Chapter 3

Compositional Modelling of Stochastic Hybrid Systems

Stefan Strubbe
University of Twente

Arjan van der Schaft
University of Groningen

3.1 Introduction

Stochastic hybrid systems often have a complex structure, meaning that they consist of many interacting components. Think for example of an Air Traffic Management system where multiple aircraft, multiple humans, etc., are involved. These systems are too complex to be modelled in a monolithic way. Therefore, for these systems there is a need for compositional modelling techniques, where the system can be modelled in a stepwise manner by first modelling all individual components and secondly by connecting these components to each other.

In this chapter we present the framework of CPDPs (Communicating Piecewise Deterministic Markov Processes), which is a compositional modelling framework for stochastic hybrid systems of the PDP type. (For the PDP model we refer to [4] or [3].) In this framework each component of a complex stochastic hybrid system can be modelled as a single CPDP and all these component CPDPs can be connected through a composition operator. As we will see, connecting two or more CPDP components results in another CPDP. In other words, the class of CPDP is closed under the composition operation. CPDP is an automaton framework (like the models from [1] and [10]). Another framework for compositional modelling of PDP-type systems is [5], which is a Petri-net framework.

The framework of CPDPs can be seen as an extension of the established framework of IMCs (Interactive Markov Chains, [6]). The main extension of CPDPs with

respect to IMCs is the same as the extension of a Piecewise Deterministic Markov Process with respect to a continuous-time Markov chain: it allows for a general continuous dynamics in the continuous state variables, while the jump rates of the Poisson processes may depend on these continuous state variables, and the continuous state variables are stochastically reset at event times. As a result, CPDPs cover quite a large class of stochastic hybrid systems as encountered in applications, although *diffusions* can*not* be included.

A CPDP is a syntactical object. To make clear how a CPDP behaves, we need to give a formal semantics of CPDP. In Section 3.2 we introduce some semantical models and we explain the behavior of these models. Then, in Section 3.3 we introduce the CPDP model and we give its semantics in terms of the semantical models of Section 3.2. By giving these semantics, we make clear how the CPDP behaves. At the end of Section 3.3, the CPDP model is extended to the so-called value-passing CPDP model. In this extended model there are richer interaction possibilities: CPDP components can now send information to each other concerning the continuous variables.

The contents of this chapter is based on the papers [12] and [15] and on the thesis [11]. For more material on this subject, for further explanation of the material of this chapter, and for the proofs of the theorems of this chapter we refer to the thesis [11].

3.2 Semantical Models

Semantical models are used to capture/express the behavior of a syntactical model. Semantical models are also used to compare syntactical models with each other. For example, if two syntactical objects have the same semantics, they can be regarded equivalent.

In this chapter we consider two syntactical and four semantical models. The syntactical models are: Piecewise Deterministic Markov Processes (PDPs) and Communicating Piecewise Deterministic Markov Processes (CPDPs). The semantical models are: Transition Mechanism Structures (TMSs), Non-deterministic Transition Systems (NTSs), Continuous Flow Spontaneous Jump Systems (CFSJSs), and Forced Transition Systems (FTSs). We can distinguish different levels for the semantical and syntactical models that we use. If the behavior of a semantical model M_1 can be expressed within the semantical model M_2, then we say that M_1 is a higher level semantical model than M_2. From high to low we consider the following levels.

- Syntactical level: PDP, CPDP
- High semantical level: CFSJS, NTS
- Intermediate semantical level: FTS
- Low semantical level: TMS

In this section we define all semantical models and we show how these models are related to each other, i.e., how lower semantical objects express the behavior of higher semantical objects.

All semantical models in this section will be used to capture a certain part of the behavior of PDP or CPDP-type systems. The final definition of the semantics of CPDPs in Section 3.3.2 will be done in terms only of a CFSJS and an NTS.

3.2.1 Transition Mechanism Structure

A Transition Mechanism Structure (TMS) gives us the random variables and the flow maps that are necessary to determine execution paths of a stochastic (hybrid) system. Once we know the TMS of a system, we can directly determine the stochastic process and the stochastic execution paths of the system. The stochastic process or the execution paths can be used to analyze the systems (stochastic) behavior.

A TMS consists of two parts: 1. a transition mechanism which determines the time of a transition and the target state of the transition, 2. a flow map which determines the continuous flow between two transitions. The semantical model TMS is formally defined as follows.

DEFINITION 3.1 *A* Transition Mechanism Structure *(TMS) is a tuple* (E, ξ_0, ϕ, TM). *E is a Borel state space, ξ_0 is the initial state. ϕ is a flow map, i.e., the process evolves from state ξ_0 at time zero to state $\phi(t,\xi)$ at time t if no transitions occur in the interval $[0,t]$, etc. TM is a* transition mechanism *on E. A* transition mechanism *on a Borel space E is a pair (T,Q) with $T : E \to RV(\mathbb{R}_+, \mathscr{B}(\mathbb{R}_+))$, where $RV(\Omega,\mathscr{F})$ denotes the set of all random variables (defined on any probability space) taking values in the measurable space $(\Omega,\mathscr{F})$ and where $\overline{\mathbb{R}}_+ := \mathbb{R}_+ \cup \{\infty\}$, and with $Q : E \to Prob(E)$. Here $Prob(E)$ denotes the set of all probability measures on the measurable space $(E,\mathscr{B}(E))$.*

Given a state $\xi \in E$, the transition mechanism $TM = (T,Q)$ determines a transition-time t and a transition target state ξ' by drawing a sample t from the random variable $T(\xi)$ followed by drawing a sample ξ' from the probability measure $Q(\phi(t,\xi))$. We also say that with this procedure we have drawn the sample (t,ξ') from the transition mechanism $TM(\xi)$. If the sample ∞ is drawn from $T(\xi)$, then this is not followed by drawing a sample from Q. We then say that the sample $(\infty,\emptyset)$ is drawn from $TM(\xi)$.

An execution path of a TMS (E,ξ_0,ϕ,TM) is generated as follows. Draw a sample (t_1,ξ_1) from $TM(\xi_0)$. For $t \in [0,t_1[$ the execution path has value $\phi(t,\xi_0)$. Draw a sample (t_2,ξ_2) from $TM(\xi_1)$. For $t \in [t_1,t_1+t_2[$, the execution path has value $\phi(t-t_1,\xi_1)$. Draw a sample (t_3,ξ_3) from $TM(\xi_2)$, etc.

TMS forms the lowest semantical level that we use. The TMS of a system can be derived from the CFSJS and the FTS semantics of the system. This derivation is done in Section 3.2.4.

3.2.2 Continuous Flow Spontaneous Jump System (CFSJS)

For both the syntactical models PDP and CPDP, we have that transitions, where the state instantaneously jumps to another state, can happen in two ways: 1. spontaneously (with some probability distribution), 2. forced (when the state reaches some "forbidden" area and is forced to jump to a another state). In between transitions, the state of a PDP/CPDP evolves continuously. The part of the PDP/CPDP system behavior concerning the continuous evolution and the spontaneous transitions is captured by a CFSJS. The part concerning the forced transitions is captured by an FTS.

DEFINITION 3.2 *A CFSJS is a tuple* $(E, \xi_0, \phi, \lambda, Q)$. *The state space* E *is a Borel space.* ξ_0 *is the initial state,* $\phi : \mathbb{R}_+ \times E \to E$ *is the flow map,* $\lambda : E \to \mathbb{R}_+$ *is the jump rate and* $Q : E \to \mathit{Prob}(E)$ *is the transition measure.*

The jump rate $\lambda(\xi)$ of a CFSJS at state ξ determines the probability of a spontaneous transition "near" state ξ as follows: if the system is at state ξ at time t, then the probability that a spontaneous transition occurs in the interval $[t, t+\Delta t]$ equals $\lambda(\xi)\Delta t + o(\Delta t)$, where $o(\Delta t)$ denotes a function such that $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$. In other words, for Δt small enough, the probability that a spontaneous transition occurs in the interval $[t, t+\Delta t]$ equals approximately $\lambda(\xi)\Delta t$. (This means that if the process is at state ξ at time $\hat{t}$ and the next jump happens at time $\hat{t}+t$, then t is determined by a Poisson process with intensity $\lambda(\phi(t,\xi))$, see [8].)

If a systems behavior is completely captured by a CFSJS, i.e., if there are no forced transitions, then the CFSJS completely determines the stochastic executions of the system. By determining the TMS of a CFSJS, we indirectly determine the stochastic process/executions of the CFSJS.

DEFINITION 3.3 *The TMS (Transition Mechanism Structure) of a CFSJS* $(E, \xi_0, \phi, \lambda, Q)$ *is defined as* $(E, \xi_0, \phi, (T, Q))$, *where, for* $\xi \in E$, *the* survivor function $\Psi_{T(\xi)}(t)$ *of* $T(\xi)$, *is defined as*

$$\Psi_{T(\xi)}(t) = \mathrm{e}^{-\int_0^t \lambda(\phi(s,\xi))\mathrm{ds}}. \tag{3.1}$$

The survivor function $\Psi_{T(\xi)}(t)$ *is by definition equal to* $\mathrm{P}(\mathrm{T}(\xi) > \mathrm{t})$ *and thus expresses the probability that* $T(\xi)$ *"survives" be the time instant* t, *or, in other words, expresses the probability that a transition does not occur until time* t.

We show that (3.1) indeed expresses that if at time zero, i.e., the time of the previous transition, the process is at state $\hat{\xi}$ and at time t the process is at state ξ, then, given that no jump occurred in the interval $[0,t]$, the probability that a spontaneous transition occurs in the interval $[t, t+\Delta t]$ equals $\lambda(\xi)\Delta t + o(\Delta t)$.

$$\mathrm{P}(T(\hat{\xi}) \in [t, t+\Delta t] \mid T(\hat{\xi}) > t) = \frac{\Psi_{T(\hat{\xi})}(t) - \Psi_{T(\hat{\xi})}(t+\Delta t)}{\Psi_{T(\hat{\xi})}(t)} =$$

$$1-\mathrm{e}^{-\int_0^{t+\Delta t}\lambda(\phi(s,\hat{\xi}))\mathrm{d}s+\int_0^t\lambda(\phi(s,\hat{\xi}))\mathrm{d}s}=1-\mathrm{e}^{-\int_0^{\Delta t}\lambda(\phi(s+t,\hat{\xi}))\mathrm{d}s},$$

which, after Taylor expansion, equals $\lambda(\xi)\Delta t+o(\Delta t)$.

