HYBRIDGE

Distributed Control and Stochastic Analysis of Hybrid Systems
Supporting Safety Critical Real-Time Systems Design

WP4: Compositional Specification of Stochastic Hybrid Systems

CPDP: A compositional framework for stochastic hybrid systems of the PDP type

S.N. Strubbe and A.J. van der Schaft[1]

February 12, 2004

Version:	*0.4*
Task number:	*4.2*
Deliverable number:	*D4.2*
Contract:	*IST-2001-32460 of European Commission*

[1] University of Twente

DOCUMENT CONTROL SHEET

Title of document:	*CPDP: A compositional framework for stochastic hybrid systems of the PDP type*
Authors of document:	*S.N. Strubbe and A.J. van der Schaft*
Deliverable number:	*D4.2*
Contract:	*IST-2001-32460 of European Comission*
Project:	*Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design (HYBRIDGE)*

DOCUMENT CHANGE LOG

Version #	Issue Date	Sections affected	Relevant information
0.1	5-6-2003		
0.2	1-7-2003	Chpt. 2,4,5,6,7	Comments from NLR
0.3	29-1-2004	Chpt. 1,2,4,5,6	Comments from INRIA
0.4	12-2-2004	Chpt. 4	Comments from external reviewers

Version 1.0		Organisation	Signature/Date
Authors	Stefan Strubbe	Twente Univ.	
	Arjan van der Schaft	Twente Univ.	
Internal reviewers	Henk Blom	NLR	
	Stefan Haar	INRIA	

CPDP: A compositional framework for stochastic hybrid systems of the PDP type[1]

S.N. Strubbe and A.J. van der Schaft

June 2003

[1]Public deliverable D4.2 for the EU project HYBRIDGE

Contents

Chapter 1

Introduction

The main target of WP4 of the Hybridge project is to develop a formalism which can be used for compositional specification and analysis of stochastic hybrid systems. The class of stochastic hybrid systems we aim at in this project is the class of PDPs (Piecewise Deterministic Markov Processes) which was introduced by Davis [2]. In [3], this class has been recognized as very useful for example in the Air Traffic Management context. In [3] it was also shown that PDP systems can be represented in a Petri Net framework (DCPN). With DCPN it is possible to specify PDPs with the compositional power of Petri Nets. The compositional framework that we want to develop should be powerful enough to model at least the class of PDP systems.

In this report we want to start with a discussion about compositionality. It is not always clear what people mean when they talk about compositionality, compositional modelling and analysis, or compositional frameworks. Therefore we should make ourselves clear by what we mean with the term compositional. Other words that are connected to compositionality are hierarchy (hierarchical modelling) and modularity (modular modelling). As we see it, modularity and compositionality are equivalent. We prefer to use the word compositional instead of modular. Hierarchy however, is a different notion. We will come back to the notion of hierarchy later in this introduction.

Many real life systems that we want to study these days are large scale (hybrid) systems with an inherent high complexity. An example is Air Traffic Management (ATM) for Safety Risk Assessment. An ATM-system can consist of many components of many different types. Some components may represent aircrafts, while other components represent human behavior (pilots, ground controllers) or logistical systems (runway management). It is desirable that it is possible to model such systems block by block or component by component. If we then model one of the components, it should not be necessary to have an overview of specific details of the whole of the system. For example, it should not be necessary to need knowledge about component A when we want to model component B. This is one aspect of compositional modelling which we consider important.

We distinct two types of composition: Parallel composition and Sequential composition. If the components of a system are running in parallel or, in other words, simultaneously (like components representing different aircrafts in an ATM-system), then the components are composed in a parallel way. If the components of a system are active one after the other (like different control modes in a control-system), then the components are composed in a sequential way. Classical systems theory was mainly concerned with parallel composition in the sense that a controller is active simultaneously with the plant. In discrete event systems theory (including automata and Petri nets), we see that both forms of composition are present: One automaton can have different states or locations (sequential) and multiple automata can run concurrently while communicating with each other via shared labels (parallel). For several kinds of automata (finite state automata, timed automata, hybrid automata) *parallel operators* are designed in the literature. These operators take two automata as their arguments and have the composed automaton as the result. This composed automaton behaves as the parallel composition of multiple automata, but is in fact just one automaton. This single automaton (assumed that this automaton is not an intermediate step of a modelling process but is the complete and final system) can have many locations (sequentially composed), but does not have parallelism (or communication) in it anymore, although it expresses the behavior of multiple automata communicating with each other. The composition is not truly concurrent anymore. In Petri nets (or in a composition of Petri nets) there can be multiple tokens present in parallel. Petri nets are said to be truly concurrent therefore.

In the framework that we want to develop, both sequential and parallel composition should be possible. The idea of sequential composition is already present in the structure of PDPs (they have different locations with their own dynamics). We could try to copy this structure (as far as it concerns the sequential composition) to the new, to be made, framework. The idea of parallel composition is not present in the original PDP structure. For that, we should look at other sources for inspiration. The transition semantics can also be seen as a compositionality issue. We will see that the transition structure of the PDP will be decomposed into different parts in the to be developed compositional framework. Later in this report we introduce what we call the CPDP framework. It is our first attempt for developing a compositional framework for PDP systems. In this attempt the above notions of compositionality are incorporated.

In this report, we will give a small overview of the frameworks from the literature that involve compositionality issues. This means that we will look at automata frameworks which all have a clear notion of parallel composition. We will also look at Petri Net formalisms, which have the notion of true concurrency (or true parallelism). Statecharts will also be treated. In statecharts, it is the notion of hierarchy that is interesting.

With hierarchical modelling we mean that we can specify models at different levels of abstraction. Hierarchy is thus a totally different notion than compositionality, although they might be connected in certain situations. It could for example be that we have one component at a certain level of abstraction, which appears to be a composition of two different components at one level of abstraction lower. One of these two components can appear to be a composition again if we zoom in at two levels of abstraction lower, etc. We will discuss whether we consider this notion of hierarchy as important and valuable or not. We discuss the potential role of this in the new framework.

For the parallel composition in for example automata and Petri nets, we are particularly interested in how the different components can influence each other. In other words, how do they interact or how do they communicate? We can divide the different formalizations of communication into two types: The one-way and the two-way kinds of communication. In the one-way communication case one component can influence other components, but other components cannot influence this one component. In the two-way communication case, two interacting components can influence one another.

An important concept in the discrete event systems literature is *process algebra*. We will contemplate on this concept for a while and we discuss whether the advantages of using process algebra for system specification can possibly carry over to the context of hybrid (PDP) systems. Can we expect great benefits from introducing a process algebra language for specification of CPDP systems?

After this we will discuss the potential role of (classical) systems theory in the context of our new CPDP framework. Is it possible and interesting to use (classical) control concepts in the context of hybrid systems?

All these things above concerning the (hybrid) systems literature, will be treated in the second chapter. The evaluation at the end of that chapter should make clear in what direction we want to develop the new framework. Then in the third chapter we introduce the CPDP framework, which is a first attempt in the development of the compositional framework. This framework is very close to the original PDP framework. We show this in the fourth chapter, where we can find a (technical) proof for the equivalence of the classes of systems brought forth by PDPs and CPDPs. This means that we can model PDP systems and only PDP systems in the CPDP framework. We think that the compositional power of this CPDP framework is still a bit restricted, therefore we do an attempt in the fifth chapter to refine the framework with more compositional power. The consequence of this refinement is that the class of systems brought forth by the new framework is broader than the class of PDPs. We chose to introduce the 'old' CPDP framework as well in this report, because we can use it as a reference point: If under certain conditions we can show that a CPDP (from the new framework) can also be modelled in the old framework, then we know (since we have the equivalence proof) that this CPDP behaves as a PDP. In the sixth chapter, we give an example of how the refined CPDP framework can be used to compositionally specify a *forest landscape model* (which is supposed to be a real-life example). In the last chapter we tell our ideas about the future research in WP4 of the Hybridge project.

Chapter 2

Formalisms from the literature

In this chapter we explore the literature that is relevant for our purposes. We treat Automata, Petri nets, Statecharts, Process Algebra and Systems theory. We look at them all in the light of our quest: compositional specification of PDP systems. At the end of this chapter we will have a short evaluation where we point out which elements of this literature are potentially interesting for the development of our compositional framework. Before we start the literature overview, we give a short description (in automata jargon) of PDP systems. We skip the technical details here, for a complete definition of PDPs we refer to [2].

A PDP is characterized by the elements L, Inv, f, λ and Q. L is a countable set of locations. $Inv(l)$ is an open subset of $\mathbb{R}^{d(l)}$, where $d(l)$ is the dimension of the continuous statespace that is associated with location l. $f(l)$ (or f_l) is a vectorfield and is a locally Lipschitz continuous function. With every location l, we associate the ordinary differential equation (ODE) $\dot{x} = f_l(x)$.

Suppose the initial hybrid state is (l_0, x_0). Then the evolution of the hybrid process state is determined by the ODE associated with l_0, until a jump occurs. A jump can be caused in two ways: Firstly, when the continuous state x reaches the boundary of $Inv(l_0)$. Secondly, when the PDP Poisson process generates a jump. This Poisson process is always active with parameter $\lambda(l, x)$. This means that the first time-instant-of-jumping generated by this Poisson process has probability density function

$$\lambda(l(t), x(t))\mathrm{e}^{-\int_0^t \lambda(l(t),x(t))\mathrm{d}t}.$$

Whenever a jump occurs, the transition measure Q determines the target hybrid state by means of a probability measure: Q assigns to each reachable hybrid state (including boundary states) a probability measure on the Borel sets of the hybrid state space $\mathcal{H} = \{(l, x) | l \in L, x \in Inv(l)\}$. $Q(A, (l, x))$ then means the probability of jumping into the set $A \subset \mathcal{H}$ if the jump is taken from hybrid state (l, x). After a jump we take the target state as new initial value and repeat as above.

2.1 Automata formalisms

2.1.1 Formalism descriptions

In discrete event systems literature we find the very popular automata. An automaton typically has a set of locations and a set of labelled transitions. At any point in time, the process that is described by the automaton is in one of the locations. The process can jump to another location by means of a transition. Suppose this transition has label a, then when the transition is taken, we say that event a is executed. Continuous dynamics play no role here. In Figure 2.1, we see the graphical representation of automaton $\mathcal{A}$.

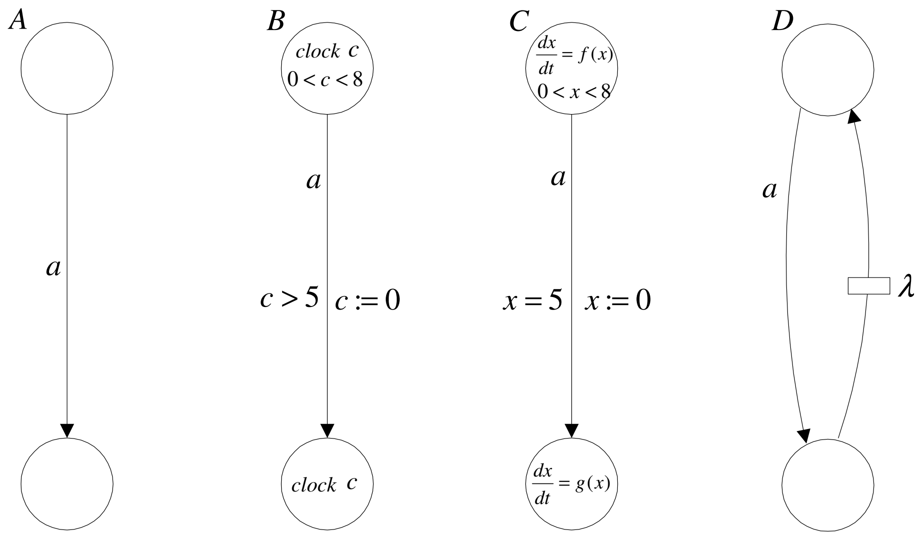

Figure 2.1: Examples: Automaton, Timed Automaton, Hybrid Automaton, IMC

Another formalism called *timed automata* can be seen as a first step in the direction of hybrid systems. Here we also have locations and transitions, but in each location there can be one or more clocks. Whenever the process jumps to a new location, the clocks in this location are reset and start running. In timed automata a location can have an invariant. An invariant is defined in terms of the clocks. If c is for example a clock in location l which is reset to zero every time the process jumps to location l, then $0 < c < 8$ can be an invariant. The meaning of this invariant is that the process can dwell in location l for at most 8 seconds. After 8 seconds the process has to jump to another location. The transitions in timed automata can be guarded. If for example transition t has guard $c > 5$, then this means that the transition can only be taken when the clock c in the location has values larger than 5. In Figure 2.1, we see the graphical representation of timed automaton $\mathcal{B}$.

