Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Semantics of Stochastic Gamma Composition Language

Author
Simon József Nagy

Author
Kristóf Marussy

October 24, 2022

Contents

Chapter 1

Introduction

In this section the formal semantics of Stochastic Gamma Composition Language is presented. The semantics of the Stochastic Gamma Composition Language defines the mathematical behavior of the dependability models using the tools of measure and probability theory. The structure and deterministic definitions in the SGCL semantics are based on the semantics of GCL and broaden with stochastic modeling. The semantics of SGCL can be interpreted as a refinement of the semantics of GCL, where the non-specified, nondeterministic behavior is resolved with stochastic rules.

The stochastic definitions of the SGCL semantics are defined using the definitions within the semantics of the Church probabilistic programming language and Gamma Composition Language. As a result, the SGCL semantics can be viewed as probabilistic program semantics, where the instructions are defined by statechart composition. The semantics of stochastic composition extends the semantics of deterministic Gamma composition. Thus a model transformation algorithm can construct a probabilistic program, which has the same semantics as the SGCL model, as it can be seen in Chapter **??**.

In this chapter, we use boxes to highlight and discuss theoretical questions which are relevant to the development of SGCL semantics.

Meaning of *nondeterminism*

The usage of the word *nondeterminism* is different in different scientific works. In [**?**], the nondeterminism defines a behavior, which is completely unknown. In this case, nondeterministic models can specify optimization problems, worst-case scenarios, or two games. Therefore nondeterminism cannot be used to describe stochastic phenomenon since the stochastic phenomenon is determined by a measurable space and a probability measure. Contrarily, in [**?**], nondeterminism is used to describe stochastic phenomena. In this paper, we do not use nondeterminism for stochastic phenomena, and we clearly distinguish stochastic and nondeterministic behavior.

Symbol	Explanation
2^X	Power set of X is the set of subsets $SX \subseteq X \to SX \in 2^X$
X	Set of possible outcomes of a random phenomena
$\mathcal{F}$	Sigma algebra
$X \times Y$	Product of two sets
$X \uplus Y$	Disjoint union of two sets
$f \colon X \to Y$	Function
$f \colon X \xrightarrow{\mathrm{m}} Y$	Measureable function
M_X	Measureable space $M_X = (X, \mathcal{F})$
μ	Probability measure $\mu \in \mathcal{D}(X)$
$\mathcal{D}(X)$	Set of all distributions on X.
ε_F	Evaluation function of distributions: $\varepsilon_F \colon \mathcal{D}(X) \to [0; 1]$
$\delta_X(x)$	Dirac measure
$\mu \times \lambda$	The product measure of two probability measure
$f_*(\mu)$	Pushforward, induced distribution $f_*(\mu) = \mu \circ f^{-1} \colon Y \to \mathbf{R}^+$
$\mu \gg\!= f$	Binding of distributions

Table 1.1: Symbols of probability theory

Symbol	Explanation
e	Event.
inst(e)	Set of instances of an event.
ei	Instance of an event $ei \in inst(e)$
$Dom(e)$	Domain of the event parameters
$Dom(e) = \top$	e has no parameter
$\mathcal{B}$	Borel σ-algebra generated by the open subsets of the real numbers.
E	Set of events.
$\perp$	Absence of an event instance, an event is not occurred.
$\perp_E$	Empty event vector.
v_E	Event vector. $v_E : E \to \cap_{e \in E} Dom(e)$
V_E	Set of all possible event vector.
$\mathcal{F}(V_E)$	$\prod_{e \in E} \mathcal{F}(inst_\perp(e))$
$v_1 \triangleright v_2$	Overlay of two event vectors: $v_2(e)$ if not null and $v_1(e)$ otherwise.
$\ominus$	Synchronous component.
π^0	Initial state probability distribution.
S	Measurable space of potential states
I and O	Finite set of input and output events.
T	Transition function.
$\circledS$	Synchronous composite component.
$\rightleftharpoons$	Channel function: $\widehat{I} \setminus I \to \widehat{O}$
$\widehat{I}$	Exported input event set.
$\widehat{O}$	Exported output event set.
$\circledC$	Cascade composite component.
$arm(e)$	Set of arming event instanve of an event $\mathbb{R}^+ \times inst(e)$.
aei	Arming event instance $aei \in arm(e)$
av	Arming vector $av \in Arm(e)$
$\pm_E$	Empty arming vector: $\pm_E(e) = \perp$ for all $e \in E$
A	Set of alert input events.
$Prio_A$	Alert input event sequence.
$\oplus$	Asynchronous adapter.
$\circleddash$	Asychronous component.
$\circledA\rangle\circleddash$	Asynchronous composite component.
$lift_i(\mu)$	Lift the distribution of a subcomponent to the component level.
$step(s, av, q)$	A stochastic component processes an event and makes a step.

