operation has often been regarded as trivial, but a quite different attitude is found in the discrete domain where several definitions of composition have been proposed and different semantics have been developed. The nontriviality of composition carries over from discrete systems to hybrid systems. The paper discusses the compositionality issue in the context of discrete, continuous, and hybrid systems, mainly on the basis of a number of examples.

Keywords: modularity, non-deterministic systems, multitime semantics, jump rules

1 Introduction

It is by now widely accepted that the modeling of complex technical systems requires some form of object orientation. To model a complex system, one needs to break it up into parts, which again may be broken up into smaller parts, until a level is reached where elements are simple enough to be described directly. The element models are then combined into a system model by stagewise composition. The importance of modularity and hierarchy as tools for managing complexity has long been recognized in computer science; but also in the modeling of continuous systems the concept of "network modeling" has become a standard, as is seen for instance from the popularity of simulation languages such as Simulink and Modelica which allow the user to put together a system model by connecting elements taken from a library provided by the software. The role of composition is expected to be emphasized even more in the modeling of realistic-size hybrid systems.

The aim of network modeling may be summarized as: inferring relevant properties of a system as a whole from information provided by subsystems. The operation of composition that is applied should be strong enough to support this process but should not needlessly limit flexibility in subsystem descriptions. In principle therefore the question must be posed what operation of composition is most suitable, given a specification of what properties of the system as a whole are of interest and what information can be provided by subsystems. Questions of this type have been studied much more intensely in the domain of discrete systems than in the domain of continuous systems. In computer science, several different notions of composition are used and are related to various notions of subsystem behavior. To continuous-system analysts working with differential equations however, system interconnection simply means joining the equations of all subsystems together.

In this paper we discuss notions of composition in the context of discrete-event systems, continuous systems and hybrid systems. With some abuse of notation, the composition operator will in each case be denoted by the symbol $\|$. The foremost properties that should be satisfied by the composition operator are *commutativity* and *associativity*, i. e. for all A, B, C one should have $A \parallel B = B \parallel A$ and $(A \parallel B) \parallel C = A \parallel (B \parallel C)$. The symbols A, B, and C that appear here denote subsystem descriptions in some specified format, and the notion of equality that is used is interpreted in the sense of equivalence with respect to some *semantics* that is specified along with the description format. Usually one associates to a given subsystem description a set of associated *solutions*, which may be referred to by various names such as "traces", "trajectories", or "executions", depending on the context. Again depending on the context, the collection of all solutions corresponding to a given subsystem description may be called for instance a "language" or a "behavior". These objects are sets and in particular we can apply the operation of intersection to them. Denoting the collection of solutions corresponding to a given description A by $L(A)$, we say that the *rule of modular behavior* holds if for all A and B we have

$$L(A \parallel B) \;=\; L(A) \cap L(B). \tag{1.1}$$

In many applications it is important to consider also an *inclusion* or *implementation* relation. Such a relation may be used to express that a given system satisfies certain requirements. These requirements are often referred to as *specifications*; they may be expressed in the same format as is used for subsystem descriptions, or in a format of their own (a *specification language*). The notion of composition may be also applied to specifications. If the implication

$$A_1 \sqsubset S_1, \quad A_2 \sqsubset S_2 \quad \Rightarrow A_1 \parallel A_2 \sqsubset S_1 \parallel S_2 \tag{1.2}$$

holds, where the symbol $\sqsubset$ denotes an implementation relation, then we say that the *rule of modular control* applies. This rule guarantees that controlled subsystems not only satisfy their control specifications when disconnected from each other, but also when they are in actual interconnection with each other. For example, it guarantees that whenever the composition of an uncontrolled system P with a controller (or supervisor) system C is such that $P \parallel C \sqsubset S$ for a certain specification process S, then $P \parallel C$ may actually *replace* the specification process S in any larger system (see e.g. [1, 2, 3] for further motivation and background).

The formal consideration of implementation relations is well developed in computer science but has received less attention in continuous system theory. Only recently, fundamental studies of interconnection of continuous systems have been performed in the behavioral framework, see e. g. [4, 5].

The fact that composition is a major motive in the discrete world suggests that the relatively low profile of this issue in the continuous setting will not be maintained in the context of hybrid systems. Quite to the contrary, it is to be expected that the richness of the study of compositionality in the discrete domain will be inherited and perhaps even enhanced in the domain of hybrid systems. In this paper we make some comparisons between compositionality issues in discrete, continuous, and hybrid systems. We do not attempt to present any definitive theory; instead we present a number of illustrative examples which may help to focus ideas. Our main purpose is to formulate some concrete questions for research. Furthermore, we present a multitime semantics for event-flow formulas (as introduced for a modular description of hybrid systems in [6]), allowing composition of hybrid systems by simple conjunction without imposing global synchronization. The focus of the paper is on modeling rather than on control.

The paper is structured as follows. We first illustrate the nontriviality of composition in a discrete context (Section 2). Then, in Section 3, we turn to continuous systems. Compositionality in hybrid systems is investigated in Sections 4 and 5, with Section 4 presenting the multitime semantics of event-flow formulas, and Section 5 discussing compositionality of jump rules. Conclusions follow in Section 6.

2 Composition in discrete systems

The theory of modelling and control of discrete-event systems has been a vital research area in systems and control theory over the last fifteen years. An informative survey, including many references to the extensive literature, can be found in [7]. Since we wish to highlight compositionality issues we confine ourselves in this section to basic notions of composition and implementation of (non-deterministic) automata and processes, which are well-known in computer science; see e.g. [8, 9, 10]. A treatment of non-deterministic discrete-event systems which emphasizes compositionality issues can be found in [2, 3]; see also [11, 12] for other developments in the control of non-deterministic systems. Here we summarize the main line of development in the theory of automata as it can be found for instance in [13, 10, 8, 3].

An *automaton* is defined in the usual way by a triple (Σ, X, E). Here X is the state space of the automaton (the set of nodes in a graph representation of the automaton), Σ is the *alphabet* whose elements are called event labels of *symbols*, and $E \subset X \times \Sigma \times X$ is the transition rule (the set of edges). Often X and Σ are assumed to be finite sets.

The set E determines a (possibly partial) *transition function* $\delta : X \times \Sigma \to \mathcal{P}(X)$ by

$$\delta(x, \sigma) = \{y \in X \mid (x, \sigma, y) \in E\}. \tag{2.3}$$

where $\mathcal{P}(X)$ denotes the collection of all subsets of X. Furthermore, usually a subset $X_0 \subset X$ is identified as the set of *initial* states of the automaton. The *language* $L(A)$ generated by an automaton $A = (\Sigma, X, E)$ is the set of all possible finite strings of symbols (*traces*) corresponding to the transition rule E and initial states X_0, that is, all sequences

$$s = \sigma_0 \sigma_1 \cdots \sigma_n \tag{2.4}$$

for which there exists a state space sequence $x_0, x_1, \cdots, x_n$, such that $x_0 \in X_0$, and

$$(x_i, \sigma_i, x_{i+1}) \in E, \quad i = 0, 1, \cdots, n-1. \tag{2.5}$$

The automaton (Σ, X, E) is called *deterministic* if for every (x, σ) the set $\delta(x, \sigma)$ is at most a singleton. Furthermore, it is often required that X_0, the set of initial states, is a *singleton.* Two *deterministic* automata are considered *equivalent* if they generate the same language, cf. [8].