The *Hybrid Automata* formalism can be seen as an extension of timed automata. In Hybrid automata there are also locations and (labelled and guarded) transitions. In each location there are continuous variables whose dynamics are defined in terms of differential equations and algebraic equations. Whenever a transition between two locations takes place, the continuous state may jump as well. The transitions are guarded and the guards are defined in terms of the continuous variables instead of the clocks in the case of timed automata. In Figure 2.1, we see the graphical representation of hybrid automaton $\mathcal{C}$.

As far as the automata formalisms above are concerned, stochastics are not involved. An interesting automata formalism for stochastic discrete event systems is IMC (Interactive Markov Chains) which was introduced by Hermanns [7]. In IMC there are locations and two kinds of transitions: Interactive transitions and Markovian transitions. Interactive transitions are labelled and non-stochastic. Markovian transitions are not labelled. A Markovian transition is governed by a Poisson process with jump rate λ. Suppose τ is a sample from the probability distribution of the Poisson process, then the Markovian transition is taken after τ time units, unless some other transition was taken earlier. In Figure 2.1, we see the graphical representation of IMC $\mathcal{D}$, which can jump from the top location to the bottom location via an interactive transition labelled a and can jump back via a Markovian transition with rate λ.

2.1.2 Composition in automata formalisms

In the automata literature, the notion of composition has had much attention for a long time. The term *communication* is used to denote that two systems can influence each other within a composition context. Communication typically takes place by means of synchronizing transitions. This communication typically needs three ingredients: the two systems to be composed and a set of synchronization labels. Semantically this means that when two systems A and B are composed, then a transition in system A with (synchronization) label a can only be executed if at the same time a transition with label a in system B is executed, and vice versa. We call this kind of communication *two way synchronization* because we have that the one who can execute a the earliest, has to wait for the other to be ready to execute a. In other words, the one can not go without the other and the other can not go without the one. These notions of composition and communication bring about *concurrency*, i.e. components may act in parallel and synchronize over some joint events, or exchange of messages, etc. See Milner's work [13] for concurrency in the context of automata and process algebra.

Composition as described above (two way) has been formalized for Automata, Timed Automata, Interactive Markov Chains and Hybrid Automata (as it is done for example in Section 3 of [10], or in [6]).

Composition as one way synchronization is treated in the I/O Automata [12] and HIOA (Hybrid I/O Automata) frameworks [10],[11]. In these frameworks, the external variables which are meant for communication are partitioned into input and output variables. The idea is that output variables are to be controlled by the (sub)system while the input variables are free and can be controlled by other systems. A system is supposed to be able to react on all inputs. In I/O automata, the variables are discrete events. In HIOA, the variables can be both discrete events and continuous variables.

2.1.3 Discussion

We will now discuss whether or not the automata frameworks described above seem to be suitable for compositional specification of PDP systems.

First of all, we have to conclude that the idea of *synchronizing events* has become very popular. This is mainly because the composition operator, which is the formalization of this idea, has very nice compositional properties like associativity and commutativity and the class of systems is closed under the composition operator. This gives good reasons for using the idea of synchronizing events also in a hybrid systems setting (as in [10]).

IMC seems to be a good source for inspiration and probably for more. In [7] the choice is made to split all transitions into two kinds, interactive and Markovian transitions. Then, Hermanns proves that in this way a composition operator can be defined with the three compositional properties above and furthermore, he develops a sound and complete process algebra for IMC. Thus, from a process algebraic and compositionality point of view, it seems sensible to follow this choice (of the two transitions) in a stochastic hybrid systems setting.

In **I/O Automata** and **HIOA** the idea of one way synchronization is used. In the approach that we will take, we also choose one way synchronization. The reason for that will become clear in the next section, when we treat the DCPN framework including an example of one-way synchronization.

2.2 Petri Net formalisms

From a pictorial point of view, Petri Nets have some things in common with automata. There are places (the equivalents of locations) and transitions. Furthermore, there are tokens. Tokens reside in places and determine in that way which transitions are active. There can be more than one token in a Petri Net and they can be in different places. This means that multiple places can be active at the same time. This is opposed to automata, where only one location can be active. In several new Petri Net formalisms, tokens can be colored or dynamically colored. In case of dynamical coloring, this means that a token color is a process which can evolve in time. One can look at a Petri Net as a given infrastructure which determines the flow and token color evolutions, where the colors are the running processes. If there are multiple colored tokens in different places, then these colors can be seen as concurrently evolving processes.

Many Petri Net formalisms appeared in the literature for both discrete event systems and hybrid systems. In this report we focus on the DCPN formalism, because this framework generates stochastic processes of the type PDP.

DCPN

DCPN stands for Dynamically Colored Petri Net. In [3], the DCPN formalism is introduced as a compositional framework for modelling complex PDP systems. We evaluate the compositionality of this framework by discussing the ATM example that is given in [3].

In the example there are three so called local Petri Nets which are interacting with each other. One Petri Net models the engine of an aircraft, another Petri Net models the navigation system of the aircraft and the third Petri Net models the evolution of the position and the velocity of the aircraft. In Figure 2.2 we see the three Petri Nets pictured.

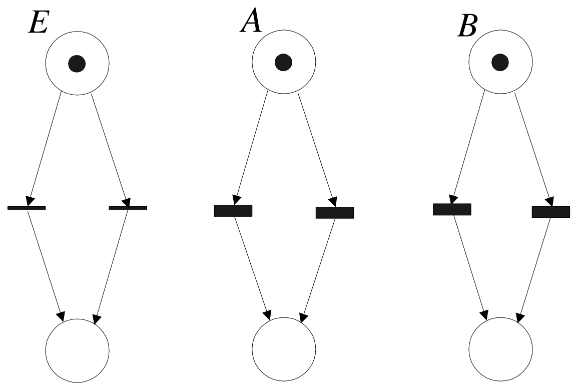

Figure 2.2: Petri Nets: Evolution ($\mathcal{E}$), Engine ($\mathcal{A}$), Navigation ($\mathcal{B}$)

The engine Petri Net and the navigation Petri Net are independent of their environments. The evolution Petri Net however, is depending on both the other Petri Nets: If one of the two systems is not working properly, then the evolution dynamics will change. The effect of the two Petri Nets on the evolution Petri Net is made explicit by drawing extra arcs from specific places in the two independent Petri Nets to specific transitions in the dependent Petri Net. (Arcs are always drawn from places to transitions or from transitions to places. For the exact description of the evolution of DCPNs we refer to [3]). In this way, the three local Petri Nets, which are connected by arcs, form a new Petri Net which is pictured in Figure 2.3. Notice that one of the transitions of the evolution Petri Net is split into two transitions. That the three local systems are running concurrently, is made explicit by assigning one token to each of the three local Petri Nets. The type of communication in this example is one-way synchronization: Two Petri Nets influence another Petri Net, but these two Petri Nets themselves are not influenced by their environments.

This example from [3] shows clearly that in DCPN, complex systems can be modelled by composing them through different parts and by specifying how these parts influence each other.

2.3 Statecharts

Statecharts were introduced by Harel in 1987 as a graphical formalism for modelling complex systems [5]. Also in this formalism we have states (locations) and transitions between states. In statecharts, states are hierarchic. This means that a state can consist of many substates. These substates are either connected via transitions (as in automata) or they run parallel. A state of the first (serial) type is called an OR state and a state of the second (parallel) type is called an AND state. Substates themselves are again OR or AND states and consist of several subsubstates etc. In this way the statespace is thus hierarchical. Graphically a specific state can be pictured abstractly (we don't see the substates) or a state can graphically be refined

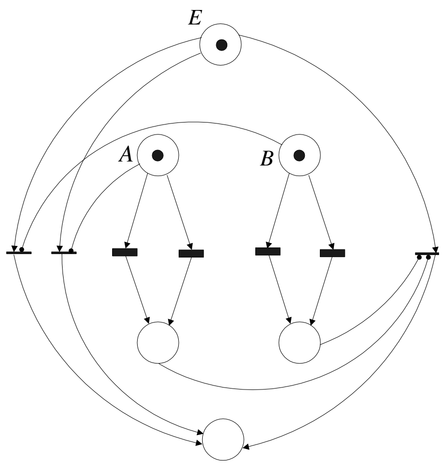

Figure 2.3: Composed Petri Nets

and then we see the substates and their interrelation. The substates of an AND state are called orthogonal states. Orthogonal states are concurrent processes which can communicate with each other via a mechanism called broadcast communication. In Figure 2.4 we see $State1$ in a refined version (right) and a more abstract version (left). The states $State1$ and A are OR states. State B is an AND state, this means that processes $B1$ and $B2$ run parallel.

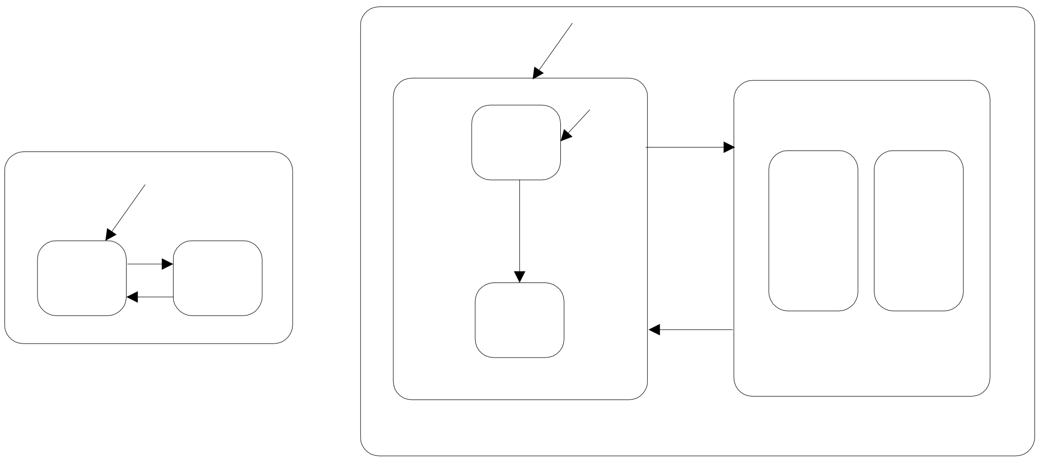

Figure 2.4: Refinement of states in Statecharts

Several variants for Statecharts have already been proposed. Also one which can represent hybrid systems [4]. The modelling language Modelica was used to implement a specific class of these hybrid Statecharts.

The idea of hierarchy is new in the sense that it is not present in the other classical formalisms. In the context of analysis and specification of complex systems hierarchy might become an important notion. This also explains the popularity that Statecharts already gained.

Besides hierarchy, Statecharts have a similar structure as automata and we think that therefore the concept of hierarchy can also be incorporated into other automata frameworks.

Especially in hybrid systems the notion of abstraction (which is closely related to hierarchy in statecharts) might be valuable. It might for example be necessary to abstract certain parts of a complex system for ease of computation in formal problems like the reachability problem. Hybrid systems can have a very complex structure because of the mix of discrete locations and continuous dynamics. This complexity can make it impossible to solve reachability problems (in [6] it is proven for example that many formal problems are undecidable for hybrid systems). If we abstract certain parts of the system, this normally means that the structure of the abstraction is much less complicated than the structure of the real system (we can for example abstract away all continuous dynamics from a hybrid system, then the abstraction will be a finite state automaton). Of course it depends on the kind of application, which parts can be abstracted and which parts can not.

2.4 Process Algebra

So far, process algebraic frameworks were mainly developed for discrete event systems (as for example Milner did in [13]). With process algebra it is possible to describe a system in one mathematical formula. This formula is called an element of the language of the process algebra. This language is constructed by a grammar over a set of variables and a set of events (or labels). The grammar says for example that if A is a process (or element of the language) and B is a process, then $A + B$ is also a process. Here $+$ is the choice operator. This means that $A + B$ describes the process where a choice has to be made between A and B. If the choice is A, then the process will evolve as A, if the choice is B, the process will evolve as B. If A is for example equal to $a.0$, which means that the event a can be executed after which the process terminates (0 symbol), and B equals $b.a.0$, which means that an event b can be executed and after that an a can be executed after which the process terminates, then $A+B$ equals $(a.0) + (b.a.0)$, or $a.0 + b.a.0$ for short, which means that the process can do a after which termination follows or b after which a can be done and then termination. A, B and $A + B$ are pictured in Figure 2.5.

Thus, one advantage of process algebra is that it makes it possible to represent systems with a compact mathematical formula. Another advantage is the use of process algebra in the theory of equivalence relations. Two systems are equivalent only with respect to some equivalence notion. Many equivalence notions were designed in the discrete event systems literature. A famous one is called *trace equivalence*. Two systems are trace equivalent if they bring forth the same set of traces. The processes $a.b.0+a.b.0$ and $a.b.0$ are trace equivalent for example because they both have $\{a, ab\}$ as their trace sets (note that a is here what is called an incomplete trace). Another very important equivalence relation is *bisimulation*, which we will not treat here because it is too complicated for short treatment.