Table 1.2: Symbols of SGCL

Symbol	Explanation
$+\!\!+$	Operator of concatenating sequences.
q	Queue of event instances $q = \langle ei_1, ei_2, \ldots ei_K \rangle$.
$es = (s, av_A, ei)$	Extended state of a stochastic asynchronous component.
S_{Ext}	$S_{Ext} = S \times Arm_A \times O$ set of extended states of a $\ominus$.
$F_{es} \subset S_{Ext}$	Measurable set of extended states.
ρ	Path of a component $\rho = \langle es_0 \ldots es_K \rangle$.
$Path^{\ominus}$	The set of all (finite or infinite) paths of a $\ominus$.
$Path^{\ominus}_{\leq L}$	The set of all paths of a $\ominus$ with at most L states.
$Cyl_L(\rho)$	Cylinder set of a given path $\langle I_{es\,0} \ldots I_{es\,K} \rangle = Cyl_L(\rho) \subseteq Path^{\ominus}$
$\mathcal{F}(\rho_L)$	$\mathcal{F}(\rho_L)$ is the smallest σ-algebra of all traces with a given length.
tr	Trace of a component: $tr = \langle ei_0, \tau_1, \ldots, \tau_K, ei_K \rangle$.
$Trace^{\ominus}$	The set of all (finite or infinite) traces of a component $\ominus$.
$Trace^{\ominus}_{\leq L}$	The set of all traces of a component $\ominus$ with at most L event instances.
$Cyl_L(tr)$	Cylinder set of a given trace with prefix $Cyl_L(tr) \subseteq Trace^{\ominus}(O)$
$\mathcal{F}(tr_L)$	$\mathcal{F}(tr_L)$ is the smallest σ-algebra of all traces with a given length.
$Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_L))$	Probability measure of an L long path.
$Pr^{\ominus}_{tr}(\mathcal{F}(tr_L))$	Probability measure of an L long trace.

Table 1.3: Symbols of paths and traces

Chapter 2

Mathematical background

2.1 Measure theory

In this section, we present the definitions of measure theory. we use the toolset of measure theory to describe stochastic phenomena. All definition in this section is based on [**?**] and [**?**].

Measurable spaces are often used to describe the set of outcomes in a stochastic phenomenon. For instance, in the case of the stochastic delay in Model **??** the set of outcomes is the set of positive real numbers $t \in \mathbf{R}^+$. The sigma-algebra on a measurable denotes the set of intervals on the set of outcomes on which one can define a probability measure. In the stochastic delay component in Model **??**, the sigma-algebra of the outcomes is the Borel sigma-algebra on $\mathbf{R}^+$.

Definition 1 (Measurable space) *A* measurable space *from probability theory is a pair $M_X = (X, \mathcal{F})$, where X is a set of* outcomes *and $\mathcal{F} \subseteq 2^X$ is a σ-algebra. For conciseness, we will use the notation X to refer to both the set of outcomes and the space M_X when the meaning is clear from context, and use $\mathcal{F}(X)$ to refer to the associated σ-algebra.*

Definition 2 (Trivial measurable space) *The* trivial measurable space *is $\top = (\{\top\}, \{\emptyset, \{\top\}\})$, which is the (only) measurable space associated with the one-element set.*

If there are two or more random phenomena, the product of the measurable spaces of the two phenomena defines the outcome space of the joint occurrence. Additionally, the disjoint union of the two measurable spaces is the outcome space of random phenomena when one of the two phenomena occurs randomly.

Definition 3 (Operations on measurable spaces) *We will use the notations $X \times Y$ and $X \uplus Y$ to refer to products and disjoint unions of measurable spaces, i.e., $\mathcal{F}(X \times Y) = \mathcal{F}(X) \otimes \mathcal{F}(Y)$ is the smallest σ-algebra containing the sets $\{F \times G \mid F \in \mathcal{F}(X), G \in \mathcal{F}(Y)\}$ and $\mathcal{F}(X \uplus Y) = \{F \uplus G \mid F \in \mathcal{F}(X), G \in \mathcal{F}(Y)\}$.*

2.2 Probability measures

Probability measures assign are functions which assign a positive real value to each interval on the set of outcomes. A probability measure can be a probability or a probability density value in the case of discrete and continuous measurable spaces, respectively.