In the non-deterministic case, however, automata that generate the same language may have different properties, especially with respect to *deadlock* behavior. One way to characterize such properties of non-deterministic automata is provided by the notion of *failure semantics*, as introduced in [10]. An automaton A is said to *refuse* a symbol subset $R \subset \Sigma$ after a string $s \in L(A)$ if it can reach a state x

by executing string s, and it can*not* execute any symbol in the set R in this state. The subset R is called a *refusal*, and the collection of refusals is called the refusal set $\mathrm{ref}(A, s) \subset \mathcal{P}(\Sigma)$. For a deterministic automaton A the following equivalence holds:

$$R \in \mathrm{ref}(A, s) \Leftrightarrow R \subset \{\sigma \in \Sigma \mid s\sigma \notin L(A)\} \tag{2.6}$$

for all $s \in L(A)$. Conversely, any automaton with this equivalence property is deterministic. However, for a non-deterministic automaton the implication $\Rightarrow$ need not hold. For instance, it may happen that by executing a certain string s the automaton ends up in a deadlock state, while there is another path in the automaton that generates the same string s but that leads to another state (node) not exhibiting deadlock behavior. Clearly, in this case the observation of the generated trace does not uniquely determine the state of the automaton after execution of this trace, and in this sense non-determinism is related to a lack of *observability* in the sense of behavioral theory [14]: the paths of the external variables (symbols) do not uniquely determine the paths of the internal variables (states).

Non-deterministic automata can be considered to be *equivalent* if they generate the same language *and* their refusal sets are the same. The corresponding semantics is known as *failure semantics*. Let us add here that apart from failure semantics there are other (and stronger) semantics that can be used to reason about non-deterministic systems, such as bisimulation semantics [9].

Given two automata $A = (\Sigma_A, X_A, E_A)$ and $B = (\Sigma_B, X_B, E_B)$ with $\Sigma_A = \Sigma_B$, the *synchronous* (parallel) *composition* $A \parallel B$ is defined as the automaton with state space $X_A \times X_B$, alphabet $\Sigma := \Sigma_A = \Sigma_B$ and transition rule E given by

$$\begin{aligned} E \ = \ \big\{ \big(\big(x_1^A, x_1^B\big), \sigma, \big(x_2^A, x_2^B\big)\big) \in X_A \times X_B \times \Sigma \times X_A \times X_B \mid \\ \big(x_1^A, \sigma, x_2^A\big) \in E_A, \big(x_1^B, \sigma, x_2^B\big) \in E_B \big\} \end{aligned} \tag{2.7}$$

It follows readily that the language generated by the composition $A \parallel B$ satisfies the *rule of modular behavior* (1.1)

$$L(A \parallel B) = L(A) \cap L(B). \tag{2.8}$$

For deterministic automata this property can alternatively be taken as the *definition* of composition, in the sense that the *canonical* deterministic automaton determined by the language $L(A) \cap L(B)$ (using Nerode equivalence) is equivalent to $A \parallel B$. However, for *non-deterministic* automata the situation becomes different. It is easy to see that the synchronous composition $A \parallel B$ of two non-deterministic automata A and B has the following property with respect to its refusal sets

$$\mathrm{ref}(A \parallel B, s) = \{R_a \cup R_b \mid R_a \in \mathrm{ref}(A, s), R_b \in \mathrm{ref}(B, s)\} \tag{2.9}$$

This property should be explicitly taken into account, in addition to the intersection property (2.8), when defining composition of non-deterministic automata on the *process* level (that is, on the level of the generation of the *symbols*).

The same issues apply to the *inclusion* condition $A \sqsubset B$ of two automata A and B (sometimes called an implementation relation, cf. [10]). In the deterministic case it makes sense to define $A \sqsubset B$ by the condition

$$L(A) \subset L(B), \tag{2.10}$$

implying that $A \sqsubset B$ together with $B \sqsubset A$ yields $L(A) = L(B)$, so that A and B are equivalent. In the non-deterministic case, however, condition (2.10) is not strongh enough for $A \sqsubset B$, since we should also require that

$$\mathrm{ref}(A, s) \subset \mathrm{ref}(B, s), \qquad \text{for all } s \in L(A). \tag{2.11}$$

Indeed, if we define the inclusion $A \sqsubset B$ by conditions (2.10) *and* (2.11), then $A \sqsubset B$ and $B \sqsubset A$ imply that $L(A) = L(B)$ and $\mathrm{ref}(A, s) = \mathrm{ref}(B, s)$ for all $s \in L(A) = L(B)$, so that A and B are equivalent non-deterministic automata (in the sense of failure semantics).

With respect to the *rule of modular control* (1.2) this discussion implies the following. In the deterministic case the property $P \sqsubset S$ for automata P, S easily implies that $G \parallel P \subset G \parallel S$ for any other automaton G, by the simple set-theoretic inclusion condition (2.10). For the non-deterministic case the rule of modular control (1.2) still can be shown to hold, see e.g. [13], if we let as above the inclusion $\sqsubset$ be defined by (2.10) and (2.11).

3 Composition in continuous systems

A central idea in systems theory is to view a dynamical system as the "*interconnection*" or "*composition*" of subsystems, and to relate the properties of the overall system to the properties of the individual subsystems. Notwithstanding this central role a *formalization* of the notion of composition or interconnection in continuous systems and control theory has been somewhat lacking. Instead, the definition of interconnection has been often taken for granted, by relying on standard feedback interconnections and well-established interconnection structures like Kirchhoff's laws. As we have seen in the previous section this is in contrast with the situation in automata theory and communicating processes where the notion of composition has received wide attention.

Recently, a formalization of the notion of composition or interconnection of continuous systems has been elaborated within the behavioral approach to systems theory as advocated by Willems and co-workers, see e.g. [4, 5, 14]. In this set-up the variables used in the description of a system are split into two classes: the *manifest* or external variables $w \in W$, and the *latent* or internal variables $\lambda \in \Lambda$. The *behavior* $\mathcal{B}$ of a system is then defined as the set of all possible time trajectories of the external variables w, where in the continuous time case the time-axis is equal to the whole real line $\mathbb{R}$ (or e.g. the interval $[0, \infty)$). Note that this definition of the behavior of a continuous time system is very similar to the notion of the *language* generated by an automaton, if we identify the symbol (event labels) with the values taken by the external variables.

The notion of *interconnection* of two continuous systems Σ_1 and Σ_2 with shared space of external variables W, and behaviors $\mathcal{B}_1$, respectively $\mathcal{B}_2$, is then simply defined as the system with behavior $\mathcal{B}$ given by the intersection

$$\mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2, \tag{3.12}$$

just as in the case of the composition of two deterministic automata, as treated in the previous section. Hence, by definition, the rule of modular behavior (1.1) holds. Similarly, the inclusion or implementation relation between two continuous systems Σ_1 and Σ_2 with shared space of external variables W is simply given by the inclusion of their behaviors: Σ_1 implements Σ_2 ($\Sigma_1 \sqsubset \Sigma_2$) if

$$\mathcal{B}_1 \subset \mathcal{B}_2. \tag{3.13}$$

As in the case of deterministic automata, it immediately follows from basic set-theoretic operations that for the above composition (interconnection) and implementation (inclusion) rules the rule of modular control (1.2) is valid. The paper [5] then continues with interesting developments concerning possible *feedback* interpretations of interconnection.

