hybr2d Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

**The present work was submitted to the LuFG Theory of Hybrid Systems**

BACHELOR OF SCIENCE THESIS

# A COMPOSITIONAL MODELING LANGUAGE FOR STOCHASTIC HYBRID SYSTEMS

**Nadja Scherer**

*Examiners:*
Prof. Dr. Erika Ábrahám
Prof. Thomas Noll

*Additional Advisor:*
Sasan Vakili

Aachen, March 30. 2022

**Abstract**

Stochastic Hybrid Automata are an extension of Hybrid Automata to resolve nondeterminism and model hybrid processes by introducing stochastics. Existing modeling approaches specify for each system state a single "global" probability distribution, which fixes the behavior of all stochastic processes. Whereas this has advantages for verification, this formalism is often disadvantageous from the modeling perspective.

This thesis presents a modeling language that models stochastic processes as first-class citizens: "local" state-dependent stochastic distributions are specified per stochastic process and each such process is modeled by a set of edges. We present syntax and semantics for our proposed language, including probability calculations for symbolic paths of the model.

Our modeling language is designed to support compositional modeling, allowing natural formalizations of both stochastically dependent as well as stochastically independent processes. To keep modeling intuitive, we introduce syntactical restrictions to avoid the introduction of unwanted stochastical dependencies as well as deadlocks and timelocks. Finally, we discuss alternative design options and their advantages and disadvantages for the modeler and the system's safety.

# Contents

# Chapter 1

# Introduction

Hybrid systems unite continuous dynamics and discrete dynamics in a model that is commonly used to model, analyze and control complex systems. To further expand the modeling scope, we can introduce uncertainty to the model, resolving the non-determinism. Uncertainty can have different sources: jump time, jump choice, reset choice, dynamics uncertainties, and initial state choice. The stochastic models differ mainly in source and type of random behavior introduction [LP10]. For jump time decisions, most literature uses a single "global" probability distribution for the locations or states of the model ([FHH$^+$11, BBB$^+$14, LP10]. Generally, in those models, we sample the jump time first and then choose a suitable jump enabled at that time. A generalization of these stochastic automata contains a transition rate function for time selection and a transition kernel for jump selection [LP10]. When we add random behavior, the question of safety is now no longer whether we can reach a bad state but whether the probability of getting to a bad state is acceptable [FHH$^+$11]. To get the model checking problem to be decidable, one approach is to restrict the stochastic model further and use only finite discrete distributions [Spr00]. Another method is to only consider timed automata to simplify model checking and introduce probability distributions only for delays and discrete choice of jumps [BBB$^+$14].
In contrast to the above systems, in this work we want to define a model with "local" state-dependent probability distributions for the processes. We then assign each stochastic process to edges. Furthermore, we want our model to be modeling-friendly and compositional, i.e., stochastic independence of two processes, but also dependencies, are easy to model. A problem for the existing approaches with global distributions is that often it is not straightforward to use them for parallel composition since we have to come up with new distributions for each location in the composition. Making the framework applicable for parallel construction and abstraction can help us to model larger and more complex applications. It also facilitates model analysis, as we can divide the system into subsystems for the analysis [LP10].
There already exists some related works in the direction of local distributions for the system's edges, for example by adding a set of probability distributions to the states [KNSS00]. Here first, a satisfying distribution is chosen non-deterministically. Next, the probability of going to one of the states is calculated with the selected distribution. For our model, we want to eliminate all non-determinism and have the distributions assigned to an edge. For Hybrid Petri Nets, a similar approach exists for so-called 'race models.' Each enabled transition has a firing delay, which runs down

and competes with the other enabled candidates. The transition fires when a firing delay expires and its transition is still enabled [BK02, HPS+21]. In this thesis, we will introduce clocks to the edges of Hybrid Automata for our compositional modeling language in a similar way. We will further define well-formed models and their probabilities to execute symbolic paths.

# Chapter 2

# Preliminaries

Hybrid systems combine discrete transitions, as used in discrete transition systems, and continuous behaviors, i.e., continuous time evolution. An example is a heater in a room with two modes ‚off' and ‚on', where it switches from one mode to another according to a particular room temperature threshold. The room temperature evolves continuously, while the heater determines its discrete model operation according to the temperature value. In this work, we use the Hybrid Automata modeling framework, which allows us to analyze the interplay of continuous and discrete activities of a hybrid system. An advantage of this modeling is its graphical representation for discrete and continuous activities via edges and vertices, respectively.

## 2.1 Hybrid Automata

For a function $f : A \to B$ and some $A' \subseteq A$ we denote by $f|_{A'}$ the function $f|_{A'} : A' \to B$ with $f|_{A'}(x) = f(x)$ for all $x \in A'$.
By $\mathbb{N}$ and $\mathbb{R}$ we denote the set of all natural numbers including 0 and respectively the real numbers.

**Definition 2.1.1** (Syntax of Hybrid Automata)**.** *A Hybrid Automaton [ACH+95] is a tuple* $\mathcal{H} = (Loc, Con, NCon, Edge, Act, Inv, Init)$ *with*

- *Loc is a finite non-empty set of locations;*
- *Con is a finite set of controlled variables;*
- *NCon is a finite set of non-controlled variables where* $Con \cap NCon = \emptyset$*;*
  *We define* $Var = Con \cup NCon$*;*
  Valuations $\nu : Var \to \mathbb{R}$*, V is the set of valuations;*
  *For* $v \in Var, \nu \in V$ *we define* $\nu[v \mapsto t]$ *as* $\nu[v \mapsto t](x) = \begin{cases} \nu(x) & \text{if } x \neq v \\ t & \text{if } x = v; \end{cases}$
- *Edge is a finite set of edges with* $Edge \subseteq Loc \times 2^{V^2} \times Loc$ *such that for all* $e = (l, \mu, l') \in Edge$ *and for all* $(\nu_1, \nu_2) \in \mu$ *it holds*
  1. $\nu_2 \in Inv(l')$,
  2. *for all* $\nu_1' \in V$ *if* $\nu_1|_{Con} = \nu_1'|_{Con}$ *then* $(\nu_1', \nu_2) \in \mu$ *and*
  3. $\nu_1|_{NCon} = \nu_2|_{NCon}$*;*

- *Act is a function assigning a set of activities* $f : \mathbb{R}_{\geq 0} \to V$ *to each location* $l \in Loc$. *The activity sets are time-invariant, i.e.* $f \in Act(l)$ *implies* $(f + t) \in Act(l)$, *where* $(f + t)(t') = f(t + t')$ *f.a.* $t' \in \mathbb{R}_{\geq 0}$.
  *Furthermore, for each* $\nu \in V$ *and each* $l \in Loc$ *there exists exactly one* $f \in Act(l)$ *such that* $f(0) = \nu$; *we denote this unique activity* $f$ *as* $f_{l,\nu}$;

- *Inv is a function assigning an invariant* $Inv(l) \subseteq V$ *to each location* $l \in Loc$ *with for all* $\nu \in Inv, \nu' \in V$, *if* $\nu|_{Con} = \nu'|_{Con}$ *then also* $\nu' \in Inv$;
  $\Sigma$ *is the set of states* $\sigma = (l,\nu) \in Loc \times V$ *for which* $Inv(l)$ *holds;*

- $Init \subseteq \Sigma$ *is a set of initial states.*

We modified the syntax of Hybrid Automata from [ACH+95] by dividing the variable set into controlled and uncontrolled variables. Further, we add constraints to the edge and activity functions. We extended the original edge definition by three restrictions. Firstly, the invariant in the source location and the guard of the edge implies the invariant's satisfiability in the target location. Secondly, edge guards are limited to controlled variables, and variables of *NCon* do not influence the guards. Lastly, we have a controlled reset, meaning uncontrolled variables do not change their values when taking a jump. Furthermore, we restrict *Act* to be deterministic, such that we have a unique activity function for each location and valuation.