3.2.2.1 Memoryless Property of the Jump Times

Let X be a TMS with transition mechanism (T,Q) and flow map ϕ. We can execute X as described in Section 3.2.1. Suppose that during such an execution we lose at some time $\hat{t}$, while the process is at state $\xi_{\hat{t}}$, the information of the last drawn sample from T. Can we now continue the execution path from $\xi_{\hat{t}}$ in a correct way, or do we have to start a new execution path from ξ_0? Let t_l denote the time of the previous transition (before $\hat{t}$) and let ξ_{t_l} be the state of the execution path at time t_l, which is the target state of the transition at time t_l. If t_l and ξ_{t_l} are known, then it is correct to continue the stochastic execution from $\hat{t}$ as follows: draw a sample $\tilde{t}$ from $\tilde{T}$, where $\tilde{T}$ is a random variable such that

$$\mathrm{P}(\tilde{T}>t)=\mathrm{P}(T(\xi_{t_l})>\hat{t}-t_l+t|T(\xi_{t_l})>\hat{t}-t_l).$$

Now let the execution path flow from state $\xi_{\hat{t}}$ to state $\phi(\tilde{t},\xi_{\hat{t}})$ and switch at state $\phi(\tilde{t},\xi_{\hat{t}})$ according to the measure $Q(\phi(\tilde{t},\xi_{\hat{t}}))$. From the new state we again draw a new sample from the transition mechanism, etc.

Now we show that we can determine $\mathrm{P}(\tilde{T}>t)$ without knowing t_l and ξ_{t_l}, and consequently we can conclude that we can correctly continue the execution path from state $\hat{\xi}$ without having any information except that the process is at state $\hat{\xi}$.

The transition mechanism (T,Q) of a CFSJS has a special structure expressed by the following property.

$$\mathrm{P}(T(\xi)>\hat{t}+t|T(\xi)>\hat{t})=\mathrm{P}(T(\phi(\hat{t},\xi))>t). \tag{3.2}$$

This property expresses the fact that the jump times are memoryless. Because of this property we have

$$\mathrm{P}(\tilde{T}>t)=P(T(\hat{\xi})>t).$$

Thus, if during the execution of a CFSJS, we loose the information of the last drawn sample before time $\hat{t}$ and state $\xi_{\hat{t}}$, then because of property (3.2), we can continue the execution by considering $\xi_{\hat{t}}$ as a state right after some switch. This means that we draw a sample $\tilde{t}$ from $T(\xi_{\hat{t}})$, followed by drawing a sample from $Q(\phi(\tilde{t},\xi_{\hat{t}}))$, etc. As we will see in the next section, this observation makes it possible that the behavior of a system that consists of two CFSJSs executed at the same time can be expressed as a single CFSJS.

3.2.2.2 Representing Two Parallel CFSJSs as a Single CFSJS

Let $X=(E_X,\xi_{X,0},\phi_X,\lambda_X,Q_X)$ and $Y=(E_Y,\xi_{Y,0},\phi_Y,\lambda_Y,Q_Y)$ be two CFSJSs. Assume that at time t_0 both the processes X and Y are started. Let $\xi_X:\mathbb{R}_+\to E_X$ be an execution path generated by the TMS of X and let $\xi_Y:\mathbb{R}_+\to E_Y$ be an execution path generated by the TMS of Y. Then we call $\xi:\mathbb{R}_+\to E_X\times E_Y$, where $\xi(t)=(\xi_X(t),\xi_Y(t))$, an execution path of the simultaneous execution of X and Y

on the combined state space $E_X \times E_Y$. We show that these combined execution paths can be generated by the TMS of a single CFSJS denoted as $X|Y$. In Section 3.3 we need this result when two components (i.e., two CPDPs) that are executed in parallel, need to be represented as a single component (i.e., as a single CPDP). The state space of $X|Y$ is the product space $E_X \times E_Y$.

Suppose that after the start at t_0, X switches for the first time at t_1 at state $\xi_{X,1}$ and Y does not switch before t_1. Then at t_1, the state of X is reset by the measure $Q(\xi_{X,1})$. Let $\xi_{Y,1}$ be the state of Y at time t_1. The state of Y is not reset at time t_1, but from Section 3.2.2.1 we know that the stochastic behavior of Y will not change if we reset the state of Y at time t_1 with probability one to the same state. (Equivalently, the reset measure is the Dirac measure concentrated at the current state.) Then for the execution of Y, the state does not change at time t_1, but a new sample is drawn from the transition mechanism at state $\xi_{Y,1}$, which does not influence the stochastic execution according to Section 3.2.2.1.

We define a transition mechanism (T,Q) on the state space $E_X \times E_Y$ such that generating an execution path of (T,Q) is equal to generating a combined execution path for X and Y as described above.

Let for all $(\xi_X,\xi_Y) \in E_X \times E_Y$ the random variable $T(\xi_X,\xi_Y)$ be equal to

$$\min\{T_X(\xi_X), T_Y(\xi_Y)\}.$$

Then T determines the jump time of either X or Y. It can be seen that the survivor function of $T(\xi_X,\xi_Y)$ equals

$$\Psi_{T(\xi_X,\xi_Y)}(t) = \mathrm{e}^{-\int_0^t (\lambda_X(\phi(s,\xi_X)+\lambda_Y(\phi(s,\xi_Y))\mathrm{ds}}. \tag{3.3}$$

If a switch happens at combined state (ξ_X,ξ_Y), then it can be seen that the probability that this switch is a switch of X is equal to $\frac{\lambda_X(\xi_X)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)}$ and the probability that this switch is a switch of Y is equal to $\frac{\lambda_Y(\xi_Y)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)}$.

If X switches at state (ξ_X,ξ_Y), the reset measure $Q_X(\xi_X) \times Id(\xi_Y)$ is used and if Y switches at state (ξ_X,ξ_Y), the reset measure $Id(\xi_X) \times Q_Y(\xi_Y)$ is used. Then we get for Q

$$Q(\xi_X,\xi_Y) = \frac{\lambda_X(\xi_X)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)} Q_X(\xi_X) \times Id(\xi_Y) + \frac{\lambda_Y(\xi_Y)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)} Id(\xi_X) \times Q_Y(\xi_Y). \tag{3.4}$$

Define CFSJS $X|Y$ as $(E_X \times E_Y, (\xi_{X,0},\xi_{Y,0}), (\phi_X,\phi_Y), \lambda, Q)$, where

$$\lambda(\xi_X,\xi_Y) = \lambda_X(\xi_X) + \lambda_Y(\xi_Y).$$

The TMS of $X|Y$ equals (T,Q) and therefore CFSJS $X|Y$ generates the same execution paths (with the same probabilities) as the combination of execution paths of X and Y.

If $|$ denotes the operator that maps two CFSJSs to the combined CFSJS, then it can be seen that $|$ is associative and the combination of X,Y and Z can be expressed as either $(X|Y)|Z$ or $X|(Y|Z)$.

3.2.3 Forced Transition Structure (FTS)

If a system has forced transitions, then the behavior of the system concerning these forced transitions can be captured as an FTS.

DEFINITION 3.4 *An FTS is a tuple $(E, \mathscr{T})$, where the state space E is a Borel space, and $\mathscr{T} \subset E \times \mathsf{Prob}(E)$ is the transition relation. For each $\xi \in E$, there exists at most one measure m such that $(\xi, \mathsf{m}) \in \mathscr{T}$. If a state ξ is such that there exists an m such that $(\xi, \mathsf{m}) \in \mathscr{T}$, then we call ξ an* enabled state *of the FTS.*

If a system X has corresponding FTS $(E, \mathscr{T})$, then if $(\xi, m) \in \mathscr{T}$ means that if X reaches state ξ at some time t, then X is forced to switch at this state and the target state of the switch is determined by measure m.

3.2.4 CFSJS Combined with FTS

The behavior of a PDP and, under certain conditions, the behavior of a CPDP can be captured as a combination of a CFSJS and an FTS. In fact, this combination means that the process runs as the CFSJS until an enabled state of the FTS is reached. Then the forced transition is executed, and the CFSJS execution continues from the state right after the forced transition. We now show how this combination of CFSJS and FTS behaves in terms of TMS.

Let (X_C, X_F), where $X_C = (E, \xi_0, \phi, \lambda, Q)$ is an CFSJS and $X_F = (E, \mathscr{T})$ is an FTS, be the combined semantics of a system X with state space E. For each $\xi \in E$ we define $t_*(\xi)$ as

$$t_*(\xi) := \begin{cases} \inf\{t \geq 0 | \phi(t, \xi) \text{ is an enabled state of } X_F\} \\ \infty \text{ if no such time exists.} \end{cases}$$

Thus, $t_*(\xi)$ is the maximum time before a jump surely occurs from the moment that the process is in state ξ. Either a jump occurs before time $t_*(\xi)$ because of the CFSJS part or a forced jump happens at time $t_*(\xi)$.

The transition mechanism structure of (X_C, X_F) is then equal to $(E, \xi_0, \phi, (T, \tilde{Q}))$, where $\tilde{Q}(\xi)$ equals $Q(\xi)$ if ξ is not an enabled state of X_F and $\tilde{Q}(\xi)$ equals m if ξ is an enabled state of X_F, where m is such that $(\xi, m) \in \mathscr{T}$. The survivor function of T (whose definition we take from [4]) equals

$$\Psi_{T(\xi)}(t) = I_{(t < t_*(\xi))} \mathrm{e}^{-\int_0^t \lambda(\phi(s,\xi))\mathrm{d}s}. \tag{3.5}$$

In [11] it is described how the semantics of a PDP or a CPDP (that has no non-determinism) can be given as a CFSJS together with an FTS. In this chapter the CPDPs do have non-determinism. Then the semantics can not be expressed by a CFSJS together with an FTS, but instead will be expressed by a CFSJS together with an NTS.

3.2.5 Non-deterministic Transition System (NTS)

Besides spontaneous and forced transitions, we can also distinguish *non-deterministic transitions* . We call a ξ-enabled transition *non-deterministic* if, when a process reaches state ξ, the process has the potential to execute the transition but it is not forced to execute the transition. In other words, it is not determined whether the process should execute the transition. Forced transitions are clearly not non-deterministic since the process has no choice but is forced to execute a ξ-enabled forced transition when its state reaches ξ. Execution of spontaneous transitions is determined by random variables, spontaneous transitions are not non-deterministic therefore. We now define the semantical model NTS (Non-deterministic Transition Structure), whose transitions are non-deterministic.