Nevertheless, also for continuous systems it is quite possible to encounter situations where the above implementation and composition rule has unsatisfactory features, as was the case for non-deterministic automata or processes. As a simple mathematical example consider the following continuous system:

$$\begin{array}{rcl} \dot{x} & = & \lambda_1 \\ w & = & \lambda_2 x \end{array} \tag{3.14}$$

with external variable w and internal variables λ_1, λ_2 and x (regarded as a *state* variable). Clearly, the behavior $\mathcal{B}$ of this system is equal to the behavior of the trivial system with external variable w where we do not impose any restrictions on w. On the other hand, we would like to distinguish, certainly for interconnection purposes, this system from the trivial system, since if the state x happens to be driven to zero by the latent variable λ_1, then the system can only generate the zero trajectory $w(t) = 0$. Note that this situation is very similar to the situation that may arise in non-deterministic automata, for instance when it possible to arrive at a state from which deadlock occurs, while the same trace can be also generated while ending up in a different state. Also note that in the above example the internal variables λ_1, λ_2 and x are clearly not *observable* from the external variable w. These considerations strongly suggest to develop also for continuous systems a certain notion of failure semantics (or, bisimulation semantics), together with a corresponding notion of implementation and composition, such that the rules of modular control and modular behavior remain valid in this context. This would entail a refinement and extension of behavioral systems theory, by not only considering the ensemble of all possible traces (the behavior $\mathcal{B}$ in the sense of Willems, cf. [5]), but $\mathcal{B}$ endowed with an additional "refusal" structure. It goes without saying that these considerations become even more pressing in the study of *hybrid systems*, as discussed in the next section. Note that it is not immediate how to properly generalize the definition of refusal sets for discrete systems to the realm of continuous-time systems. In this context we remark, without going into details, that for *linear systems* as studied in [5, 14] such a refusal structure does not add any information. Indeed, a notion of failure semantics for continuous systems seems tied up with *nonlinearity*, as is already suggested by the simple example given above.

Finally we mention that in the theory of interconnection of continuous systems *initialization* of the state variables may play an important role. Indeed, interconnection constraints on the shared external variables w will often induce *algebraic constraints* on the state or internal variables of the connected systems. This is the reason that a proper framework for the modular representation of continuous systems is given by DAE's (differential and algebraic equations). It means that for interconnecting continuous systems the state variables of the connected systems have to be (re-)initialized. Although similar considerations hold for discrete systems the situation in the continuous case seems more complex.

4 Event-flow formulas

Many formalisms for the description of hybrid systems have been proposed in the literature; see for instance [15, 16, 17, 18, 19, 20, 21]. A description format for hybrid systems that is close to the usual modeling of continuous systems by means of differential equations was proposed in [6]. In this format a hybrid system is represented by a so-called "event-flow formula" (EFF) which consists of a set of equations connected by Boolean operators. The equations in an EFF may include differential equations, update equations, and algebraic equations and inequalities. An operation of composition is defined in [6] by which a new EFF can be formed from two or more given ones. This composition makes use of so-called "empty events" to express the non-synchronicity of events in different subsystems. Here we present an alternative approach which makes non-synchronicity more explicit. The somewhat artificial device of empty events is removed in this way, at the expense of introducing a more complicated notion of time.

One should distinguish between the *syntax* of a description format (the rules that determine what is to be considered as a well-formed description) and its *semantics*, which in the case of dynamical systems can be interpreted as the notion of solution. In principle different semantics can be attached to the same syntax. We now first describe the syntax of EFFs.

4.1 Definition of EFFs

We start with a finite index set V whose elements are called *variables*. The set V is the disjoint union of four subsets denoted by X, P, W, and S. The variables in X are called *continuous state variables*, those in P are called *discrete state variables*, those in W are *continuous communication variables*, and those in S are *discrete communication variables*. To each of the variables there is an associated *range space*. For the continuous variables we let this space be the real line; it would not be difficult to generalize to the case of differentiable manifolds but we do not aim here for the greatest possible generality. The range spaces of the discrete variables are finite sets which are denoted by L_i $(i = 1, \dots, k)$ in the case of state variables and by A_i $(i = 1, \dots, r)$ in the case of communication variables. The sets L_i are sometimes referred to as sets of *locations*, whereas the sets A_i are usually called *alphabets*.

For each continuous state variable $x \in X$ we introduce a new variable denoted by $\dot{x}$ which also has the real line as its range space; as the notation suggests, the symbol will be used in the semantics to express differentiation with respect to time. The set of new variables that is obtained in this way will be denoted by $\dot{X}$. Likewise, for each continuous state variable $x \in X$ and each discrete state variable $p \in P$ we introduce new variables $x^\sharp$ and $p^\sharp$ which will be used to express update operations. Both new variables have the same range space as the variables from which they are derived. The new sets of variables that are thus obtained will be denoted by $X^\sharp$ and $P^\sharp$. We write $V' := V \cup \dot{X} \cup X^\sharp \cup P^\sharp$.

Let V_0 be a subset of V'. A *valuation* of V_0 is a mapping that assigns to each element of V_0 an element of its associated range space. If the elements of V_0 are given a fixed order, then valuations of V_0 can be written as vectors whose length is the number of elements of V_0.

A *clause* over V_0 is a mapping that assigns to each valuation of V_0 the value TRUE or FALSE. In applications, a clause is typically given by an arithmetic or logical expression. As a trivial example, if $V_0 = \{x, x^\sharp\}$ (taken in this order), then a clause over V_0 is for instance given by the expression $x^\sharp = x + 1$, which returns TRUE for the valuation $(0, 1)$ and FALSE for the valuation $(0, 0)$. The semantics to be developed below is based in particular on clauses over variables in $X \cup P \cup W \cup \dot{X}$ (*flow clauses*) and clauses over variables in $X \cup P \cup S \cup X^\sharp \cup P^\sharp$ (*event clauses*). If ϕ is a clause over V_0 then we also say that V_0 is the *span* of ϕ, and we write span(ϕ)= V_0.

Finally we can express the notion of an event-flow formula.

Definition 4.1 An *event-flow formula*, or EFF, is a Boolean formula whose terms are clauses.

4.2 A multitime semantics for EFFs

Event-flow formulas are intended to represent the set of possible evolutions of systems that are partly described by differential equations and partly by update operations. Updates will take place at *event times* that may be different for different evolutions. Moreover, not all variables in a hybrid system need to be updated at the same time; it may happen that an event is local to some subsystem. As a consequence, we need a concept of time that is considerably more complicated than the usual model based on the real line. We shall take a fairly radical point of view here and equip each variable with its own time axis. Therefore we begin with defining a suitable notion of time axis for a single variable. Then we proceed to joint evolutions of all variables that occur in a given EFF and we define in what sense an EFF can be satisfied by such a joint evolution. In the process we obtain an overall time axis which however is not in general a totally ordered set; this reflects the idea of *partial synchronization*.