Process algebra can be used for axiomization of an equivalence relation. This means that

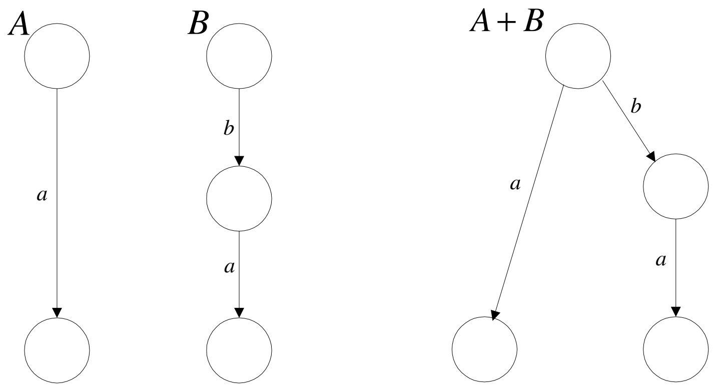

Figure 2.5: Automata representation of processes A, B and $A + B$

a set of axiomatic rules can be found which define the equivalence relation. The set with as only rule

$$A + B = B + A,$$

defines a very trivial equivalence relation, where for example $a.b + a.c$ and $a.c + a.b$ are equivalent. The use of process algebra in combination with equivalence relations has proven to be very valuable (see for example [7] for case studies). (Computer) tools have been developed, which can find equivalent systems for a particular to be analyzed system. If they find an equivalent system which is less complex in its structure, then this equivalent system can be used instead of the original system and the analysis will be easier. The kind of equivalence relation used depends of course on the kind of application.

Process algebra's have been developed for automata, stochastic and timed automata ([1]). As far as we know, no process algebra's were developed for hybrid automata ([15] seems to be one of the first attempts (in development) for hybrid systems, but here the hybrid system is modelled with a transition system structure rather than with an automata structure).

If we could develop a process algebra for CPDPs, such that the CPDPs can be represented by compact formula's, then that would already be a nice result on itself. But, since the structure of CPDPs is so much more complex than the structure of discrete event systems, it seems not very realistic to expect that popular process algebra applications as equivalence checkers, proof checkers, state space reduction etc. will be operational on the level of (stochastic) hybrid systems as well. Perhaps that under specific conditions and abstractions, operational tools can be developed. At least, at this moment it seems to be a bit out of scope.

2.5 Systems Theory

Systems in the classical mathematical systems theory are typically input/output systems. The system accepts an input signal and produces an output signal. Specification of systems is done by differential or difference equations (like the state space approach) or by transfer functions. Composition in systems theory is typically the interconnection of a plant and a controller in a feedback way. This feedback interconnection assumes an input/outout flow direction. People started realizing that the partitioning of signals into input and output signals is not always natural and therefore input/output-free notions were developed for interconnection. This was and is mainly done in the behavioral approach of mathematical systems ([16], [19]).

In Hybrid Automata, in DCPN and in Hybrid Statecharts, the continuous dynamics are normally represented by differential and algebraic equations (as in [18]). In HIOA, the continuous dynamics is given as a set of trajectories. This is similar to the behavioral approach where they have the philosophy that systems are characterized by their trajectories rather than by a representation like differential equations. In the HBA (Hybrid Behavioral Automata) framework, a first attempt was made for a behavioral approach to hybrid systems [9]. In HBA, the notion of interconnection as developed in the behavioral framework, is carried over to hybrid systems. This opens the possibility of continuous control, explored to some extent in [8].

The continuous dynamics of PDP systems is represented by differential equations [2]. It seems natural then, to use differential equations also in any (graphical) framework that is supposed to generate PDP systems. This is how it is done in the DCPN framework and in the CPDP framework which will be treated in the next chapter. On the other hand, it is not really a must. It might be better to switch to a behavioral or another approach for the continuous dynamics in case we want to do continuous interconnection like control (this is what is done in a refined version of the CPDP framework, introduced in Chapter 5).

2.6 Evaluation

Our goal is to develop a framework for compositional specification of complex PDP systems. With this goal and the things written above in mind, we make the following choices:

- We have chosen to develop an automata framework for the compositional specification of PDP systems. We think that PDP systems can be represented properly within an automata framework (PDPs are in [2] also defined in a structure which is very automata-like). To show this, we have taken the first steps in the development of the CPDP framework (see next chapter). We think that the nice compositional features of automata will apply also to CPDP.

- We think that it is interesting to carry over ideas from IMC to a hybrid setting. We will use the idea of splitting the transitions into interactive transitions and stochastic transitions. In IMC, the stochastic transitions are called Markovian jumps. This idea falls naturally into a PDP context since "Markovian jumps" are also present in PDP systems. In the PDP context, "Markovian jump" can give rise to some confusion, therefore we will call these jumps Poisson jumps (since the jumps are generated by a Poisson process). Also from a process algebraic point of view, this choice of splitting the transitions was proven to be a good choice [7].

- In first instance, the continuous dynamics will be represented by differential equations as in the PDP description itself. We want to stay flexible here. In chapter 5 we introduce a more refined version of CPDP, there we use, for compositionality purposes, a behavioral description of the continuous dynamics.

- We do not want to focus on *hierarchy* in the first phase of the project. This is a notion that might be interesting later on in the project. Then, inspiration could be found in the Statecharts literature. We think that many frameworks can be upgraded with the notion of hierarchy. This means that it is not necessary to take hierarchy into account from the first moments of the development of a new framework.

We already took some concrete steps in the direction of a new compositional framework using the choices above. This resulted so far in the CPDP framework, which will be the subject of the next chapter.

Chapter 3

The CPDP framework

In this chapter we introduce the 'old' CPDP framework. We call this old, because we will also introduce a refined CPDP framework later. First we define the CPDP in detail and we make clear how the semantics is in terms of the execution of a CPDP. After that we talk about (parallel) composition of CPDPs. It is of course very important that CPDPs can be composed in some way, since we are trying to develop a compositional framework. We show that CPDPs can indeed be composed. We formalize this by introducing a composition operator, symbolically written as $||$. In the last section we give an example of the composition of two CPDPs in the old CPDP framework.

3.1 Definitions and notations

In this section we introduce the CPDP formalism. First we will formally define its structure and after that we will explain how the execution of a CPDP takes place.

A CPDP $\mathcal{A}$ is a 9-tuple $(L, d, Inv,\ A, C, B, M, P,\ G)$, where

- L is a countable set of locations.
- $d : L \to \mathbb{N}$ is a mapping, which maps each location to the dimension of the continuous state space in that location.
- Inv maps each location to its invariant set. This means that for each $l \in L$, $Inv(l)$ is an open subset of $\mathbb{R}^{d(l)}$. We also define the boundary set $\partial Inv(l) := \overline{Inv(l)} \backslash Inv(l)$ of $Inv(l)$.
- A is the set of labels.
- C is the transition-choice function. $C(b, l, x) \in [0, 1]$ is the probability of executing the boundary-hit transition b (see next item) from the boundary state (l, x). $C(b, l, x)$ is defined on all $l \in L$ and all $x \in \partial Inv(l)$ and all $b \in B$ that are outgoing transitions of l. Furthermore $\sum_{b \in B_{l\to}} C(b, l, x) = 1$, where $B_{l\to}$ is the set of all elements of B that are outgoing transitions of l.
- B is the set of boundary-hit transitions. Each element b of B is a quadruple (l, a, l', R), where l is the origin location, a is the label of the jump, l' is the target location, and R is the reset map of the jump. $R(A, x) \in [0, 1]$ is the probability of jumping into the

set A when the transition b is taken from boundary state x. $R(A, x)$ is defined for all $x \in \partial Inv(l)$ with $C(b, l, x) > 0$ and for all Borel subsets A of $Inv(l')$.

- M is the set of Poisson transitions (the equivalent of Markovian transitions in Interactive Markov Chains in [7]). Each element m of M is a pentuple (l, a, l', R, λ), where l is the origin location, a is the label of the jump, l' is the target location, R is the reset map of the jump, and λ is the jump rate. $R(A, x) \in [0, 1]$ is the probability of jumping into the set A when the transition m is taken from state x. $R(A, x)$ is defined for all $x \in Inv(l)$ and for all Borel subsets A of $Inv(l')$. $\lambda : Inv(l) \to \mathbb{R}_+$ is a bounded Borel measurable function, and determines the rate of jumping of the process.
- P is the set of passive transitions. Each element p of P is a quadruple (l, a, l', R), where l is the origin location, a is the label of the jump, l' is the target location, and R is the reset map of the jump. $R(A, x) \in [0, 1]$ is the probability of jumping into the set A when the transition b is taken from state x. $R(A, x)$ is defined for all $x \in Inv(l)$ and for all Borel subsets A of $Inv(l')$.
- G determines the flow of the continuous states within the locations. For each $l \in L$, $G(l)$ is a locally Lipschitz continuous function from $\mathbb{R}^{d(l)}$ to $\mathbb{R}^{d(l)}$, and is the vectorfield for the continuous flow in location l.

A CPDP $\mathcal{A}$ as defined above, generates a stochastic process. To give insight to the execution of a CPDP process, we will now describe how sample paths of this stochastic process can be generated. As we will see later, the passive transitions play a role in communication with the outside world. For the generation of a sample path, we assume that the outside world is silent, in other words, no communication takes place and therefore, the passive transitions play no role in this sample path generation.

Execution of a CPDP

We assume that an initial hybrid state $\xi_0 = (l_0, x_0)$ is given. The flow of the continuous state x in location l_0 is determined by the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}x = G_{l_0}(x),$$

$G(l)$ is written here as G_l. Suppose that $x(t)$ reaches the boundary $\partial Inv(l_0)$ at time τ_b and suppose that location l_0 has n outgoing Poisson transitions. During the continuous flow, for every outgoing Poisson transition, a stochastic process of the Poisson type is active, which can cause a jump to another hybrid state. The probability density functions of these processes equal

$$\lambda_i(x(t))\mathrm{e}^{-\int_0^t \lambda_i(x(t))\mathrm{d}t},$$

where λ_i is the jump rate of the i-th outgoing Poisson transition. For each outgoing Poisson transition m_i, we draw a sample τ_i from its Poisson probability distribution. This means that m_i would cause a jump at time τ_i if no other jump occurred before τ_i, therefore the most relevant Poisson transition is the one corresponding to

$$\tau_M := \min_{i=1..n} \tau_i.$$

Now there are two possibilities. Firstly, if $\tau_b < \tau_M$, the boundary is reached before any Poisson transition is about to be executed, which means that a boundary-hit transition is executed at time τ_b. Secondly, if $\tau_M < \tau_b$, the Poisson transition corresponding to τ_M causes a jump at time τ_M before the boundary is reached.

Boundary-hit transition

A boundary-hit transition at time τ_b from the boundary state $x(\tau_b) \in \partial Inv(l_0)$ is executed as follows. It could be that multiple boundary-hit transitions are active in state $x(\tau_b)$, therefore we use the choice function C to choose one of the active transitions. We draw a sample $b \in B_{l_0\rightarrow}$ from the probability measure determined by $C(\cdot, l_0, x(\tau_b))$. Now, $b = (l_0, a_b, l'_b, R_b)$ is the transition that will be executed. The target location is l'_b and the target state x' within l'_b is drawn from the probability measure $R_b(\cdot, x(\tau_b))$. With the new hybrid state (l'_b, x') at time τ_b, we can repeat the algorithm above to continue the sample path.

Poisson transition

A Poisson transition $m = (l_0, a_m, l'_m, R_m, \lambda_m)$ at time τ_M from the state $x(\tau_M) \in Inv(l_0)$ is executed as follows. The target location is l'_m and the target state x' within l'_m is drawn from the probability measure $R_m(\cdot, x(\tau_M))$. With the new hybrid state (l'_m, x') at time τ_M, we can repeat the algorithm above to continue the sample path.

With the sample path description above, we informally described the stochastic processes that correspond to the CPDPs. From the definition of PDPs in [2] it can be seen that the stochastic processes of CPDPs are much like the PDP stochastic processes. In fact it was shown that the class of CPDPs and the class of PDPs are equivalent (with respect to a strong equivalence notion called *transition equivalence*). This result is the central part of the next chapter.