The current state of a Hybrid Automaton depends on both discrete and continuous behavior. A state change is possible either discretely when taking an edge $e = (l,\mu,l') \in Edge$ (jump) or continuously when letting the time elapse. This is modeled by the continuous change of variables $x \in Var$ according to the activity $Act(l)$ in the current location $l \in Loc$ (flow). For an edge to be taken, the guard of this edge needs to be satisfied. Respectively for time to elapse, the invariant of the current location must hold. To model parallel composition of systems, we additionally introduce environment steps, where another system can change certain variables while it is running. We use the term 'environment' for these other components with which the system is composed. Only the system itself has writing access for its controlled variables. Therefore we restrict all other components to be only able to change variables that are not controlled. To achieve this, the sets of controlled variables of the systems need to be disjoint for composition. After composing, we want to obtain a closed system, i.e., the environment cannot change the variables in the system anymore. We then define the semantics only over closed systems.

**Definition 2.1.2** (Closed). *A Hybrid Automaton is closed if* $NCon = \emptyset$.

**Definition 2.1.3** (Semantics of Hybrid Automata). *The semantics for a closed Hybrid Automaton* $\mathcal{H} = (Loc,Con,NCon,Edge,Act,Inv,Init)$ *consists of discrete instantaneous steps (jumps), continuous time steps (flow) and environmental steps (*$\tau$*-steps):*

1. *Discrete step*

$$\frac{e = (l,\mu,l') \in Edge \quad (\nu,\nu') \in \mu}{(l,\nu) \xrightarrow{e} (l',\nu')} Rule_{discrete}$$

2. *Time step*

$$\frac{f \in Act(l) \quad f(0) = \nu \quad f(t) = \nu' \quad t \geq 0 \quad \forall 0 \leq t' \leq t . f(t') \in Inv(l)}{(l,\nu) \xrightarrow{t} (l,\nu')} Rule_{time}$$

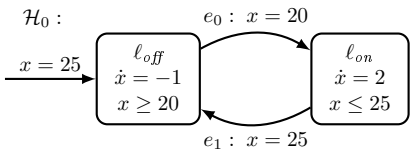



Figure 2.1: Thermostat modeled as Hybrid Automaton

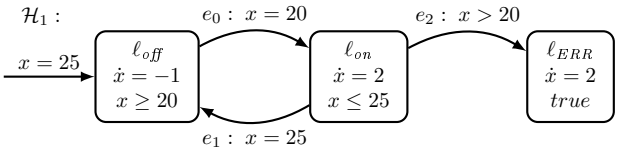



Figure 2.2: Thermostat with error state modeled as non-deterministic Hybrid Automaton

*3. Environmental step*

$$\frac{\nu|_{Con} = \nu'|_{Con}}{(l,\nu) \xrightarrow{\tau} (l,\nu')} Rule_{environment}$$

**Definition 2.1.4** (Execution step)**.** *We define* $\rightarrow = (\bigcup_{e \in Edge} \xrightarrow{e}) \cup (\bigcup_{t \geq 0} \xrightarrow{t}) \cup \xrightarrow{\tau}$.

**Definition 2.1.5** (Path, reachability)**.** *A path for a Hybrid Automaton (Loc,Con, NCon,Edge,Act,Inv,Init) is an infinite sequence of transitions:* $\sigma_0 \rightarrow \sigma_1 \rightarrow \sigma_2 ...$ *with* $\sigma_0 \in Init$.
*A state is called reachable if there is a path leading to it.*

**Example 2.1.1.** *In Figure 2.1 a basic Hybrid Automaton* $\mathcal{H}_0 = (Loc,Con,NCon,Edge, Act,Inv,Init)$ *is shown with*

- $Loc = \{\ell_{on},\ell_{off}\}$
- $Con = \{x\}$
- $NCon = \emptyset,\ Var = \{x\}$
- $Edge = \{e_0 = (\ell_{off}, \{(\nu,\nu') \,|\, \nu(x) = 20 \wedge \nu'(x) = \nu(x)\},\ell_{on}),$
  $e_1 = (\ell_{on}, \{(\nu,\nu') \,|\, \nu(x) = 25 \wedge \nu'(x) = \nu(x\},\ell_{off})\}$
- $Act\,(\ell_{off}) = \{f : \mathbb{R}_{\geq 0} \rightarrow V \,|\, \forall t \in \mathbb{R}_{\geq 0}.f(t) = f(0) - t\},$
  $Act\,(\ell_{on}) = \{f : \mathbb{R}_{\geq 0} \rightarrow V \,|\, \forall t \in \mathbb{R}_{\geq 0}.f(t) = f(0) + 2t\}$
- $Inv\,(\ell_{off}) = \{\nu \in V \,|\, \nu(x) \geq 20\}, Inv\,(\ell_{on}) = \{\nu \in V \,|\, \nu(x) \leq 25\}$
- $Init = \{(\ell_{off}, \nu_s) \,|\, \nu_s(x) = 25\}$.

*We look at examples for the preceding definitions.* $\mathcal{H}_0$ *is a closed system because* $Con = Var$*, meaning that the environment is not allowed to change the variable* $x$ *and can only read it. The reachable states are* $\{(\ell_{off},\nu) \,|\, \nu(x) \geq 20\} \cup \{(\ell_{on},\nu) \,|\, \nu(x) \leq 25\}$. *To look at an example of flow and jump, we start in the initial state* $(\ell_{off},\nu_s)$ *with* $\nu_s(x) = 25$. *Here only flow is possible: the automaton can stay in location* $\ell_{off}$ *for*

*a time duration t if the invariant $x \geq 20$ is satisfied for each time point in [0,t]. However, a jump is possible if we are in location $\ell_{off}$ and guard $x = 20$ is satisfied (after time elapsed). We can then take the edge $e_0$ to go to location $\ell_{on}$. A path in the automaton $\mathcal{H}_0$ looks like $(\ell_{off},\nu_0) \xrightarrow{5} (\ell_{off},\nu_1) \xrightarrow{e_0} (\ell_{on},\nu_1) \xrightarrow{2.5} (\ell_{on},\nu_0) \xrightarrow{e_1} (\ell_{on},\nu_0)...$ with $\nu_0(x) = 25$ and $\nu_1(x) = 20$. In Figure 2.2, the Hybrid Automaton is extended by another edge leading to an error state. Here we have non-deterministic behavior, meaning e.g. it is possible from state $(\ell_{on},\nu_0)$ to take both edges $e_1$ and $e_2$. Therefore it is decided non-deterministically which one to take.*

## 2.2 Definitions

We need to define a probability space and probability distributions to define Stochastic Hybrid Automata. For the definition of the probability space, we are following [Áb21].

**Definition 2.2.1** (Experiment)**.** *A random experiment is an act with an uncertain outcome.*

**Definition 2.2.2** (Sample space)**.** *The sample space $\Omega$ of an experiment is the set of all possible outcomes of the experiment.*

**Definition 2.2.3** (Event)**.** *Given a sample space $\Omega$, an $\Omega$-event is a subset of $\Omega$.*

**Definition 2.2.4** ($\sigma$-algebra)**.** *A $\sigma$-algebra $\mathcal{F} \subseteq 2^\Omega$ for a sample space $\Omega$ is a set of $\Omega$-events containing the maximal event $\Omega$ and being closed under complement and countable union.*

**Definition 2.2.5** (Probability measure, measurable space)**.** *Given $(\Omega, \mathcal{F})$, a probability measure for $(\Omega, \mathcal{F})$ is a function $Pr : \mathcal{F} \to [0,1] \subseteq \mathbb{R}$ with*

- $Pr(\Omega) = 1$;
- $Pr(E) = 1 - Pr(\overline{E})$ *for all* $E \in \mathcal{F}$;
- $Pr(\bigcup_{i=0}^{\infty} E_i) = \sum_{i=0}^{\infty} Pr(E_i)$ *where* $E_i \in \mathcal{F}$ *and* $E_i \cap E_j = \emptyset$ *for all* $i,j \in \mathbb{N}, i \neq j$.

*If such a function exists then $(\Omega, \mathcal{F})$ is also called a measurable space.*

**Definition 2.2.6** (Probability space)**.** *A probability space is a triple $(\Omega, \mathcal{F}, Pr)$ with*

- $(\Omega, \mathcal{F})$ *is a measurable space and*
- $Pr$ *a probability measure for* $(\Omega, \mathcal{F})$.