4.2.1 Time axes for single variables

To model the events that affect a certain variable, we need time axes in which certain points may have multiplicity in a suitable sense. For this we use the following model. Given an interval T of the real line, an *enrichment of* T is a combination $((\mathbb{T}, \leq), \pi)$ satisfying the following axioms:

(i) $\mathbb{T}$ is a totally ordered set with order relation $\leq$;

(ii) π is a surjective and order-preserving mapping from $\mathbb{T}$ to T;

(iii) for each $t \in T$, the set $\{\tau \in \mathbb{T} \mid \pi(\tau) = t\}$ is well-ordered with respect to the order $\leq$ and has a maximum.

By abuse of notation, we shall also write $(\mathbb{T}, \pi)$ or even just $\mathbb{T}$ instead of $((\mathbb{T}, \leq), \pi)$; in these cases the order relation $\leq$ and the mapping π are taken to be understood. An enriched time interval will also sometimes be called a *rich time axis*.

To explain the definition, we note that the mapping π serves to establish contact between the enrichment $\mathbb{T}$ and the "physical time" expressed through T. The interval T may be the whole real line, a finite interval, or even just a single point in $\mathbb{R}$. The latter case may be thought of as being degenerate in the sense that it leaves no room for continuous dynamics.

The order preservation property of the mapping π is taken in the sense that, for all τ_1 and τ_2 in $\mathbb{T}$, $\tau_1 \leq \tau_2$ implies $\pi(\tau_1) \leq \pi(\tau_2)$ where the latter inequality refers, of course, to the usual ordering of the real line. In particular the mapping π need not be one-to-one. If $t \in T$ is such that the set $\pi^{-1}[\{t\}] := \{\tau \in \mathbb{T} \mid \pi(\tau) = t\}$ (the *fibre of* t) contains more than one point, then t is said to be an *event time* with respect to $\mathbb{T}$. The set of event times of $\mathbb{T}$, which is a subset of T, will be denoted by $\mathcal{E}(\mathbb{T})$.

In axiom (iii) it is required that fibres are well-ordered; this means that every nonempty subset of $\pi^{-1}[\{t\}]$ must have a minimal element. In particular the fibre $\pi^{-1}[\{t\}]$ itself has a minimum, which we shall denote by t^-. The maximum of the fibre, whose existence is required by axiom (iii) as well, will be denoted by t^+. From the well-ordering it also follows that each element τ of the fibre of t, except t^+, has a successor; the successor of τ will be denoted by $\tau^\sharp$.

We equip the rich time axis $\mathbb{T}$ with the order topology. That is to say, a basis for the topology of $\mathbb{T}$ is given by the open intervals, i. e. subsets of the form $(\tau_1, \tau_2) := \{\tau \in \mathbb{T} \mid \tau_1 < \tau < \tau_2\}$. The surjectivity and monotonicity of the mapping π ensure its continuity with respect to this topology on $\mathbb{T}$ and the usual topology on the interval T.

In addition to continuity we will also need a notion of differentiability. Given a rich time axis $(\mathbb{T}, \pi)$, we denote by $\mathbb{T}^c$ (the *continuous part* of $\mathbb{T}$) the set of all points of $\mathbb{T}$ that have no neighborhood on which the mapping π is constant. We equip this set with the toplogy that it inherits from $\mathbb{T}$. A real-valued function x

defined on $\mathbb{T}$ is said to be *differentiable* in $\tau_0 \in \mathbb{T}_c$ with *derivative* $\dot{x}(\tau_0)$ if for all positive ε there is a neighborhood N of τ_0 in $\mathbb{T}$ such that for all $\tau \in N$ we have

$$|x(\tau) - x(\tau_0) - \dot{x}(\tau_0)(\pi(\tau) - \pi(\tau_0))| \;\leq\; \varepsilon\, |\pi(\tau) - \pi(\tau_0)|. \tag{4.15}$$

The function x is said to be *differentiable* if it is differentiable for all $\tau \in \mathbb{T}^c$; if the resulting function $\dot{x}(\cdot)$ is continuous on $\mathbb{T}^c$, then x is said to be *continuously differentiable*.

A rich time axis $\mathbb{T}$ having the property that the interval (t^-, t^+) is empty for all $t \in \mathcal{E}(\mathbb{T})$ will be called a *simply punctuated time axis*. A *timed event set* defined on a given interval T of the real line is a collection of well-ordered sets parametrized by a the points in a subset $\mathcal{E}$ of T.

4.2.2 Evolutions

An *evolution* of a discrete or continuous state variable v is a pair consisting of a rich time axis $\mathbb{T}$ and a mapping from $\mathbb{T}$ to the range space associated with v. Evolutions of continuous communication variables are defined similarly, with the additional condition that the time domain $\mathbb{T}$ should be a simply punctuated time axis. For discrete communication variables, evolutions are again defined in the same way, but replacing the rich time axis by a timed event set. So evolutions can be thought of as time-trajectory pairs. By abuse of notation, trajectories will be indicated by the same symbol as the variable to which they refer; context will make clear whether a symbol, say, v denotes an element of the index set V or a mapping from a time axis $\mathbb{T}$ to the range space associated with v.

A *global evolution* is a mapping that assigns to each $v \in V$ an evolution $(\mathbb{T}_v, v)$ which is such that all time domains $\mathbb{T}$ correspond to the same interval T of the real axis. Discrete communication variables are in fact only defined on part of T. Actually the framework presented here might easily be extended to allow also other variables to be defined only on part of T, in the spirit of Benveniste's "presences" [20]. Although such a feature is certainly useful in some applications, we shall refrain here from extending the framework in this way.

4.2.3 Multitimes

Let a global evolution $\bar{v}$ over V be given, with physical time domain T. This provides us in particular with a collection of time axes $\{(\mathbb{T}_v, \pi_v) \mid v \in V\}$. A *multitime* is a pair $(V_0, \bar{\tau})$ where V_0 is a subset of V and $\bar{\tau}$ is a mapping from V_0 to the product space $\Pi_{v \in V_0} \mathbb{T}_v$ such that there exists a point $t \in T$ such that for all $v \in V_0$ we have $\pi_v(\tau(v)) = t$. Since V_0 is a finite set, we can think of the mapping $\bar{\tau}$ as a vector of times $(\tau_{v_1}, \ldots, \tau_{v_k})$ indexed by the variables appearing in V_0. With slight abuse of notation we shall denote a multitime usually by the vector $\bar{\tau}$. We say that $\bar{\tau}$ is a multitime *over* V_0, or also that the set V_0 is the *span* of the multitime $\bar{\tau}$.

4.2.4 Evaluation of clauses

Let ϕ be a clause defined over a subset V_1 of V'; recall that V' is the original set of variables V extended with symbols of the form $\dot{v}$ (for continuous state variables v) and $v^\sharp$ (for continuous or discrete state variables v). Let V_1^r denote the set of variables $v \in V$ for which one or more of the variables v, $\dot{v}$, $v^\sharp$ appear in V_1, and write $V_1^a := V_1 \cup V_1^r$. Given a global evolution $\bar{v}$ and given a multitime $\bar{\tau}$ whose span contains V_1^r, we can, under some conditions to be specified below, define valuations of all variables in the span of ϕ. This is done as follows. In the spirit of the abuse of notation that was already announced before, we denote by v the trajectory associated to the variable v by the global evolution $\bar{v}$.