In the next section we use the following shorthand notation. We write $l \xrightarrow{a,R} l'$, $l \overset{a,R}{\dashrightarrow} l'$, $l \overset{a,R,\lambda}{\longmapsto} l'$ to denote the existence of respectively boundary-hit, passive and Poisson jumps from l to l' with label a, reset map R and, in the case of a Poisson jump, jump rate λ. Furthermore, $l \rightarrow l'$ denotes the existence of a boundary-hit jump from l to l' and $l \rightarrow$ denotes the existence of a boundary-hit jump outgoing from l. For Poisson and passive jumps we use equivalent notations. Finally, we also have negated versions like $l \not\rightarrow$, which means that l has no outgoing boundary-hit transitions.

3.2 Composition of CPDPs

In this section we introduce a composition operator $||$ on CPDPs. In CPDPs, communication or interaction takes place via the passive transitions. In a context with two composed CPDPs, one of the CPDPs can execute a passive transition with label a if and only if at the same time the other CPDP executes an active event with label a. We call all boundary-hit and all Poisson transitions active transitions. The execution of an active transition always happens independently from the other systems in the composition context. This notion of composition is formally defined below.

Suppose the CPDPs $\mathcal{A}_i = (L_i, d_i\ , Inv_i, A, C_i, B_i, M_i, P_i, G_i)$ are given for $i = 1, 2$. Then the composition $\mathcal{A} = \mathcal{A}_1||\mathcal{A}_2$ is characterized by the 9-tuple $(L, d, Inv, A, C, B, M, P, G)$, where

$$L = L_1 \times L_2,$$

$$d(l_1, l_2) = d_1(l_1) + d_2(l_2),$$

$$Inv(l_1, l_2) = Inv_1(l_1) \times Inv_2(l_2)$$

and

$$G(l_1, l_2) = \begin{pmatrix} G_1(l_1) \\ G_2(l_2) \end{pmatrix},$$

which means that the vectorfield $G(l_1, l_2)$ of location (l_1, l_2) is the stacked vector containing the vectorfields of the original locations l_1 and l_2. In the rules **r1** till **r6** below, l_1 and l_1' are in L_1 and l_2 and l_2' are in L_2.

$b \in B$, $m \in M$ and $p \in P$ if b, m and p can be derived from the following rules, where rule $\frac{A,B}{C}$ means that "A holds and B holds" implies that C holds.

$$\textbf{r1.}\ \frac{l_1 \xrightarrow{a,R_1} l_1', l_2 \overset{a}{\not\dashrightarrow}}{(l_1,l_2) \xrightarrow{a,R} (l_1', l_2)}, \textbf{r2.}\ \frac{l_1 \xrightarrow{a,R_1} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2'}{(l_1,l_2) \xrightarrow{a,R} (l_1', l_2')},$$

$$\textbf{r3.}\ \frac{l_1 \overset{a,R_1,\lambda_1}{\longmapsto} l_1', l_2 \overset{a}{\not\dashrightarrow}}{(l_1,l_2) \overset{a,R,\lambda}{\longmapsto} (l_1', l_2)}, \textbf{r4.}\ \frac{l_1 \overset{a,R_1,\lambda_1}{\longmapsto} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2'}{(l_1,l_2) \overset{a,R,\lambda}{\longmapsto} (l_1', l_2')},$$

$$\textbf{r5.}\ \frac{l_1 \overset{a,R_1}{\dashrightarrow} l_1', l_2 \overset{a}{\not\dashrightarrow}}{(l_1,l_2) \overset{a,R}{\dashrightarrow} (l_1', l_2)}, \textbf{r6.}\ \frac{l_1 \overset{a,R_1}{\dashrightarrow} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2'}{(l_1,l_2) \overset{a,R}{\dashrightarrow} (l_1', l_2')}.$$

We will now specify R in **r1** till **r6**. Here, we make use of the analysis result that for two measures M_1 on Σ_1 and M_2 on Σ_2, there is a unique measure M on $\Sigma_1 \times \Sigma_2$ (called the product measure), such that $M(A_1 \times A_2) = M_1(A_1)M_2(A_2)$ for all Borel subsets A_1 and A_2. We also make use of the indicator function $1_A(x)$ which equals one if $x \in A$ and zero if $x \notin A$. Then in **r1** till **r6**, R is the product measure where $R(A_1 \times A_2, (x_1, x_2)) = R_1(A_1, x_1)1_{A_2}(x_2)$ (case **r1**, **r3**, **r5**) and $R(A_1 \times A_2, (x_1, x_2)) = R_1(A_1, x_1)R_2(A_2, x_2)$ (case **r2**, **r4**, **r6**) for all $x_1 \in \partial Inv(l_1)$ (case **r1**,**r2**) or all $x_1 \in Inv(l_1)$ (case **r3**, **r4**, **r5**, **r6**), all $x_2 \in Inv(l_2)$ and all Borel subsets A_1 of $Inv(l_1')$ and A_2 of $Inv(l_2)$. Furthermore, in both **r3** and **r4**, $\lambda(x_1, x_2) = \lambda_1(x_1)$ for all $x_1 \in Inv(l_1)$ and all $x_2 \in Inv(l_2)$.

Besides the rules **r1** till **r6**, there are the rules **r1'** till **r6'** which are the mirrored versions of **r1** till **r6**. This means that

$$\textbf{r1'.}\ \frac{l_2 \xrightarrow{a,R_2} l_2', l_1 \overset{a}{\not\dashrightarrow}}{(l_1,l_2) \xrightarrow{a,R} (l_1, l_2')}, \textbf{r2'.}\ \frac{l_2 \xrightarrow{a,R_2} l_2', l_1 \overset{a,R_1}{\dashrightarrow} l_1'}{(l_1,l_2) \xrightarrow{a,R} (l_1', l_2')},$$

etc. Transitions are elements of B, M or P, if and only if they follow from the rules **r1** till **r6** and **r1'** till **r6'**.

At last, we have to specify C. Take any $l = (l_1, l_2) \in L$ and any $b : (l_1, l_2) \to (l'_1, l'_2) \in B$. b has been derived from one of the following four cases:

$$\textbf{c1:}\ \underbrace{l_1 \to l'_1}_{t_1}, l_2 \not\dashrightarrow, l_2 = l'_2\ (\mathbf{r1}),$$

$$\textbf{c2:}\ \underbrace{l_1 \to l'_1}_{t_2}, l_2 \dashrightarrow l'_2\ (\mathbf{r2}),$$

$$\textbf{c3:}\ \underbrace{l_2 \to l'_2}_{t_3}, l_1 \not\dashrightarrow, l_1 = l'_1\ (\mathbf{r1'}),$$

$$\textbf{c4:}\ \underbrace{l_2 \to l'_2}_{t_4}, l_1 \dashrightarrow l'_1\ (\mathbf{r2'}).$$

Now for all $(x_1, x_2) \in \partial Inv(l_1, l_2)$, C is defined as

$$C(b, l, (x_1, x_2)) =$$

$$\begin{cases} C_1(t_1, l_1, x_1) & ; \text{ case } \mathbf{c1}, x_1 \in \partial Inv_1(l_1), \\ & \quad x_2 \in Inv_2(l_2), \\ C_1(t_2, l_1, x_1) & ; \text{ case } \mathbf{c2}, x_1 \in \partial Inv_1(l_1), \\ & \quad x_2 \in Inv_2(l_2), \\ C_2(t_3, l_2, x_2) & ; \text{ case } \mathbf{c3}, x_1 \in Inv_1(l_1), \\ & \quad x_2 \in \partial Inv_2(l_2), \\ C_2(t_4, l_2, x_2) & ; \text{ case } \mathbf{c4}, x_1 \in Inv_1(l_1), \\ & \quad x_2 \in \partial Inv_2(l_2), \\ \text{undefined} & ; x_1 \in \partial Inv_1(l_1), \\ & \quad x_2 \in \partial Inv_2(l_2), \\ 0 & ; \text{ else.} \end{cases}$$

We will say some words about the meaning of the rules **r1,r1'** till **r6,r6'**. If one CPDP can execute an active event and the other CPDP does not have a matching passive partner, then in the composition context, the active CPDP executes the transition while the other CPDP stays in the same location (rules **r1,r1',r3,r3'**). If the first CPDP can execute an active transition and the second CPDP has a matching passive partner, then in the composition context, both CPDPs execute respectively the active and passive transition at the same time (rules **r2,r2',r4,r4'**). If the first CPDP has a passive transition with label a and the second CPDP has no passive transition with label a, then the composed system has a passive transition with label a outgoing from the joint location, which gives the possibility to interact with other systems in another composition context (rules **r5,r5'**). If both CPDPs have a passive transition with the same label, then the composed system also has a passive transition with this label. This means that both CPDPs can execute the passive transitions at the same time in another composition context where a third party executes an active transition with the same label (rules **r6,r6'**).

We see in the above definition that C is undefined for all (x_1, x_2) where both x_1 and x_2 are boundary points. We call such a state a *double boundary state*. In the execution of the system, these points play a role only when the two components reach their boundaries at exactly the same time τ_b. It is not evident what to do in such a situation. Two boundary-hit transitions should be executed at the same time. These transitions may have different labels and then a simultaneous execution of these transitions gives problems from a compositionality point of view: If a third party has both labels available in passive transitions, which passive transition should be chosen? However, for many composed systems these problems will not be present because the probability that two separate CPDPs reach their boundary at exactly the same time will be zero. For now, we leave the choice of what to do in double boundary states open and we say that the composed CPDP is undefined on the double boundary states.

The composition of two CPDPs is again a CPDP (under certain conditions). For the proof of this statement we refer to [17]. We now give an example of the composition of two CPDPs.

3.3 Example of a composed CPDP

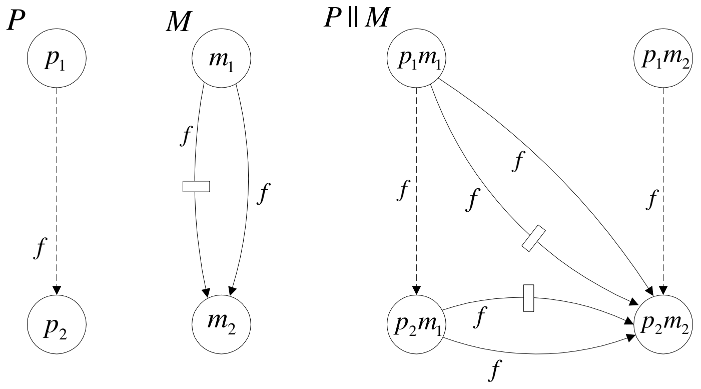

Figure 3.1: Two CPDP automata and their composition

In Fig 3.1 we see the graphical representation of three CPDPs, $\mathcal{P}$, $\mathcal{M}$ and the composition $\mathcal{P}||\mathcal{M}$. Boundary-hit, passive and Poisson transitions are respectively pictured as solid arrows, dashed arrows and solid arrows with a little box in the middle. This graphical representation has its origins in Interactive Markov Chains [7]. We give the following interpretation, which is a simplification of the ATM example from [3]. $\mathcal{P}$ is a process which depends on a machine $\mathcal{M}$. $\mathcal{P}$ has two working-modes, a nominal desired mode (p_1) and a failure mode (p_2). The machine $\mathcal{M}$ also has a nominal (m_1) and a failure (m_2) mode. $\mathcal{M}$ is an autonomous machine which can break down in two ways, via a boundary-hit of the continuous process or via a Poisson process. The interrelation between $\mathcal{P}$ and $\mathcal{M}$ is that $\mathcal{P}$ breaks down whenever $\mathcal{M}$ breaks down. This is a one-way synchronization which is reflected by the passive transition.

Now we give the formal description of $\mathcal{P}$, $\mathcal{M}$ and $\mathcal{P}||\mathcal{M}$ where we omit the specification of the continuous processes for simplicity reasons.

$\mathcal{P} = (L_P,\, d_P,\, Inv_P,\, A,\, C_P,\, B_P,\, M_P,\, P_P,\, G_P)$, where $L_P = \{p_1, p_2\}$, $A = \{f\}$, $B_P = M_P = \emptyset$, $P_P = \{(p_1, f, p_2, R_P)\}$ with R_P some reset map.

$\mathcal{M} = (L_M, d_M, Inv_M, A, C_M, B_M, M_M, P_M, G_M)$ where $L_M = \{m_1, m_2\}$, $B_M = \{(m_1, f, m_2, R_{M,1})\}$, $M_P = \{(m_1, f, m_2, R_{M,2}, \lambda_M)\}$, $P_P = \emptyset$. $R_{M,1}$ and $R_{M,2}$ are proper reset maps and λ_M a proper jump-rate-function.

