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

BACHELOR OF SCIENCE THESIS

COMPARING TWO MODELING FORMALISMS FOR STOCHASTIC HYBRID SYSTEMS

Matthias Appenzeller

Examiners:
Prof. Dr. Erika Ábrahám
Prof. Dr. ir. Dr. h. c. Joost-Pieter Katoen

Additional Advisor:
Dr. Stefan Schupp

Aachen, 11. Mai 2021

Abstract

In this thesis, we introduce stochastics to rectangular automata models and discuss two generalized formalisms for stochastic hybrid automata. Stochastic hybrid automata are based on probability distributions, that globally decide the timing when some transition is taken. We try to give an intuitive reasoning why those two stochastic hybrid automaton models using global probability distributions are equivalent. In contrast to the global models, we introduce a stochastic hybrid automaton model which is based on local probability distributions, using an approach similar to stochastic hybrid Petri nets. Instead of a global probability distribution for all transitions, this model uses a local probability distribution for each jump. Since this model is not very practical, problems of using the local approach are highlighted.

Erklärung

Hiermit versichere ich, dass ich die vorgelegte Arbeit selbstständig verfasst und noch nicht anderweitig zu Prüfungszwecken vorgelegt habe. Alle benutzten Quellen und Hilfsmittel sind angegeben, wörtliche und sinngemäße Zitate wurden als solche gekennzeichnet.

Aachen, den 11. Mai 2021

Contents

Chapter 1

Introduction

Stochastic hybrid automata allow modeling of complex real-time systems that describe the interaction between discrete-continuous behaviour and random behaviour influencing the discrete and continuous choices that are made in such a model. The model checking problem for stochastic hybrid automata asks the question, whether the probability to reach states with certain properties is within some bounds [FHH$^+$11].

The need for stochastic hybrid automata arises from the issue of nondeterminism in hybrid automata. Hybrid automata involve uncertainities which need to be resolved. There have been several approaches to resolve this nondeterminism by probability distributions already. Jeremy Sproston for example solved the nondeterminism, when multiple transitions are enabled at the same time, by adding discrete probability distributions [Spr00]. The issue, when a transition can be taken and what transition can be taken after that time has been analyzed for stochastic timed automata [BBB$^+$14]. For rectangular automata, no stochastic model existed yet. The nondeterministic choices in rectangular automata involving when a discrete transition will be taken, with what rate the continuous time evolves, what transition will be taken and how the values of variables are reset, still need to be resolved. This thesis will deal with this topic by summarizing existing hybrid automaton models to then be able to introduce a new model, stochastic rectangular automata. Stochastic rectangular automata are a combination of the existing rectangular automata and stochastic timed automata and will be used to find a general definition of stochastic hybrid automata using global probability distributions. Due to numerous different existing formalisms, our objetive is to provide a unification that compares these formalisms and highlights problems.

Other related work dealt with hybrid Petri nets [HPS$^+$19]. In that model, stochastics have been introduced not by global probability distributions as in the hybrid automata cases, but instead using local probability distributions. Due to the control modes in Petri nets running in parallel, probability distributions needed to be introduced locally for each event. This raised the question, whether a local approach could be used to model stochastic hybrid automata too and will be dealt with in this thesis.

Chapter 2

Preliminaries

2.1 Timed Automata

In hybrid automata, both discrete modes and continuous time is used to model real-time systems. The following preliminaries , especially syntax and semantics of several modeling formalisms that will be introduced, are taken from the lecture "Modeling and Analysis of Hybrid Systems" [Ábr12] to build the foundation of this thesis.
The model consists of discrete modes and a continuous time evolution within those modes. With discrete transitions or jumps, the discrete mode can be changed. Clocks are used to measure continuous time and automata will be used to model the discrete modes of the system. Each clock has a value that continuously develops with time, the change over time is defined by a real-valued flow function. For a clock x, the flow is denoted as its derivative $\dot{x}$; hence, the flow in the below timed automata definition is implicitely set to $\dot{x} = 1$ to indicate a clock rate of 1.
Timed automata use clocks from a set $\mathcal{C}$ to model real-time systems. The clock valuation is a map $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$ assigning a real value to each clock. In the following thesis, we also write $g \subseteq \mathbb{R}^n$ and we also write $\nu \in g$ to denote that the real vector of variables values defined by ν is in g, assuming a silently fixed order of the variables. Let x be a clock and $c \in \mathbb{N}$ a number. The value of x can be compared to c, denoted by $x \sim c$ with $\sim$ being an element of $\{<, \leq, =, \geq, >\}$. We call comparisons atomic and conjunctions of those atomic clock constraints build a set of clock constraints $\mathrm{CC}(\mathcal{C})$. Jumps are guarded by clock constraints and might reset part of the clocks to zero. Discrete modes are restricted by clock constraints called invariants.
In the following definition, a labelling of atomic propositions to model parallel construction will be left out. For timed automata, it is easy to introduce but it is not trivial anymore for following models using stochastics.

Definition 2.1.1 (Syntax of Timed Automata). *A timed automaton (TA) is a tuple* $A = (Loc, \mathcal{C}, Edge, Inv, Init)$ *with*

- *Loc is a finite set of locations;*
- *$\mathcal{C}$ is a finite set of real-valued clocks. Let V denote the set of clock valuations* $\nu : \mathcal{C} \to \mathbb{R}_{\geq 0}$.

  A state is a pair (l,ν) *where* $l \in Loc$ *and* $\nu \in V$. Σ *denotes the set of states;*

- *Edge is a finite set of transitions. A transition $\epsilon \in$ Edge is a triple (ℓ, ρ, ℓ') such that ℓ, $\ell' \in$ Loc and ρ is the edge relation: $\rho \in CC(\mathcal{C}) \times 2^{\mathcal{C}}$ such that every edge consists of a clock constraint(guard) and a set of clocks to be reset to zero upon taking the edge;*

- *Inv is a labeling function, assigning an invariant $Inv(l) \subseteq V$ to each location $l \in$ Loc;*

- *Init is a set of initial states Init $\subseteq \Sigma$ with $\forall$ $(l, \nu) \in$ Init and $x \in \mathcal{C}$. $\nu(x) = 0$.*

In the graphical interpretation of A, vertices represent pairs of a location and its invariant, edges model the transitions between the vertices and clocks generate constraints for enabling or disabling edges.
In a timed automaton, there are two possibilities for a clock constraint: it can either be used as an invariant for locations to set a condition for staying in a location or it can be used as a guard for edges in order to form an enabling/disabling criterion for edges. The flow for all clocks $x \in \mathcal{C}$ is limited to $\dot{x} = 1$. All clocks proceed at rate 1, thus a flow function can be omitted in the definition.

Example 2.1.1. *This figure shows a timed automaton A_1 = (Loc_1, C_1, $Edge_1$, Inv_1,$Init_1$) where:*