Definition 4 (Set of all distributions) *Given a measurable space X, let $\mathcal{D}(X)$ denote the set of all* probability measures *(also called* distributions*) on X. We use $\mu \in \mathcal{D}(X), \mu\colon X \to \mathbf{R}^+$, to define a distribution on X.*

Modeling deterministic behavior with Dirac distribution

Some parts of the SGCL models may have purely deterministic behavior. Dirac measure and Dirac distribution are defined to model the semantics of deterministic behavior. Dirac distribution models random phenomena, the output of which is always the same. Therefore the Dirac measure assigns 1.0 to the deterministic otcome and 0.0 to anything else.

Definition 5 (Dirac measure) *The* Dirac measure $\delta_X(x) \in \mathcal{D}(X)$ *at $x \in X$ is $\delta_X(x)(F) = 1$ if $x \in F$, 0 otherwise. We will omit the subscript X if it is clear from context.*

In the case of stochastic phenomena, the product of probability measures is a probability measure of the random phenomenon, where all the joint stochastic phenomena occur independently from each other. The product of probability measure is also called joint distribution in the case of independent random phenomena.

Definition 6 (Product of measures) *The* product measure $\mu \times \lambda\colon \mathcal{F}(X \times Y) \to [0;1]$ *of $\mu\colon \mathcal{F}(X) \to [0;1]$ and $\lambda\colon \mathcal{F}(Y) \to [0;1]$ is the unique measure such that $(\mu \times \lambda)(F \times G) = \mu(F) \cdot \lambda(G)$ for all $F \in \mathcal{F}(X)$ and $G \in \mathcal{F}(Y)$.*

In measure theory the measurable functions are the generalization of the continuous functions. Measurable functions can describe mathematical relations and transformation on set of outcomes of random phenomenon.

Definition 7 (Measurable function) *The function $f\colon X \to Y$ is* measurable *if $f^{-1}(F) = \{x \in X \mid f(x) \in F\} \in \mathcal{F}(X)$ for all $F \in \mathcal{F}(Y)$. We will write $f\colon X \overset{\text{m}}{\to} Y$ to emphasize that f is measurable.*

In probability theory measurable functions are used to apply transformations on (the outcomes of) random phenomena. The probability measure of the transformed phenomena is called induced or pushforward probability measure.

Definition 8 (Induced/Pushforward probability measure) *If there is $f\colon X \overset{\text{m}}{\to} Y$ a measurable function and $\mu\colon X \to \mathbf{R}^+$ a probability measure on X then let be $f_*(\mu) = \mu \circ f^{-1}\colon Y \to \mathbf{R}^+$ is the induced or pushforward probability measure of μ and f, where $\nu = f_*(\mu)$ and $\nu(F) = \mu \circ f^{-1}(F) = \mu(f^{-1}(F))$ for all $F \in \mathcal{F}(Y)$.*

Measurable space of distributions

Let be $\mathcal{D}(X)$ the set of distributions over measurable space X. $\mathcal{D}(X)$ itself forms a measurable space [**?**], where $\mathcal{F}(\mathcal{D}(X))$ is the smallest σ-algebra such that the *evaluation maps* $\varepsilon_F\colon \mathcal{D}(X) \to [0;1]$, $\varepsilon_F(\mu) = \mu(F)$ are measurable for all $F \in \mathcal{F}(X)$.

Using the fact that $\mathcal{D}(X)$ is measurable, we can conclude that the outcome of a random phenomenon can be another stochastic phenomenon. For instance, one randomly draws out a coin from a bag full of biased coins, and the head-tail probabilities of the coins are

different. The probability measure of such a phenomenon is the special case of the pushforward measures when the measurable function assigns another probability measurable to the outcome of a random phenomenon. This type of pushforward probability measure is called *distribution binding*.