According to the definition of $||$ we find $\mathcal{P}||\mathcal{M} = (L, d, Inv, A, C, B, M, P, G)$ where $L = \{(p_1, m_1), (p_1, m_2), (p_2, m_1), (p_2, m_2)\}$, $B = \{b_{t,1}, b_{t,2}\}$, $M = \{m_{t,1}, m_{t,2}\}$ and $P = \{p_{t,1}, p_{t,2}\}$. From the composition rules **r1,r1'** till **r6,r6'** we find

$$b_{t,1} = ((p_1, m_1), f, (p_2, m_2), R_{b,1}) \ (\textbf{r2'}),$$

$$b_{t,2} = ((p_2, m_1), f, (p_2, m_2), R_{b,2}) \ (\textbf{r1'}),$$

$$m_{t,1} = ((p_1, m_1), f, (p_2, m_2), R_{m,1}, \lambda_1) \ (\textbf{r4'}),$$

$$m_{t,2} = ((p_2, m_1), f, (p_2, m_2), R_{m,2}, \lambda_2) \ (\textbf{r3'}),$$

$$p_{t,1} = ((p_1, m_1), f, (p_2, m_1), R_{p,1}) \ (\textbf{r5}),$$

$$p_{t,2} = ((p_1, m_2), f, (p_2, m_2), R_{p,2}) \ (\textbf{r5}),$$

where all reset-maps and jump-rate-functions follow from the rules.

We make some remarks according to this example. Since $\mathcal{P}$ has no boundary-hit transitions, $\mathcal{P}$ has no boundary points at all because a CPDP has at least one boundary-hit transition active for each boundary point. This means that $\mathcal{P}$ and $\mathcal{M}$ can not reach their boundaries at the same time and since this is the only case of underspecification, $\mathcal{P}||\mathcal{C}$ is not underspecified and is therefore a (fully specified) CPDP.

The second remark is on initial states. We did not use the concept of initial states in the definition of CPDPs and therefore any hybrid state is potentially an initial state. In the example above, it might be desirable to indicate location $p_1 \in L_P$ and $m_1 \in L_M$ as initial locations. In that case location (p_1, m_1) would be the initial location of $\mathcal{P}||\mathcal{M}$ and location (p_1, m_2) and its outgoing passive transition would be useless. In a CPDP definition with initial state(s), the definition of $||$ could then be adjusted in such a way that these "useless" or unreachable locations and transitions are discarded.

The last remark is on other composition contexts. The composed system $\mathcal{P}||\mathcal{M}$ still has passive transitions which can be used for interaction in other composition contexts. There can for example be a second autonomous machine $\mathcal{M}_2$ which executes the label f and can therefore bring the process $\mathcal{P}$ into failure mode. If we compose $\mathcal{P}||\mathcal{M}$ in this new context with $\mathcal{M}_2$, we would get the process $(\mathcal{P}||\mathcal{M})||\mathcal{M}_2$. It is intuitively clear that a good composition operator should be commutative and associative, i.e. $\mathcal{P}||\mathcal{M}$ should be equivalent to $\mathcal{M}||\mathcal{P}$ and $(\mathcal{P}||\mathcal{M})||\mathcal{M}_2$ should be equivalent to $\mathcal{P}||(\mathcal{M}||\mathcal{M}_2)$. We are convinced that $||$ satisfies these properties, but it still has to be proven.

Chapter 4

Equivalence of PDPs and (closed) CPDPs

In this chapter we will prove the equivalence between the class of PDPs and the class of closed CPDPs, i.e. the class of CPDPs not containing passive transitions. A major part of this chapter concerns the technical proof of the equivalence theorem. This chapter can be skipped without losing grip on the main line of this report. We chose to include this result in this report because it justifies in some way the use of the automata framework CPDP for the modelling of PDP systems.

Notice that PDPs and closed CPDPs have very similar structures. For both PDPs and closed CPDPs we have: The process concerns a hybrid state (l, x) which is piecewise deterministically evolving. The piecewise deterministic part is determined by an ODE. A Poisson process (PDP) or multiple Poisson processes (CPDP) can cause a hybrid jump. Also a boundary hit can cause a hybrid jump. The target state of the jump is determined by a probability measure (transition measure Q for PDPs and reset maps R for CPDPs). We prove equivalence between PDPs and CPDPs in the strong sense of *transition equivalence* as defined below. Transition equivalence is a structural notion of equivalence. In future research, we would like to extend this notion to structural bisimulation of CPDPs. Then we will also pay more attention to the resulting notion of stochastic equivalence.

Definition: A PDP $\mathcal{A}_P$ and a CPDP $\mathcal{A}_C$ are *transition equivalent* if they have the same location set, the same invariant spaces, the same vectorfields and if $\mathcal{A}_P$ and $\mathcal{A}_C$ start in the same initial hybrid state, then

$$\mathrm{P}_{\mathcal{A}_P}(A) = \mathrm{P}_{\mathcal{A}_C}(A),$$

for every A from $\mathcal{B}(HT)$, where

$$HT := \{(t, x, l) | t \in \mathbb{R}_+, l \in L, x \in Inv(l)\},$$

and $\mathcal{B}(HT)$ is the set of Borel subsets of HT. If we denote by t the time instant of the first transition and we denote by (l, x) the target hybrid state of that transition, then $\mathrm{P}_{\mathcal{A}_P}(A)$ means the probability that $(t, l, x) \in A$ for PDP $\mathcal{A}_P$ and $\mathrm{P}_{\mathcal{A}_C}(A)$ means the probability that $(t, l, x) \in A$ for CPDP $\mathcal{A}_C$. $\square$

If a PDP and a CPDP are equivalent (from now on we mean transition equivalent with this), then this means that when they both are in the same hybrid state at some time instant t_i, then the probability distributions of the time instant t_f of the first transition after t_i are the same and the transition measures which determine the target hybrid state after the jump, are the same.

It is evident that if the probability distribution on the hybrid statespace is equivalent after the first transition, then it is also equivalent after the second transition, the third transition etc. because we can take the target state after the first transition as new initial state and apply the definition again. This gives the following corollary.

Corollary: Suppose PDP $\mathcal{A}_P$ generates the stochastic process $\{Y_t\}$ and CPDP $\mathcal{A}_C$ generates the stochastic process $\{Z_t\}$. Y_t and Z_t take value in the hybrid state spaces of $\mathcal{A}_P$ and $\mathcal{A}_C$ respectively. If $\mathcal{A}_P$ and $\mathcal{A}_C$ are transition equivalent, then

$$\mathrm{P}(Y_t \in A | Y_s = (l,x)) = \mathrm{P}(Z_t \in A | Z_s = (l,x)),$$

for all $s > 0$, $t > s$, hybrid states (l,x) and for all Borel sets A of the hybrid statespace. $\square$

We prove transition equivalence of the class of PDPs and the class of CPDPs in two parts. In the next section we prove that for every closed CPDP we can find an equivalent PDP. In the section thereafter we prove that for every PDP we can find an equivalent closed CPDP.

4.1 Closed CPDP into PDP

Suppose we have a closed CPDP with boundary hit transition set B, Poisson transition set M and choice function C. We now define a PDP with the same location set, invariants and vectorfields and furthermore

$$\lambda(l,x) = \sum_{m \in M_{l\to}} \lambda_m(x),$$

which means that λ is the sum of the rates of all Poisson transitions that leave location l. We define Q only on Borel sets that are a subset of an invariant of some location. From the axioms of probability measures, Q can easily be extended to all Borel sets of the hybrid state space $\mathcal{H}$. Suppose A is a Borel subset of $Inv(l')$ for some location l', then we define for all $l \in L$ and all $x \in Inv(l)$,

$$Q(A,(l,x)) := \frac{\sum_{m \in M_{l\to l'}} \lambda_m(x) R_m(A,x)}{\sum_{n \in M_{l\to}} \lambda_n(x)}.$$

Now we define Q for the boundary-states, that is for all l and all $x \in \partial Inv(l)$. Suppose A is a Borel subset of $Inv(l')$ for some location l', then

$$Q(A,(l,x)) := \sum_{b \in B_{l\to l'}} C(b,l,x) R_b(A,x).$$

We have now specified the PDP completely and we are ready to verify that the CPDP and the newly defined PDP are equivalent.

Take any $A \in \mathcal{B}(HT)$. Suppose both the PDP and the CPDP start at $t = 0$ in state (l_0, x_0). We assume that for every $(t, l, x) \in A$, l equals some constant l'. Thus, we restrict to one target location. With the axioms for measure and integral theory, we can easily extend the proof to the multiple target location case. We define

$$T := \{t | (\exists l, x)((t, l, x) \in A)\},$$

and

$$S_t := \{(l, x) | (t, l, x) \in A\}.$$

Then the probability of A for the PDP equals

$$P_{\mathcal{A}_P}(A) = \int_T \lambda(t) \mathrm{e}^{-\int_{T_{<t}} \lambda(\tau) \mathrm{d}\tau} Q(S_t, t) \mathrm{d}t.$$

Here we abused some notation. $\lambda(t)$ is in fact $\lambda(l_t, x_t)$, where (l_t, x_t) is the evolved state at time t assumed that there were no transitions before t. $Q(S_t, t)$ is then in fact $Q(S_t, (l_t, x_t))$ and $T_{<t} = \{\bar{t} \in T | \bar{t} < t\}$.

For $P_{\mathcal{A}_C}(A)$ we get multiple integrals, one for each Poisson transition going from l_0 to l'. By X_i we denote the stochastic variable for the transition time instant generated by the Poisson process of the ith Poisson transition outgoing from l_0. By n we denote the number of Poisson jumps outgoing from l_0. Then we get,

$$\begin{aligned} &P_{\mathcal{A}_C}(A) = \\ &\sum_{m_i \in M_{l_0 \to l'}} \int_T \lambda_i(t) \mathrm{e}^{-\int_{T_{<t}} \lambda_i(\tau) \mathrm{d}\tau} \mathrm{P}(X_1 > t, \cdots, X_{i-1} > t, X_{i+1} > t, \cdots, X_n > t) R_i(S_t, t) \mathrm{d}t \\ &= \sum_{m_i \in M_{l_0 \to l'}} \int_T \lambda_i(t) \mathrm{e}^{-\int_{T_{<t}} \lambda_i(\tau) \mathrm{d}\tau} \prod_{j \neq i} \{\mathrm{e}^{-\int_{T_{<t}} \lambda_j(\tau) \mathrm{d}\tau}\} R_i(S_t, t) \mathrm{d}t \\ &= \sum_{m_i \in M_{l_0 \to l'}} \int_T \lambda_i(t) \mathrm{e}^{-\int_{T_{<t}} \lambda(\tau) \mathrm{d}\tau} R_i(S_t, t) \mathrm{d}t \\ &= \int_T \sum_{m_i \in M_{l_0 \to l'}} \{\lambda_i(t) R_i(S_t, t)\} \mathrm{e}^{-\int_{T_{<t}} \lambda(\tau) \mathrm{d}\tau} \mathrm{d}t \\ &= \int_T \lambda(t) Q(S_t, t) \mathrm{e}^{-\int_{T_{<t}} \lambda(\tau) \mathrm{d}\tau} \mathrm{d}t = P_{\mathcal{A}_P}(A), \end{aligned}$$

from which we conclude that the PDP and the CPDP are equivalent as far as it concerns the case where the first transition is caused by a Poisson process.

For the case that the first transition is caused by a boundary-hit: It is evident (since the invariants and vectorfields are the same) that the boundary-hit transition happens at the same time in the PDP and the CPDP. Suppose a hybrid jump occurs from hybrid state (l_0, x), where $x \in \partial Inv(l_0)$. For the CPDP, the probability of jumping into the Borel set $A \subset Inv(l')$ equals

$$\sum_{b \in B_{l \to l'}} C(b, l, x) R_b(A, x),$$

and this equals $Q(A, (l, x))$ of the PDP. So also the transition measure is the same. The PDP and the CPDP are therefore equivalent.

4.2 PDP into closed CPDP

Suppose we have a PDP with transition measure Q and Poisson rate function λ. We will translate Q and λ into boundary-hit transitions, Poisson transitions and the choice function of a closed CPDP. Again, we translate locations, vectorfields and invariants to the CPDP structure. For every location l and l' we define a boundary-hit transition b from l to l' with arbitrary label and with reset map

$$R_b(A, x) = \frac{Q(A, (l, x))}{Q(Inv(l'), (l, x))},$$

for every Borel set $A \subset Inv(l')$ and every $x \in \partial Inv(l)$. For every $x \in \partial Inv(l)$ and every boundary hit transition $b : l \to l'$ we define the transition choice function

$$C(b, l, x) = Q(Inv(l'), (l, x)).$$

Also for every location l and every location l' we define a Poisson transition m from l to l' with arbitrary label and with rate

$$\lambda_m(x) = Q(Inv(l'), (l, x))\lambda(l, x),$$

for every $x \in Inv(l)$, and with reset map

$$R_m(A, x) = \frac{Q(A, (l, x))}{Q(Inv(l'), (l, x))},$$

for every Borel set $A \subset Inv(l')$ and every $x \in Inv(l)$.