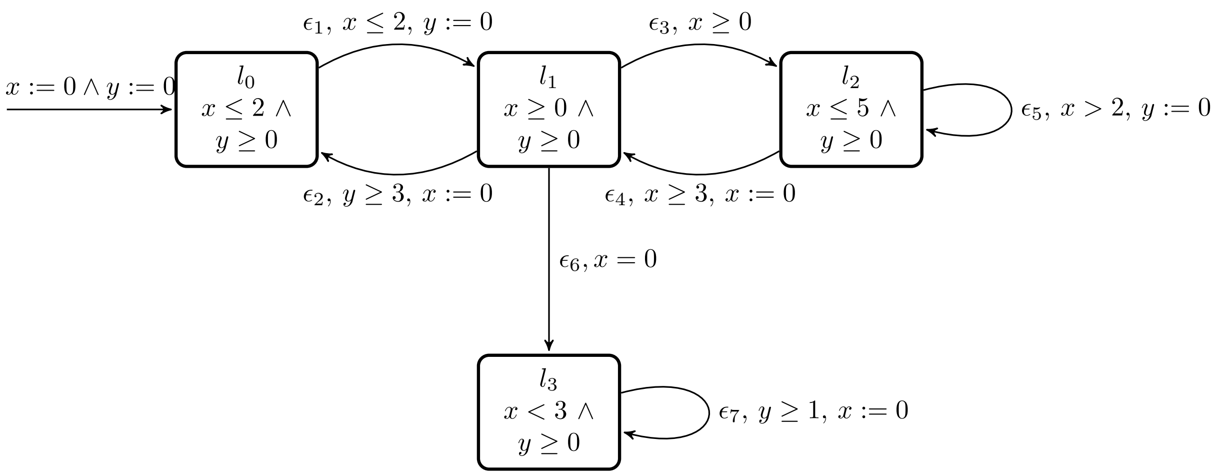
- *Loc_1 = {ℓ_0, ℓ_1, ℓ_2, ℓ_3 },*

- *C_1 = {x, y},*

- *$Edge_1$ = {*
  (ℓ_0, ϵ_1, $x \leq 2$, {y}, ℓ_1),
  (ℓ_1, ϵ_2, $y \geq 3$, {x}, ℓ_0),
  (ℓ_1, ϵ_3, $x \geq 0$, {}, ℓ_2),
  (ℓ_2, ϵ_4, $x \geq 0$, {x}, ℓ_1),
  (ℓ_2, ϵ_5, $x > 2$, {y}, ℓ_2),
  (ℓ_1, ϵ_6, $x = 0$,{}, ℓ_3),
  (ℓ_3, ϵ_7, $y \geq 1$, {x}, ℓ_3)
  }

- *$Inv_1(\ell_0) = x \leq 2 \wedge y \geq 0$, $Inv_1(\ell_1) = x \geq 0 \wedge y \geq 0$, $Inv_1(\ell_2) = x \leq 5 \wedge y \geq 0$, $Inv_1(\ell_3) = x < 3 \wedge y \geq 0$,*

- *$Init_1$ = (ℓ_0, $x=0 \wedge y=0$).*



This timed automaton models a system with one bad location ℓ_3. Since ℓ_3 only has one outgoing transition ϵ_7 that is a self loop, once the system reaches ℓ_3 it can not get out. The transitions describe the discrete jumps between locations. For example, the system can go from ℓ_0 to ℓ_1 with transition ϵ_1, if $x \leq 2$ is satisfied and the invariant of the target location is satisfied. Since the invariant of ℓ_0 requires $x \leq 2$, the system has to take the transition ϵ_1 within 2 time steps, because the invariant of a location may not be violated. Upon taking the transition ϵ_1 $y := 0$ is executed, meaning the clock value of y is reset to 0.

Now that the syntax of timed automata has been introduced, semantics are necessary. Since timed automata are a subclass of rectangular automata, the semantics will be omitted here and introduced in the next section.

2.2 Rectangular Automata

With timed automata, the clock valuations were limited to linear evolution, especially only allowing $\dot{x} = 1$ for a clock $x \in \mathcal{C}$. This limitation changes with rectangular automata.
Rectangular automata are timed automata with two major extensions: The clock value can change within values from a rectangular interval and the clock reset can set clocks to arbitrary values, not only to zero. All invariants, activities and valuations are described by rectangular sets. A set $R \subset \mathbb{R}^n$ is called rectangular if it is a cartesian product of (possibly unbounded) intervals, all of whose finite endpoints are rational. $\mathcal{R}^n$ denotes the set of all n-dimensional rectangular sets [Ábr12].

Definition 2.2.1 (Syntax of Rectangular Automata). *A rectangular automaton (RA) $\mathcal{A}$ is a tuple $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ with*

- *Loc is a finite set of locations;*
- *$\mathcal{C}$ is a finite set of real-valued variables.*
  A state is a pair (l,ν) where $l \in Loc$ and $\nu \in V$. Σ denotes the set of states;
- *$Edge \subseteq Loc \times \mathcal{R}^n \times \mathcal{R}^n \times 2^{\mathcal{C}} \times Loc$ is a set of transitions;*

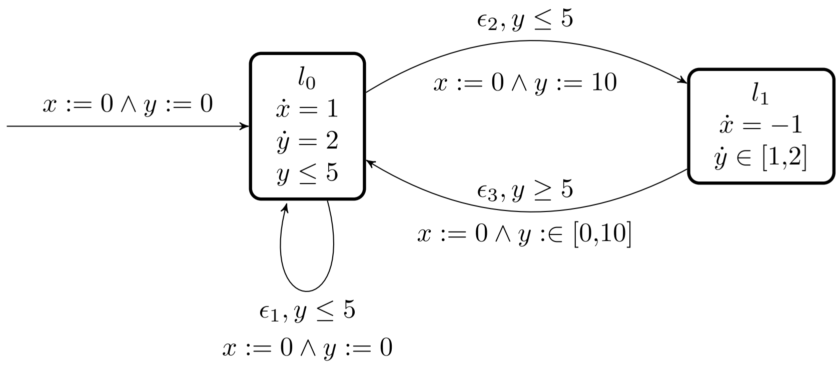
- *Act: Loc* $\rightarrow \mathcal{R}^n$ *is a flow function*

- *Inv is a labeling function, assigning an invariant* $Inv(l) \in \mathcal{R}^n$ *to each location* $l \in Loc$*;*

- *Init is a set of initial states Init* $\subseteq \Sigma$ *with* $\bigcup_{s \in Init}(l,\nu)$ *such that* $\nu \in \mathcal{R}^n$ *and* $\nu \in Inv(l)$*.*

The definitions of timed and rectangular automata are similar, but with rectangular automata the flow function Act is added to the definition. Act(ℓ) consists of first time derivatives of the flow trajectories of all variables in location $\ell \in Loc$. A transition $\epsilon \in Edge$ is a triple (ℓ, ρ, ℓ') such that $\ell, \ell' \in Loc$ and ρ is the edge relation: $\rho \in \mathcal{R}^n \times \mathcal{R}^n \times 2^{\mathcal{C}}$. In contrast to timed automata, jumps can reset values to arbitrary values of a rectangular set, not only to zero. A jump $(l, g, r, jump, l')$ is enabled if the valuation satisfies the guard g, the successor valuation of variables in r satisfies the invariant of l' and the values of variables not in r do not change.

Example 2.2.1. *This example shows the difference between timed automata and rectangular automata. The difference is the set of values for all activities and transitions. In Example 2.1.1, all clock derivatives were restricted to 1. With rectangular automata, for example in* ℓ_0*, derivatives like* $\dot{y}$*=2 are allowed. Furthermore, for a transition, the reset set is allowed to be arbitrary. Upon taking a transition, a variable does not have to be reset to 0, but can be reset to any value like the value of y is set to 10 in* ϵ_2*.*



2.2.1 Semantics of Rectangular Automata

Now that the base model for this thesis has been introduced, operational semantics of rectangular are necessary to allow describing executions on rectangular automata. Furthermore, a path definition needs to be defined.

Definition 2.2.2 (Semantics of Rectangular Automata)**.** *The semantics of a rectangular automaton $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ consists of discrete instantaneous steps (jumps) and continuous time steps (flow):*

1. *Discrete step semantics*

$$\frac{\begin{array}{c} e = (\ell, (g, r), jump, \ell') \in Edge \\ \nu \in g \quad \nu' \in r \quad \forall i \notin jump. \nu(i)' = \nu(i) \quad \nu' \in Inv(l') \end{array}}{(\ell, \nu) \xrightarrow{e} (\ell', \nu')} Rule_{discrete}$$

for state $s = (l, \nu) \in \Sigma$ and for each discrete transition $e = (l, (g, r), jump, l') \in Edge$ of l we say that e is enabled if and only if the valuation ν satisfies the guard g, the system may take the transition e and meanwhile update the value of variables in $jump$ to r only if updated valuation ν' does not violate the invariant of target location.