**Definition 2.2.7** (Random variable)**.** *Assume a probability space $(\Omega, \mathcal{F}, Pr)$ and a measurable space $(S,\Sigma)$.*

- *For $X : \Omega \to S, s \in S$ and $\sigma \in \Sigma$ we define*
  - $X = s$ *to be* $\{\omega \in \Omega \mid X(\omega) = s\}$ *and*
  - $X^{-1}(\sigma) = \bigcup_{s \in \sigma}(X = s)$
- *$X$ is measurable if $X^{-1}(\sigma) \in \mathcal{F}$ for all $\sigma \in \Sigma$*
- *A random variable is a measurable function $X : \Omega \to S$.*

The *supp*$(f)$ of a function is the set of all elements from $S$, which are mapped to a positive value.

**Definition 2.2.8** (Support)**.** *For a function $f : S \to \mathbb{R}_{\geq 0}$ its support is defined as $supp(f) = \{s \in S \mid f(s) > 0\}$.*

**Definition 2.2.9** (Probability distributions)**.** *A function $f : S \to \mathbb{R}_{\geq 0}$ is*

- *a discrete probability distribution if $supp(f)$ is countable and $\sum_{s \in supp(f)} f(s) = 1$. The codomain is reduced to $[0,1] \subseteq \mathbb{R}_{\geq 0}$ in that case;*
- *a continuous probability distribution if $f$ is continuous, $supp(f)$ is uncountable and $\int_{s \in S} f(s)ds = 1$.*

A probability distribution is either a discrete or a continuous probability distribution. It is is a function showing the distribution of probabilities allowing us to compute the probability measure of an experiment.
In the following, we consider S to be either $\mathbb{R}_{\geq 0}$ the Euclidean space $\mathbb{R}^n$ or the location set of a Hybrid Automaton.

# Chapter 3

# Stochastic Hybrid Automata

We present Stochastic Hybrid Automata intending to resolve non-determinism by introducing probabilities. In our approach, we use probability distributions for each stochastic process locally, allowing easy model composition. To define semantics and probabilities we assume from now on that the Stochastic hybrid system is closed. We also assume that the enabled time interval for an edge is a closed set, meaning all guards and invariants are closed sets.

## 3.1 Syntax of Stochastic Hybrid Automata

**Definition 3.1.1** (Syntax of Stochastic Hybrid Automata)**.** *A Stochastic Hybrid Automaton (SHA) is a tuple* $\mathcal{A} = (\mathcal{H}, Lab, Edge_r, JmpT, Wgt, Rst, Init_{loc}, Init_{val})$ *where* $\mathcal{H} = (Loc, Con, NCon, Edge, Act, Inv, Init)$ *is a Hybrid Automaton with the following components:*

- $Lab$ *is a set of labels specifying the random processes;*
- $Edge_r : Edge \to Lab$ *is a function assigning a random process to each edge;*
- $JmpT(r,\sigma) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *is a function assigning a probability distribution to each label* $r \in Lab$ *and each state* $\sigma \in \Sigma$*;*
- $Wgt : Edge \to \mathbb{N}_{>0}$ *is a function assigning a weight to each jump;*
- $Rst(e,\nu) : V \to \mathbb{R}_{\geq 0}$ *is a function assigning a probability distribution to each edge and each valuation.*
  *For* $e = (l, \mu, l') \in Edge$ *and* $\sigma = (l,\nu)$ *we require* $supp(Rst(e,\nu)) \subseteq \{\nu \in V \mid (\nu,\nu') \in \mu\}$*;*
- $Init_{loc} : Loc \to \mathbb{R}_{\geq 0}$ *is a discrete probability distribution with* $supp(Init_{loc}) \subseteq \{l \in Loc \mid (l,\nu) \in Init\}$*;*
- $Init_{val}(l) : V \to \mathbb{R}_{\geq 0}$ *is a function assigning a continuous distribution to each location* $l \in Loc$ *with* $supp(Init_{val}) \subseteq \{\nu \in V \mid (l,\nu) \in Init\}$.

Additional to the Hybrid Automaton definition, we now introduce labels to assign a random process to each edge. We also add a distribution for each label $r$ to determine when the corresponding jumps should be taken, which we then store in a random

variable $c_r$. Furthermore, we use distributions to reset system variables and choose the initial state's location and valuation. In the following Section 3.3 the individual functions and backgrounds of the elements of an SHA are explained in more detail.

## 3.2 Semantics of Stochastic Hybrid Automata

The following probability distributions are part of the model:

- $JmpT(r,\sigma)$ stochastically determines how long jumps of the process $r$ need to be enabled before one of them is taken.
  We can model urgent jumps, that are taken almost surely, immediately when they are enabled, by setting $JmpT(r,\sigma)(0) = 1$.
  For a discrete probability distribution, the probability that a jumps of process $r$ is taken after an enabling duration between a and b is defined with $P_e(a \leq c_r \leq b) = \sum_{i=a,i\in supp(JmpT(r,\sigma))}^{b} JmpT(r,\sigma)(i)$. Otherwise the probability can be calculated using $P_e(a \leq c_r \leq b) = \int_a^b JmpT(r,\sigma)(x)dx$.
- $Rst(e,\nu)$ resolves non-determinism on reset values. Only resets, that are included in the transition relation, can be part of the support.
- $Init_{loc}$ determines the initial location.
- $Init_{val}$ determines the initial valuation.
  If the domain is discrete then the distribution needs to be discrete, otherwise it can be discrete or continuous.

The transition relation $\mu \in 2^{V^2}$ of an edge is a set of pairs of valuations $(\nu,\nu') \in \mu$. The first valuation $\nu$ represents the guard and the second one $\nu'$ represents the reset. We introduce a new variable set for random variables $Var_r$ and we further use a transition relation $\mu_r$ and valuations $\nu_r$ for the random variables. We also introduce a notation to combine the pair of valuations for system and random variables. Accordingly, we define a valuation by assigning each system variable its value according to $\nu$ and each random variable its value according to $\nu_r$.

**Definition 3.2.1** (Random variables). *Let $Var_r = \{c_r \mid r \in Lab\}$ with $Var \cap Var_r = \emptyset$ contain a unique freh variable for each random process. We call the elements of $Var_r$ random variables and define $V_r$ to be the set of all valuations $\nu_r : Var_r \to \mathbb{R}$.*

**Definition 3.2.2** (Guard). *For a given transition relation $\mu \subseteq V \times V$ the guards are defined as the set*
$g_\mu = \{\nu \in V \mid \exists \nu' \in V.\, (\nu,\nu') \in \mu\}$.

**Definition 3.2.3** (isEn). $isEn(g_\mu,t,f) = true \Leftrightarrow f(t) \in g_\mu$.

**Definition 3.2.4** (enablednessInv). $enablednessInv_{(l,t,f,\nu_r,\nu'_r)} = true$ *iff*
$\forall e = (l,\mu,l*) \in Edge.\Big[\big[\nu'_r(c_r) = \nu_r(c_r) - t \geq 0 \wedge \forall 0 < t' < t.isEn(g_\mu,t',f)\big] \vee$
$\big[\nu'_r(c_r) = \nu_r(c_r) \wedge \forall 0 < t' < t.\neg isEn(g_\mu,t',f)\big]\Big]$.

When time evolves, we need to make sure that random variables are running down as they should. During the whole enabling time of an edge, its process should be running, meaning its random variables run down without interruption.Therefore

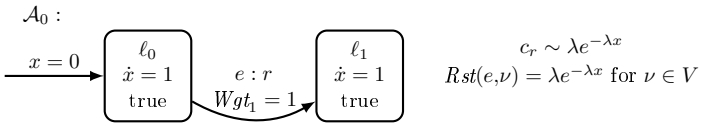



Figure 3.1: Stochastic Hybrid Automaton example

during a timestep the guards of all outgoing edges should be either invariantly true or invariantly false. All guards that were enabled with the old valuation $\nu$ need to remain enabled after time $t$ passed with the new valuation $\nu'$, and all guards that were not enabled, also need to stay disabled. Predicate '*enablednessInv*' ensures that a jump cannot change its enabledness within the time (0,$t$). And if so, we take another timestep from that time point on.

The Semantics for a Stochastic Hybrid Automaton $\mathcal{A} = (\mathcal{H}, Lab, Edge_r, JmpT, Wgt, Rst, Init_{loc}, Init_{val})$ where $\mathcal{H} = (Loc, Con, NCon, Edge, Act, Inv, Init)$ is only defined for closed systems and consists of discrete instantaneous steps (jumps), continuous time steps (flow) and environmental steps ($\tau$-steps):
First we initialize all random variables from their distribution in the initial state. For all $r$ we sample $c_r \sim JmpT(r,\sigma)$ with $\sigma$ is the initial state.