DEFINITION 3.5 *An NTS is a tuple $(E,\Sigma,\mathscr{T})$. The state space E is a Borel space. Σ is a set of labels. $\mathscr{T} \subset E \times \Sigma \times \mathit{Prob}(E)$ is the transition relation.*

(At this level of generality Σ is an arbitrary set, although in most situations it will actually be a finite set.) We write $\xi \xrightarrow{\sigma} m$ for $(\xi,\sigma,m) \in \mathscr{T}$. If a system X has corresponding NTS $(E,\Sigma,\mathscr{T})$, then $\xi \xrightarrow{\sigma} m$ has the meaning that if X reaches state ξ at some time t, then X has the possibility/potential to jump at this point on action σ and the target state of the jump is determined by measure m. Whether the transition is really executed at state ξ is not-determined. If there are multiple ξ-enabled transitions, then the process has, at state ξ, the potential to execute either one of them or to execute none of them. In the concurrent processes literature non-determinism is often used in a stricter sense than here. A ξ-enabled transition with label σ is then called non-deterministic if there is another ξ-enabled transition with the same label σ.

Actions σ are used for interaction between systems. In Section 3.3 we show how this interaction is established in the context of CPDPs.

If a system has non-deterministic transitions, then the system is open in the sense that its behavior can be influenced by other systems. Therefore, a system with non-deterministic transitions, if the non-determinism is not resolved by for example a scheduler, can not be stochastically executed. This means that we cannot determine the TMS of such a system.

We now give two simple examples that show how the behavior of a stochastic system can be captured in terms of CFSJS, FTS, and NTS. In the first example the behavior is captured as a CFSJS together with an FTS, in the second example the behavior is captured as a CFSJS together with an NTS.

EXAMPLE 3.1 The state x of system X takes value in $\mathbb{R}$ and evolves continuously as described by the ordinary differential equation $\dot{x} = 1$. The initial state is $x_0 = 0$. A spontaneous transition may happen and the time of such a transition is exponentially distributed with parameter λ. The target state of such a transition is chosen with uniform distribution in $[0,1]$. If x

reaches the value 1, a transition is forced to happen.

The behavior of X can be captured as a CFSJS together with an FTS. The CFSJS equals $(\mathbb{R}, 0, \phi, \lambda, Q)$, where $\phi(t,x) = x+t$ for all states x and Q equals $U[0,1]$, i.e., the uniform distribution on $[0,1]$, for all states x. The FTS equals $(\mathbb{R}, \mathscr{T}_F)$, with $\mathscr{T}_F = \{(1, U[0,1])\}$.

The TMS that corresponds to the combination of this CFSJS and FTS equals $(\mathbb{R}, 0, \phi, (T, Q))$, where for all states $x < 1$

$$\mathrm{P}(T(x) > t) = I_{(t<1-x)} \mathrm{e}^{-\lambda t}.$$

■

EXAMPLE 3.2 The state x of system X takes value in $\mathbb{R}$ and evolves continuously as described by the ordinary differential equation $\dot{x} = 1$. The initial state $x_0 = 0$. A spontaneous transition may happen and the time of such a transition is exponentially distributed with parameter λ. The target state of such a transition is chosen with uniform distribution in $[0,1]$. If $x \geq 1$, a transition is allowed but not forced to happen. The action of the transition is τ.

The behavior of X can be captured as a CFSJS together with an NTS. The CFSJS equals the CFSJS of Example 3.1. The NTS equals $(\mathbb{R}, \{\tau\}, \mathscr{T}_N)$, with $\mathscr{T}_N = \{(x, \tau, U[0,1]) | x \geq 1\}$. Note that because of the presence of nondeterminism, the behavior of the CFSJS together with the NTS cannot be captured as a TMS. ■

3.3 Communicating PDPs

With PDPs we can model a broad class of stochastic systems. However, because PDPs do not allow modelling in a compositional way, the modelling process becomes nearly impossible if systems have a (very) complex structure. In this section we introduce the automaton model CPDP (communicating PDP), which makes it possible to model PDP-type systems in a compositional way.

3.3.1 Definition of the CPDP Model

DEFINITION 3.6 *A CPDP is a tuple* $(L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$, *where*

- L *is a set of locations.*
- V *is a set of state variables. With* $d(v)$ *for* $v \in V$ *we denote the dimension of variable* v. $v \in V$ *takes its values in* $\mathbb{R}^{d(v)}$.

- W *is a set of output variables. With* $d(w)$ *for* $w \in W$ *we denote the dimension of the variable* w. $w \in W$ *takes its values in* $\mathbb{R}^{d(w)}$.
- $\nu : L \to 2^V$ *maps each location to a subset of* V, *which is the set of state variables of the corresponding location. We call the valuation space of the variables of* $\nu(l)$ *the state space of location* l.
- $\omega : L \to 2^W$ *maps each location to a subset of* W, *which is the set of output variables of the corresponding location. We call the valuation space of the variables of* $\omega(l)$ *the output space of location* l.
- F *assigns to each location* l *and each* $v \in \nu(l)$ *a mapping from* $\mathbb{R}^{d(v)}$ *to* $\mathbb{R}^{d(v)}$, *i.e.,* $F(l,v) : \mathbb{R}^{d(v)} \to \mathbb{R}^{d(v)}$. $F(l,v)$ *is the vector field that determines the evolution of* v *for location* l *(i.e.,* $\dot{v} = F(l,v)$ *for location* l*).*
- G *assigns to each location* l *and each* $w \in \omega(l)$ *a mapping from* $\mathbb{R}^{d(v_1)+\cdots+d(v_m)}$ *to* $\mathbb{R}^{d(w)}$, *where* v_1 *till* v_m *are the state variables of location* l. $G(l,w)$ *determines the output equation of* w *for location* l *(i.e.,* $w = G(l,w)$*).*
- Σ *is the set of communication labels.* $\overline{\Sigma}$ *denotes the "passive" mirror of* Σ *and is defined as* $\overline{\Sigma} = \{\overline{a} | a \in \Sigma\}$.
- $\mathscr{A}$ *is a set of active transitions and consists of five-tuples* (l, a, l', G, R), *denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with communication label* $a \in \Sigma$, *guard* G *and reset map* R. G *is a subset of the* valuation space *of* l, *which notion is introduced in Section 3.3.1.1. The reset map* R *assigns to each point in* G *for each variable* $v \in \nu(l')$ *a probability measure on* $\mathbb{R}^{d(v)}$, *i.e.,* $R(g,v) \in \mathit{Prob}(\mathbb{R}^{d(v)})$ *for all* $g \in G$ *and all* $v \in \nu(l')$.
- $\mathscr{P}$ *is a set of passive transitions of the form* $(l, \overline{a}, l', R)$. R *is defined on the valuation space of* l *as the* R *of an active transition is defined on the guard space.*
- $\mathscr{S}$ *is a set of spontaneous transitions and consists of four-tuples* (l, λ, l', R), *denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with jump-rate* λ *and reset map* R. *The jump rate* λ *(i.e., the Poisson rate of the Poisson process of the spontaneous transition) is a mapping from the state space of* l *to* $\mathbb{R}_+$. R *is defined on the state space of* l *as it is done for passive transitions.*

The graphical notation of a CPDP is as follows. The locations are pictured as circles (see for example Figure 3.1). The differential and output equations that belong to a location l are written inside the circle of l. A transition from location l to location l' is drawn as an arrow from l to l'. The communication label from Σ, the reset map and the guard of the transition are written above (or next to) the arrow. A passive transition is pictured as an active transition, except that the label is now from

$\bar{\Sigma}$ and there is no guard. A spontaneous transition is pictured as an arrow with a little box in the middle. This notation is chosen in line with the notation used for IMCs (Interactive Markov Chains, cf. [6]. The jump rate and reset map of a spontaneous transition are written above or next to this little box.

EXAMPLE 3.3 In Figure 3.1 we see the CPDP X, which models a flying aircraft. The initial location of X is l_1. The initial state x_0 in l_1 represents the position and velocity of the aircraft at initial time t_0. Since the aircraft flies in three dimensional space, the state space of variable x is $\mathbb{R}^6$. Location l_1 represents a flying mode. This means that in l_1 the aircraft is somewhere up in the sky and not at the ground. The dynamics of the aircraft in flying mode l_1 is determined by the vector field f_1. In this model we do not discriminate between state and output, therefore in all locations the output is chosen to be a copy of the state, i.e., $y = x$. Location l_2 represents a non-nominal flying mode. In this mode the aircraft is flying while there is a defect. (This mode represents for example that the navigation system is not working properly.) In this non-nominal flying mode the dynamics is determined by vector field f_2. In location l_1, the time till the defect occurs, is exponentially distributed with parameter λ_1. This is expressed by the spontaneous transition from l_1 to l_2 with jump rate λ_1. When this transition is executed at some state $(l_1, \{x = x_1\})$, i.e., when the defect occurs at this state, the state of l_2 is reset by $R_1(\{x = x_1\})$, which is a probability measure on $\mathbb{R}^6$, the state space of x. Since the position and velocity of the aircraft do not change when a switch from l_1 to l_2 happens, $R_1(\{x = x_1\})$ equals the Dirac measure at x_1, which assigns probability one to the singleton set $\{x = x_1\}$. We call R_1 an identity reset map. In the non-nominal mode l_2, repair activities are undertaken. The time till the defect is repaired is exponentially distributed with parameter λ_2. This is expressed by the spontaneous transition from l_2 to l_1 which has reset map R_2, which also equals the identity reset map.