2. *Time step semantics*

$$\frac{(t = 0 \wedge \nu = \nu') \vee (t > 0 \wedge (\nu' - \nu)/t \in Act(l)) \quad \nu' \in Inv(l)}{(l, \nu) \xrightarrow{t} (l, \nu')} Rule_{time}$$

for state $s = (l, \nu) \in \Sigma$ the system may stay at location for t time steps and update valuation to ν' only if either t is zero and the clock valuation does not change or the ratio of difference between ν' and ν to t meets the continuous activities given by the function Act(l) and ν' does not violate the invariant of l.

Given a rectangular automaton $\mathcal{H} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ with states set Σ, a finite run [AD94] ϱ of $\mathcal{H}$ is a sequence of transitions

$$\varrho = s_0 \xrightarrow{t_1, e_1} s_1 \xrightarrow{t_2, e_2} \dots \xrightarrow{t_n, e_n} s_n$$

where $n \in \mathbb{N}$, $(s_i) = (l_i, \nu_i) \in \Sigma$ with $0 \leq i \leq n$, such that for all $i \in [1, n]$ there is a σ_i such that $\sigma_i \xrightarrow{t_i} \sigma_i' \xrightarrow{e_i} \sigma_{i+1}$. We say that n is the length of the run ϱ. We use $Run_f(\mathcal{H}, s_0)$ to denote the set of all finite runs from s_0 in $\mathcal{H}$. Similarly an infinite run [AD94] ϱ of $\mathcal{H}$ is a sequence of transitions without end state

$$\varrho = s_0 \xrightarrow{t_1, e_1} s_1 \xrightarrow{t_2, e_2} s_2 \xrightarrow{t_3, e_3} \dots$$

with $(s_i) \in \Sigma\ \forall i \in \mathbb{N}$, $(t_i)_{i \in \mathbb{N}^+} \in \mathbb{R}^{\geq 0}$ and $(e_i)_{i \in \mathbb{N}^+} \in Edge$, satisfying the above requirements. We use $Run(\mathcal{H}, s_0)$ to denote the set of all infinite runs from s_0.

Definition 2.2.3 (Direct Successors [BK08]). *Let $\mathcal{H} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a rectangular automaton. For $s \in \Sigma$ and $e \in Edge$, the set of direct e-successors of s is defined as:*

$$Post(s,e) = \{s' \in \Sigma \mid \exists t \in \mathbb{N}. s \xrightarrow{t,e} s'\},$$

and the set of direct successors of s is defined as:

$$Post(s) = \bigcup_{e \in Edge} Post(s,e).$$

Given a rectangular automaton $\mathcal{H} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ with states set Σ, a finite path $\widehat{\pi}(s_0, e_1 \dots e_n)$ of $\mathcal{H}$ is a set of all finite runs starting from s_0 by taking $e_1, e_2, \dots, e_n$, i.e. $\widehat{\pi}(s_0, e_1 \dots e_n) = \{s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} \dots \xrightarrow{t_n,e_n} s_n \in Run_f(\mathcal{H}, s_0)\}$. An infinite path $\pi(s_0, e_1 e_2 \dots)$ of $\mathcal{H}$ is a set of all infinite runs that start from s_0 and take $e_1, e_2, \dots$ in sequence, we define $\pi(s_0, e_1 e_2 \dots) = \{s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} s_2 \xrightarrow{t_3,e_3} \dots \in Run(\mathcal{H}, s_0)\}$.
These definitions will be used later in this thesis to calculate path probabilities for stochastic hybrid automata.

2.3 Probabilistic Rectangular Automata

A problem with rectangular automata involves nondeterminism. When multiple outgoing transitions from one location are enabled at the same time, one transition is taken nondeterministically. To solve this nondeterminism, probabilistic rectangular automata extend rectangular automata with discrete probability distributions. In that case, a random factor decides which transition is taken.

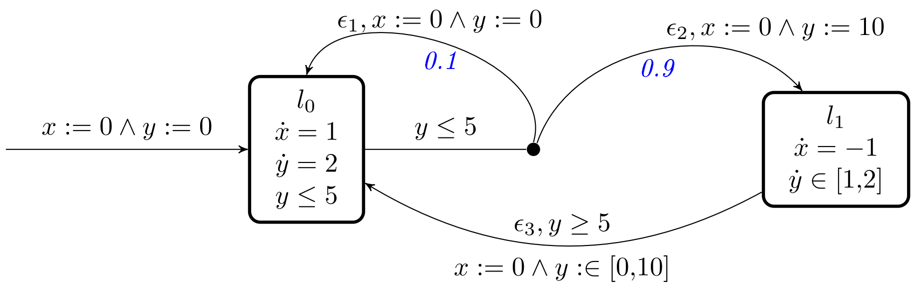
Definition 2.3.1 (Syntax of Probabilistic Rectangular Automata). *A probabilistic rectangular automaton (PRA) is a tuple $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, P)$ where*

- *(Loc, $\mathcal{C}$, Edge, Act, Inv, Init) is a rectangular automaton.*
- *P is a transition probability function $P : Edge \to [0,1]$ such that:*

$$\forall l \in Loc\, \forall g \in \{g \in R^n \mid \exists e = (l,g,r,jump,l') \in Edge\}. \left(\sum_{e=(l,g,r,jump,l') \in Edge} P(e) = 1 \right)$$

Example 2.3.1. *This example illustrates the difference between rectangular automata and probabilistic rectangular automata. In Example 2.2.1, the rectangular automaton had a nondeterministic choice. At location ℓ_0, two discrete transitions can be enabled at the same time. Discrete transitions or jumps in probabilistic rectangular automata are urgent, meaning if a jump is enabled, some jump must be taken. If a jump should be taken, then the probabilistic automaton resolves this nondeterminism by a probability distribution, marked with blue in the graphical representation. The two outgoing transitions of ℓ_0 now have a probability, meaning that with a probability of 0.9, the system will take transition ϵ_2 and with a probability of 0.1, it will take transition ϵ_1*

and stay in ℓ_0.



Now that the syntax of probabilistic rectangular automata has been introduced, semantics are necessary. For the semantics, urgency states that when a transition is enabled, some jump is taken. When multiple different guards are enabled at the same time, a random guard is chosen.
Since probabilistic rectangular automata can still contain nondeterminism in the derivatives of variables and in the reset sets, we can not clearly define semantics yet. In the above example, in location l_1, the derivative of y can be within the rectangular set [1,2]. This nondeterminism can not be resolved by using the discrete probability distributions of probabilistic rectangular automata, hence a model using continuous probability distributions needs to be introduced.

Chapter 3

Global Probability Distributions

3.1 Stochastic Rectangular Automata

In this section, the main model of this bachelor thesis - stochastic rectangular automata - will be introduced. Stochastic rectangular automata are based on rectangular automata [Ábr12] and stochastic timed automata [BBB$^+$14]. It involves both nondeterministic and probabilistic behaviour. Stochastic Rectangular Automata are an extension of Probabilistic Rectangular Automata to continuous state spaces and continuous probability measures. The motivation behind introducing them is that with timed automata, jumps are not always urgent. This is not always the case in practice, for example the measuring of time with digital clocks might be less precise than the timed automaton demands. With continuous probability measures, this behaviour can be approximated so that stochastic hybrid automata provide a less strict handling than timed automata.

This thesis restricts the flow of clocks within stochastic rectangular automata to linear behaviour, meaning derivatives of clock valuations are only allowed to be constant upon entering a location. In this case, for each location, each outgoing transition is only enabled for one continuous, possibly infinitely large interval. That is due to the convex guards of a transition only being able to be satisfied at most once, and then disabled at most once.
When allowing non-linear behaviour for stochastic rectangular automata, the interval of enabledness for a transition is not necessarily continuous. A flow $\dot{x}$ = [-1, 1] for example allows to enable and disable a transition multiple times. This would create a nondeterminism that cannot be solved with here proposed formalisms.