1. For variables $v \in V \cap V_1^r$, the valuation of v at $\bar{\tau}$ determined by $\bar{v}$ is $v(\tau_v)$.
2. For variables $v^\sharp \in V_1$, the valuation of $v^\sharp$ at $\bar{\tau}$ determined by $\bar{v}$ is $v(\tau_v^\sharp)$, if τ_v has a successor; otherwise the valuation of $v^\sharp$ is not defined.
3. For variables $\dot{v} \in V_1$, the valuation of $\dot{v}$ at $\bar{\tau}$ determined by $\bar{v}$ is $\dot{v}(\tau_v)$, if v is differentiable at τ_v; otherwise the valuation of $\dot{v}$ is not defined.

If all variables are valuated, then we say that the given clause can be *evaluated* at the multitime $\bar{\tau}$ in the evolution $\bar{v}$.

4.2.5 The solution concept

From single clauses, we now go to collections of clauses. Recall that event-flow formulas have been defined as Boolean combinations of clauses. We denote by $C(\bar{\phi})$ the set of clauses that appear in a given EFF $\bar{\phi}$. Every assignment of values TRUE or FALSE to the clauses in $C(\bar{\phi})$ will generate a value TRUE or FALSE for the EFF as a whole. As we think of EFFs as representing systems consisting of subsystems that evolve partly independently, we will often want to draw conclusions about the truth value of an EFF based on only partial information from the constituent clauses. We shall say that an EFF $\bar{\phi}$ is *validated* by a given evaluation of a subset C_0 of $C(\bar{\phi})$ (so an assignment of the value TRUE or FALSE to all clauses in C_0) if the value of $\bar{\phi}$ is TRUE for the given values of the clauses in C_0 and for all values of the clauses in $C(\bar{\phi}) \setminus C_0$. We shall say that $C_0 \subset C(\bar{\phi})$ is *supporting* for a global evolution $\bar{v}$ at a multitime $\bar{\tau}$ if all clauses in C_0 can be evaluated in the given evolution at the given multitime, and $\bar{\phi}$ is validated by the resulting evaluation.

We need a few more definitions before we arrive at the final solution concept. Let a global evolution $\bar{v}$ be given. Recall that a multitime $\bar{\tau}$ assigns to each variable v in a given subset V_0 of V an element of the time axis $\mathbb{T}_v$ that is associated to v by the global evolution $\bar{v}$. If $\bar{\tau}_1$ and $\bar{\tau}_2$ are multitimes defined over the same set of variables V_0, then we say that $\bar{\tau}_1$ *precedes* $\bar{\tau}_2$, and we write $\bar{\tau}_1 < \bar{\tau}_2$, if for all $v \in V_0$ we have $\bar{\tau}_1(v) < \bar{\tau}_2(v)$ where in the latter expression the ordering is derived from the ordering of $\mathbb{T}_v$. We shall say that a collection $\bar{\mathbb{T}}$ of multitimes over a fixed set of variables V_0 is *strictly ordered* if for all $\bar{\tau}_1 \neq \bar{\tau}_2 \in \bar{\mathbb{T}}$ we have either $\bar{\tau}_1 < \bar{\tau}_2$ or $\bar{\tau}_2 < \bar{\tau}_1$. If $\bar{\tau}$ is a multitime over V_0 and $V_1 \subset V_0$, then the *restriction* of $\bar{\tau}$ to V_1 is denoted by $\bar{\tau}|_{V_1}$.

Given an EFF $\bar{\phi}$, a *$\bar{\phi}$-certificate* for an evolution $\bar{v}$ is a pair $(\bar{\mathbb{T}}, \bar{C})$ consisting of a collection $\bar{\mathbb{T}}$ of multitimes and a mapping $\bar{C}$ from $\bar{\mathbb{T}}$ to the collection of subsets of $C(\bar{\phi})$, such that the following conditions are satisfied:

(i) for each $\bar{\tau} \in \bar{\mathbb{T}}$, the set of clauses $\bar{C}(\bar{\tau})$ is supporting for $\bar{v}$ at $\bar{\tau}$

(ii) for each $\phi \in C(\bar{\phi})$, the set of multitimes $\bar{\mathbb{T}}_\phi := \{\bar{\tau} \mid \exists \tilde{\tau} \in \bar{\mathbb{T}} \text{ s.t. } \phi \in C(\tilde{\tau}) \text{ and } \bar{\tau} = \tilde{\tau}|_{\mathrm{span}(\phi)}\}$ (the set of multitimes at which ϕ is called) is strictly ordered

(iii) for all variables $v \in V$ and for all $\tau \in \mathbb{T}_v$ there exists exactly one multitime $\bar{\tau} \in \bar{\mathbb{T}}$ such that $\bar{\tau}_v = \tau$.

Finally, we say that the evolution $\bar{v}$ is a *solution* (or an *execution*) of the EFF $\bar{\phi}$ if it allows a $\bar{\phi}$-certificate.

4.3 Regularity classes

In the study of continuous dynamical systems it is common to work with various function classes in which solutions are considered. These classes are sometimes called regularity classes. Similarly, it seems reasonable to distinguish several regularity classes for solutions of EFFs, which may be useful in particular contexts. The following notions may play a role in defining such regularity classes.

In connection with the nature of fibres, we say that $\mathbb{T}$ has *finite multiplicity* if all fibres have finite cardinality, and that it has *maximal multiplicity* m if none of the fibres contains more than $m+1$ points. The rich time axis $\mathbb{T}$ is said to have *ω-bounded multiplicity* if the ordinal number of fibres is at most $\omega+1$. Concerning event times, the rich time axis $\mathbb{T}$ is said to have *positive minimum dwell-time* if there exists $\delta > 0$ such that $|t_1 - t_2| \geq \delta$ for all $t_1, t_2 \in \mathcal{E}(\mathbb{T})$ with $t_1 \neq t_2$; $\mathbb{T}$ is said to be *non-Zeno* if $\mathcal{E}(\mathbb{T})$ has no limit points; $\mathbb{T}$ is said to be *left-Zeno* if for every t in the closure of $\mathcal{E}(\mathbb{T})$ there is a $t' \in T$ with $t' > t$ such that $(t, t') \cap \mathcal{E} = \emptyset$.

Regularity constraints on trajectories may for instance require that continuous state trajectories are continuously differentiable on $\mathbb{T}^c$ and continuous on $\mathbb{T}$; in case there are events of multiplicity ω (corresponding to a fibre of order type $\omega+1$), the latter requirement implies that the values of the state variable at successive events taking place at physical time t must converge to the value of the state variable at time t^+. Continuity is also a natural requirement to put on trajectories of continuous communication variables as well as on trajectories of discrete state variables. Because the discrete state variables map to sets that carry the discrete topology, requiring contintuity for the trajectories of these variables implies that the discrete variables must be constant between events.

An EFF does not directly provide a recipe for generating solutions; it is rather a testing device that determines whether a proposed solution is valid or not. In fact an EFF is just a list of all the laws satisfied by a given system. In the terminology of [4], EFFs are kernel representations rather than image representations. Descriptions of this form are user-friendly in the sense that they facilitate specification, but they do pose a challenge to the developers of simulation software.