1. Discrete step

$$\frac{(l,\nu) \xrightarrow{e} (l',\nu') \quad r = Edge_r(e) \quad \nu_r(c_r) = 0 \quad \nu' \in supp(Rst(e,\nu)) \quad \nu'_r(c_r) \in supp(JmpT(r,(l',\nu')))}{(l,\nu,\nu_r) \overset{e}{\Rightarrow} (l',\nu',\nu'_r)} Rule_{discrete}$$

2. Time step

$$\frac{(l,\nu) \xrightarrow{t} (l,\nu') \quad enablednessInv_{(l,t,f,\nu_r,\nu'_r)}}{(l,\nu,\nu_r) \overset{t}{\Rightarrow} (l,\nu',\nu'_r)} Rule_{time}$$

3. Environmental step

$$\frac{(l,\nu) \xrightarrow{\tau} (l,\nu')}{(l,\nu,\nu_r) \overset{\tau}{\Rightarrow} (l,\nu',\nu_r)} Rule_{environment}$$

## 3.3 Explanation

With the help of example Stochastic Hybrid Automaton shown in Figure 3.1, we explain the individual probability distributions and elements of a Stochastic Hybrid Automaton.
$\mathcal{A}_0$ is a SHA with

- $Lab = \{r\}$
- $Edge_r(e) = r$
- $JmpT(r,\sigma) = \lambda e^{-\lambda x}$

- $Wgt(e) = 1$
- $Rst(e_1,\nu) = \lambda e^{-\lambda x}$
- $Init_{loc} = \delta(\ell_0)$ Dirac distribution $Init_{loc}(\ell_0) = 1$
- $Init_{val} = \delta(\nu)$ Dirac distribution with $Init_{val}(\ell_0)(\nu) = 1$ for $\nu(x) = 0$

To resolve non-determinism for edge decisions, we introduce state-dependent probability distributions $JmpT(r,\sigma)$ for stochastic processes and assign processes to the edges. For each jump, a non-negative value of its distribution gets randomly chosen and stored in the random variable $c_r$. These random variables are backward running clocks only decreasing when a jumps of its random process is enabled. We can take the jump as soon as the variable gets to 0, and we reset it after to a new random value. A second distribution $Rst(e,\nu)$ also allows resetting the system variables. We exclude some complications, e.g., the user defining a distribution for reset outside of the possible valuation values. To do so, we restrict the support of reset to only go over the transition relation. To build a meaningful and working system, the modeler should choose the distributions wisely. In that sense, a discrete jump distribution fits if an edge is enabled at a countable number of discrete time points and a continuous distribution if its guard is satisfied at uncountable many time points. Although the distribution should be in a way, so it is impossible to sample a too large value, we do not exclude this possibility. In this case, we can rescale the probability if it is possible to take another edge.

We expanded the three semantical rules from Hybrid Automata for the semantics of Stochastic Hybrid Automata. Starting with the discrete step, we assumed that guard and reset of jump directly imply the invariant of the following location to make the model easier. The following must hold to take a given edge $e$ with its random label $r$: the guard is enabled, the valuation is part of the transition relation (same as in Hybrid Automata), the random clock reaches 0, and we sample a suitable value for the reset of the system and the random variable. Next, we take a look at the time steps. We start in time point 0 with valuation $\nu$, and after $t$ time, we get the valuation $\nu'$. With the attribute 'enablednessInv', we make sure that the following conditions are met: The invariant holds for the whole time step, the same edges are enabled, and the random clocks for these enabled edges run down. It is important to note that the time only elapses as long as the enabling conditions do not change. When one jump's enabling value changes, we update its random variable, i.e., it starts or stops running. This way, we can have multiple time steps after each other. Every time a condition for a jump is enabled or disabled, we take a new time jump to ensure the random clock $c_r$ counts its assigned edge's total time of enabledness.

## 3.4 Well-formed Automata

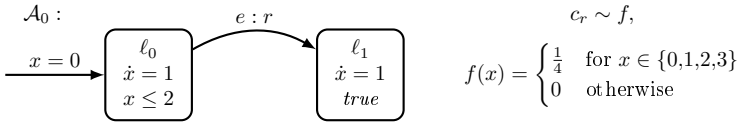
To protect the probability space, we need to assure that the control always (almost surely) leaves a location before the location's invariant gets violated. If it is possible to leave before the invariant gets violated, this will occur with a probability of 1 per definition. We will neglect the whole path where the random variable is assigned to a value to violate the invariant. The whole probability will split between the possible paths; therefore, the probability gets rescaled. If we cannot leave the location at any time and the invariant gets violated unavoidably, then the model is not meaningful.

**Definition 3.4.1** (Time convergence)**.** *For a Stochastic Hybrid Automaton* $\mathcal{A} = (\mathcal{H}, Lab, Edge_r, JmpT, Wgt, Rst, Init_{loc}, Init_{val})$ *where* $\mathcal{H}$ = (*Loc,Con,NCon,Edge, Act,Inv,Init*) *we define*
$ExecTime : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *with ExecTime(d)=d for* $d \in \mathbb{R}_{\geq 0}$ *and ExecTime(e)=0 for* $e \in Edge \cup \{\tau\}$. *Furthermore, for* $\rho = \sigma_0 \overset{\alpha_0}{\Longrightarrow} \sigma_1 \overset{\alpha_1}{\Longrightarrow} \sigma_2 ...$ *we define* $ExecTime(\rho) = \sum_{i=0}^{\infty} ExecTime(\alpha_i)$.
*A path is called time-divergent iff* $ExecTime(\rho) = \infty$, *and time-convergent otherwise. A path is called zeno iff it is time-divergent and contains infinitely many discrete steps.*

The probability to reach a time-convergent path should be 0 in a well-defined automaton, therefore the probability for an infinite time-divergent path should be 1 at all times. This implies (almost sure) the property of non-zenoness, and freedom of time- and deadlocks.

**Definition 3.4.2** (Timelock, deadlock)**.** *State* $\sigma$ *has a timelock if there is no time-divergent path starting in* $\sigma$.
*State* $\sigma$ *has a deadlock if no transition can be taken from* $\sigma$.

Example of an automaton with a timelock:




*JmpT*($r$,$\sigma$) is a discrete uniform distribution from 0 to 3. If a time point over 2 is sampled e.g. $X_r = 3$, it is not possible to take the edge at time 3, so we should stay in location $\ell_0$ forever, but it is not possible. We have a timelock.
An automaton is timelock-free if no reachable state contains a timelock with a positive probability. Following, we describe two sufficient conditions for a timelock-free automaton. The first one is that we can always leave each location, i.e., for all locations $l$ and each valuation $\nu$, the guard of at least one edge with source location $l$ is satisfied by $\nu$. An automaton is also timelock-free if all locations have the trivial invariant 'true', and therefore, it is possible to stay in each location forever.

**Definition 3.4.3** (Well-formed automata)**.** *A SHA is well-formed if the underlying hybrid automaton is free of timelocks, it has no zeno paths, and both the invariants and the guards are closed sets.*

In the following we only consider well-formed automata.

## 3.5 Probabilities of Stochastic Hybrid Automata

We now introduce a way to calculate the probability of taking specific paths in a Stochastic Hybrid Automaton. We note that the probability of reaching a single state will typically be 0. However, the likelihood of getting to a state set can be positive. Therefore, we define finite symbolic paths, which start at a fixed state and follow a sequence of edges. Symbolic paths represent the set of all paths for which the edges are the same, but the time steps can differ. We use the attribute fireable, which means that not only is the edge's guard enabled at the current state, but also its random variable has reached 0.

**Definition 3.5.1** (Fireable). *fireable*$((l,\nu,\nu_r),e) = true$ *iff* $e = (l,\mu,l') \wedge r = Edge_r(e) \wedge \nu \in g_\mu \wedge \nu_r(c_r) = 0$.

**Definition 3.5.2** (Execution step). $\Rightarrow = (\bigcup_{e \in Edge} \stackrel{e}{\Rightarrow}) \cup (\bigcup_{t \geq 0} \stackrel{t}{\Rightarrow}) \cup \stackrel{\tau}{\Rightarrow}$

**Definition 3.5.3** (Symbolic paths). *We define a finite symbolic path* $\pi = (\sigma, e_1, ..., e_n)$ *with* $\sigma \in Init$ *and* $e_1,...,e_n \in Edge$.