In the following paragraphs, the definitions of stochastic timed automata [BBB$^+$14] will be used and adapted to rectangular automata to define models for stochastic rectangular automata, which resolve all possible nondeterministic choices. The possible nondeterministic choices are the same as in [BBB$^+$14], namely when a transition is taken and which transition is taken. Since we introduce stochastics to rectangular automata and not timed automata, derivatives of variables and resets can be within rectangular sets. This yields two new nondeterministic possibilities that need to be solved, random derivatives for variables and random resets. First, the intervals of enabledness for transitions will be defined to build a foundation for a probability distribution over delays, deciding when some transition is taken. As in [Spr00], this interval will be extended from a time interval to a state set. We will define a set of states that can be reached with any derivative in any amount of time, such that some jump from those states can be taken.

Definition 3.1.1 (Interval of enabled transitions).
Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a rectangular automaton with $s \in \Sigma$ a state of $\mathcal{A}$ and $e \in Edge$. The set $I(s,e) = \{s' \in \Sigma \mid \exists s'' \in \Sigma . s \xrightarrow{t} s' \xrightarrow{e} s''\}$ describes all states which can be reached from a state s by waiting an amount of time with arbitrary derivatives of the variables, such that a discrete transition can be taken from those states. $I(s) = \bigcup_{e \in Edge} I(s,e)$ is a finite union of the sets over all outgoing edges describing all possible intervals with all possible derivatives where a jump can be taken.

To define the stochastic processes behind stochastic rectangular automata, an intuitive understanding of the behaviour of such an automaton is helpful. Stochastic rectangular automata involve both continuous flows and discrete transitions. The stochastic process, from a state s, first chooses a random delay with random variable derivatives to stay inside the same location, then a random transition out of all at that state enabled transitions is chosen and randomly reset according to the transition relation. Now, probability distributions over delays, which is a combination of when a jump will be taken and the derivatives of the variables, will be defined.

Definition 3.1.2 (Probability Distribution over Delays).
Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a non-blocking rectangular automaton and let $s \in \Sigma$ be a state of $\mathcal{A}$. The probability distribution on state s over delays is a probability measure μ_s over $\mathbb{R}^{\geq 0}$ such that:

(H.1) $\mu_s(I(s)) = \mu_s(\mathbb{R}^{\geq 0}) = 1$, where $I(s)$ is not empty as $\mathcal{A}$ is non-blocking. For a state s, the probability of eventually taking an arbitrary enabled transition is 1. The probability of taking a transition outside of those intervals is 0, meaning $\mu_s(\mathbb{R}^{\geq 0} \setminus I(s)) = 0$;

(H.2) Let λ be the standard Lebesgue measure on $\mathbb{R}^{\geq 0}$. If $\lambda(I(s)) > 0$, then μ_s is equivalent to λ on $I(s)$. Otherwise, μ_s is equivalent to the uniform distribution over points of $I(s)$.

Notice that the probability measure μ_s is defined on Borel $\sigma - algebra$.

In this thesis, we decided to choose the global probability distribution according to the intervals for transitions. Depending on the interval, the probability distribution will be one of three cases:

- The interval contains finitely many points $\rightarrow$ the probability distribution is only based on weights on transitions, so it will be computed like the probabilistic case.
- The interval is dense, but bounded $\rightarrow$ the probability distribution is the uniform distribution.
- The interval is unbounded $\rightarrow$ the probability distribution is the exponential distribution.

This global probability distribution imtroduces probabilities to the nondeterministic choices of derivatives and time passage. To resolve the nondeterminism of choosing a transition when multiple transitions are enabled at the same time, a probability distribution over transitions using weights will be defined. By dividing the weight of the discrete transition to be taken with the sum of weights of all enabled transitions, a normalized factor for the probability distributions can be found.

Definition 3.1.3 (Probability Distribution over Transitions).
Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a non-blocking rectangular automaton and let $s \in \Sigma$ be a state of $\mathcal{A}$. The probability distribution p_s over transitions, such that for every transition e, $p_s(e) > 0$ iff e is enabled in s, is given by weights on transitions. By assigning a weight $w(e) > 0$ to each transition $e \in Edge$ the probability of taking e is defined as:

$$e \textit{ is enabled in } s \iff p_s(e) = \frac{w(e)}{\sum_{e' \textit{ is enabled in } s} w(e')} > 0.$$

Now that probability distributions for time passage, variable derivatives and transitions have been defined, the only possibility for nondeterminism in rectangular automata is given by resets on transitions.
After a discrete transition out of all enabled transitions is chosen, the continuous state can randomly be reset to values from a rectangular set. To solve this nondeterminism, another probability distribution over resets has to be introduced, which will be done by defining a probability distribution over the set of all possible resulting states, after a reset was executed.

Definition 3.1.4 (Successor States).
Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a rectangular automaton with $s = (l,\nu) \in \Sigma$ a state of $\mathcal{A}$ and $e = (l,g,r,jump,l') \in Edge$. Let $s' \in I(s,e)$. The set of successor states $Succ(s,e) = \{(l', \nu')$ with $\nu' \in V \mid \nu' \in r \wedge \forall c \in \mathcal{C} \setminus \{jump\}. \nu'(c) = \nu(c) \wedge \nu' \in Inv(l')\}$ is the set of possible resulting states after a discrete transition is taken and a reset is executed.

Definition 3.1.5 (Probability Distribution over Resets).
Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ be a non-blocking rectangular automaton. Let $s \in \Sigma$ be a state of $\mathcal{A}$ and $e \in Edge$. The probability distribution on state s and discrete transition e over resets is a probability measure $\psi_{s,e}$ such that $\psi_{s,e}(Succ(s,e)) = 1$. For a state s, the probability of eventually resetting to an arbitrary successor state is 1. The probability of taking a transition outside of this set is 0. The probability for each successor state $s' \in Succ(s,e)$ is uniformly distributed.

With probability measures over delays, transitions and resets of rectangular automata, a model for stochastic rectangular automata can be defined.

Definition 3.1.6 (Syntax of Stochastic Rectangular Automata)**.** *A stochastic rectangular automaton (SRA) is a tuple $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \mu, W, \psi, \iota_{init})$ with states set Σ where:*

- *$(Loc, \mathcal{C}, Edge, Act, Inv, Init)$ is a rectangular automaton;*
- *μ is a set of probability distributions over delays and derivatives: $\mu = \bigcup_{s\in\Sigma}\{\mu_s\}$ where μ_s is defined as in Definition 3.1.2;*
- *W is a set of weight functions: $W = \bigcup_{e\in Edge}\{w(e)\}$, the probability $p_s(e)$ is defined as in Definition 3.1.3;*
- *ψ is a set of probability distributions over resets: $\psi = \bigcup_{s\in\Sigma, e\in Edge}\{\psi_{s,e}\}$ where $\psi_{s,e}$ is defined as in Definition 3.1.5;*
- *ι_{init} is an initial distribution over the initial states Init.*

To the definition of rectangular automata, a set of global probability distributions for each state has been added, a set of weight functions, adding weights to each transition and a set of probability distributions over resets. To understand executions of stochastic rectangular automata, operational semantics for rectangular automata are defined as in [Ábr12].
Operational semantics of stochastic rectangular automata are similar to operational semantics of rectangular automata. The only differences to the operational semantics of rectangular automata are that the continuous flow within a location is limited to being a constant value for each clock with an additional stochastic part that defines probabilities of paths. To make use of the probability distributions and weights, a formula to calculate probabilities for paths will be introduced as in [BBB+14], with the extension of using rectangular automata instead of timed automata.