We completely specified the CPDP. Now we can check whether the PDP and newly defined CPDP are equivalent. Following the lines from "closed CPDP into PDP", we first look ate the case where the first transition is caused by the Poisson process.

Take any $A \in \mathcal{B}(HT)$. Suppose both the PDP and the CPDP start at $t = 0$ in state (l_0, x_0). We assume that for every $(t, l, x) \in A$, l equals some constant l'. Thus, we restrict to one target location. With the axioms for measure and integral theory, we can easily extend the proof for the multiple target location case. We define

$$T := \{t | (\exists l, x)((t, l, x) \in A)\},$$

and

$$S_t := \{(l, x) | (t, l, x) \in A\}.$$

Then the probability of A for the PDP equals

$$P_{\mathcal{A}_P}(A) = \int_T \lambda(t) \mathrm{e}^{-\int_{T<t} \lambda(\tau)\mathrm{d}\tau} Q(S_t, t)\mathrm{d}t.$$

For the CPDP we now also have one integral because we defined the CPDP such that it had exactly one Poisson transition from l_0 to l'. We assume that this transition is transition i from a total of n Poisson transitions leaving location l_0.

$$P_{\mathcal{A}_C}(A) = \int_T \lambda_i(t)\mathrm{e}^{-\int_{T<t}\lambda_i(\tau)\mathrm{d}\tau}\mathrm{P}(X_1 > t, \cdots, X_{i-1} > t, X_{i+1} > t, \cdots, X_n > t) R_i(S_t, t)\mathrm{d}t$$

$$= \int_T \lambda_i(t)\mathrm{e}^{-\int_{T<t}\lambda_i(\tau)\mathrm{d}\tau}\prod_{j\neq i}\{\mathrm{e}^{-\int_{T<t}\lambda_j(\tau)\mathrm{d}\tau}\}R_i(S_t, t)\mathrm{d}t$$

$$= \int_T Q(Inv(l'), t)\lambda(t)\mathrm{e}^{-\int_{T<t}Q(Inv(l'),\tau)\lambda(\tau)\mathrm{d}\tau}\prod_{j\neq i}\{\mathrm{e}^{-\int_{T<t}Q(Inv(l_j),\tau)\lambda(\tau)\mathrm{d}\tau}\}\frac{Q(S_t, t)}{Q(Inv(l'), t)}\mathrm{d}t,$$

where l_j is the target location of Poisson transition j, then the above equation equals

$$\int_T \lambda(t)\mathrm{e}^{-\int_{T<t}\sum_{l\in L}\{Q(Inv(l),\tau)\}\lambda(\tau)\mathrm{d}\tau}Q(S_t, t)\mathrm{d}t,$$

and because $\sum_{l\in L}\{Q(Inv(l), \tau)\} = 1$, this equals

$$\int_T \lambda(t)\mathrm{e}^{-\int_{T<t}\lambda(\tau)\mathrm{d}\tau}Q(S_t, t)\mathrm{d}t = P_{\mathcal{A}_P}(A).$$

For the case that the first transition is caused by a boundary-hit: It is evident (since the invariants and vectorfields are the same) that the boundary-hit transition happens at the same time in the PDP and in the CPDP. Suppose a hybrid jump occurs from hybrid state (l_0, x), where $x \in \partial Inv(l_0)$. For the CPDP, the probability of jumping into the Borel set $A \subset Inv(l')$ equals

$$\sum_{b\in B_{l\to l'}} C(b, l, x)R_b(A, x) = C(b, l, x)R_b(A, x),$$

because there is exactly one boundary-hit transition b from l to l'. Then we get,

$$C(b, l, x)R_b(A, x) = Q(Inv(l'), (l, x))\frac{Q(A, (l, x))}{Q(Inv(l'), (l, x))}$$

and this equals $Q(A, (l, x))$ of the PDP. So also the transition measure is the same. The PDP and the CPDP are therefore equivalent.

Chapter 5

The refined CPDP framework

In this chapter we introduce a refined version of the CPDP framework. First we will discuss why and where we think that the old framework needs to be refined. The compositional power of the old CPDP framework is restricted to one-way communication via synchronizing discrete transitions. For many applications, especially control applications as feedback control, it is necessary that continuous variables interact with each other (in feedback, interaction would mean that the output of one system equals the input of the other system and vice versa). In the new CPDP framework this is possible. The continuous dynamics will be described now in a behavioral setting (behavioral as in [16]) rather than as differential equations because we think that the behavioral setting has nicer compositional properties (see [16] for the introductory book on behavioral systems).

Another refinement we make has to do with the synchronizing discrete transitions. Only one-way communication was allowed in the old framework. We want to retain this one-way principle (expressed via active and passive interactions) in the new framework, but we also want to add the possibility of two-way interaction. Two-way interaction is very popular in automata theory and it is used for many applications. For that reason and for the reason that we are able to include it in the framework without destroying the one-way interaction principle, we chose to extend the CPDP framework with two-way interaction. Two-way interaction means (in our setting) that a transition with label a (assumed that a is a two-way synchronization label) can only go as soon as all the other components are ready to execute an a as well. In this way the components have to wait for each other. Therefore, one transition in a component can influence the other components, and the other components can also influence this same transition.

We include two-way interaction by means of synchronizing boundary-hit transitions. This means that a CPDP synchronizing on a can execute a boundary-hit transition labelled a only when the boundary is reached and all the other components reached their boundaries at exactly the same time and have boundary-hit transitions labelled a available at these boundaries. With the old concept of boundary, where the jump takes place as soon as the boundary of the invariant is reached, two-way interaction seems to be a bit meaningless since it is very unlikely that multiple components reach the boundary of their invariants at exactly the same time. Therefore, we will use a new concept of "boundary" in the new framework. In this new concept the boundary is a specific area in the invariant, rather than "the boundary of the invariant". If the process state is in the boundary area at some moment in time, then the process is allowed to make a boundary-jump. The word "boundary" is not really appropriate now anymore, therefore we will use the word "guard" for this concept. Guards are generally used in automata frameworks. For boundary-hit transition we now choose the name interactive transition. With this concept of guards as areas in the invariant, two-way interaction makes good sense since it is possible that at some time instant all components are in their guard areas. And if this is the case, the interactive transitions (with the same synchronization label) are allowed to be taken.

In the next section we will define the new CPDP framework. In the sections thereafter we define the new composition operator (which makes one-way and two-way interaction possible) and give examples of discrete (transition) interaction and continuous (variable) interaction. We end with a little discussion about the chosen structures for the CPDP framework.

5.1 Definition CPDPs

In this section we introduce the new CPDP formalism. We make use of the following notation: If S is a set of variables, then with $\mathbf{S}$ we denote the valuation space of S. A CPDP $\mathcal{A}$ is a 10-tuple (L, W, *Active*, *Reset*, *Inv*, A, I, M, P, $\mathfrak{B}$), where

- L is a countable set of locations.
- W is the set of continuous variables. A continuous variable w takes its values in $\mathbb{R}^{d(w)}$, where $d(w)$ denotes the dimension of w.
- *Active* assigns to each element in L a subset of W, which denotes the set of active variables.
- *Reset* assigns to each element in L a subset of W, which denotes the set of variables that should be reset as soon as the location is entered.
- *Inv* maps each location to its invariant set. This means that $Inv(l)$ is an open subset of $\mathbf{W}$. We also define the boundary set $\partial Inv(l) := \overline{Inv(l)} \backslash Inv(l)$ of $Inv(l)$.
- A is the set of labels.
- I is the set of interactive transitions. Each element i of I is a pentuple (l, a, l', G, R), where l is the origin location, a is the label of the jump and l' is the target location. $G \subset \mathbf{W}$ is the guard of the jump. R is the reset map of the jump. $R(A, x)$ is the probability of jumping into the set $A \subset \mathbf{Reset}(l')$ when the transition i is taken from $x \in \mathbf{Active}(l)$. $R(A, x)$ is defined for all $x \in (Inv(l) \cap G(l)) \downarrow Active(l)$ and for all Borel subsets A of $Inv(l') \downarrow Reset(l')$. With $\mathbf{S} \downarrow S'$ we mean $\mathbf{S}$ restricted to the variables in S'.
- M is the set of Poisson transitions. Each element m of M is a pentuple (l, a, l', R, λ), where l is the origin location, a is the label of the jump, l' is the target location, R is the reset map of the jump, and λ is the jump rate. $R(A, x) \in [0, 1]$ is the probability of jumping into the set A when the transition m is taken from state x. $R(A, x)$ is defined for all $x \in Inv(l) \downarrow Active(l)$ and for all Borel subsets A of $Inv(l') \downarrow Reset(l')$.

$\lambda : Inv(l) \rightarrow \mathbb{R}_+$ is a bounded Borel measurable function, and determines the rate of jumping of the process.

- P is the set of passive transitions. Each element p of P is a quadruple (l, a, l', R), where l is the origin location, a is the label of the jump, l' is the target location, and R is the reset map of the jump. $R(A, x) \in [0, 1]$ is the probability of jumping into the set A when the transition b is taken from state x. $R(A, x)$ is defined for all $x \in Inv(l) \downarrow Active(l)$ and for all Borel subsets A of $Inv(l') \downarrow Reset(l')$.
- $\mathfrak{B}$ maps each location to its continuous behavior. $\mathfrak{B}$ is a subset of $\mathbf{W}^{\mathbb{R}}$.

In the next section we use the following shorthand notation. We write $l \xrightarrow{a,R} l'$, $l \overset{a,R}{\dashrightarrow} l'$, $l \overset{a,R,\lambda}{\longmapsto} l'$ to denote the existence of respectively interactive, passive and Poisson jumps from l to l' with label a, reset map R and, in the case of a Poisson jump, jump rate λ. Furthermore, $l \rightarrow l'$ denotes the existence of an interactive jump from l to l' and $l \rightarrow$ denotes the existence of an interactive jump outgoing from l. For Poisson and passive jumps we use equivalent notations. Finally, we also have negated versions like $l \not\rightarrow$, which means that l has no outgoing interactive transitions.

5.2 Composition of CPDPs

In this section we introduce a composition operator $||_S$ on CPDPs. In CPDPs, communication or interaction takes place between two interactive, between an interactive and a passive, or between a Poisson and a passive transition. In a context with two composed CPDPs, one of the CPDPs can execute a passive transition with label a if and only if at the same time the other CPDP executes an active event with label a. We call all interactive and all Poisson transitions active transitions opposed to passive transitions. The execution of a Poisson transition always happens independently from the other systems in the composition context. Interaction via two interactive transitions happens by synchronizing on labels from S, which is the set of synchronization-labels. This notion of composition is formally defined below. The main difference with the composition operator from Section 3.2 is that the refined operator also composes continuous variables (expressed as the intersection of behaviors). The composition rules **r0** till **r6** are similar to the ones from Section 3.2, but now we have to pay more attention to the reset maps because now the continuous variables also play a role in the reset maps.

Suppose the CPDPs $\mathcal{A}_i = (L_i, W, Active_i, Reset_i, Inv_i, A, I_i, M_i, P_i, \mathfrak{B}_i)$ are given for $i = 1, 2$. Then the composition $\mathcal{A} = \mathcal{A}_1 ||_S \mathcal{A}_2$, where S is a subset of A, is characterized by the 10-tuple $(L, W, Active, Reset, Inv, A, I, M, P, \mathfrak{B})$, where

$$L = L_1 \times L_2,$$

$$Active(l_1, l_2) = Active_1(l_1) \cup Active_2(l_2),$$

$$Reset(l_1, l_2) = Reset_1(l_1) \cup Reset_2(l_2),$$

$$Inv(l_1, l_2) = Inv_1(l_1) \cap Inv_2(l_2),$$

$$\mathfrak{B}(l_1, l_2) = \mathfrak{B}_1(l_1) \cap \mathfrak{B}_2(l_2).$$

In the rules **r0** till **r6** below, l_1 and l_1' are in L_1 and l_2 and l_2' are in L_2.

$i \in I$ if i can be derived from the following three rules,

$$\textbf{r0.}\ \frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2', a \in S}{(l_1,l_2) \overset{a,G_1\cap G_2,R}{\longrightarrow} (l_1',l_2')},$$

$$\textbf{r1.}\ \frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a}{\not\dashrightarrow}, a \notin S}{(l_1,l_2) \overset{a,G_1,R}{\longrightarrow} (l_1',l_2)}, \textbf{r2.}\ \frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2', a \notin S}{(l_1,l_2) \overset{a,G_1,R}{\longrightarrow} (l_1',l_2')}.$$

We will now specify R in **r0**, **r1** and **r2**. For **r0**, **r1** and **r2** we have that R is respectively the product measure $R_1 \times R_2$, $R_1 \times 1$ and $R_1 \times R_2$, where 1 denotes the indicator function. In the case of **r0** and **r2** we assume that $Reset_1(l_1')$ and $Reset_2(l_2')$ are disjoint, otherwise the product measure would be ill-defined.