4.4 Composition of EFFs

The solution concept that was proposed above is more complicated than the one in [6] because here we chose to work with different time axes for each variable rather than with a uniform time axis. The benefit is that we can now define composition in a simpler way than in [6].

Definition 4.2 The *parallel composition* of two EFFs $\bar{\phi}_1$ and $\bar{\phi}_2$ is defined by $\bar{\phi}_1 \parallel \bar{\phi}_2 := \bar{\phi}_1 \wedge \bar{\phi}_2$.

Clearly, in this way one may in fact unambiguously define the parallel composition of an arbitrary number of EFFs.

As a simple example of a hybrid system specification by means of compositions of EFFs, consider the standard example of a hybrid system, namely the thermostat. The behavior of room temperature θ may be given by an equation of the form

$$\dot{x} \;=\; f(x, w, H), \quad \theta \;=\; g(x) \tag{4.16}$$

where x is a suitable physical state variable through which the thermal properties of the room are expressed, H denotes the mode of the heater (`on` or `off`), and w represents external variables such as the outdoor temperature. Assuming that the thermostat is supposed to switch on when the room temperature goes down to 19 degrees and switches off when the room temperature has reached 20 degrees, the behavior of the thermostat may be described by a disjunction of the flow condition

$$\{\{\theta \leq 20\} \wedge \{H = \texttt{on}\}\} \vee \{\{\theta \geq 19\} \wedge \{H = \texttt{off}\}\} \tag{4.17}$$

and the event condition

$$\{\{\theta = 19\} \wedge \{H^+ = \mathtt{on}\}\} \vee \{\{\theta = 20\} \wedge \{H^+ = \mathtt{off}\}\}. \tag{4.18}$$

The equation (4.16) can be read as an EFF with an empty event condition; note that the comma is a notational device indicating conjunction. The behavior of the system as a whole may now be described by a composition of the subsystem "room" described by (4.16) and the subsystem "thermostat" given by (4.17) and (4.18). Note for instance that any trajectory in which the temperature drops below 19 degrees but the heating doesn't switch on cannot be a solution because the flow condition (4.17) would be violated. In this way the flow condition may *force* an event to take place. On the other hand the event condition (4.18) specifies the situation in which a certain event may take place; in this sense the event conditions may be seen as *enabling* conditions.

The above example is too small to show the full power of EFFs; for instance no use is made of discrete communication variables. Also it would be easy to write a description for the system as a whole directly, without making use of the operation of composition. For a complex system, however, the notion of composition is indispensible.

For a more extensive example of an EFF, consider the class of *linear complementarity systems* that has been described for instance in [22, 23]. Systems in this class are obtained by imposing complementarity characteristics as are well-known for instance in the Karush-Kuhn-Tucker conditions of mathematical programming to each input-output pair of a standard input-state-output system with equal numbers of inputs and outputs. A complementarity system with k inputs and outputs has 2^k modes which are most easily labeled by the subsets of the set $\{1, \ldots, k\}$. The mode corresponding to $I \subset \{1, \ldots, k\}$ is characterized by the conditions

$$y_i = 0 \quad (i \in I), \quad y_i \geq 0 \quad (i \notin I), \quad u_i = 0 \quad (i \notin I), \quad u_i \geq 0 \quad (i \in I). \tag{4.19}$$

To obtain a complete description, one needs a mechanism governing changes of modes and a mechanism governing resets of the continuous state. The latter can be based on the observation that the linear constraints in (4.19) constitute, together with the given linear dynamics, a differential-algebraic system for which "fast modes" and "slow modes" have been discussed extensively in the literature (see for instance [24]). In particular, if there is a unique solution for each initial condition then the impulsive part of the solution has the effect of projecting the given initial condition to the subspace of consistent states along a complementary subspace of jump directions; for details see for instance [25] or [6]. The projection corresponding to mode I will be denoted by P_I. Concerning the rule for changing modes one can use the so-called *rational complementarity problem* (RCP) that is extensively discussed in [26]. The RCP determines admissible modes for a given initial condition by requiring that the Laplace transforms of the inputs and outputs in a candidate mode should be nonnegative for sufficiently large real values of the transform parameter. A general description for the dynamics of a linear complementarity system can now be given in the form of an EFF by the flow conditions

$$\dot{x} = Ax + Bu, \; y = Cx + Du, \; y_I = 0, \; y_{I^c} \geq 0, \; u_{I^c} = 0, \; u_I \geq 0 \tag{4.20}$$

(where y_I is short for the vector with components y_i $(i \in I)$, I^c denotes the set $\{1, 2, \ldots, k\} \setminus I$, and the inequalities are understood componentwise) and the event condition

$$I^\sharp \in S(x), \quad x^\sharp = P_{I^\sharp} x \tag{4.21}$$

where $S(x)$ denotes the collection of modes that are selected by the rational complementarity problem. For conditions under which the above EFF has unique solutions, see for instance [25, 26, 6]. A system of the form (4.20–4.21) would be quite awkward to describe in terms of a "flat" hybrid automaton.

5 Modularity and jump rules

In this section we consider some of the problems that may arise when state-dependent switching is combined with modeling according to the principle of modularity. In particular we focus on the question if and how the *jump rules* for a physical hybrid system may be derived from the jump rules for its constitutive parts. We do not aim at presenting a general theory; instead we illustrate the questions on a very simple example (taken from [27]): an integrator coupled to a complementarity characteristic. The integrator may be described in standard state space notation by the equations

$$\dot{x} \;=\; u, \quad y \;=\; x \tag{5.22}$$

whereas the complementarity characteristic is given by the disjunction

$$\{\; y \geq 0, \quad u = 0 \;\} \quad \vee \quad \{\; y = 0, \quad u \geq 0 \;\}. \tag{5.23}$$

A physical example of such a coupling is provided by an electrical circuit in which a capacitor is connected to an ideal diode; the variable y is then interpreted as the voltage drop across the capacitor, which is equal to minus the voltage drop across the diode, and u is the current through the capacitor and the diode.

We consider the above system with an initial condition x_0 at time $t = 0$, and we shall be interested in solutions defined from that time point on. From physical considerations, the solution that one would hope to obtain from an appropriately defined solution concept is the following. If x_0 is nonnegative, no event occurs at time 0 and the solution is given by

$$y(t) \;=\; x_0, \quad u(t) \;=\; 0 \quad (0 \leq t < \infty). \tag{5.24}$$

If x_0 is negative, an event takes place at time 0 which causes the state x to jump to zero immediately; after that we have the zero solution

$$y(t) \;=\; 0, \quad u(t) \;=\; 0 \quad (0 < t < \infty). \tag{5.25}$$

In the capacitor-diode interpretation, the first situation corresponds to the blocking mode of the diode, while in the second situation the diode is conducting and the capacitor discharges instantly. The problem that we wish to consider is to find a *formal* specification that (i) produces the above solutions, and (ii) is modular in nature. The modularity requirement means that we are looking for a specification that can be written as the parallel composition of two subspecifications, which refer to the capacitor (5.22) and the diode (5.23) respectively.