Definition 3.1.7 (Probability Distribution over Finite Paths of SRA)**.** *Let $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \mu, W, \psi, \iota_{init})$ with states set Σ be a stochastic rectangular automaton and let $\widehat{\pi}(s_0, e_1 \dots e_n)$ be a finite path of A. The probability of $\widehat{\pi}(s_0, e_1 \dots e_n)$ starting from s_0 and taking $e_1, e_2, \dots, e_n$ in sequence is defined as:*

$$\mathbb{P}_A(\widehat{\pi}(s, e_1 e_2 \dots e_n)) = \int_{s'\in I(s,e_1)} \mu_s(s') \cdot p_{s'}(e_1) \cdot \int_{s''\in Succ(s',e_1)} \psi_{s',e_1}(s'') \cdot \mathbb{P}_A(\widehat{\pi}(s'', e_2 \dots e_n)) ds' ds''$$

where s' is the state that results by staying at the location of s for a certain time and changes the valuation according to the derivatives of the location. Initially we define $\mathbb{P}_A(\widehat{\pi}(s_0)) = 1$. The integrals are to be understood as polytopes.

For each consecutive transition in the path $\widehat{\pi}$, two new integrals will be added. The first integral describes the state $s' \in I(s)$, with s being the current state of the system, solving the nondeterminism of when a transition is taken and the derivative of variables, and a probability distribution over normalized weights randomly chosing what transition will be taken. The second integral describes the possible resulting states s'' and gives a probability for the resets. Iterating over all transitions yields a probability of taking the path $\widehat{\pi}$.

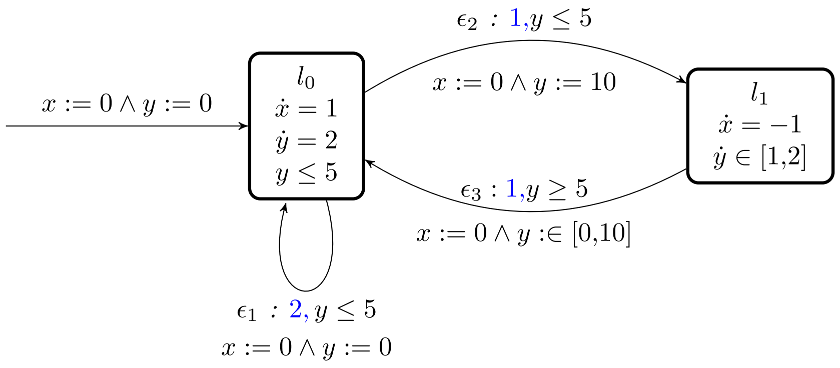
Example 3.1.1. *This figure shows a simple stochastic rectangular automaton* $A = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \mu, W, \psi, \iota_{init})$*. Assume that* $s_0 = (l_0, \{x = 0, y = 0\})$ *is a state of* A*,* μ_{s_0} *is the uniform distribution over* $I(s_0)$ *and* $\iota_{init}(s_0) = 1$*. The weight of each transition is marked with numbers in blue. The probability of starting from* s_0 *and taking* ϵ_1, ϵ_1 *in sequence is:*

$$\mathbb{P}_A((s_0, \epsilon_1\epsilon_1)) = \iota(s_0) \cdot \mathbb{P}_A(\widehat{\pi}(s_0, \epsilon_1\epsilon_1))$$

Using 3.1.7 yields following calculation:

$$=1 \cdot \int_{s_1 \in I(s_0,\epsilon_1)} \mu_{s_0}(s_1) \cdot p_{s_1}(\epsilon_1) \cdot \int_{s_2 \in Succ(s_1,\epsilon_1)} \psi_{s_1,\epsilon_1}(s_2) \cdot \mathbb{P}_A(\widehat{\pi}(s_2, \epsilon_1)) ds_1 ds_2$$
$$= \int_{s_1 \in I(s_0,\epsilon_1)} \frac{1}{|I(s_0)|} \cdot \frac{w(\epsilon_1)}{w(\epsilon_1) + w(\epsilon_2)} \cdot 1 \cdot \mathbb{P}_A(\widehat{\pi}(s_2, \epsilon_1)) ds'$$
$$= \int_{s_1 \in I(s_0,\epsilon_1)} \frac{1}{2.5} \cdot \frac{2}{3} \cdot \mathbb{P}_A(\widehat{\pi}(s_2, \epsilon_1) ds'$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \int_{s_1 \in I(s_0,\epsilon_1)} \cdot \left(\int_{s_3 \in I(s_2,\epsilon_1)} \mu_{s_2}(s_3) \cdot p_{s_3}(\epsilon_1) \cdot \int_{s_4 \in Succ(s_3,\epsilon_1)} \psi_{s_3,\epsilon_1}(s_4) \cdot \mathbb{P}_A(\widehat{\pi}(s_4)) ds_3 ds_4 \right) ds_1$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \int_{s_1 \in I(s_0,\epsilon_1)} \cdot \left(\int_{s_3 \in I(s_2,\epsilon_1)} \frac{1}{|I(s_2)|} \cdot \frac{w(\epsilon_1)}{w(\epsilon_1) + w(\epsilon_2)} \cdot 1 \cdot \mathbb{P}_A(\widehat{\pi}(s_4)) ds_3 \right) ds_1$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \int_{s_1 \in I(s_0,\epsilon_1)} \cdot \left(\int_{s_3 \in I(s_2,\epsilon_1)} \frac{1}{2.5} \cdot \frac{2}{3} \cdot \mathbb{P}_{A_2}(\widehat{\pi}(s_4)) ds_3 \right) ds_1$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \int_{s_1 \in I(s_0,\epsilon_1)} \cdot \left(\frac{1}{2.5} \cdot \frac{2}{3} \int_0^{2.5} 1 ds_3 \right) ds_1$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \int_0^{2.5} \cdot \left(\frac{1}{2.5} \cdot \frac{2}{3} \cdot 2.5 \right) ds_1$$
$$= \frac{1}{2.5} \cdot \frac{2}{3} \cdot 2.5 \cdot \frac{2}{3}$$
$$= \frac{4}{9}$$

The probability of starting from s_0 *and taking transition* ϵ_1 *n-times is* $\mathbb{P}_{A_2}((s_0, \epsilon_1^n)) = 1 \cdot \left(\frac{2}{3}\right)^n$ *with* $n \in \mathbb{N}$*. The higher* n *is, the lower is the probability to continuously take* ϵ_1*.*



Due to the resets and derivatives in this example being deterministic, the calculation is easier. The second integral for the reset always evaluates to 1 and the probability distribution μ over the time and the derivatives evaluates to a probability distribution purely based on the time interval, in this case [0,2.5].

3.2 Stochastic Hybrid Automata with Global Probability Distributions

In the first half of this chapter, the model of stochastic rectangular automata was described with a global probability distribution that will randomly sample a derivation and a delay among all possible derivations and delays, then pick a transition out of all enabled transitions and then reset the variables according to the transition relation. In this section, a generalized definition to stochastic hybrid automata will be introduced using two different modeling formalisms.
The first modeling approach is inspired by the stochastic rectangular automaton model. The definitions are almost the same, the only difference being that the underlying automaton is not restricted to being rectangular, but it can be an arbitrary hybrid automaton. The difference to rectangular automata is that all values and variables which were given by rectangular sets in rectangular automata can now be arbitrary values from a set of real values.
For simplicity, in the following section we restrict the model to unique derivatives and resets.