If the aircraft approaches the airport, the flying mode will change to a landing mode where the aircraft prepares for landing. These modes are represented by l_3 and l_4, where l_3 represents nominal landing and l_4 represents landing while there still is an unrepaired defect. l_1 switches to l_3 as soon as the altitude drops below a certain level h. Let x_1 till x_6 be the components of variable x and let x_3 denote the altitude of the aircraft. The guard G_1 of the transition from l_1 to l_3 is equal to $\{x = (x_1, x_2, \cdots, x_6) | x_3 \leq h\}$. This expresses that this transition is allowed to be executed as soon as $x \in G_1$, i.e., as soon as $x_3 \leq h$. In fact, we want the transition executed at the first time instant where $x \in G_1$. Later we will see that we can express this by assuming maximal progress, which means, roughly said, that active transitions should be executed as soon as the guard is satisfied. When this active transition is executed at some time t_2, then the identity reset map R_3 (also an identity reset map) resets the state and the discrete event *land* is executed at time t_2. The discrete event (and its transition) is executed instantaneously, i.e., does

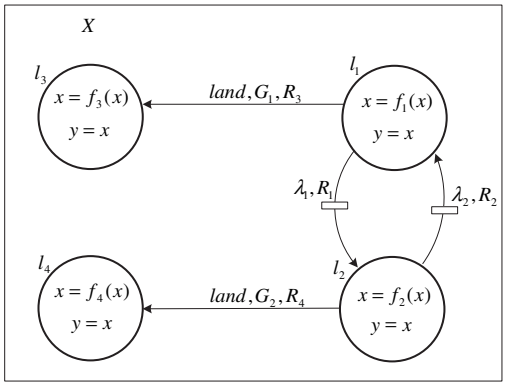

FIGURE 3.1: Landing aircraft modelled as CPDP.

not consume time. Later we will see that these discrete events can be used for communication between CPDPs. In location l_3 the continuous dynamics (expressing the landing phase) is determined by vector field f_3. In the same way the transition from l_2 to l_4 with identity reset map R_4 represents the transition from flying mode to landing mode, but now while there is a defect present.

In this example the jump rates λ_1 and λ_2 are constants, expressing exponentially distributed times. However, the CPDP model also allows λ_1 and λ_2 to depend on the state, and therefore indirectly depend on the time. If for example λ_1 depends on x_3 such that $\lambda_1(x_1, x_2, \cdots, x_6) > \lambda_1(x'_1, x'_2, \cdots, x'_6)$ when $x_3 > x'_3$, then this would express that the rate of switching is larger for great altitudes, i.e., at great altitudes it is more likely that a defect occurs than at small altitudes. ■

One feature of the CPDP model, the passive transition, is not explained in the above example. The meaning of passive transitions becomes apparent in the context of communication between multiple CPDPs and is explained and illustrated in Section 3.3.3.

3.3.1.1 The State and Output Space of a CPDP

The state of a CPDP is hybrid; it consists of a location on the one hand and of values for the continuous variables on the other hand.

DEFINITION 3.7 *Let X be a CPDP with location set L, set of state variables V, set of output variables W, and for each $l \in L$ the sets of active state and output variables $\nu(l) \subset V$ and $\omega(l) \subset W$. The (hybrid) state space of X is*

defined as

$$\{(l, val) \mid l \in L, val \in vs(l)\},$$

where $vs(l)$ *denotes the valuation space of location* l*, which in case* $\nu(l) = \{\nu_1, \nu_2, \cdots, \nu_n\}$*, is given as*

$$\mathbb{R}^{d(\nu_1)} \times \cdots \times \mathbb{R}^{d(\nu_n)}$$

and in case $\nu(l) = \emptyset$ *is defined as* $\{0\}$*. The output space of* X *is defined as*

$$\{(l, val) \mid l \in L, val \in os(l)\},$$

where $os(l)$ *denotes the output space of location* l*, which in case* $\omega(l) = \{w_1, \cdots, w_m\}$ *is defined as* $\mathbb{R}^{d(w_1)} \times \cdots \times \mathbb{R}^{d(w_m)}$ *and in case* $\omega(l) = \emptyset$ *is given as* $\{0\}$*. The output value* 0 *is used for CPDP states where no output is defined.*

EXAMPLE 3.4 The state space of CPDP X of Figure 3.1 equals

$$\{(l, val) \mid l \in \{l_1, l_2, l_3, l_4\}, val \in \mathbb{R}^6\}.$$

The output space of X equals $\mathbb{R}^6$. ■

REMARK 3.1 We allow that for CPDP locations l we have $\nu(l) = \emptyset$, i.e., we allow that locations do not have continuous variables attached to it. We call these locations *empty locations*. According to Definition 3.7, the valuation space of an empty location l equals $\{0\}$ and therefore l contributes one state to the state space of the CPDP; the state $(l, 0)$. The guard of an active transition α with origin location l is then equal to $\{0\}$, which means that at an empty location, active transitions are always enabled. Spontaneous transitions at empty locations assign a constant λ to the single state of the valuation space. This means that the jump time of such a spontaneous transition is exponentially distributed with parameter λ. A reset map of a transition whose target location is an empty location assigns probability one to the single state 0 of the valuation space of that empty location.

We also allow for CPDP locations such that $\omega(l) = \emptyset$. This means that no output dynamics is defined for such locations. The output at states of these locations will later be defined as 0. ■

Let $\alpha = (l, a, l', G, R)$ be an active transition. Then we we define the mappings *oloc* (origin location), *lab* (label), *tloc* (target location), *guard*, and *rmap* (reset map) as: $oloc(\alpha) = l$, $lab(\alpha) = a$, $tloc(\alpha) = l'$, $guard(\alpha) = G$, $rmap(\alpha) = R$. These mappings, except for *guard*, are also defined in the same way for passive transitions. *oloc*, *tloc*, and *rmap* are also defined in the same way for spontaneous transitions. Furthermore, let $\xi = (l, val)$ be some hybrid state. Then $loc(\xi) := l$ maps ξ to its discrete part, and $val(\xi) := val$ maps ξ to its continuous part.

We define the flow map $\phi : \mathbb{R}_+ \times E \to E$ of a CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$ with state space E. $\phi(t, \xi)$ is determined by the differential equations

$$\dot{x}_1 = F(l, x_1), \;\; \dot{x}_2 = F(l, x_2), \cdots \;\; \dot{x}_n = F(l, x_n), \tag{3.6}$$

where $l = loc(\xi)$ and $\nu(l) = \{x_1, x_2, \cdots, x_n\}$. Thus, for $t \geq 0$ and $\xi = (l, \{x_1 = r_1, \cdots, x_n = r_n\}) \in E$, $\phi(t, \xi)$ equals $\xi' = (l, \{x_1 = r_1', \cdots, x_n = r_n'\})$, where $r_1, \cdots, r_n$ are the solutions of (3.6) for $x_1 \cdots x_n$ at time t where $x_1 \cdots x_n$ at time zero have values $r_1 \cdots r_n$. For empty locations l we define the flow map as $\phi(t, (l, 0)) := (l, 0)$ for all $t \geq 0$.

3.3.2 Semantics of CPDPs

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$ be a CPDP with state space E, flow map ϕ and initial state ξ_0. We define the semantics of X as the combination of a CFSJS and an NTS. Let X_C denote the CFSJS of X and let X_N denote the NTS of X. Then, $X_C = (E, \xi_0, \phi, \lambda, Q)$, where

$$\lambda(l, val) := \sum_{\alpha \in \mathscr{S}_{l\rightarrow}} \lambda_\alpha(val),$$

where $\mathscr{S}_{l\rightarrow}$ denotes the set of all spontaneous transitions with origin location l, and for all $A \in \mathscr{B}(E)$,

$$Q(l, val)(A) = \sum_{\alpha \in \mathscr{S}_{l\rightarrow}} \frac{\lambda_\alpha(val)}{\lambda(l, val)} R_\alpha(A)$$

and $X_N = (E, \Sigma \cup \overline{\Sigma}, \mathscr{T})$, where

- $(\xi, a, m) \in \mathscr{T}$ if and only if there exists an $\alpha \in \mathscr{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$, $val(\xi) \in guard(\alpha)$, and $rmap(\alpha)(\xi) = m$.
- $(\xi, \bar{a}, m) \in \mathscr{T}$ if and only if there exists an $\alpha \in \mathscr{P}$ such that $lab(\alpha) = \bar{a}$, $oloc(\alpha) = loc(\xi)$, and $rmap(\alpha)(\xi) = m$.

Note that the CFSJS defined above expresses correctly that in each location there is a "race" between the spontaneous transitions enabled at that location just as the "race" between the spontaneous transitions of two CFSJSs that are running in parallel as described in Section 3.2.2.2. That the λ and Q of the CFSJS correctly express this "race" can, mutatis mutandis, also be found in Section 3.2.2.2.

EXAMPLE 3.5 The semantics of CPDP X of Figure 3.1 with state space E, flow map ϕ, and initial state ξ_0 is as follows. $X_C = (E, \xi_0, \phi, \lambda, Q)$, where for $\xi = (l, val) \in E$,

$$\lambda(\xi) = \begin{cases} \lambda_1 \text{ if } l = l_1, \\ \lambda_2 \text{ if } l = l_2, \\ 0 \;\; \text{if } l \in \{l_3, l_4\} \end{cases}$$

and for all $B \in \mathscr{B}(E)$

$$Q(\xi)(B) = \begin{cases} R_1(\xi)(B) \text{ if } l = l_1, \\ R_2(\xi)(B) \text{ if } l = l_2, \\ \text{undefined if } l \in \{l_3, l_4\}. \end{cases}$$

$X_N = (E, \Sigma \cup \overline{\Sigma}, \mathscr{T})$, where

$$\mathscr{T} = \{(\xi, land, m) | \xi \in G_1, m = R_3(\xi)\} \cup \{(\xi, land, m) | \xi \in G_2, m = R_4(\xi)\}.$$

■

We cannot give the execution of a CPDP X with active/passive transitions in terms of a transition mechanism system, because it is not determined when transitions from $\mathscr{T}$ are executed. However, we can describe the execution of a general CPDP $X = (L, V, v, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$ as follows. Let $X_C = (E, \xi_0, \phi, \lambda, Q)$ be the CFSJS of X and let $X_N = (E, \Sigma, \mathscr{T})$ be the NTS of X. Then, the execution of X can be seen as the execution of X_C while at every state ξ the process has the potential to switch with measure m if $(\xi, \sigma, m) \in \mathscr{T}$ for some $\sigma \in \Sigma \cup \overline{\Sigma}$.

3.3.2.1 Output Semantics

The CFSJS and NTS do not capture the complete behavior of a CPDP. At every state $\xi \in E$ of a CPDP, the CPDP also has an output value which lies in its output space E_O and which is determined by the output mapping $G : E \to E_O$. Therefore we could say that the complete behavior of a CPDP is captured by its CFSJS, its NTS and its output mapping.

3.3.3 Composition of CPDPs

Now we will define how CPDPs can be composed. The composition operator that we use, which can be seen as a generalization of the composition operator for Interactive Markov Chains from [6], will be denoted by $|^P_A|$. We do not have the space here to explain the full interaction-potential of this operator. We now give informally the main features of $|^P_A|$. For a full explanation we refer to [11] or [13].