The capacitor-diode example belongs to the class of linear complementarity systems as treated at the end of the previous subsection. In this example we only have two modes, corresponding to the conditions "$y = 0$" and "$u = 0$"; we shall refer to these modes simply as mode 0 and mode 1 respectively. The only consistent initial state for mode 0 is the zero state and so the projection corresponding to this mode maps any state to zero. On the other hand, in mode 1 all initial states are consistent and so the corresponding projection is the identity mapping. The following formal specification of our capacitor-diode system as the composition of the following EFFs results:

$$\dot{x} = u, \quad y = x, \quad \{P = 0,\ y = 0,\ u \geq 0\} \vee \{P = 1,\ u = 0,\ y \geq 0\} \tag{5.26}$$

$$x_0 \leq 0, \quad P^\sharp = 0, \quad x^\sharp = 0. \tag{5.27}$$

Solutions of the composed EFF (5.26)$\|$(5.27) can be sought for instance in the space NZ/1/C^1/C^0 of evolutions with non-Zeno simply punctuated time axes and trajectories that are continuous for all variables and continuously differentiable for the continuous state variables. In this space (as well as in spaces with richer time) there is a unique solution, which is the one that we already described above. So we obtain the "right" solution. However, although our description uses composition of two EFFs, the separate terms (5.26) and (5.27) do not correspond to the separate elements "capacitor" and "diode" in our system. Hence this specification is *not* modular.

Therefore, let us now discuss an alternative method which starts from modeling the two elements separately. The behavioral approach (see for instance [4]) would suggest to determine behaviors (that is, sets of compatible trajectories for the external variables) for each of the elements and then to take intersection in order to obtain the behavior of the system as a whole. To carry out this program, one first has to specify the function spaces from which the trajectories will be taken. The simplest choice is to look at smooth (say, C^∞) solutions. However the intersection of the smooth behaviors of the capacitor equation (5.22) and the diode equation (5.23 contains only the trajectories $y(t) = c$, $u(t) = 0$, where c is a nonnegative constant; and none of these trajectories are compatible with a negative value of the initial condition.

Such a negative result might be expected since we need solutions with jump components. So, instead of behaviors in C^∞, let us look at solutions in the space of distributions of the form $a\delta + f$ where a is a constant, δ is the Dirac delta distribution with support at time 0 and f is a smooth function defined on $(0, \infty)$. The nonnegativity constraint can be imposed on the smooth parts in the usual way; for the distributional part it seems natural to require that the constant a multiplying δ should be nonnegative. In this space, the behavior corresponding to the capacitor equation (5.22) is

$$\mathcal{B}_\mathrm{D} \;:=\; \{(y,u) \mid y \in C^\infty,\;\; \lim_{t\downarrow 0} y(t) = x_0 + a,\;\; u = a\delta + \dot{y}\} \tag{5.28}$$

and the behavior corresponding to the diode equation (5.23) is

$$\begin{aligned}\mathcal{B}_\mathrm{C} \;:=\; \{(y,u) \mid y = a_1\delta + f_1,\;\; & u = a_2\delta + f_2,\;\; a_1 \geq 0,\;\; a_2 \geq 0,\;\; a_1a_2 = 0,\\ & f_1 \in C^\infty(0,\infty),\;\; f_2 \in C^\infty(0,\infty),\\ & f_1(t) \geq 0,\;\; f_2(t) \geq 0,\;\; f_1(t)f_2(t) = 0 \text{ for all } t > 0\}.\end{aligned} \tag{5.29}$$

The intersection of the two behaviors is

$$\mathcal{B}_\mathrm{CD} \;=\; \{(y,u) \mid y \in C^\infty,\;\; \lim_{t\downarrow 0} y(t) = x_0 + a,\;\; u = a\delta + \dot{y},\;\; a \geq 0\}. \tag{5.30}$$

Unfortunately this intersection contains more solutions than we want; a condition is missing that would specify that the constant a can only be nonzero if x_0 is negative, and in that case it must be equal to $-x_0$.

We find ourselves therefore in a situation in which we have on the one hand a description that produces the "right" solutions but that does not have the desired modular structure, and on the other hand a description that does have modularity as desired but that doesn't lead to the solution set that we would like to see. How can we resolve this dilemma? Of course, the diode as we described it is an idealization. The mode switching that is implied by the diode equations could be described on another level of representation as a smooth process that takes a little time to go from a situation in which the current is approximately zero to one in which the voltage vanishes approximately. The ideal diode is a simplified description of these phenomena. It is received wisdom, however, that such neglect of details (coupled to appropriate robustness considerations) is actually a key factor in building useful models of physical systems. Indeed, an important way of constructing manageable models is to identify a time scale of interest so that variables that change on a slower time scale can be treated as constants (i. e. as parameters rather than as variables) and changes that take place on a faster time scale can be described as occurring instantaneously, that is, as jumps.

The idea of replacing fast motions by jumps, attractive as it may be from a computational point of view, is not without difficulties. First of all it may not be possible to take limits in an unequivocal way, and secondly it is not clear *a priori* that such a jump rule can be formulated on a modular level. To argue this last point, consider first the situation on a slow timescale where motion is described in terms of differential equations. The differential equations may well be described in a modular way; one might say that there is “infinitesimal modularity”. This modularity still holds approximately for time steps of the order that would typically be used in integration routines (or, conversely, the time step in an integration routine is taken sufficiently small so that modularity approximately holds.) However, on larger time intervals one cannot simulate each subsystem separately and combine the results to obtain a simulation of the system as a whole. Nevertheless, a modular jump rule attempts to do just that. In fact, the time interval connected to a jump may be small with respect to the chosen time scale of interest, but it is not small with respect to the fast motion that the jump rule is aiming to represent. This argument suggests that in general it will not be possible to represent a global jump rule by a set of local rules.

Let us now return to our example. It has been argued above that we should extend the description of the capacitor (5.22) with a jump rule for the short-circuit condition $y = 0$, leading to the EFF model

$$\{\dot{x} = u_{\mathrm{C}},\ y_{\mathrm{C}} = x\} \vee \{y_{\mathrm{C}}^{\sharp} = 0\}. \tag{5.31}$$

Furthermore, in the specific example at hand, the following EFF description of the diode element (5.23), including a *local* jump rule, can be postulated:

$$\{\{u_{\mathrm{D}} \geq 0,\ y_{\mathrm{D}} = 0\} \vee \{y_{\mathrm{D}} \geq 0,\ u_{\mathrm{D}} = 0\}\} \vee \{y_{\mathrm{D}}^{\sharp} \geq y_{\mathrm{D}}\}. \tag{5.32}$$

For the connection we may write a separate EFF:

$$\{y_{\mathrm{D}} = y_{\mathrm{C}}\} \wedge \{u_{\mathrm{D}} = u_{\mathrm{C}}\}. \tag{5.33}$$

The composition of the capacitor model, the diode model, and the connection then leads to an EFF which has the “right” solutions, and satisfies the rule of modular behavior (1.1), i. e., the behavior of the system as a whole is the intersection of the behaviors of the components.