Definition 3.2.1 (Syntax of Stochastic Hybrid Automata with Global Probability Distributions). *A stochastic hybrid automaton (SHA) using global probability distributions is a tuple $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \mu, W, \psi, \iota_{init})$ with states set Σ where:*

- *$(Loc, \mathcal{C}, Edge, Act, Inv, Init)$ is a hybrid automaton with*
  - *Loc is a finite set of locations;*
  - *$\mathcal{C}$ is a finite set of real-valued variables;*
  - *$Edge \subseteq Loc \times 2^V \times (2^V \to 2^V) \times Loc$ is a set of transitions;*
  - *Act: $Loc \to (\mathbb{R}^n \to \mathbb{R}^n)$ is a flow function;*
  - *Inv is a labeling function, assigning an invariant $Inv(l) \in \mathbb{R}^n$ to each location $l \in Loc$;*
  - *Init is a set of initial states Init $\subseteq \Sigma$ with $\bigcup_{s \in Init}(l,\nu)$ such that $\nu \in \mathbb{R}^n$ and $\nu \in \mathrm{Inv}(l)$.*
- *μ is a set of probability distribution over delays: $\mu = \bigcup_{s \in \Sigma} \mu_s$;*
- *W is a set of weight functions: $W = \bigcup_{e \in Edge} w(e)$ with probability $p_s(e)$;*
- *ι_{init} is an initial distribution over the initial states Init.*

As depicted above, this definition is similar to the stochastic rectangular automaton definition, the only difference is the underlying hybrid automaton. All variables in the rectangular automata model which could be values from rectangular sets can now be arbitrary real values. The variables evolve according to the activity function

Act, which is typically specified as an ordinary differential equation. In the following paragraph, we will give operational semantics for this model.

Definition 3.2.2 (Semantics of Stochastic Hybrid Automata)**.** *The semantics of a stochastic hybrid automaton $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init)$ consists of discrete instantaneous steps (jumps) and continuous time steps (flow):*

1. *Discrete step semantics*

$$\frac{\begin{array}{c} e = (\ell, g, r, \ell') \in Edge \\ \nu \in g \quad \nu' \in r(\nu) \quad \nu' \in Inv(l') \end{array}}{(\ell, \nu) \xrightarrow{e} (\ell', \nu')} Rule_{discrete}$$

2. *Time step semantics*

$$\frac{\begin{array}{c} (t = 0 \wedge \nu = \nu') \vee \\ (\exists f : [0,t] \rightarrow V. f(0) = \nu \wedge f(t) = \nu' \wedge \forall \tau \in (0,t). \frac{df}{dt}(\tau) = Act(l)(f(\tau)) \\ \forall\ 0 \leq \tau \leq t.\ \nu'(\tau) \in Inv(l) \end{array}}{(l,\nu) \xrightarrow{t} (l,\nu')} Rule_{time}$$

John Lygeros, Maria Prandini and Manuela L. Bujorianu used a different approach to define generalized stochastic hybrid systems. The main idea of these papers will be adapted in the following paragraphs to model stochastic hybrid automata using global probability distributions [LP10] [BL06]. In these papers, the continuous evolution of variables is given by stochastic differential equations, but this will be omitted in this thesis since we use the action function in our stochastic hybrid automata models.
The foundation is similar to the first approach, which was derived from stochastic rectangular automata. There exists a probability distribution, which determines a time delay after which a transition will be taken and after that time, a transition among all enabled transitions will be chosen. Instead of adding weights to the transitions, this approach requires the modeler to declare a transition kernel.

Definition 3.2.3 (Syntax of Stochastic Hybrid Automata with Global Probability Distributions and Transition Kernel)**.** *A stochastic hybrid automaton (SHA) using global probability distributions and a transition kernel is a tuple*
$\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \lambda, R, \iota_{init})$ *with states set* Σ *where:*

- $(Loc, \mathcal{C}, Edge, Act, Inv, Init)$ *is a hybrid automaton;*
- $\lambda : \Sigma \rightarrow \mathbb{R}_+$ *is a transition rate function;*
- *R is a transition kernel;*
- *ι_{init} is an initial distribution over the initial states Init.*

This stochastic hybrid automaton model requires a hybrid automaton, a transition rate function and a transition kernel. The transition rate function λ will be used to simulate a new clock that will determine the jump times of transitions and the

transition kernel R will decide, which transition is taken after the clock ran out.
The execution of such a model can informally be defined similar to the previous models. Continuous time evolves according to the activity function Act and discrete states change when a transition occurs. In this model, as declared in [LP10], we define two types of jump times which as a result, yield a transition:

- forced transitions: τ_f = inf{ t $\geq 0 \mid (\ell, \nu(t)) \notin$ *Inv*$(l)\}$
  Forced transitions occur, when the continuous state reaches the boundary of a location, that is given by invariants and guards. The continuous state would leave the state set Σ, so a transition has to be taken beforehand.
- spontaneous transitions occur according to a Poisson arrival process with a time varying rate of λ.

Forced transitions just specify the boundary of a location, but spontaneous transitions can execute at every time where they are enabled. To find the jump time for the next transition, an auxiliary continuous state variable $x_{n+1} \in \mathbb{R}$ is introduced that is evolving with a given transition rate function λ. This variable will be used to simulate an independent clock, such that whenever the clock runs out of time, the hybrid automaton reaches the jump time for a spontaneous transition.

Definition 3.2.4 (Spontaneous Jump Time)**.** *A spontaneous jump time is a time* $\tau_s \in \mathbb{R}$ *such that:*

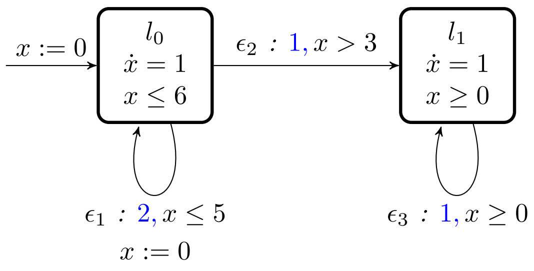
- x_{n+1} *is the auxiliary continuous state variable;*
- $x_{n+1}(0) = \ln(z_0)$*, where* z_0 *is uniformly distributed in (0,1);*
- $\frac{dx_{n+1}}{dt}(t) = \lambda(\ell, \nu(t))$*;*
- $\tau_s = inf\{\ t \geq 0 \mid x_{n+1} \geq 0\}$*.*

The clock x_{n+1} is initialized with a uniformly distributed random variable z_0, and is a negative value. This clock simulates the Poisson arrival process, it evolves with the negative exponentially distributed transition rate λ. The spontaneous jump time is given as the time t, at which that clock, that is starting from a random negative value and evolves with λ, reaches the value 0.
The system now has a jump time for forced transitions upon leaving the boundary of a location and a jump time of spontaneous transitions when the Poisson process event occurs [LP10].

Definition 3.2.5 (Next Jump Time)**.** *The next jump time, when the system takes a transition, is given by* $\tau_{next} = min\{\tau_f, \tau_s\}$*.*

The next jump time is given by the minimum of forced and spontaneous jump time. If the Poisson process shoots fast enough, a spontaneous transition can be taken. Once a transition takes place at time τ_{next}, the variable x_{n+1} will be reset to a new negative value $\ln(z_1)$, where z_1 is again unformly distributed in (0,1).
The system now has a jump time, when a transition will be taken. Which transition will be taken, is given by the transition kernel R. At each jump time, R states a probability for each transition to be taken. That can be a priority list, a weight function or similar and is given by the modeler. We use the notion of the Poisson process P and the transition kernel R as in [LP10].

Example 3.2.1. *We illustrate the above definitions on the hybrid automaton below. The Poisson process P is running parallel to the system, according to $P_\lambda(1) = \lambda \cdot e^{-\lambda}$, and yields spontaneous jump times. Location ℓ_0 is bounded by $x \leq 6$. Let the transition kernel R be given by weights, depicted in blue color in the graphical representation. An exemplary path could be $\pi(s_0, \epsilon_1\epsilon_1\epsilon_2)$, depending on the random behaviour of the system.*
Assuming the system starts with x=0 in ℓ_0, P could shoot for the first time after t=2 time steps. Now $\tau_s = 2$ and $\tau_f = 6$, so the next jump time is 2. Since only ϵ_1 is enabled, it will be taken. The auxiliary clock will be reset and P starts running again. The next spontaneous jump time will be sampled, for example $\tau_s = 4$. Since $\tau_s = 4 < 6 = \tau_f$, the next jump time will be 4. The transition kernel R decides based on probabilities for all enabled edges, in this case ϵ_1 and ϵ_2. Assume the system takes ϵ_1, the auxiliary clock will be reset. Now the Poisson process could shoot at time t=7. It holds that $\tau_s = 7 > 6 = \tau_f$, so a forced transition will be taken. Since only ϵ_2 is enabled in ℓ_0 at t=6, ϵ_2 will be taken.