First we discuss the distinction between active and passive transitions. Active transitions can be executed independently from passive transitions. Passive transitions can only be executed when they are triggered by active transitions in another component. If CPDPs X and Y are composed, and CPDP component X executes an a-transition, then this transition will trigger (if available) a passive $\overline{a}$-transition in component Y. We could also say that the $\overline{a}$-transition of Y observes the a-transition of X.

In $|^P_A|$, A, which is a subset of Σ, is the set of active events that should synchronize. This means that if CPDPs X and Y are composed through operator $|^P_A|$, and if $a \in A$, then an a-transition of X can be executed only if at the same time an a-transition of Y is executed (and vice versa). If CPDPs X and Y are composed through $|^P_A|$ and $a \in A$,

then an a-transition of X cannot trigger a $\bar{a}$-transition of Y. In other words, the events from A are used for active-active synchronization and the events from $\Sigma\backslash A$ are used for active-passive synchronization.

P, which is a subset of $\bar{\Sigma}$, is the set of all passive events that should synchronize. Briefly said, an event $\bar{a} \in P$ is such that multiple $\bar{a}$-transitions can be triggered by a single a-transition. This means that an a-transition of X can trigger a passive $\bar{a}$-transition in all of the other components Y, Z, etc. If $\bar{a} \notin P$, then an a-transition of X can trigger a $\bar{a}$-transition in only one of the other components Y, Z, etc.

In the definition of composition of CPDPs, communication is expressed through synchronization of transitions and not through the sharing of continuous variables. Therefore, each component should have its own continuous variables, i.e., the intersection of the sets of continuous variables of the two components should be empty. If this is not the case, then the two components are not compatible for composition.

We now give the definition of composition of CPDPs. Afterwards we briefly explain the composition rules and therewith explain how interaction is expressed in this definition of composition.

DEFINITION 3.8 *Let* $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathscr{A}_X, \mathscr{P}_X, \mathscr{S}_X)$ *and* $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathscr{A}_Y, \mathscr{P}_Y, \mathscr{S}_Y)$ *be two CPDPs such that* $V_X \cap V_Y = W_X \cap W_Y = \emptyset$. *Then* $X|^P_A|Y$ *is defined as the CPDP* $(L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$, *where*

- $L = \{l_1|^P_A|l_2 \mid l_1 \in L_X, l_2 \in L_Y\}$.
- $V = V_X \cup V_Y$, $W = W_X \cup W_Y$.
- $\nu(l_1|^P_A|l_2) = \nu(l_1) \cup \nu(l_2)$, $\omega(l_1|^P_A|l_2) = \omega(l_1) \cup \omega(l_2)$.
- $F(l_1|^P_A|l_2, \nu)$ *equals* $F_X(l_1, \nu)$ *if* $\nu \in \nu_X(l_1)$ *and equals* $F_Y(l_2, \nu)$ *if* $\nu \in \nu_Y(l_2)$.
- $G(l_1|^P_A|l_2, w)$ *equals* $G_X(l_1, w)$ *if* $w \in \omega_X(l_1)$ *and equals* $G_Y(l_2, w)$ *if* $w \in \omega_Y(l_2)$.
- $\mathscr{A}$, $\mathscr{P}$ *and* $\mathscr{S}$ *are the least relations satisfying the rules r1, r2, r2′, r3, r3′, r4, r4′, r5, r6, r6′, r7 and r7′, defined below*

$$r1.\ \frac{l_1 \xrightarrow{a,G_1,R_1} l_1', l_2 \xrightarrow{a,G_2,R_2} l_2'}{l_1|^P_A|l_2 \xrightarrow{a,G_1 \times G_2, R_1 \times R_2} l_1'|^P_A|l_2'} (a \in A).$$

$$r2.\ \frac{l_1 \xrightarrow{a,G_1,R_1} l_1', l_2 \xrightarrow{\bar{a},R_2} l_2'}{l_1|^P_A|l_2 \xrightarrow{a,G_1 \times vs(l_2), R_1 \times R_2} l_1'|^P_A|l_2'} (a \notin A).$$

$$r2'.\ \frac{l_1 \xrightarrow{\bar{a},R_1} l_1', l_2 \xrightarrow{a,G_2,R_2} l_2'}{l_1|^P_A|l_2 \xrightarrow{a,vs(l_1) \times G_2, R_1 \times R_2} l_1'|^P_A|l_2'} (a \notin A).$$

$$r3.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(a\notin A).$$

$$r3'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(a\notin A).$$

$$r4.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a}\notin P),\quad r4'.\frac{l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(\bar{a}\notin P)$$

$$r5.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(\bar{a}\in P).$$

$$r6.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a}\in P),\quad r6'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(\bar{a}\in P)$$

$$r7.\frac{l_1 \overset{\lambda_1,R_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{\hat{\lambda}_1,R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2},\quad r7'.\frac{l_2 \overset{\lambda_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\hat{\lambda}_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'},$$

where $\hat{\lambda}_1$ *and* $\hat{\lambda}_2$ *are defined as* $\hat{\lambda}_1(\xi_1,\xi_2) := \lambda_1(\xi_1)$ *and* $\hat{\lambda}_2(\xi_1,\xi_2) := \lambda_2(\xi_2)$.

We briefly explain how the composition ruler r1 till r7 should be interpreted. r1 says that if $a \in A$ and both $l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1'$ and $l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'$ are true, i.e., if X has an a-transition from location l_1 to location l_1' with guard G_1 and reset map R_1 and if Y has an a-transition from location l_2 to location l_2' with guard G_2 and reset map R_2, then CPDP $X|_A^P Y$ has an a-transition from location $l_1|_A^P|l_2$ to location $l_1'|_A^P|l_2'$ with guard $G_1 \times G_2$ and with reset map $R_1 \times R_2$ (where in the latter $\times$ denotes the product probability measure). Rule r1 expresses that a-transitions with $a \in A$ should synchronize. In the same way rule $r2$ expresses that for $a \notin A$, an a-transition of X synchronizes with (i.e., triggers) a $\bar{a}$-transition of Y (and vice versa with rule r2$'$). Note that $vs(l_2)$ denotes the whole state space of location l_2. Rule r3 expresses that if no $\bar{a}$-transition is present in Y, then the a-transition of X will be executed on its own. Note that Id denotes the identity reset map (i.e., the Dirac probability measure). Rules r4 and r4$'$ express that for $\bar{a} \notin P$, passive $\bar{a}$-transitions do not synchronize and are therefore executed on their own. Rule $r5$ expresses that for $\bar{a} \in P$, $\bar{a}$-transitions synchronize. Rules r6 and r6$'$ express for $\bar{a} \in P$, that if one of the components does not have a $\bar{a}$-transition, then the other component can still execute its passive $\bar{a}$-transition. This expresses (in the context of three components) that an a-transition of X can trigger a $\bar{a}$-transition of Y also when Z does not have a $\bar{a}$-transition enabled. Rules $r7$ and $r7'$ express that all spontaneous transitions remain (unchanged) in the composition.

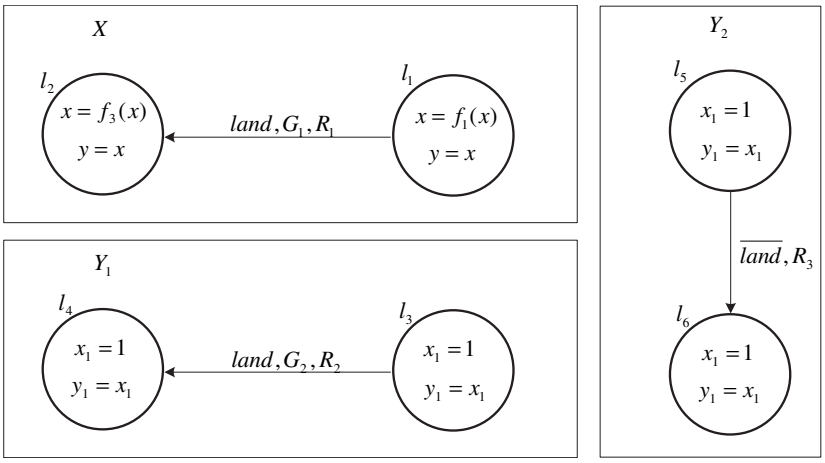

FIGURE 3.2: Landing aircraft and control tower modelled as interacting CPDPs

In [11], the composition operator $|^P_A|$ is also defined on the semantical level of NTS. Then the following result holds, which shows that a composed CPDP correctly expresses the behavior (as an NTS and CFSJS) of the interaction of the component CPDPs.

THEOREM 3.1 *Let X and Y be two CPDPs with semantics (X_C, X_N) and (Y_C, Y_N) respectively, where X_C and Y_C are CFSJSs and X_N and Y_N are NTSs. Let $(X|^P_A|Y_C, X|^P_A|Y_N)$ be the semantics of CPDP $X|^P_A|Y$. Then,*

$$(X|^P_A|Y_C, X|^P_A|Y_N) = (X_C|Y_C, X_N|^P_A|Y_N).$$

Also we have the following result.

THEOREM 3.2 *$|^P_A|$ for CPDPs is commutative for all A and P. $|^P_A|$ for CPDPs is associative if and only if for all events a we have $a \notin A \Rightarrow \overline{a} \in P$.*

EXAMPLE 3.6 The CPDP X of Figure 3.2 models a flying aircraft and has initial state $\xi_{X,0} = (l_1, x_0)$. Note that, for reasons of simplicity, the non-nominal locations from Figure 3.1 are not modelled here. CPDP Y_1 of Figure 3.2 models a control tower at an airport that can communicate with the aircraft modelled by X. Location l_3 is the location where Y_1 waits for a signal from X. The dynamics of l_3 is a clock dynamics expressing the time that Y has to wait before X sends a signal. Therefore, the initial valuation of initial

location l_3 equals $\{x_1 = 0\}$. Location l_4 is the location where Y_1 'knows' that X has send a signal. The dynamics of this location is again a clock dynamics. If Y_1 enters location l_4, then the timer is reset to zero, which means that reset map R_2 assigns for each value of x_1 in l_3 probability one to the Borel set $\{\{x_1 = 0\}\}$.