**Definition 3.5.4** (Probability of symbolic paths). *The probability for a symbolic path* $\pi = (\sigma, e_1, ..., e_n)$ *with a known* $\sigma = (l,\nu), e_1 = (l, \mu, l'), Lab = \{r_1,...,r_k\}, Edge_r(e_1) = r$ *is defined as* $P(\sigma) = 1$ *for* $n = 0$ *and otherwise for* $n \geq 1$*:*

$$P((\sigma, e_1, ..., e_n)) = \int_{t_1=0}^{\infty} JmpT(r_1,(l,\nu))(t_1) \cdot ... \cdot \int_{t_k=0}^{\infty} JmpT(r_k,(l,\nu))(t_k) \cdot P'(\sigma',e_1,...,e_n)dt_k...dt_1$$

*with* $\sigma' = (l,\nu,\nu_r)$, *with* $\nu_r$ *is the valuation assigning the value* $t_i$ *to* $c_{r_i}$ *for each* $i = 1,...,k$.

$$P'((\sigma', e_1, ..., e_n)) = P'' \cdot Remainder + [\prod_{e \in Edge} (1 - fireable(\sigma',e))] \cdot Fin \cdot P'(succ_t(\sigma'),e_1,...,e_n)$$

$$P'' = \begin{cases} 0 & \text{if } \sum_{e \in Edge, fireable(\sigma',e)} Wgt(e) = 0 \\ \frac{fireable((l,\nu,\nu_r),e_1) \cdot Wgt(e_1)}{\sum_{e \in Edge, fireable(\sigma',e)} Wgt(e)} & \text{otherwise} \end{cases}$$

$$Remainder = \begin{cases} \int_{\nu' \in V} Rst(e_1,\nu)(\nu') \cdot \int_{t \geq 0} JmpT(r,(l',\nu'))(t) \\ \quad \cdot P'((l', \nu', \nu_r[c_r \mapsto t]),e_2,...,e_n)dt\, d\nu' & \text{if } n \geq 2 \\ 1 & \text{otherwise} \end{cases}$$

$$Fin = \begin{cases} 1 & \text{if maximal time duration in } \sigma \text{ is finite} \\ 0 & \text{otherwise} \end{cases}$$

$$succ_t(\sigma) = \{\sigma' \mid \sigma \stackrel{t}{\Rightarrow} \sigma', t \text{ is maximal}\}$$

*If the distributions used above are continuous, we use the given integrals. If the distributions are discrete, we replace the integrals with (potentially infinite) sums. The weight only influences the probabilities if two edges are fireable simultaneously. Yet, the likelihood to choose the same point when at least one distribution has countable support is 0. For this reason, weights only affect the non-continuous case, i.e., discrete distributions with finite support, and we can omit them otherwise from the probability calculation.*

**Definition 3.5.5** (Probability for sequence of edges). *For* $e_1 = (l,\mu,l')$ *the probability of starting at any initial state such that* $(e_1,...,e_n)$ *can be executed is:*

$$P(e_1,...,e_n) = Init_{loc}(l) \cdot \int_{\nu \in V} Init_{val}(l)(\nu) \cdot P((l,\nu),e_1,...,e_n)d\nu$$

To calculate the probability of symbolic paths, we first need the probability of initializing every single random process. For this, we assign a duration $t_i$ to each random value. The valuation $\nu$ denotes the valuation at the very beginning, where

only the system variables are initialized. Now we define $\nu_r$ assigning the value $t$ to each random variable. After initialization, we recursively calculate the probability for each of the jumps with $P'$ for the given $t_i$. This way we get the probability that for this $t_1$,...,$t_k$ we will execute $e_1$ to $e_n$. $P'$ consists of two summands. The first one represents the probability that $e_1$ happens now: If $e_1$ is fireable, we take the probability that we take $e_1$, considering the weight of all other enabled edges ($P''$). Then we take the jump, so we reset the system variables with *Rst*, after that we reset the random clock to $t$ with the distribution $JmpT$ for the new state. The edge we take tells us in which location we go and $\nu'$ denotes the valuation after the reset. Finally, we multiply the probability of the remaining path, considering that we took edge $e_1$ and we have new values for system variables and the clock. The predicate *fireable* has the output true or false, but in $P''$ we consider it a pseudo boolean with the value 0 for false or 1 for true. With the second summand of $P'$, we calculate the probability of taking $e_1$ later because it is not fireable now. We figure if none of the edges in the current state are enabled with the product. If one is fireable, the product is 0. Otherwise we make the longest time step possible, and if it is finite, we take $P'$ for the successor of the longest step. The longest step exists, because we assumed that the enabled time interval for an edge is a closed set. The flag *Fin* ensures that we get the probability 0 if the maximum time to stay in the location is infinite. This happens if no edge will ever be enabled in the future. But if it is possible to take a jump in the future, then the time cannot elapse forever, so a maximum for $\text{succ}_t$ exists. When we look at the last edge of a sequence, that means $n = 1$, and we either take the final step in $P'$ or let time elapse.
It is challenging to compute these integrals, and in general, this task is undecidable because the reachability problem for Hybrid Automata is undecidable. Here we show how we calculate the probabilities for a simple example.

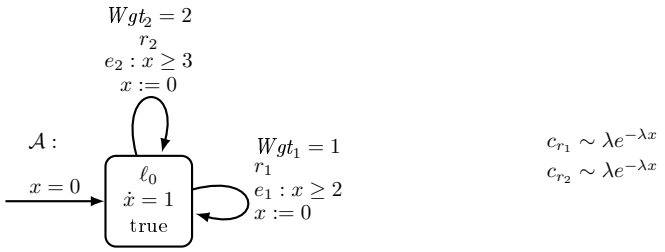



Figure 3.2: A simple Stochastic Hybrid Automaton, for which we compute the probabilities of symbolic paths

**Example 3.5.1.** *The figure 3.2 shows a SHA* $\mathcal{A} = (\mathcal{H}, Lab, Edge_r, JmpT, Wgt, Rst, Init_{loc}, Init_{val})$ *where* $\mathcal{H} = (Loc, Con, NCon, Edge, Act, Inv, Init)$ *with:*

- $Loc = \{\ell_0\}$
- $Con = \{x\}$
- $NCon = \emptyset,\ Var = \{x\}$
- $Edge = \{e_1 = (\ell_0, \{(\nu,\nu') \mid \nu(x) \geq 2, \nu' = \nu[x \mapsto 0]\}, \ell_0),$
  $e_2 = (\ell_0, \{(\nu,\nu') \mid \nu(x) \geq 3, \nu' = \nu[x \mapsto 0]\}, \ell_0)\}$

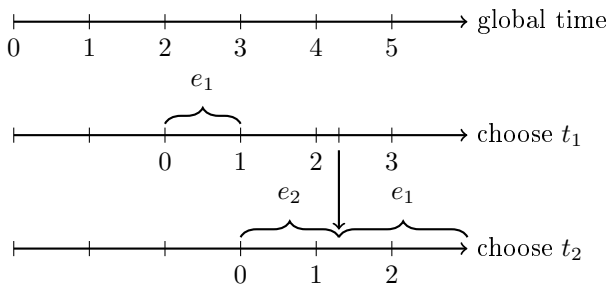



Figure 3.3: Timelines for the clocks of edge $e_1$ and $e_2$:
$e_1$ is taken as first edge, if $t_1$ is sampled between 0 and 1
$e_2$ is taken as first edge, if $t_1$ is sampled above 1 and $t_2$ is sampled between 0 and $t_1 - 1$
$e_1$ is taken as first edge, if $t_1$ is sampled above 1 and $t_2$ is sampled between $t_1 - 1$ and $\infty$

- $Act(\ell_0) = \{f : \mathbb{R}_{\geq 0} \to V \mid \forall t \geq 0. f(t) = f(0) + t\}$
- $Inv(\ell_0) = V$
- $Init = \{(\ell_0, \nu) \in \Sigma \mid \nu(x) = 0\}$
- $Lab = \{r_1, r_2\}$
- $Var_r = \{c_{r_1}, c_{r_2}\}$
- $Edge_r(e_1) = r_1$, $Edge_r(e_2) = r_2$
- $JmpT(r_1,\sigma) = JmpT(r_2,\sigma) = \lambda e^{-\lambda x}$ *continuous exponential distribution, with rate parameter* $\lambda = 2$
- $Wgt(e_1) = 1$, $Wgt(e_2) = 2$
- $Rst(e_i,\nu) = \delta(\nu')$ *Dirac distribution with* $\nu'(x) = 0$ *for* $i = 1,2$
- $Init_{loc} = \delta(\ell_0)$ *Dirac distribution*
- $Init_{val}(\ell_0) = \delta(\nu)$ *Dirac distribution with* $\nu(x) = 0$

We are first going to calculate the probability of taking only $e_1$ and only $e_2$ using Definition 3.5.4. To do so, we insert a arbitrary edge $e$ first and then split these calculations into the choice to take either $e_1$ or $e_2$. Going further, we calculate the probability of taking a sequence of two edges. We always split the integrals during calculation until we get the corresponding path we want to compute. We illustrated the values where we need to split the integrals for the probabilities for one execution in the timelines in Figure 3.3.