3.2.1 Comparing the Global Approaches

We summarized two existing models of stochastic timed automata [BBB$^+$14] and stochastic hybrid systems [LP10] and defined two generalized stochastic hybrid automaton definitions based on them. The foundation of both models is an underlying global probability distribution, which yields a time when a transition will be taken and after that time, a decision will be made which transition to take. This leaves space for the question, where the differences between those two models are or whether they are equivalent.
The differences are easy to determine. Whereas the first approach uses a global probability distribution for each state of the system to sample a transition delay, the second approach uses an independent Poisson process. However, in our stochastic hybrid automaton model, we distinguish between different probability distributions depending on the enabled intervals of transitions. In the other approach [LP10], the jump time is always given according to a negative exponentially distributed Poisson process.
The second difference is given by the weight function in the first approach and the transition kernel in the second approach. The transition kernel R is very powerful, the modeler is given a lot of freedom and decisive power over the system. That makes the approach easy to simulate but less useful for model checking. By only assigning weights to transitions, a probability for paths is easy to calculate, hence reachability problems that require a certain minimal probability are more simple to verify [BBB$^+$14].

The differences between both models are little, but equivalence still needs to be verified. Intuitively, their essence is to rely on a probability distribution to give a jump time and after that jump time, a transition will be taken. To formally prove this, an algorithm needs to be found that translates each generalized stochastic hybrid automaton model from one approach to the other approach and vice versa. Since the Poisson process is a fixed independent system from the stochastic hybrid automaton, the translation from the first approach to the second approach needs to be coded into the transition kernel R. Due to R being so powerful, it could be possible to code all information on weights and different probability distributions for intervals into the transition kernel. For simple intervals where only one transition is enabled, this might be easy because then the transition kernel would be exactly the probability distribution over the interval from the first approach. But once the system gets more complex, this does not need to be the case anymore. Hence the intuitive reasoning to explain equivalence is not enough, and future work will have to verify it.

Chapter 4

Local Probability Distributions

In this chapter, an example will be given of how stochastic hybrid automata can be modeled with local probability distributions instead of a global probability distribution. This main idea is derived from stochastic hybrid Petri nets and will be used to not give a definition, but an intuition of such a model.
Stochastic hybrid Petri nets, just like stochastic hybrid automata, are a subclass of stochastic hybrid systems. For a more detailed understanding of hybrid Petri nets, we refer to [HPS$^+$19].
In essence, in a stochastic hybrid Petri net, each transition can take an action independent from the other transitions, but the result of an action can affect other transitions. A simple example would be one location with one token and two outgoing transitions. Upon execution of one transition, the token moves from the previous location to the resulting location of this transition. If one transition shoots, the token moves and the other transition is blocked because the precondition of an existing token is not satisfied anymore.

4.1 Stochastic Hybrid Automata with Local Probability Distributions

This idea, each transition having an impact on the random behaviour of the stochastic hybrid system will be used in this modeling attempt. Instead of one global probability distribution that randomly picks a delay when a transition should be taken, this approach uses a local probability distribution for each transition or possible jump that can be taken from a state. Each probability distribution yields a jump time for a transition, when it would be taken. The jump time indicates the resulting state.
In the following section, a modeling idea of stochastic hybrid automata with local probability distributions will be given that neither claims to be complete, nor completely correct. It is meant to give an intuition on modeling with local probability distributions and highlight problems with this approach. In the scope of this thesis, we underestimated the complexity of this issue. We could not find a satisfactory definition, but will instead give our modeling attempt.

4.1.1 Modeling Approach

Proposition 4.1.1 (Syntax of Stochastic Hybrid Automata with Local Probability Distributions)**.** *A stochastic hybrid automaton (SHA) with local probability distributions is a tuple*
$\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, Inv, Init, \Lambda)$ *with states set* Σ *where:*

- $(Loc, \mathcal{C}, Edge, Act, Inv, Init)$ *is a hybrid automaton;*
- Λ *is a set of probability distributions over jumps:* $\Lambda = \bigcup_{s\in\Sigma, e\in Edge} \lambda_e(s)$.

This exemplary definition of stochastic hybrid automata with local probability distributions is similar to the definition of stochastic hybrid automata with global probability distributions. The difference is that there are no weights anymore and the probability distributions now exist for each jump, so for each enabled transition e in each state. To be able to use this as a model, an algorithm of executions on stochastic hybrid automata with local probability distributions needs to be given. As indicated earlier, the local probability distributions yield a jump time for each possible jump. The shortest jump time of that system will be executed. The question is, how jump times can be determined.

Example 4.1.1 (Determining Jump Times)**.** *This paragraph is an example for calculating the jump times of each transition. The idea is to simulate a clock for each transition, a timer that is running out. When the timer runs out, the jump time is reached.*

- *For each transition* $e_1, ..., e_n \in Edge$ *that is enabled in a state s=(l,v), an auxiliary continuous variable* $\nu_i \in \mathbb{R}$ *with* $\frac{d\nu_i}{dt}(t) = \lambda_i(\ell, \nu(t))$ *for each* $i \in [1,n]$ *will be added, with* λ *being a probability distribution over each transition.*
- *Each of those clocks will be initialized with* $\nu_i(0) = \ln(z_0)$*, where* z_0 *is uniformly distributed in (0,1). The advantage of using the natural logarithm of a value between zero and one is that the result is always a negative value.*
- *To simulate a timer, the valuations of these negative clocks evolve according to* $\nu_i(t) = \nu_i(\tau_{last}) + \int_{\tau_{last}}^{t} \lambda_i(\ell, v(t))dt$ *with* τ_{last} *being the time the last jump was taken and* $\nu_i(\tau_{last})$ *being the randomly sampled negative value with the natural logarithm. This formula is a negative value getting increased by the integral of a probability distribution.*
- *The clocks are negative timers that run out when the time reaches 0. The resulting jump time for each transition is then given by* $\tau_{si} = inf \{t \geq 0 \mid \nu_i(t) \geq 0\}$*, so the smallest amount of time t, such that the clock* ν_i *of a transition* e_i *reaches 0. For each enabled transition* $e_1, ..., e_n$*, the algorithm yields a possible jump time* $\tau_{s1}, ..., \tau_{sn}$*, a spontaneous jump time for each transition.*
- *The next transition to be executed is given according to the minimal possible jump time of all spontaneous and the forced transition* $\tau_{next} = min(\tau_{s1}, ..., \tau_{sn}, \tau_f)$*. The resulting state is given by the transition relation of the transition with minimal jump time. After each jump, all clocks will be reset to a random negative value* $\nu_i(0) = \ln(z_{last})$*, where* z_{last} *is uniformly distributed in (0,1). Boundary hitting transitions* τ_f *similar to the second global approach will not be considered any further.*

In essence, this exemplary algorithm to calculate jumps of a stochastic hybrid automaton with local probability distributions simulates negative clocks, whose values evolve with a rate that is given by the local probability distributions. The first clock that expires determines the transition that will be executed by the system. For completeness, possible operational semantics for such a system will be given. As a reminder, this chapter only claims to be an intuitive modeling idea for using local probability distributions and no formal definition of a correctly running system.