We connect X and Y_1 via composition operator $|^P_A|$, where $A = \{land\}$ (P is not relevant here). This means that the signal/label *land* is used as a shared synchronization action between X and Y_1. Then, Y_1 can execute the *land* transition only when at the same time X executes its *land* transition. We want to model that X can execute its *land* transition independently from Y. Once this happens, this transition should be communicated to Y. We can express this via the guards G_1 and G_2. G_1 equals G_1 from Example 3.3, expressing that this switch may happen as soon as the altitude of the aircraft drops under a certain level h. G_2 equals the whole valuation space of location l_3. This expresses that this transition can always be taken and consequently it expresses that this transition cannot block the *land* transition of X. We assume maximal progress. Then, the synchronized *land* transition is executed as soon as guard G_1 is satisfied. After the synchronized *land* transition, Y_1 is in location l_4. We could say that the information "X switched to landing mode", which is received by Y_1, is stored in the discrete component of the hybrid state of Y_1. In other words, discrete state l_4 of Y_2 has the meaning "X is in landing mode".

The CPDP $X|^P_A|Y_1$, which expresses the composite system of X interconnected with Y_1, is pictured in Figure 3.3. According to composition rule r1, the guard G_3 equals $G_1 \times G_2$ and the reset map R_4 equals $R_1 \times R_2$. If we look at the behavior of CPDP $X|^P_A|Y_1$ under maximal progress, then we will see that this CPDP indeed expresses the communication from X to Y_1 that we wanted to model: the initial hybrid state of $X|^P_A|Y_1$ equals $(l_1|l_3, \{x = x_0, x_1 = 0\})$. The continuous state variables x and x_1 evolve along vector fields f_1 and 1 respectively until guard G_3 is satisfied. G_3 is satisfied when the vertical position of x reaches the level h. Then, the *land* transition is executed and the state variables x and x_1 are reset by R_4, which means that x is reset by R_1 and x_1 is reset by R_2. Thus, we see that at the moment that X switches to landing mode, Y_1 switches to location l_4, which indeed establishes the communication that we intended.

Now we show how the aircraft/control tower system can be modelled by using a passive transition. For this example, we think that modelling the communication with a passive transition is more natural, since there is a clear distinction between an active system (the aircraft which sends the information of the switch) and a passive system (the control tower which receives the information). Now, the control tower is modelled as the CPDP Y_2 of Figure 3.2. Y_2 is exactly the same as Y_1, except that the active transition is replaced by a passive transition with label $\overline{land}$. This passive transition expresses that as soon as a *land* signal is received (from X), the passive transition is executed and reset map R_3 (whose action equals the action of R_2) resets the timer x_1

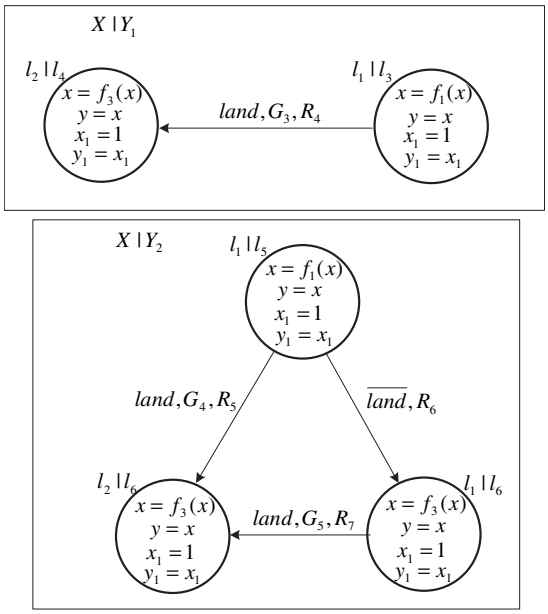

FIGURE 3.3: Composite CPDP of landing aircraft and control tower

to zero. Since *land* is not a synchronization action here, we connect X and Y_2 via $|^P_A|$, where $A = \emptyset$ (P is not relevant). The resulting CPDP $X|^P_A|Y_2$ is pictured in Figure 3.3. It can be seen from rule r2, that guard G_4 is equal to G_3 and reset map R_5 is equal to R_4. This means that as far as locations $l_1|l_3$ and $l_2|l_4$ / $l_1|l_5$ and $l_2|l_6$ are concerned, $X|^P_A|Y_1$ and $X|^P_A|Y_2$ have the same behavior. The difference between $X|^P_A|Y_1$ and $X|^P_A|Y_2$ lies in the fact that $X|^P_A|Y_2$ can switch to location $l_1|l_6$ via a passive transition, while $X|^P_A|Y_1$ cannot do this. The meaning of this switch to $l_1|l_6$ becomes apparent in a composition context with more than two components. A third component could by means of executing an active *land*-transition then trigger this passive transition. ■

EXAMPLE 3.7 In Figure 3.4, a repair shop system is modelled as the composition of CPDPs M_1,M_2 and R. CPDPs M_1 and M_2 model two machines and CPDP R models a repair shop. M_1 initially starts in location $l_{1,0}$ with a clock dynamics for its state variable x_1. M_1 can break down with state dependent jump rate λ_1. This is modelled by the spontaneous transition to $l_{1,1}$. $l_{1,1}$ is an empty location, therefore the spontaneous transition to $l_{1,1}$ has a trivial reset map that assigns probability 1 to state $(l_{1,1}, 0)$. This reset map is not pictured in Figure 3.4. From $l_{1,1}$ an active transition is executed to

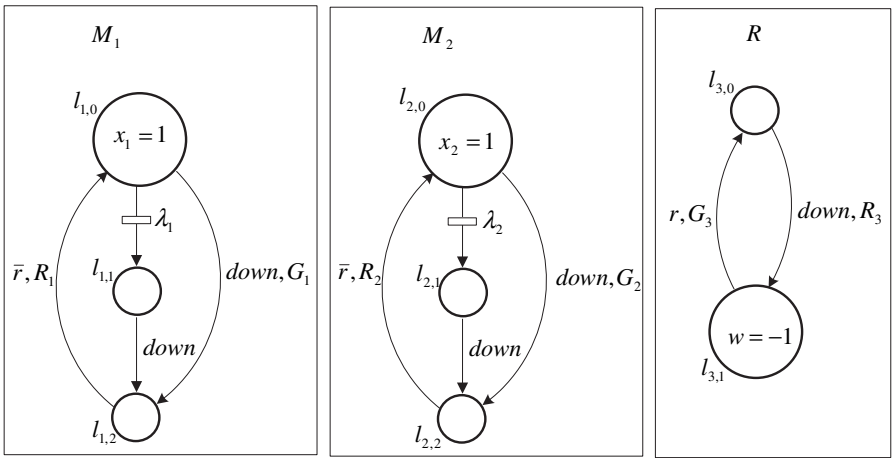

FIGURE 3.4: CPDP model of the repair shop system

$l_{1,3}$, with label *down*. We want to model that this *down*-transition is executed immediately after location $l_{1,1}$ is reached. Then, the *down* signal is executed exactly when M_1 breaks down. In the next chapter we will see that with maximal progress M_1 indeed models that no time is consumed in location $l_{1,1}$. If the machine does not break down via the spontaneous transition before s_1 time units, i.e., the maximal age of the machine, then the machine should be taken out of order to the repair shop. This is modelled by the *down*-transition from $l_{1,0}$ to $l_{1,2}$ with guard G_1 equal to $x_1 \geq s_1$. In location $l_{1,2}$, machine M_1 waits for an r signal. This is expressed by the passive $\bar{r}$-transition. This r signal will be sent by the repair shop, indicating that the machine has been repaired. Reset map R_1 resets state variable x_1 to zero, which expresses that the machine starts brand new. Machine M_2 is modelled likewise.

The repair shop CPDP R starts in empty location $l_{3,0}$. Here it waits until one of the machines needs to be repaired. The switch to repair mode $l_{3,1}$ is modelled by the active *down*-transition. We define *down* to be a synchronization action and therefore this *down*-transition synchronizes with either a *down*-transition of M_1 or a *down*-transition of M_2. Due to this synchronization, R switches to repair mode $l_{3,1}$ exactly when one of the machines need to be repaired. Reset map R_3 resets state variable w with a uniform distribution on the interval $[t_1, t_2]$, determining the time needed to repair the machine. In $l_{3,1}$, w counts down to zero, expressed by the dynamics $\dot{w} = -1$. If w has been counted down to zero, R switches back to $l_{3,0}$ where it waits for a new machine to be repaired. This switch is modelled by the active r transition. The guard G_3 of this transition equals $w = 0$. The passive $\bar{r}$ transitions of M_1 and M_2 can synchronize with this active r-transition, therefore these passive $\bar{r}$-transitions

are executed exactly when the machine is repaired.

From the description above, we get that *down* is an interleaving action between M_1 and M_2, *down* is a synchronization action between R and M_1 or M_2, and r is an interleaving action between R and M_1 or M_2. The passive action $\overline{r}$ may be chosen interleaving or synchronizing. This choice does not influence the behavior since M_1 and M_2 will never visit their locations $l_{1,2}$ and $l_{2,2}$ at the same time (i.e., joint location $(l_{1,2}, l_{2,2})$ is never reached). This gives that the total repair shop system is modelled as the CPDP

$$(M_1|^{\emptyset}_{\emptyset}|M_2)|^{\emptyset}_{down}|R.$$

■

3.3.4 Value Passing CPDPs

In this section we extend the CPDP model to *value passing CPDPs*. For CPDPs, interaction is established through synchronization of transitions. This means that the information that one CPDP can obtain concerning other CPDPs in the composition is, first, which active actions are executed and, second, at which times these actions are executed. For example, via a passive $\overline{a}$-transition, a CPDP "knows" when another CPDP executes an a-transition. With value passing CPDPs we extend the CPDP model such that it is possible for one CPDP to obtain information about the values of the output variables of other CPDPs. The moments where this information is communicated from one CPDP to another CPDP are the moments where the transitions synchronize. In other words, this communication of output information is expressed through synchronization of transitions. This idea of passing values through synchronizing transitions is called *value passing* in the literature and has been developed for example for the specification languages LOTOS [2, 9] and CSP [7].

This section is organized as follows. First we define the value passing CPDP model and we give the CFSJS/NTS semantics of a value passing CPDP. Then we define the composition operator $|^P_A|$ for value passing CPDPs. As in the case of CPDPs, we will see that the behavior of two interacting value passing CPDPs X and Y is equal to the CFSJS and NTS of the value passing CPDP $X|^P_A|Y$. Finally, we give some examples illustrating the expressiveness of value passing in the context of value passing CPDPs.