$m \in M$ if m can be derived from the following two rules,

$$\textbf{r3.}\ \frac{l_1 \overset{a,R_1,\lambda_1}{\longmapsto} l_1', l_2 \not\dashrightarrow}{(l_1,l_2) \overset{a,R,\lambda}{\longmapsto} (l_1',l_2)}, \textbf{r4.}\ \frac{l_1 \overset{a,R_1,\lambda_1}{\longmapsto} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2'}{(l_1,l_2) \overset{a,R,\lambda}{\longmapsto} (l_1',l_2')}.$$

In **r3** R is the measure with $R(A_1 \times A_2, (x_1,x_2)) = R_1(A_1,x_1)1_{A_2}(x_2)$ for all $x_1 \in Inv(l_1)$, all $x_2 \in Inv(l_2)$ and all Borel subsets A_1 of $Inv(l_1')$ and A_2 of $Inv(l_2)$. In **r4** R is the measure with $R(A_1 \times A_2, (x_1,x_2)) = R_1(A_1,x_1)R_2(A_2,x_2)$ for all $x_1 \in Inv(l_1)$, all $x_2 \in Inv(l_2)$ and all Borel subsets A_1 of $Inv(l_1')$ and A_2 of $Inv(l_2')$. In both **r3** and **r4**, $\lambda(x_1,x_2) = \lambda_1(x_1)$ for all $x_1 \in Inv(l_1)$ and all $x_2 \in Inv(l_2)$.

$p \in P$ if p can be derived from the following two rules,

$$\textbf{r5.}\ \frac{l_1 \overset{a,R_1}{\dashrightarrow} l_1', l_2 \not\dashrightarrow}{(l_1,l_2) \overset{a,R}{\dashrightarrow} (l_1',l_2)}, \textbf{r6.}\ \frac{l_1 \overset{a,R_1}{\dashrightarrow} l_1', l_2 \overset{a,R_2}{\dashrightarrow} l_2'}{(l_1,l_2) \overset{a,R}{\dashrightarrow} (l_1',l_2')}.$$

In **r5** R is the measure with $R(A_1 \times A_2, (x_1,x_2)) = R_1(A_1,x_1)1_{A_2}(x_2)$ for all $x_1 \in Inv(l_1)$, all $x_2 \in Inv(l_2)$ and all Borel subsets A_1 of $Inv(l_1')$ and A_2 of $Inv(l_2)$. In **r6** R is the measure with $R(A_1 \times A_2, (x_1,x_2)) = R_1(A_1,x_1)R_2(A_2,x_2)$ for all $x_1 \in Inv(l_1)$, all $x_2 \in Inv(l_2)$ and all Borel subsets A_1 of $Inv(l_1')$ and A_2 of $Inv(l_2')$.

Beside the rules **r0** till **r6**, there are the rules **r0'** till **r6'** which are the mirrored versions of **r1** till **r6**. This means that

$$\textbf{r3'.}\ \frac{l_2 \overset{a,R_2}{\longmapsto} l_2', l_1 \not\dashrightarrow}{(l_1,l_2) \overset{a,R}{\longmapsto} (l_1,l_2')}, \textbf{r4'.}\ \frac{l_2 \overset{a,R_2}{\longmapsto} l_2', l_1 \overset{a,R_1}{\dashrightarrow} l_1'}{(l_1,l_2) \overset{a,R}{\longmapsto} (l_1',l_2')},$$

etc. Transitions are elements of B, M or P, if and only if they follow from the rules **r0** till **r6** and **r0'** till **r6'**.

5.3 Examples - Discrete interaction

We give three examples of composition of CPDPs. In Figure 5.1 we see the first example. Here B executes an (inter)active transition and A follows with the passive transition. We see $A||B$, which is the result of the composition. No guard is indicated for the interactive transition. This means that the transition is always enabled to go, or in other words, the guard is equal to $\mathbf{W}$. In Figure 5.2 we see an example of interaction via two way communication. The two interactive transitions synchronize on event b. In Figure 5.3, we see an example where there is both active / passive communication and interactive (two way) communication. The Poisson transition in B can trigger the passive transition in A via signal a. Event b is used for the two way interactive synchronization.

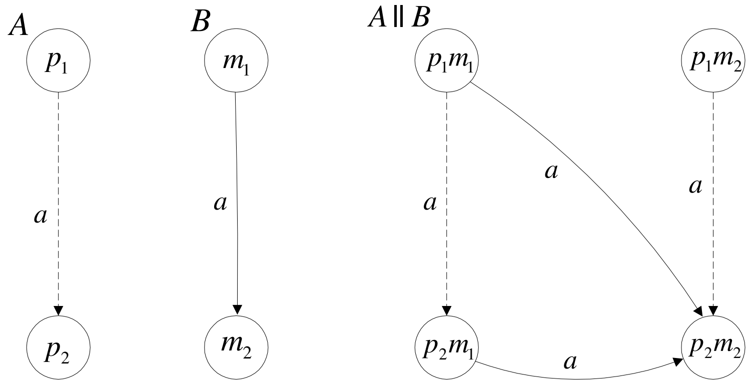

Figure 5.1: Active / passive composition

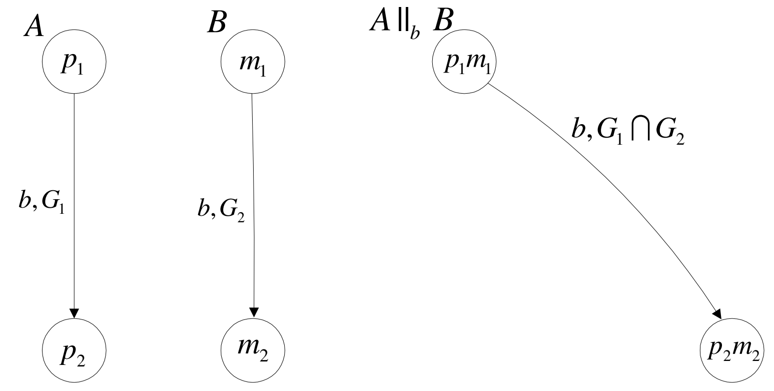

Figure 5.2: Interactive composition

5.4 Example - Continuous interaction

We consider the example of Figure 5.1, nut now we will look at the continuous dynamics. Suppose that in the locations of A there is a continuous variable y active and in the locations

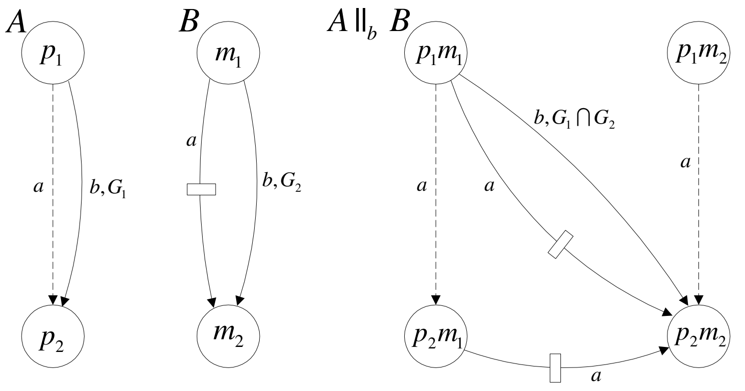

Figure 5.3: Active / passive and interactive composition

of B there is a continuous variable x active. CPDP B is responsible for the dynamics of x, but we want the resetmap in A to depend on the value of x. To accomplish this, we could use the following (partial) specifications:

$$Active_A(p_1) = \{x, y\}, Active_B(m_1) = \{x\},$$

$$Reset_A(p_2) = \{x\}, Reset_B(m_2) = \{y\},$$

$$\mathfrak{B}_A(p_1) = \{(x(t), y(t)) | \dot{y} = f_y(y), x(t) \in \mathbf{x}^{\mathbb{R}}\},$$

$$\mathfrak{B}_B(m_1) = \{(x(t), y(t)) | \dot{x} = f_x(x), y(t) \in \mathbf{y}^{\mathbb{R}}\}.$$

The elements in $Active_A(p_1)$ determine which variables have influence on the resetmap. Since $x \in Active_A(p_1)$, we opened the possibility to let x, the dynamics in B, influence the reset in CPDP A.

5.5 Discussion

We introduced the new structure for CPDPs. Opposed to the former structure, we can now model two way communication and we can make use of guards which introduce nondeterminism. Nondeterminism means that certain things are not specified. In this case it is not specified at which point in the guard area the transition will actually take place. The consequence of this is that the class of CPDPs is now broader than the class of PDP systems. So, we can no longer speak of a one-to-one relation with PDP systems. On the other hand, we can try to figure out which assumptions should be made to ascertain that a CPDP behaves like a PDP.

Concerning the continuous dynamics, we chose to make all existing continuous variables available in all locations. In the composition of two CPDPs, this means that both CPDPs have the same set of continuous variables, which results in easy and understandable notations (we can simply take intersections etc.). This choice does not seem to be restrictive, if in some location we only consider part of the variables, we can leave the rest unspecified. In behavioral terms this means that the behavior for these variables equals the space of all possible trajectories.

The class of systems we can model now with the new CPDP framework, is broader than IMC, Timed Automata and PDP. It seems to be a superset for these three classes of systems. If we focus on PDP systems, we should find strategies to close down CPDP systems such that they represent PDP behavior. For resolving nondeterminism introduced by (synchronization of) guarded interactive transitions, we could for example use a maximal progress strategy (as in IMC). For now, we omitted the choice function C, which was present in the old structure. In cases where a boundary point has multiple interactive transitions, we should reintroduce a kind of choice function to resolve this kind of nondeterminism.

Chapter 6

Compositional specification of a forest landscape model using CPDPs

6.1 Introduction

In this chapter we will give a more or less real-life example of compositional modelling within the new CPDP framework. The example is based on the article *Cell interaction in semi-Markov forest landscape models* ([14]). In this article, a landscape is modelled as a square of m times m plots. Each plot influences the evolution of its neighbor-plots. The total process is modelled as a Piecewise Deterministic Markov Process. In this article we show that the modelling can be done in a compositional way using CPDPs. At the end of the chapter we have a little discussion about the benefits of using a compositional framework for the kind of models of this example.

We simplify the model by using a cell of m times one plots, where each plot has two neighbors (See Figure 6.1 where we see an m times m cell and a m times one cell with $m = 8$). Each plot can be in three different states. In the original model each plot can be in five different states.

6.2 The PDP model

A model is made of a forest-cell. A cell consists of n plots. This means that each plot, except for the plots on the boundaries, has two neighbors which can influence the evolution of the plot. Each plot can be in one of three locations. While time progresses, a plot can change to a different location. A location represents a specific kind of forest, depending on species dominance and tree height. If plot i is in state k, then $p^i_{k,j}$ is the probability that the next location for plot i is j. This $p^i_{k,j}$ depends on the locations of the neighbors. More about these interdependencies follows later.

Given that plot i is in state k and given that it will jump to state j, the random time that plot i spends in state k before it jumps, is distributed with density $f^i_{k,j}$ and distribution

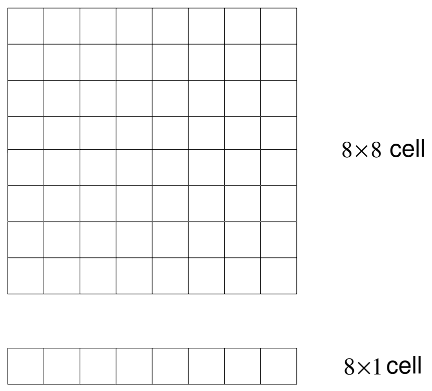

Figure 6.1: Forest cells

function $F^i_{k,j}$.

This can be turned into a PDP. The following formulas come from [14]. For plot i, if it is in location k, we have a Poisson process with rate

$$\lambda^i_k(u) = \frac{\sum_j p^i_{k,j}(u) f^i_{k,j}(u)}{\sum_j p^i_{k,j}(u)(1 - F^i_{k,j}(u))}, \tag{6.1}$$

where u is the time spent in location k. This means that u has a clock dynamics: $\dot{u} = 1$. And we have transition measures

$$Q^i_{k,j}(u) = \frac{p^i_{k,j}(u) f^i_{k,j}(u)}{\sum_j p^i_{k,j}(u) f^i_{k,j}(u)}, \tag{6.2}$$

which equals the probability of jumping to location j if a jump occurs from location k at time u. After each jump, the continuous variable u is reset to zero. Jumps only occur as consequence of the Poisson process. This means that there are no boundaries and consequently no boundary-hits.