Definition 4.1.1 (Semantics of Stochastic Hybrid Automata with Local Probality Distributions)**.** *The semantics of a stochastic hybrid automaton* $\mathcal{A} = (Loc, \mathcal{C}, Edge, Act, \text{Inv}, \text{Init})$ *with local probability distributions consists of discrete instantaneous steps (jumps) and continuous time steps (flow):*

1. *Discrete step semantics*

$$\frac{\begin{array}{c} e = (\ell, g, r, \ell') \in Edge \\ \nu \in g \quad \nu' \in r(\nu) \quad \nu' \in Inv(l') \end{array}}{(\ell, \nu) \xrightarrow{e} (\ell', \nu')} Rule_{discrete}$$

2. *Time step semantics*

$$\frac{\begin{array}{c} (t = 0 \wedge \nu = \nu') \vee \\ (\exists f : [0,t] \to V. f(0) = \nu \wedge f(t) = \nu' \wedge \forall \tau \in (0,t). \frac{df}{dt}(\tau) = Act(l)(f(\tau)) \\ \forall\ 0 \leq \tau \leq t.\ \nu'(\tau) \in Inv(l) \end{array}}{(l,\nu) \xrightarrow{t} (l,\nu')} Rule_{time}$$

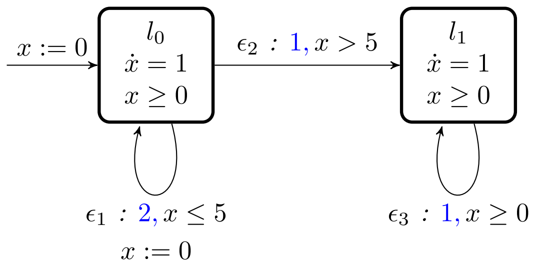
4.2 Problems of Using Local Probability Distributions

Common literature rarely uses local probability distributions to model stochastic hybrid systems. Modeling with global probability distributions is highly preferred, because there are several issues making the understanding and execution of the local approach harder than the global approach.
The biggest problem with local probability distributions is usability for model checking. Whereas with stochastic hybrid systems using global probability distributions, finding probabilities for a certain path was given by one formula $\mathbb{P}_A(\widehat{\pi}(s, e_1e_2 \ldots e_n))$ for a finite path $\widehat{\pi}(s, e_1e_2 \ldots e_n)$, it is not that simple using local probability distributions. The global approach only considers the probability of taking a transition e_i that is given by the global probability distribution, but in the local approach, one would need to consider the probabilities of all transitions. To calculate the probability of one path, the probabilities of all other transitions for each step need to be calculated. The probability of taking a transition does not only depend on the own jump time, but the jump times of all other enabled transitions, because only the transition with the lowest jump time will be executed. This makes it very difficult to find a formula for calculating path probabilities. In fact, it is not only hard to calculate path probabilities. This model does not have the linear restriction of derivatives within locations as introduced in stochastic rectangular automata, so the intervals used in

the definitions might not be continuous. We can define semantics for such a model, but without continuous intervals of enabledness for discrete transitions, the model checking problem is undecidable.

Another problem with model checking is reachability. The modeling decision to choose the transition with the minimal jump time can be very limiting, and in extreme situations it can even block transitions completely.

Example 4.2.1. *This example shows a simple stochastic hybrid automaton to visualize the problem with reachability. The interesting parts are the transitions ϵ_1 and ϵ_2. Assuming this automaton is executing according to the local approach, the minimal jump time always wins. If the system is in location ℓ_0 with $x = 0$, both transitions will always yield a jump time. Because of the given guards on those transitions, the jump time of ϵ_1 will always be less than 5, and of ϵ_2 always be greater than 5. Using local probability distributions, the system would stay in ℓ_0 forever and do the self loop. Assuming this system is executing according to the global approach, first a random delay when a transition is taken will be sampled, and then a transition will be taken according to weights. In that case, the system could sample a delay of $t > 5$, so ϵ_2 would be taken.*



This example does not mean that the global approach is more expressive than the local approach, but it shows the limits of reducing the next jumps to the minimal jump time. That problem could be avoided by several ways, that again yield new problems:

- Change the minimal jump time condition: If any other jump time but the minimal is picked, the same problem occurs but in a different scenario.
- Pick a random jump time out of the calculated ones: This would solve the problem, but at the cost of introducing another random variable that would need to be considered in all executions.
- Introduce a scheduler that can enable/disable transitions: That way, ϵ_1 could be enabled such that ϵ_2 could be taken, but introducing such a scheduler would make the model even more difficult.

The last problem already concurrently mentioned introducing a new random variable which addresses another problem. The stochastics behind this stochastic hybrid automaton model using local probability distributions do not only depend on the local probability distributions, but on the uniformly distributed random variable z_0/z_{last}. The probability of taking a transition is heavily influenced by this random variable, because it sets the value of the negative timers for all transitions. If the timer is small, the probability of taking the transition is very high.
Because of this, the expressivity of probabilities of such a system is difficult to interpret, because transitions that would probably never be executed could suddenly be executed every time due to the random variable being convenient. This problem could be avoided by resetting all clocks to the same value, for example $\nu_i = -0.5$ for all enabled transitions i, but then new problems appear. How to pick such a value for all clocks? The clocks do not expire with the same rate, so -0.5 could be very much for one clock, but almost nothing for another one. This could cause fairness problems.
Due to those problems, the approach with local probability distributions is less practical than the global approach and using one of the global definitions in future work is highly preferred.

Chapter 5

Conclusion

5.1 Summary

In this thesis, we summarized several already existing hybrid automaton models and introduced them in an order such that each subsequent model either is an extension to the previous model or resolves more nondeterminism. We started with a definition of the basic timed automaton model and extended the continuous time evolution to rectangular sets. To solve the nondeterminism between jumps which can occur in rectangular automata, weights were added to competing transitions to create a simple probabilistic choice.
This bachelor thesis then dealt with introducing new models based on the existing ones. The existing models of stochastic timed automata and rectangular automata have been combined to introduce a new model of stochastic rectangular automata. Due to several different modeling strategies, the issue to find a general definition of stochastic hybrid automata appeared. General definitions of stochastic hybrid automata using global probability distributions have been introduced, based on the stochastic rectangular automaton and a stochastic hybrid system definition.
Another main goal of this thesis was to find a different approach for stochastic hybrid automata, but instead using local probability distributions for each jump. We could not define such a model, but gave a modeling idea instead and discussed problems, why this approach is not very practical.

5.2 Future work

This thesis can be used as a foundation for several future topics, there are still open questions needing to be solved.
First, the stochastic rectangular automaton model might not be complete. For simplicity, this thesis restricted the continuous time evolution to linear change within a location. The question would be, if and how the model would change when more general derivatives are allowed, such that the enabled intervals for transitions are not connected anymore. On that topic, a model checking algorithm regarding reachability could be introduced for stochastic rectangular automata.

Other possible future topics refer to the generalized stochastic hybrid automaton models. For the global approaches, a formal verification to prove or disprove equivalence needs to be found. For local probability distributions, it might be possible to define a probability space over that model to work out probability formulas for paths, thus making this model more practical.
Further tasks could involve the implementation of these new models and their respective reachability analysis into the HyPro library, possibly doing benchmark analysis with tools like Uppaal or Modest Toolset.

Bibliography

[Ábr12] Erika Ábrahám. Modeling and analysis of hybrid systems. *RWTH Aachen University, Lecture Notes*, 2012.

[AD94] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[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.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[BL06] Manuela L. Bujorianu and John Lygeros. *Toward a General Theory of Stochastic Hybrid Systems*, pages 3–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[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*, page 43–52, New York, NY, USA, 2011. Association for Computing Machinery.

[HPS$^+$19] Jannik Hüls, Carina Pilch, Patricia Schinke, Joanna Delicaris, and Anne Remke. State-space construction of hybrid Petri nets with multiple stochastic firings. In David Parker and Verena Wolf, editors, *Quantitative Evaluation of Systems*, pages 182–199, Cham, 2019. Springer International Publishing.

[LP10] John Lygeros and Maria Prandini. Stochastic hybrid systems: A powerful framework for complex, large scale applications. *European Journal of Control*, 16:583–594, 11 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.