*For $e \in Edge$ we get:*

$$\begin{aligned} P(e) &= Init_{loc}(\ell_0) \cdot \int_{\nu \in V} Init_{val}(\ell_0)(\nu) \cdot P((\ell_0,\nu),e_1) d\nu \\ &= 1 \cdot \int_{\nu \in V} Init_{val}(\ell_0)(\nu) \cdot P((\ell_0,\nu),e_1) d\nu \end{aligned}$$

$$
\begin{aligned}
&P((\ell_0,\nu),e)\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot P'(\sigma',e_1)dt_2dt_1\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{P'' \cdot \mathit{Remainder}}_{=0 \text{ until an edge is fireable}} + \underbrace{[\prod_{e\in Edge} (1-\mathit{fireable}(\sigma',e))]}_{=1 \text{ until an edge is fireable}} \cdot \mathit{Fin}\\
&\quad \cdot P'(succ_t(\sigma'),e_1)dt_2dt_1\\
&\quad \text{take time steps until first edge } e \text{ is fireable}\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{P'' \cdot \mathit{Remainder}}_{>0 \text{ because } e \text{ is fireable}} + \underbrace{0}_{e \text{ is fireable}} \; dt_2dt_1\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{\frac{\mathit{fireable}((l,\nu,\nu_r),e_1) \cdot Wgt(e_1)}{\sum_{e\in Edge,\mathit{fireable}(\sigma',e)} Wgt(e)}}_{P''=1 \text{ if } e \text{ fireable now } \wedge |supp(JmpT)|=\infty} \cdot \underbrace{\mathit{Remainder}}_{=1} \; dt_2dt_1\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot P'' \cdot 1 \; dt_2dt_1\\
&\quad \text{split in different integrals}\\
&= \underbrace{\int_{t_1=0}^{1} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot P'' \; dt_2dt_1}_{\text{belongs to } P(e_1)}\\
&\quad + \underbrace{\int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{t_1-1} 2e^{-2t_2} \cdot P'' \; dt_2dt_1}_{\text{belongs to } P(e_2)} + \underbrace{\int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=t_1-1}^{\infty} 2e^{-2t_2} \cdot P'' \; dt_2dt_1}_{\text{belongs to } P(e_1)}
\end{aligned}
$$

We can now calculate the probability to take $e_1$.

$$
\begin{aligned}
&P((\ell_0,\nu),e_1)\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot P'(\sigma',e_1)dt_2dt_1\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{P'' \cdot \mathit{Remainder}}_{=0 \text{ until } e_1 \text{ is fireable}} + \underbrace{[\prod_{e\in Edge} (1-\mathit{fireable}(\sigma',e))] \cdot \mathit{Fin}}_{=1 \text{ until } e_1 \text{ is fireable or } e_2 \text{ but then } =0}\\
&\quad \cdot P'(succ_t(\sigma'),e_1)dt_2dt_1\\
&\quad \text{take time steps until } c_{r_1}=0\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{P'' \cdot \mathit{Remainder}}_{>0 \text{ because } e_1 \text{ is fireable}} + \underbrace{0}_{e_1 \text{ is fireable}} \; dt_2dt_1\\
&= \int_{t_1=0}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \underbrace{\frac{\mathit{fireable}((l,\nu,\nu_r),e_1) \cdot Wgt(e_1)}{\sum_{e\in Edge,\mathit{fireable}(\sigma',e)} Wgt(e)}}_{=P''=1 \text{ if } e_1 \text{ fireable now and } |supp(JmpT)|=\infty}\\
&\quad \cdot \underbrace{\mathit{Remainder}}_{=1} \; dt_2dt_1\\
&= \underbrace{\int_{t_1=0}^{\infty} 2e^{-2t_1}}_{\text{split at } 1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot P'' \cdot 1 \; dt_2dt_1
\end{aligned}
$$

$$= \int_{t_1=0}^{1} 2e^{-2t_1} \cdot \underbrace{\int_{t_2=0}^{\infty} 2e^{-2t_2}}_{=1,\ e_1 \text{ taken in any case}} \cdot P'' \cdot 1 \, dt_2 dt_1 + \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \underbrace{\int_{t_2=0}^{\infty} 2e^{-2t_2}}_{\text{split at } t_1 - 1} \cdot P'' \cdot 1 \, dt_2 dt_1$$

$$= \int_{t_1=0}^{1} 2e^{-2t_1} \cdot P'' \, dt_1 + \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \Big( \int_{t_2=0}^{t_1-1} 2e^{-2t_2} \cdot \underbrace{P''}_{=0 \text{ because } \neg\text{firable}(e_1)=0} dt_2$$

$$+ \int_{t_2=t_1-1}^{\infty} 2e^{-2t_2} \cdot P'' \, dt_2 \Big) dt_1$$

$$= \int_{t_1=0}^{1} 2e^{-2t_1} \cdot P'' \, dt_1 + \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=t_1-1}^{\infty} 2e^{-2t_2} \cdot P'' \, dt_2 dt_1$$

$$= (1 - e^{-2}) + \int_{1}^{\infty} 2e^{2-4t_1} dt_1$$

$$= (1 - e^{-2}) + \frac{e^{-2}}{2} \approx 0.9323$$

We have a fixed initial state in the example. Therefore, we can skip integrating over possible initial locations and valuations. We follow the formula to calculate the probability $P(e_1)$. We notice that the first summand will return 0 until $e_1$ is fireable, as this is part of the numerator of $P''$. At the same time, the second summand will recursively call $P'$ until we can take a jump. $Fin$ is always one since we can take an edge when their counter is 0. In this example, one of the edges' random variables will equal 0 after a finite time. We skip until an edge is fireable, i.e., to a time where $c_{r_1} = 0$ if we make sure that $e_1$ is the first edge enabled. Since initially $c_{r_1}$ is set to $t_1$ we wait $t_1$ $(+2)$ time until we can take the jump. Now the second summand will be 0 since an edge is fireable. In contrast, we notice that both the fraction $P''$ and the *Remainder* will equal one if $e_1$ is fireable. Thus we want to split the intervals so that $e_1$ is the first edge fireable and therefore taken. The semantics assures that edge $e_1$ is taken as long as $e_2$ is not taken before. In this example, there are two cases in which $e_1$ is the first edge fireable. For this, either its random variable sampled a value below or equal to 1, resulting in $e_1$ being fireable, no matter $e_2$'s random variable value. The other case is that $e_1$s random variable contains a value above 1, but it will still be smaller than $e_2$. Since $e_1$ is enabled one time unit before its competitor, we compare the value of $e_1$'s random variable minus 1. Therefore $c_{r_2}$ must be below $t_1 + 1$ to make sure that edge $e_1$ is taken before.

In both cases, we ensured that $e_1$ is the first edge fireable and therefore know that $P''$ and *Remainder* decompose to one. *Rst* will always reset $x$ to 0 and $P''$ will eventually be 1 if $e_1$ is enabled. We can now calculate the integrals and get a probability from 93.23 % to take $e_1$.