3.3.4.1 Definition and Semantics of Value Passing CPDPs

DEFINITION 3.9 *A value-passing CPDP is a tuple* $(L, V, W, \nu, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$, *where all elements except* $\mathscr{A}$ *are defined as in Definition 3.6 and where* $\mathscr{A}$ *is a finite set of active transitions that consists of six-tuples* (l, a, l', G, R, vp), *denoting a transition from location* $l \in L$ *to location* $l' \in L$ *with communication label* $a \in \Sigma$, *guard* G, *reset map* R *and value-passing element* vp. G *is a subset of the valuation space of* l. vp *can be equal to either* $!Y$, $?U$ *or*

$\emptyset$. *For the case* $!Y$, Y *is an ordered tuple* $(w_1, w_2, \cdots, w_m)$ *where* $w_i \in w(l)$ *for* $i = 1 \cdots m$. *If for a transition we have* $vp = !Y$ *for some* Y, *then this means that in a synchronization with other transitions, this transition passes the values of the variables in* Y *to the other transition. For the case* $?U$, *we have* $U \subset \mathbb{R}^n$ *for some* $n \in \mathbb{N}$. *If for a transition we have* $vp = ?U$, *then this means that in a synchronization with another transition that has* $vp = !Y$, *this transition receives the values from the variables of* Y *as long as these values are contained in the set* U. *If the other transition wants to pass values that do not lie in* U, *then the synchronization will not take place, i.e., it is blocked by* U. *If a transition is not used for value passing (either output* $!Y$ *or input* $?U$*), then this transition has* $vp = \emptyset$. *The reset map* R *assigns to each point in* $G \times U$ *(for the case* $vp = ?U$*) or to each point in* G *(for the cases* $vp = !Y$ *and* $vp = \emptyset$*) for each state variable* $v \in \nu(l')$ *a probability measure on* $\mathbb{R}^{d(v)}$. *Active transitions* α *with* $\omega(oloc(\alpha)) = \emptyset$, *i.e., whose origin locations have no continuous variables, have value passing element* $vp = \emptyset$.

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$ be a value passing CPDP with state space E, flow map ϕ and initial state ξ_0. We define the CFSJS and NTS semantics of X. Let X_C be the CFSJS of X and let X_N be the NTS of X. X_C is defined as in Section 3.3.2. $X_N = (E, \Sigma^{vp} \cup \Sigma \cup \overline{\Sigma}, \mathscr{T})$, where

- $\Sigma^{vp} := \{(a, r) | a \in \Sigma, r \in \mathbb{R}^n \text{ for some } n \in \mathbb{N}\}$.
- $(\xi, a, m) \in \mathscr{T}$ if and only if there exists an $\alpha \in \mathscr{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$, $val(\xi) \in guard(\alpha)$, $rmap(\alpha)(\xi) = m$ and $vp(\alpha) = \emptyset$.
- $(\xi, (a, r), m) \in \mathscr{T}$, with $r \in \mathbb{R}^n$, if and only if there exists an $\alpha \in \mathscr{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$, and $val(\xi) \in guard(\alpha)$ and

  (i) $vp(\alpha) = !(w_1, \cdots, w_k)$, where

  $$(G(loc(\xi), w_1)(val(\xi)), \cdots, G(loc(\xi), w_k)(val(\xi))) = r$$

  (i.e., the output for w_1 at ξ equals r), and $rmap(\alpha)(\xi) = m$, or

  (ii) $vp(\alpha) = ?U$ and $r \in U$ and $rmap(\alpha)(\xi, r) = m$
- $(\xi, \bar{a}, m) \in \mathscr{T}$ if and only if there exists an $\alpha \in \mathscr{P}$ such that $lab(\alpha) = \bar{a}$, $oloc(\alpha) = loc(\xi)$ and $rmap(\alpha) = m$.

EXAMPLE 3.8 Let X be a CPDP with one location l. At l there is continuous dynamics $\dot{x} = 1$ and the output map equals $y = x$. There is one active transition $\alpha = (l, a, l, G, R, vp)$ with guard G satisfied if $x \geq 1$, with reset map $R(\xi)(\{x = 0\}) = 1$, i.e., R resets x to 0 at all states $\xi = (l, \{x = r\})$ with $r \geq 1$, and with value passing element $vp = !y$.

The NTS of X, whose state space we denote by E, equals $(E, \Sigma^{vp} \cup \Sigma \cup \overline{\Sigma}, \mathscr{T})$ with $\Sigma = \{a\}$, $\Sigma^{vp} = \{(a, r) | r \in \mathbb{R}^n \text{ for some } n \in \mathbb{N}\}$ and

$$\mathscr{T} = \{((l, \{x = r\}), (a, r), m) | r \geq 1, m = \text{Dirac measure at} x = 0)\}.$$

If we have $vp = ?U$ for some $U \subset \mathbb{R}$ instead of $vp = !y$ and we have $R(\xi, r) = Id(\{x = r\}$, then we get

$$\mathscr{T} = \{((l, \{x = r\}), (a, r'), m) \mid r \geq 1, r' \in U, m = \text{Dirac measure at} x = r')\}.$$

In the latter case we have that the NTS has for states $\xi \in G$ for all $r' \in U$ a transition with label (a, r'). If another CPDP Y outputs value $r' \in U$ through an a-transition, then the NTS of Y has a transition with label (a, r'). In the NTS composition of the NTSs of X with Y, these (a, r') transitions synchronize, which expresses that X accepts the output r' of Y. X then resets its state to r' as expressed by the reset measure $Id(l, \{x = r'\})$. This idea of composition of value passing CPDPs is formally defined as follows. ■

3.3.4.2 Composition of Value Passing CPDPs

DEFINITION 3.10 *Let $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathscr{A}_X, \mathscr{P}_X, \mathscr{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathscr{A}_Y, \mathscr{P}_Y, \mathscr{S}_Y)$ be two value passing CPDPs such that $V_X \cap V_Y = W_X \cap W_Y = \emptyset$. Then $X|_A^P|Y$ is defined as the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \mathscr{A}, \mathscr{P}, \mathscr{S})$, where L, V, ν, W, ω, F, G, Σ, $\mathscr{P}$, and $\mathscr{S}$ are defined as in Definition 3.8 and $\mathscr{A}$ is the least relation satisfying the rules (note that the rules r1,r2,r2′,r,r3′ are the same as in the ordinary composition of CPDPs, cf. Definition 3.8.)*

$$r1.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1\times G_2,R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \in A).$$

$$r2.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \notin A).$$

$$r2'.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \notin A).$$

$$r3.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

$$r3'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(a \notin A).$$

$$r1data.\frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1|G_2,R_1\times R_2,v_1|v_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \in A, v_1|v_2 \neq \perp).$$

$$r2data.\frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{a,G_1\times val(l_2),R_1\times Id,v_1}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

$$r2data'.\frac{l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,val(l_1)\times G_2,Id\times R_2,v_2}{\longrightarrow} l_1|_A^P|l_2'}(a \notin A),$$

where $l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'$ *means* $(l_1,a,l_1',G_1,R_1,v_1) \in \mathscr{A}_X$ *with* $v_1 \neq \emptyset$, $l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1'$ *means* $(l_1,a,l_1',G_1,R_1,\emptyset)$, *and* $v_1|v_2$ *is defined as:*

- $v_1|v_2 := !Y$ *if* $v_1 = !Y$ *and* $v_2 := ?U$ *and* $dim(U)=dim(Y)$ *or if* $v_2 = !Y$ *and* $v_1 := ?U$ *and* $dim(U)=dim(Y)$,
- $v_1|v_2 := ?(U_1 \cap U_2)$ *if* $v_1 = ?U_1$ *and* $v_2 = ?U_2$ *and* $dim(U_1)=dim(U_2)$,
- $v_1|v_2 := \perp$ *otherwise, where* $\perp$ *means that* v_1 *and* v_2 *are not compatible.*

Furthermore, $G_1|G_2$ *is, only when* $v_1|v_2 \neq \perp$, *defined as:*

- $G_1|G_2 := (G_1 \cap U) \times G_2$ *if* $v_1 = !Y$ *and* $v_2 = ?U$,
- $G_1|G_2 := G_1 \times (G_2 \cap U)$ *if* $v_1 = ?U$ *and* $v_2 = !Y$,
- $G_1|G_2 := G_1 \times G_2$ *if* $v_1 = ?U_1$ *and* $v_2 = ?U_2$.

Here we define $G \cap U$ *as the set of all states in* G *whose output values lie in* U.

THEOREM 3.3 *Let* X *and* Y *be two value passing CPDPs with semantics* (X_C, X_N) *and* (Y_C, Y_N) *respectively. Let* $(X|_A^P|Y_C, X|_A^P|Y_N)$ *be the semantics of value passing CPDP* $X|_A^P|Y$. *Assume that there do not exist value-passing transitions* $(l_1,a,l_1',G_1,R_1,!(w_1,\cdots,w_k)) \in \mathscr{A}_X$ *and* $(l_2,a,l_2',G_2,R_2,!(\tilde{w}_1,\cdots,\tilde{w}_l)) \in \mathscr{A}_Y$ *such that* $a \in A$ *and there exist* $\xi_1 \in G_1$ *and* $\xi_2 \in G_2$ *such that* $(G_X(\xi_1,w_1),\cdots, G_X(\xi_1,w_k)) = (G_Y(\xi_2,\tilde{w}_1),\cdots,G_Y(\xi_2,\tilde{w}_l))$. *Then,*

$$(X|_A^P|Y_C, X|_A^P|Y_N) = (X_C|Y_C, X_N|_A^P|Y_N).$$

REMARK 3.2 The assumption in Theorem 3.3 says that there may not be two value passing output transitions with the same label (in A) and with the same output value for some states. Rule $r1data$ expresses that two value passing output transitions can not synchronize, which is in line with the philosophy that at any moment only one component can determine the output, while multiple components may receive this value via value passing input transitions. If the assumption is not satisfied, then on the level of composition of NTSs, there will be synchronized transition that comes from these two output

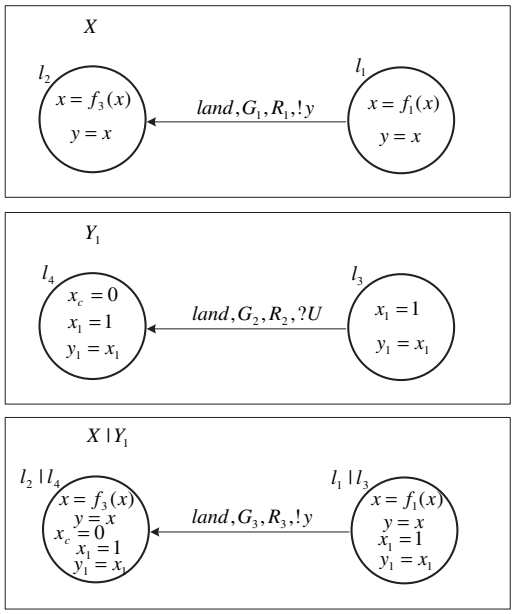

FIGURE 3.5: Value passing CPDPs.

transitions, while the NTS of the composition does not have this synchronized transition because of rule $r1data$. ■

THEOREM 3.4 *$|_A^P|$ for value passing CPDPs is commutative for all A and P. $|_A^P|$ for value passing CPDPs is associative if and only if for all events a we have $a \notin A \Rightarrow \overline{a} \in P$.*

EXAMPLE 3.9 In Figure 3.5 we see the value passing CPDPs X and Y_1. X and Y are the same as the X and Y_1 of Figure 3.2, except that here the active transitions are value passing active transitions. More specific, at the moment of switching to landing mode, the aircraft X sends the value of its state (position and velocity) to the control tower Y_1.