$p^i_{k,j}(u)$, depends on the configuration of the neighbors of plot i at time u. The interrelation is as follows.

$$p^i_{k,j}(u) = \frac{1}{2} p^{i*}_{k,j,h_l} + \frac{1}{2} p^{i*}_{k,j,h_r}, \tag{6.3}$$

where h_l is the location of the left neighbor and h_r is the location of the right neighbor and $p^{i*}_{k,j,h}$ denotes the probability for plot i to jump to j if it is in location k and if both neighbors are in location h. This probability is assumed to be known.

From the paper it is not clear to us what happens exactly to the other plots if one plot changes location. We have the idea that in their model, all clocks of all plots are reset to zero as soon as one of the plots changes location. On the other hand, we doubt on this because it does not seem very realistic. From now on we suppose that the density functions $f^i_{k,j}$ are all exponential functions, i.e. of the form $\lambda e^{\lambda u}$. In that case, the process is memory-less and the reset (if it is there at all) will not influence the process.

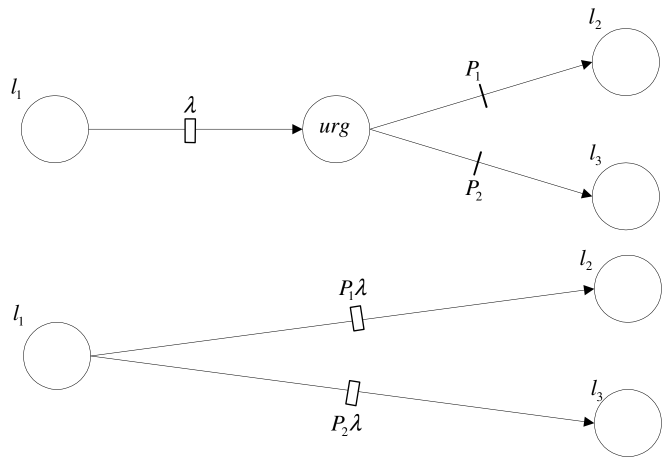

Figure 6.2: Equivalence between a PDP and a CPDP

The top part of Figure 6.2 can be seen as a graphical representation of the described PDP: 'urg' means that this is an urgent state, i.e. time is not allowed to progress here. Formally, the PDP structure does not allow 'urgent states'. However, in [14] it is explained how these urgent states can be transformed into the PDP structure. Once we reach this urgent state, we have to jump immediately to the next location. The arrows with the bars and P_1 and P_2, represent probabilistic jumps with probability P_1 or P_2. Here, P_1 equals Q_{l_1,l_2} and P_2 equals Q_{l_1,l_3}. The lower part of Figure 6.2 is a CPDP describing the same process, it can easily be proven that the processes are (transition) equivalent.

6.3 Interaction via Continuous Variables

We will model each plot as a CPDP and the whole cell as the communicating network of these CPDPs. How should communication take place? In the PDP description, we see that the location of a CPDP influences the jumping process of its neighbors. This means that for modelling interaction/communication, it is enough for a PDP to know in which locations its neighbors are. The location of a PDP is the only information that needs to be communicated to the environment. The most natural way, in our opinion, to model this is to assign a "continuous" variable l^i to each plot i, which holds the value of the current location. This means that l_i takes values in $\mathbb{R}$, but only part of the natural numbers. If plot i has three locations, l^i takes value 1 in location 1, 2 in location 2, etc. In each location the vectorfield of l^i equals zero. This continuous variable can be read by other CPDPs.

In figure 6.3 we see the CPDP of plot i. Each location has the following continuous variables: u equals the time spent in the location. u has a clock dynamics. l^i equals the value of the current location (so equals 1, 2 or 3) and has zero dynamics. l^l equals the value of the current location of the left neighbor. The dynamics is defined as free (the behavior is the set of all possible trajectories) and will be read from the left neighbor CPDP. l^r the same, now

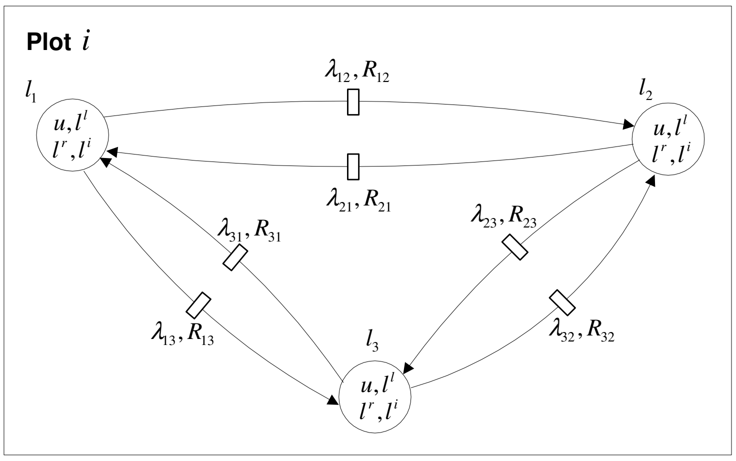

Figure 6.3: CPDP of plot i

for the right neighbor. Concerning the Poisson transitions: λ_{ij} equals $\lambda_i^i Q_{ij}^i$ from equations 6.1, 6.2 and 6.3, where h_l and h_r from equation 6.3 are now l^l and l^r respectively. Concerning the reset maps: R_{ij} resets u to zero and l^i to j.

6.4 Interaction via Active/Passive Transitions

We said that it is most natural to model the interaction via the continuous variables. However, it is also possible to do it via communication between active and passive transitions. This is much ado, because we need many transitions. Whenever plot i changes from location k to j, it should broadcast a signal s_{kj}^i which can be received by a passive transition in its neighbor CPDPs. Then, each CPDP can broadcast six different signals. All these signals are relevant for a neighbor CPDP and since every CPDP has two neighbors, every location in that CPDP should be enriched with twelve passive transitions.

In Figure 6.4 we see the CPDP where location l_1 is enriched with twelve passive transitions. Note that all six passive transitions corresponding to one neighbor are pictorially grouped to one passive transition. The outgoing transitions from locations l_1, l_2 and l_3 are enriched with labels that will be broadcast as soon as the transitions are taken. Inside each location there are three continuous variables. The fourth variable, l^i is not necessary anymore since this variable was used for communication and now communication goes via the transitions.

6.5 The benefit of compositional modelling

In the original paper that we discussed here, the Poisson rates and the transition measures are assessed for each plot from the cell. These rates and measures could easily be translated to the CPDP framework to build the constituent parts of the cell. In the original paper, these different rates and transitions still had to be translated to the single rate and to the single transition measure of the PDP that is describing the whole cell. The formulas for calculating these are given in equations (1) and (2) in the original paper [14]. One of the benefits of using a compositional framework is that after the compositional specification, the rest can be done

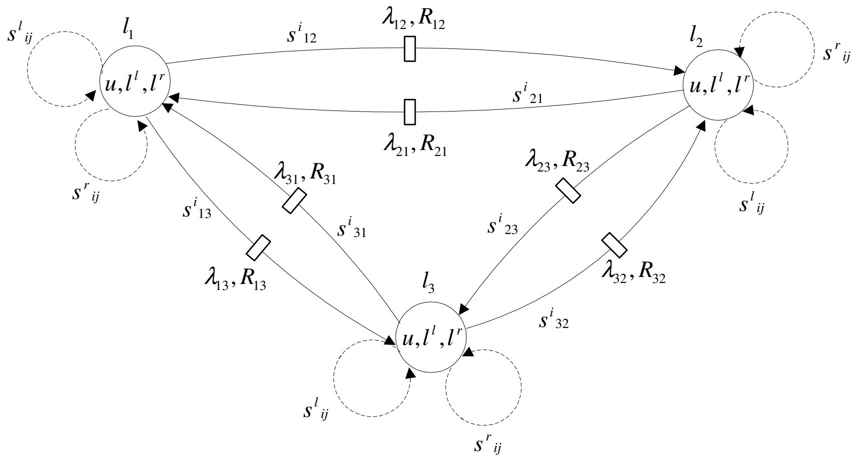

Figure 6.4: CPDP of plot i with passive transitions

automatically. At least, if these procedures are automated in for example computer tools. Finding the formulas that interrelate the different rates and measures (which becomes more difficult as the model becomes more complex) is not needed then anymore.

Chapter 7

Future research

We conclude this report by pointing out some directions that we have in mind for future research and future development of the compositional framework for (PDP like) stochastic hybrid systems.

We think it is interesting to explore the relations between CPDPs and PDPs or equivalently (and probably easier to explore) between the new CPDP and the old CPDP framework. We could try to find strategies which resolve the nondeterminism. Which strategies seem to be useful (like the maximal progress strategy of [7])? Which strategies resolve the nondeterminism such that the result is a PDP?

One of the more important questions is: Is the notion of communication as it now is in the CPDP framework, suitable for the kind of applications that we have in mind? Does the CPDP framework provide enough and the right compositional power? Before we can answer these questions we should first have to get clear for ourselves what kind of systems (for example ATM systems or operations) we want to model.

Also we have to formally justify our composition operator in the sense that we still have to prove that it is at least commutative and associative. Another thing which plays an important role in the stochastic processes, automata and process algebra literature is the concept of equivalence. Which notions of equivalence between CPDPs are interesting and useful? How are the compositional features of the composition operator in the context of substitution of equivalent components for certain components of a complex system?

Analysis of stochastic hybrid systems is an interesting topic to contemplate. What kind of analysis can we expect to be possible for such complex systems as CPDPs? Is formal analysis possible, or can we only expect that (Monte Carlo) simulations will really provide results? What benefits do we get from abstraction strategies? Do they make formal analysis possible? At least it seems to be clever to retain the (strong) Markov property, since the (strong) Markov property is very helpful for analysis of stochastic processes.

We consider all these questions written above as interesting research questions. The answers may provide us the right directions for further development of the compositional framework.

Bibliography

[1] P. R. D'Argenio, *Algebras and automata for timed and stochastic systems*, Ph.D. thesis, University of Twente, 1997.

[2] M. H. A. Davis, *Piecewise deterministic markov processes: a general class of non-diffusion stochastic models*, Journal Royal Statistical Soc. (B) **46** (1984), 353–388.

[3] M. H. C. Everdij and H. A. P. Blom, *Piecewise deterministic markov processes represented by dynamically coloured petri nets*, Tech. Report NLR-TP-2000-428, National Aerospace Laboratory NLR, Amsterdam, 2000.

[4] J.A. Ferreira and J.P. Estima de Oliveira, *Modelling hybrid systems using statecharts and modelica*, citeseer.nj.nec.com/268684.html.

[5] D. Harel, *Statecharts: a visual formalism for complex systems*, Science of Computer Programming **8** (1987), 231–274.

[6] T. A. Henzinger, *The theory of hybrid automata*, Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1996, pp. 265–292.

[7] H. Hermanns, *Interactive markov chains*, Ph.D. thesis, Universität Erlangen Nürnberg, 1998.

[8] A. A. Julius, S. N. Strubbe, and A. J. van der Schaft, *Control of hybrid behavioral automata by interconnection*, Submitted to ADHS 2003 conference.

[9] A.A. Julius, S. N. Strubbe, and A. J. van der Schaft, *Compositional modelling of hybrid systems with hybrid behavioral automata*, unpublished.

[10] N. A. Lynch, R. Segala, and F. W. Vaandrager, *Hybrid I/O automata revisited*, Lecture Notes in Computer Science **2034** (2001), 403–414.

[11] ———, *Hybrid i/o automata*, preprint, 2003.

[12] N. A. Lynch and M. R. Tuttle, *An introduction to input/output automata*, CWI Quarterly **2** (1988), no. 3, 219–246.

[13] R. Milner, *Communication and concurrency*, Prentice Hall, 1989.

[14] M.G. Monticino, T. Cogdill, and M.F. Acevedo, *Cell interaction in semi-markov forest landscape models*, Proceedings of Integrated Assessment and Decision Support (iEMSs), 2002, pp. 227–232.

[15] P.J.L.Cuijpers, M.A. Reniers, and W.P.M.H. Heemels, *Hybrid transition systems*, Tech. report, Technische Universiteit Eindhoven, 2002.

[16] J.W. Polderman and J.C. Willems, *Introduction to mathematical system theory : A behavioral approach*, Springer, New York, 1998.

[17] S. N. Strubbe, A. A. Julius, and A. J. van der Schaft, *Communicating piecewise deterministic markov processes*, Submitted to ADHS conference 2003.

[18] A. J. van der Schaft and J. M. Schumacher, *An introduction to hybrid dynamical systems*, Springer London, 2000.

[19] J. C. Willems, *On interconnections, control and feedback*, IEEE Transactions on Automatic Control **42** (1997), 326–339.