With the same calculations we obtain for $e_2$:

$$\begin{aligned} & P((\ell_0,\nu), e_2) \\ =& \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=0}^{t_1-1} 2e^{-2t_2} \cdot P'' \, dt_2 dt_1 \\ =& \int_{1}^{\infty} (2e^{2t_1} - 2e^{2-4t_1}) dt_1 \\ =& \frac{1}{2} e^{-2} \approx 0.0676 \end{aligned}$$

$$P((\ell_0,\nu), e_1) + P((\ell_0,\nu), e_2) = 1$$

Further, we compute the probability of executing a sequence of two edges. Therefore we calculate four combinations of taking the two different self-loops.

$$
\begin{aligned}
&P((\ell_0,\nu),e_1,e_1)\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P'(\sigma',e_1,e_1)dt_2dt_1\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot \underbrace{P''\cdot \mathit{Remainder}}_{=0 \text{ until } e_1 \text{ is fireable}} +\\
&\underbrace{[\prod_{e\in Edge}(1-\mathit{fireable}(\sigma',e))]\cdot \mathit{Fin}\cdot P'(\mathit{succ}_t(\sigma'),e_1,e_1)\,dt_2dt_1}_{=1 \text{ until } e_1 \text{ is fireable or } e_2 \text{ but then } =0}\\
&\text{take time steps until } c_{r_1}=0\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot \underbrace{P''\cdot \mathit{Remainder}}_{>0 \text{ because } e_1 \text{ is fireable}} + \underbrace{0}_{e_1 \text{ is fireable}} dt_2dt_1\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot \underbrace{\frac{\mathit{fireable}((l,\nu,\nu_r),e_1)\cdot \mathit{Wgt}(e_1)}{\sum_{e\in Edge,\mathit{fireable}(\sigma',e)}\mathit{Wgt}(e)}}_{=P''=1 \text{ if } e_1 \text{ fireable now and } |supp(JmpT)|=\infty}\cdot\underbrace{\int_{\nu'\in V}\delta(\nu')}_{=1}\\
&\cdot\int_{t_1'\geq 0} 2e^{-2t_1'}\cdot P'((l',\nu',\nu_r'[c_r\mapsto t_1']),e_1)dt_1'd\nu'dt_2dt_1\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''\cdot\int_{t_1'\geq 0} 2e^{-2t_1'}\cdot \underbrace{P''\cdot \mathit{Remainder}}_{\text{after timestep until } e_1 \text{ fireable again}} dt_1'dt_2dt_1\\
&=\int_{t_1=0}^{\infty} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''\cdot\int_{t_1'=0}^{\infty} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'dt_2dt_1\\
&=\underbrace{\int_{t_1=0}^{\infty}}_{\text{split this at 1}} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''\cdot\underbrace{\int_{t_1'=0}^{\infty}}_{\text{split this at 1}} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'dt_2dt_1\\
&=\int_{t_1=0}^{1} 2e^{-2t_1}\cdot\underbrace{\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''}_{=1,e_1,e_1 \text{ in any case}}\cdot\int_{t_1'=0}^{1} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'dt_2dt_1\\
&+\int_{t_1=0}^{1} 2e^{-2t_1}\cdot\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''\cdot\underbrace{\int_{t_1'=1}^{\infty} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'}_{\text{split into } \int_{t_1'=1}^{(t_2+1)}\cdots+\underbrace{\int_{t_1'=(t_2+1)}^{\infty}\cdots}_{=0}}dt_2dt_1\\
&+\int_{t_1=1}^{\infty} 2e^{-2t_1}\cdot\underbrace{\int_{t_2=0}^{\infty} 2e^{-2t_2}\cdot P''\cdot\int_{t_1'=0}^{1} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'dt_2}_{\text{split into } \underbrace{\int_{t_2=0}^{(t_1-1)}}_{=0}\cdots+\int_{t_2=(t_1-1)}^{\infty}\cdots}dt_1\\
&+\int_{t_1=1}^{\infty} 2e^{-2t_1}\cdot\underbrace{\underbrace{\int_{t_2=0}^{\infty}}_{\substack{\text{split at}\\(t_1-1)}} 2e^{-2t_2}\cdot P''\cdot\underbrace{\int_{t_1'=1}^{\infty}}_{\text{split at } (t_2-t_1+2)} 2e^{-2t_1'}\cdot P''\cdot \mathit{Remainder}\, dt_1'dt_2}_{=0 \text{ for all but } \int_{t_2=(t_1-1)}^{\infty}\cdots\int_{t_1'=1}^{(t_2-t_1+2)}\cdots}dt_1
\end{aligned}
$$

first $P''=1$ because $e_1$ is enabled and taken as first edge,

$P''\cdot \mathit{Remainder}=1$ because $e_1$ is enabled again and taken as second edge

$$
\begin{aligned}
&= \int_{t_1=0}^{1} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \int_{t'_1=0}^{1} 2e^{-2t'_1} dt'_1 dt_2 dt_1 \\
&+ \int_{t_1=0}^{1} 2e^{-2t_1} \cdot \int_{t_2=0}^{\infty} 2e^{-2t_2} \cdot \int_{t'_1=1}^{t_2+1} 2e^{-2t'_1} dt'_1 dt_2 dt_1 \\
&+ \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=(t_1-1)}^{\infty} 2e^{-2t_2} \cdot \int_{t'_1=0}^{1} 2e^{-2t'_1} dt'_1 dt_2 dt_1 \\
&+ \int_{t_1=1}^{\infty} 2e^{-2t_1} \cdot \int_{t_2=(t_1-1)}^{\infty} 2e^{-2t_2} \cdot \int_{t'_1=1}^{(t_2-t_1+2)} 2e^{-2t'_1} dt'_1 dt_2 dt_1 \\
&= (1-e^{-2})^2 \\
&\quad + (\frac{1}{2} \cdot e^{-2} \cdot (1-e^{-2})) \\
&\quad + (\frac{1}{2} \cdot e^{-2} \cdot (1-e^{-2})) \\
&\quad + \frac{1}{4} e^{-4} \\
&\approx 0.7476 \\
&\quad + 0.0585 \\
&\quad + 0.0585 \\
&\quad + 0.0046 \\
&\approx 0.8692
\end{aligned}
$$

In the following calculations we denoted $\int_{t=x}^{y} 2e^{-2t} dt$ with $\int_{x}^{y} E\ dt$ to have a better overview.

$$
\begin{aligned}
P(e_1,e_2) &= \int_0^1 E \int_0^\infty E \int_{t_2+1}^\infty E\ dt'_1 dt_2 dt_1 + \int_1^\infty E \int_{t_1-1}^\infty E \int_{t_2-t_1+2}^\infty E\ dt'_1 dt_2 dt_1 \\
&= \frac{1}{2} e^{-2} - \frac{1}{2} e^{-4} + \frac{1}{4} e^{-4} \\
&\approx 0.0631
\end{aligned}
$$

$$
\begin{aligned}
P(e_2,e_1) &= \int_1^2 E \int_0^{t_1-1} E \int_0^\infty E\ dt'_2 dt_2 dt_1 + \int_2^\infty E \int_0^{t_1-2} E \int_{t_1-t_2-2}^\infty E\ dt'_2 dt_2 dt_1 \\
&\quad + \int_2^\infty E \int_{t_1-2}^{t_1-1} E \int_0^\infty E\ dt'_2 dt_2 dt_1 \\
&= (\frac{1}{2} e^{-6} - e^{-4} + \frac{1}{2} e^{-2}) + (\frac{1}{2} e^{4} - \frac{1}{2} e^{6}) + \frac{1}{4} e^{-4} \\
&\approx 0.0631
\end{aligned}
$$

$$
\begin{aligned}
P(e_2,e_2) &= \int_2^\infty E \int_0^{t_1-2} E \int_0^{t_1-t_2-2} E\ dt'_2 dt_2 dt_1 \\
&= \frac{1}{4} e^{-4} \\
&\approx 0.0046
\end{aligned}
$$

The probability to take two executions $e$,$e' \in Edge$ after each other sums up to

$$P(\text{e,e'}) = P(e_1,e_1) + P(e_1,e_2) + P(e_2,e_1) + P(e_2,e_2) = 1$$

.

| P | e_1 | e_2 |
|---|---|---|
| e_1 | 0.869 | 0.0631 |
| e_2 | 0.0631 | 0.005 |

We have seen that the probabilities sum up to one for both one execution and two executions. Following the probability to stay in the state is 0 in this example. The reason is that we use exponential distributions that converge to 0 in this example, i.e. probability to stay forever in state is 0. It can be calculated using $1 - P(init,e_1) - P(init,e_2)$. If we do not cover the whole distribution, the remaining probability flows into other edges. If this happens for all jumps, we either stay in this location with a positive likelihood or have a deadlock if the invariant is violated.