Sending the state information is modelled as the value $!y$ for the value passing part of the transition. y is the only output variable of X and is a copy of x and contains therefore the exact information of the state. The value passing element of the transition in Y_1 equals $?U$, where $U = \mathbb{R}^6$. This means that this transition can receive all six dimensional real values. Note that if we would have $r \notin U$ for some $r \in \mathbb{R}^6$, then the transition of X would be blocked at state $\{x = r\}$.

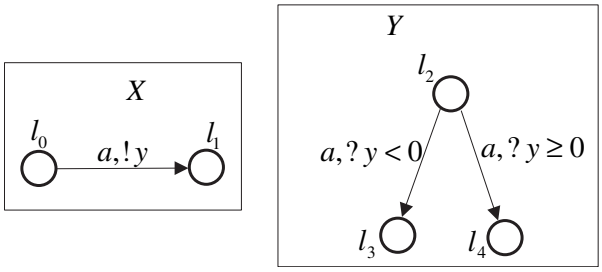

FIGURE 3.6: Value passing used to express scheduling.

Location l_4 of Y_1 has the new state variable x_c. This variable is used to store the information received from X. At the moment that X switches to l_2, Y_1 will switch to l_4 and the value of y, communicated by X, will be stored in x_c. Storing received data is done via the reset maps and in the case of Figure 3.5 it is expressed as $R(\{x_1 = r_1, x = r_2\})(\{\{x_1 = r_1, x_c = r_2\}\}) = 1$. Note that this indeed expresses that x_1 will not change by the switch and x_c holds the value of y after the switch.

■

3.3.4.3 Expressiveness of Value Passing

In Example 3.9 we have seen that value passing can express sending/receiving of the value of output variables. There are more types of communication that can be expressed by using value passing. We give two more examples which show two more types of communication: scheduling via value passing, constraint conjunction via value passing.

EXAMPLE 3.10 In this example we show how one CPDP can schedule transitions of another CPDP. In Figure 3.6, we see two CPDPs, X and Y, which are pictured without the details concerning state/output dynamics, guards and reset maps. CPDP X can switch from location l_0 to location l_1. With this switch, the value of output variable y is communicated over channel a. This value of y can be received by Y at initial location l_2. Y uses this information to schedule its two transitions at location l_2. If the value of y is smaller than zero, then the transition to location l_3 is taken, otherwise the transition to location l_4 is taken. In Figure 3.6, $?y < 0$ actually stands for $?\{r \in \mathbb{R} | r < 0\}$, and $?y \geq 0$ stands for $?\{r \in \mathbb{R} | r \geq 0\}$. In fact, these value passing transitions of Y can receive any one dimensional value that is communicated over channel a. This means that, if we compose Y with a component that sends at some time the value of some one dimensional variable y_2 over channel a, then Y can receive this value. We specifically write $y < 0$, to clarify that we

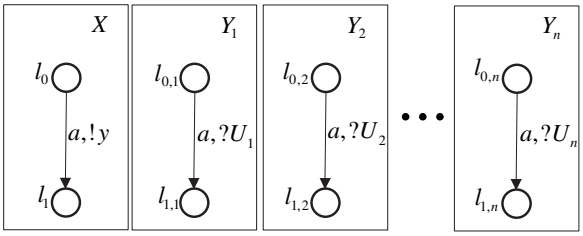

FIGURE 3.7: Value passing used for constraint conjunction.

intend that this transition is used to receive values of variable y of CPDP X.

In the composition $X|_A^P|Y$, with $A = \{a\}$ and P not relevant since no passive transitions are involved, X schedules the transitions of Y through the values of y. This method can, for example, be applied to systems where one component can perform different strategies, while the specific strategy that is chosen depends on the output variables of some other component. ■

EXAMPLE 3.11 In Figure 3.7, CPDP X can switch from initial location l_0 to location l_1. The guard of this transition (not pictured) equals the whole valuation space of l_0. If X would be executed as a stand alone CPDP, it would, because of maximal progress, switch immediately to l_1. In this example we show how other CPDPs, Y_1 till Y_n, can independently put constraints on the execution time of the active transition of X. For $i = 1 \cdots n$, U_i is the constraint put by CPDP Y_i on the execution time of the transition of X. Let y be one dimensional. Then, if $n = 2$, U_1 equals $y \geq -1$ and U_2 equals $y \leq 1$, then in $X|_A^P|Y_1|_A^P|Y_2$, with $A = \{a\}$ and P not relevant, the guard on the a-transition from location $l_0|l_{0,1}|l_{0,2}$ to location $l_1|l_{1,1}|l_{1,2}$ is equal to the part of the valuation space where $y \in [-1, 1]$. ■

3.4 Conclusions

In conclusion we summarize some aspects of the CPDP model, and describe which types of systems and which types of communication between those systems can be captured with the theory of this chapter.

A CPDP models a system with multiple locations. In each location, the continuous state of the CPDP has dynamics determined by some ordinary differential equation. The CPDP can jump from one location to another by means of a spontaneous transi-

tion or by means of a non-deterministic (or active) transition. A spontaneous transition is determined by some probability distribution. A non-deterministic (or active) transition can happen only if the continuous state lies inside the guard of that transition. However, if the process enters the guard of an active transition, then the process is not forced to execute the transition, but it is allowed to execute the transition.

Two CPDPs can communicate via the synchronization of transitions. If a is a synchronization action, then active a-transitions of the CPDPs should synchronize. This means that if one CPDP has an a-transition enabled (i.e., is inside the guard of some a-transition), and the other CPDP has no a-transition enabled, then this other CPDP blocks the enabled a-transitions of the first CPDP. We call this kind of communication *blocking interaction*. The other kind of communication that can be expressed is called *broadcasting interaction*. This happens if an active a-transition of one CPDP triggers a passive $\overline{a}$-transition of the other CPDP. Then the other CPDP "observes" that the first CPDP executes an a-transition. Thus, communication/interaction for CPDPs means that CPDPs can get knowledge about the execution of transitions in other CPDPs. Although two (or more) CPDPs cannot have shared continuous variables (as is the case in some other compositional hybrid systems frameworks), it is still possible that information concerning the continuous variables is communicated from one CPDP to the other. For this we need value-passing CPDPs, where active transitions of one CPDP can pass values (which come from the continuous variables) to active transitions in other CPDPs. These passed values can then influence the reset maps of the transitions that received these values and in that way one CPDP can get knowledge about the continuous variables of other CPDPs.

CPDPs have non-determinism. It is not determined when active transitions have to be executed and it is not determined which transition is executed at states where multiple transitions are enabled. In [11], the maximal progress assumption is used to resolve the first type of non-determinism: an active transition is executed as soon as the guard area of some transition is entered. In [11], the second type of non-determinism is resolved by defining a scheduler which probabilistically chooses which transition will be executed. Then, it is shown in [11], that a scheduled CPDP behaves under maximal progress as a PDP and an algorithm is given to transform such a scheduled CPDP into a PDP. With this equivalence result, scheduled CPDPs can be analyzed through PDP analysis techniques.

Also in [11] (or [14]), a notion of bisimulation is defined for CPDP, and an algorithm is given for finding bisimilation relations. Through bisimulation the state space of a CPDP can be reduced without changing the stochastic behavior of the CPDP.

References

[1] R. Alur, C. Coucoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid sys-

tems. *Theoretical Computer Science*, 138:3–34, 1995.

[2] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Comp. Networks and ISDN Systems*, 14:25–59, 1987.

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

[4] M. H. A. Davis. *Markov Models and Optimization*. Chapman & Hall, London, 1993.

[5] M. H. C. Everdij and H. A. P. Blom. Petri-nets and hybrid-state Markov processes in a power-hierarchy of dependability models. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 355–360, 2003.

[6] H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science*. Springer, Berlin, 2002.

[7] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Upper Saddle River, NJ, USA, 1985.

[8] J.F.C. Kingman. *Poisson Processes*. Oxford Clarendon Press, Oxford, 1993.

[9] M. Haj-Hussein, L. Logrippo, and M. Faci. An introduction to LOTOS: Learning by examples. *Comp. Networks and ISDN Systems*, 23(5):325–342, 1992.

[10] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

[11] S. N. Strubbe. Compositional Modelling of Stochastic Hybrid Systems. *Phd. Thesis*, Twente University, 2005.

[12] S. N. Strubbe, A. A. Julius, and A. J. van der Schaft. Communicating Piecewise Deterministic Markov Processes. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 349–354, 2003.

[13] S. N. Strubbe and R. Langerak. A composition operator with active and passive actions. Proc. 25th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, Taipei, 2005.

[14] S. N. Strubbe and A. J. van der Schaft. Bisimulation for Communicating Piecewise Deterministic Markov Processes (cpdps). In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 623–639. Springer, Berlin, 2005.

[15] S. N. Strubbe and A. J. van der Schaft. Stochastic semantics and value-passing for communicating piecewise deterministic Markov processes. Proc. Conf. Decision and Control, Seville, 2005.