While for this specific example it is thus possible to give a completely modular description of the system behavior (including jump rules), it remains to be seen whether this is possible in more general contexts. Specifically, it seems to be an open question whether general electrical networks with ideal diodes, switches, and linear capacitors, inductors, and resistors can be modeled on a modular basis with the electrical elements as modules. It should certainly be expected that the two variables in the ideal diode description (5.23) need to be treated symmetrically. The simple rule

$$y^{\sharp} \ \geq\ y, \quad u^{\sharp} \ \geq\ u, \tag{5.34}$$

extending the jump rule in (5.32), suggests itself and it would be nice to find conditions under which such a rule does indeed agree with the global rule (4.21). Similar questions can be posed in the context of mechanical systems subject to (multiple) collisions. In general, it is known that event handling in hybrid systems may involve “loops” in the application of local jump rules, see e.g. [28].

In summary, a research program on modularity in hybrid systems should include the identification of system classes for which either of the following holds.

(i) A jump rule can be only formulated at the global level.

(ii) The global jump rule can be derived from (a subset of) the local jump rules.

(ii) The global jump rule is equivalent to the combination of the local jump rules.

As a final remark, we note that the fast dynamics of a system may sometimes be decomposable even when the slow dynamics does not decompose. For instance, it is intuitively clear that in a mechanical systems shocks cannot propagate instantly over non-rigid interconnections. In such situations the reinitialization decision can be taken on the basis of data from only part of the system. It may be possible to set up a general theory of decompositions of this type on the basis of the *theory of structured systems* as presented for instance in the recent book by Murota [29].

6 Conclusions

The purpose of this paper has been to emphasize the importance of compositionality in hybrid system theory, and to identify some concrete questions for research in this area. The following specific problems have been formulated: (i) to develop a notion of "refusal sets" in continuous system theory, and to investigate its implications for hybrid systems theory, (ii) to determine classes of systems that allow a global rule equivalent to, or derivable from, local jump rules, and (iii) to investigate the possibility of computing reinitializations on an intermediate (between local and global) level by making use of structured system theory.

In this paper we have also presented a multitime semantics for event-flow formulas, which gives expression to the notion of partial synchronicity. As a result of the introduction of this semantics it becomes possible to define the operation of composition for hybrid system descriptions by means of conjunction, similar to the way in which traditionally continuous system descriptions are composed, without imposing global synchronization.

7 Acknowledgment

We would like to thank Jeroen Valk and the anonymous reviewers for their critical comments on an earlier version of this paper.

References

[1] Rudie K, Wonham WM. Think globally, act locally: decentralized supervisory control. IEEE Trans. Automat. Contr. 1992; 37: 1692–1708.

[2] Overkamp A. Discrete Event Control Motivated by Layered Network Architectures. PhD Dissertation, University of Groningen, 1996.

[3] Overkamp A. Supervisory control using failure semantics and partial specifications. IEEE Trans. Automat. Contr. 1997; 42: 498–510.

[4] Willems JC. Paradigms and puzzles in the theory of dynamical systems. IEEE Trans. Automat. Contr. 1991; 36: 259–294.

[5] Willems JC. On interconnections, control, and feedback. IEEE Trans. Automat. Contr. 1997; 42: 326–339.

[6] Van der Schaft AJ, Schumacher JM. An Introduction to Hybrid Dynamical Systems. Springer: London, 2000.

[7] Cassandras CG, Lafortune S, Olsder GJ. Introduction to the modelling, control and optimization of discrete event systems. Trends in Control (editor Isidori A). Springer: London, 1995; 217–291.

[8] Hopcroft JE, Ullman JD. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 1979.

[9] Milner R. A Calculus of Communicating Systems. Springer: New York, 1980.

[10] Hoare CAR. Communicating Sequential Processes. Prentice Hall, 1985.

[11] DiBenedetto MD, Saldanha A, Sangiovanni-Vincentelli A. Model matching for finite state machines. Proc. 33rd IEEE Conf. Dec. Contr. 1994: 4228–4234.

[12] Heymann M, Lin F. Discrete-event control of nondeterministic systems. IEEE Trans. Automat. Contr. 1998; 43: 3–17.

[13] Brookes SD, Hoare CAR, Roscoe AW. A theory of communicating sequential processes. Journal of the ACM 1984; 31: 560–599.

[14] Polderman JW, Willems JC. Introduction to Mathematical Systems Theory. A Behavioral Approach. Springer: New York, 1998.

[15] Antsaklis PJ, Stiver JA, Lemmon MD. Hybrid system modeling and autonomous control systems. Hybrid Systems (editors Grossman RL, Nerode A, Ravn AP, Rischel H). Lect. Notes Comp. Sci. 736, Springer: New York, 1993; 366–392.

[16] Branicky MS, Borkar VS, Mitter SK. A unified framework for hybrid control. Proc. 33rd IEEE Conf. Dec. Contr. 1994: 3117–3124.

[17] Alur R, Courcoubetis C, Halbwachs N, Henzinger TA, Ho PH, Nicollin X, Olivero A, Sifakis J, Yovine S. The algorithmic analysis of hybrid systems. Theoretical Computer Science 1995; 138: 3–34.

[18] Lynch N, Segala R, Vaandrager F, Weinberg HB. Hybrid I/O automata. Hybrid Systems III: Verification and Control (editors Alur R, Henzinger T, Sontag E). Lect. Notes Comp. Sci. 1066, Springer: New York, 1996; 496–510.

[19] Lygeros J, Godbole DN, Sastry S. Verified hybrid controllers for automated vehicles. IEEE Trans. Automat. Contr. 1998; 43: 522–539.

[20] Benveniste A. Compositional and uniform modeling of hybrid systems. IEEE Trans. Automat. Contr. 1998; 43: 579–583.

[21] Kowalewski S, Engell S, Preussig J, Stursberg O. Verification of logic controllers for continuous plants using timed condition/event-system models. Automatica 1999; 35: 505–518.

[22] Van der Schaft AJ, Schumacher JM. The complementary-slackness class of hybrid systems. Math. Contr. Signals Syst. 1996; 9: 266–301.

[23] Van der Schaft AJ, Schumacher JM. Complementarity modeling of hybrid systems. IEEE Trans. Automat. Contr. 1998; 43: 483–490.

[24] Hautus MLJ, Silverman LM. System structure and singular control. Lin. Alg. Appl. 1983; 50: 369–402.

[25] Heemels WPMH, Schumacher JM, Weiland S. Linear complementarity systems. Internal Report 97 I/01, Dept. of EE, Eindhoven Univ. of Technol., July 1997. To appear in SIAM J. Appl. Math.

[26] Heemels WPMH, Schumacher JM, Weiland S. The rational complementarity problem. Lin. Alg. Appl. 1999; 294: 93–135.

[27] Heemels WPMH, Çamlıbel MK, Schumacher JM. Dynamical analysis of linear passive networks with ideal diodes. Part I: Well-posedness. Internal Report 00 I/02, Measurement and Control Systems, Dept. of EE, Eindhoven Univ. of Technol., March 2000. Submitted for publication.

[28] Mosterman PJ. An overview of hybrid simulation phenomena and their support by simulation packages. Hybrid Systems: Computation and Control (editors Vaandrager FW, Van Schuppen JH). Lect. Notes Comp. Sci. 1569, Springer: New York, 1999; 165–177.

[29] K. Murota. Matrices and Matroids for Systems Analysis. Springer: New York, 2000.