With this example, we do not only illustrated the computations but also how we compose seemingly stochastically independent processes. We can see the example as an already composed system resulting from a parallel composition of two independent processes. We introduce stochastic dependencies between the processes by using shared variables for concurrent, seemingly stochastically independent processes. We can control the enabledness and disabledness of conditions with the shared variables.

## 3.6 Parallel Composition

We want to set up a model with the motivation of easy parallel composition. We designed the semantics in a way that it allows a composition, but we do not formulate it here. Instead, we focus on defining the conditions for parallel composition in detail. Composition $\mathcal{A}||\mathcal{A}'$ should be an SHA again, where we get the same paths from the composition as when we compose the paths of both systems separately. For this purpose, the components need to respect the controlled variables, therefore we introduce semantical composability. $\mathcal{A}$ and $\mathcal{A}'$ are semantically composable if their controlled variable and random process sets are disjoint.

**Definition 3.6.1** (Semantically composable)**.** *Two Stochastic Hybrid Automata $\mathcal{A}, \mathcal{A}'$ are semantically composable if $Con \cap Con' = \emptyset$ and $Lab \cap Lab' = \emptyset$*

We only allow controlled variables in invariants and guards, so the possible environmental behavior is fixed. Following, if $\mathcal{A}$ changed a variable's value, $\mathcal{A}'$s invariants and guards do not get violated. Even with the system only being able to change its controlled variables, we can still use our model for real-life applications. In reality, several components usually do not write the same value, but each member is an actor or a sensor. Also, we typically do not directly change a dynamic variable, and we can only change a control variable if needed. E.g., we cannot directly change the speed of an engine, but we can brake or accelerate more. There is also the possibility of implementing a writer for several components, which can write a value in a local variable. The writer can then read the local values and is the only one having writing access to the global variable.
In addition, we do not use label synchronization and only synchronize over variables. In our model, only one random process determines the time point for each synchronized jump. We could not assure this with label synchronization, then enabledness would depend on several components. The synchronized step can only have one random variable that cannot synchronize random values in that case. If we allowed only

one of the systems to act for each synchronization label and determine a random variable, then the second component might not be able to follow. To avoid these problems, we currently do not include label synchronization.
Furthermore, we want the resulting system to be well-formed and closed, so we do not lose probabilities on deadlocks and timelocks. The environment might change and read non-controlled variables in a closed system, but it doesn't affect the system's behavior anymore. The environment is fixed, non-determinism is eliminated, and the stochastic room is complete so we can define the probabilities.
A step in the parallel automaton means that one of the components does a discrete or a time step. At the same time, the other one does an environmental step, where the first component can change everything but the controlled variables. The further definition of the composition itself is left for future work.

# Chapter 4

# Alternative Design Options

Finally, we would like to present some alternative design ideas, some of which we did not choose to keep the model simple. First, we have made some changes to Hybrid Automata, which we want to discuss. One could choose not to split the variables into controlled and non-controlled. Then several components can claim write access of a variable simultaneously. We could try to restrict that using label synchronization, bringing new problems. For two jumps with the same label, it is not clear which one decides when we take the edge, as the probability that both are fireable at the same time is 0 for distributions with uncountable support.
We also restrict invariance and guard only to contain controlled variables to exclude some sources of deadlocks. If we allow non-controlled variables in both, modelers need to use them very carefully in invariants and guards not to cause any deadlocks in the composed system. We further assumed that the invariant and guard of a discrete step imply the invariant of the target location for simplicity, so we do not need to check if the invariant holds when taking an edge. We can relax this restriction by adding a suitable attribute in various places, like $guard$ and $isEn$. We have also decided to add weights to the edges in case multiple jumps are fireable at the same time.
We further do not allow the weight 0 to avoid an additional query if only edges with weight 0 are fireable, and we would have to divide by 0. Instead of using weights, we could also use priorities on the edges. We would then choose the edge with the highest priority with probability 1 —this way, we could express only taking a jump if no other jumps are enabled.
Concerning the design of the probability distribution, we decided to initialize all random variables at the very start with the initial state's distribution. Consequently, we also initialize edges that are never going to be enabled, and therefore the associated clocks will never decrease. An alternative would be to initialize them once the jump is enabled for the first time. We wanted to keep resetting consistently since we also reset them every other time after we take an edge and not before.
Last we had the idea to introduce a second distribution for each edge, which selects how many times a discrete step needs to be enabled and disabled before taking it. This way, we can model that we do not have to take an enabled jump immediately, but that an edge must be enabled several times before we take it, e.g., $e_1$ must be enabled 3 times and then we take it. In the end, we did not leave this distribution in the model to simplify it. We decided that we already had a similar effect with the race between jumps that we aimed to get with a second distribution. Moreover, we

dodge possible deadlocks happening if only a single jump is enabled, and according to the model, we should wait until it is enabled a second time.

# Chapter 5

# Conclusion

## 5.1 Summary

This thesis introduced the syntax and semantics of a modeling language for stochastic hybrid systems. In contrast to existing modeling approaches, we assign local stochastic distributions for stochastic processes and add these processes to the edges of our model. We extended Hybrid Automata to resolve non-determinism in a way to prepare the way for the composition of automata. We further reduced unwanted stochastical dependencies, deadlocks, and timelocks by introducing syntactical restrictions. We finally presented probability calculations for symbolic paths of the model. In summary, we designed a modeling language as the basis of a future definition of parallel composition.

The main challenge in this work was to find the degree between a user-friendly and simplified model and still have it as expressive and free as possible for modeling. The less error-prone we wanted to design the model, the more we had to limit it and make it more complicated. As we could not include the final composition definition, one might need to make further adjustments when defining the parallel construction. Another challenge of this work was defining the model in a way such that the probability later makes sense, and we can define it. In this course, we introduced controlled variables to the system, for example, to be able to define closed systems and get a deterministic model where the probability space is clearly defined. We have also seen that probability calculations can get complex and error-prone, even for a small example.

## 5.2 Future Work

Following this bachelor thesis, one can use the presented modeling language to define the parallel composition of models. Moreover, existing model checking algorithms can be adjusted to take our model as input language and conduct reachability analysis. Finally, it would be nice to compare the global and the local approaches in the aspects of practicality, modeling-friendliness, and expressiveness. Accordingly, it might be possible to define a way to transform one model into the other.

# Bibliography


[ACH+95] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical computer science*, 138(1):3–34, 1995.

[BBB+14] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, Quentin Menet, Christel Baier, Marcus Größer, and Marcin Jurdzinski. Stochastic timed automata. *arXiv preprint arXiv:1410.2128*, 2014.

[BK02] Falko Bause and Pieter S Kritzinger. *Stochastic Petri nets*, volume 1. Citeseer, 2002.

[FHH+11] Martin Fränzle, Ernst Moritz Hahn, Holger Hermanns, Nicolás Wolovick, and Lijun Zhang. Measurability and safety verification for stochastic hybrid systems. In *Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control*, HSCC '11, page 43–52, New York, NY, USA, 2011. Association for Computing Machinery.

[HPS+21] Jannik Hüls, Carina Pilch, Patricia Schinke, Henner Niehaus, Joanna Delicaris, and Anne Remke. State-space construction of hybrid petri nets with multiple stochastic firings. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 31(3):1–37, 2021.

[KNSS00] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *International Conference on Concurrency Theory*, pages 123–137. Springer, 2000.

[LP10] John Lygeros and Maria Prandini. Stochastic hybrid systems: a powerful framework for complex, large scale applications. *European Journal of Control*, 16(6):583–594, 2010.

[Spr00] Jeremy Sproston. Decidable model checking of probabilistic hybrid automata. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 31–45. Springer, 2000.

[Áb21] Erika Ábrahám. Lecture notes in modeling and analysis of hybrid systems. `https://ths.rwth-aachen.de/teaching/summer-term-2021/lecture-hybrid-systems/`, July 2021.