Definition 9 (Bind distributions) *Let be the measurable transformation of a distribution $\mu \gg\!= f$. If (we get a given distribution) $\mu \in \mathcal{D}(X)$ and $f\colon X \xrightarrow{\mathrm{m}} \mathcal{D}(Y)$, then we may form $\mu \gg\!= f \in \mathcal{D}(Y)$ by $(\mu \gg\!= f)(F) = \int_X f(x)(F)\, d\mu(x)$ [**?**]. The resulting distribution corresponds to the random selection of some $x \in X$ according to the distribution μ, followed by the random selection of some $y \in Y$ according to the distribution $f(x)$.*

In particular, if $g\colon X \xrightarrow{\mathrm{m}} Y$ and $f(x) = \delta_X(g(x))$, then $g_*(\mu) = \mu \gg\!= f$ is the pushforward *of μ along g, which corresponds to the random selection of some $x \in X$ according to μ followed by the application of the (deterministic) function g.*

2.3 Semantics of probabilistic programs

In computer science, probabilistic programs are often used to define the mathematical model of stochastic software behavior [**?**]. The semantics of probabilistic programs contain a set of objects, a set of transformations, and a concatenation function. The set of objects models the actual state of the software. The set of transformations on the set of objects models the code blocks within the software. An element of this set assigns a distribution of subsequent states to the actual state. The concatenation function assigns a distribution on the set of objects to an object and an object transformation function. The concatenation function models the sequential execution of the programs. The object transformation describes the state transition of the software into another software state. The concatenation function assigns an object transformation to two other object transformations. Thus the concatenation functions model the concatenatenation of two program blocks into one. In the *Church* probabilistic programming language [**?**], the state of the program is defined by a list of variable name-value pairs. The state transitions are specified by the set of *Church* expressions. The *Church* expressions can be concatenated using the distribution binding. For example, after the first *Church* expression the distribution of program states is known. The distribution is determined by the initial program state and the first expression. Then after the second expression the state distribution will be the binding distribution of the second expression and the previous state distribution.

Chapter 3

Events

An *event* is an observable phenomenon, such as an incoming message or an incoming car on the road. An event might have some parameters. Parameters define the attributes and properies of the event occurrence. A parameter can be the payload of a message or the correctness of the payload. An *event instance* is the occurrence of a given event with a given parameter. In SGCL, interfaces are syntactic sugar to group a given set of events together.

Definition 10 (Event) *Let be E a finite set of events. The parameter* domain *of the event $e \in E$ is the measurable space $Dom(e) = (X_e, \mathcal{F}_e)$. If $Dom(e) = \top$, then e has no parameter. Otherwise, e is* parameterized. *The set of all event instances for a given event e is denoted by $inst(e) = \{(e, p) \mid p \in X_e\}$. In the case when the absence of an event is of interest, $inst_\perp(e)$ is defined as $inst(e) \cup \{\perp\}$.*

In stochastic components, the event instances may occur randomly. The distribution of event instances can model random parameter values in the stochastic components. As a result, the sigma-algebra of event instances has to be defined.

Definition 11 (Sigma algebra of event) *To form measurable spaces, we define the σ-algebras $\mathcal{F}(inst(e)) = \{\{e\} \times F \mid F \in \mathcal{F}(Dom(e))\}$ and $\mathcal{F}(inst_\perp(e)) = \mathcal{F}(inst(e)) \cup \{F \cup \{\perp\} \mid F \in \mathcal{F}(inst(e))\}$, respectively. For a set of events E, we define the measurable spaces $inst(E) = \biguplus_{e \in E} inst(e)$ and $inst_\perp(E)$ similarly. Finally, let be $F_e \in \mathcal{F}(inst(e))$ a measurable set of instances of event e.*

Typical parameter types Common parameter types include finite sets X_{fin} (e.g., $\{\texttt{true}, \texttt{false}\}$ for Boolean values) with $\mathcal{F}(X_{fin}) = 2^{X_{fin}}$, integers with $\mathcal{F}(\mathbb{Z}) = 2^{\mathbb{Z}}$, and real numbers with $\mathcal{F}(\mathbb{R}) = \mathcal{B}$, where $\mathcal{B}$ is the Borel σ-algebra generated by the open subsets of the real numbers. Events with more than one parameter may be formed as the product measurable spaces of elementary parameter spaces.

Chapter 4

Semantics of stochastic synchronous components

In this section, the stochastic synchronous component is defined. Stochastic synchronous components on execution, if a trigger arrives, sample the input event instances, transition into a new state, and generate some output events. Event vectors determine which input and output events are raised during transitions. Event vectors also specify the parameter values of the generated events.

Definition 12 (Event vector) *Given a set of events E, and* event vector *v_E is a function that assigns a (possibly "null", i.e., $\perp$) event instance to every event $e \in E$ such that $v_E(e) \in inst_\perp(e)$. The set of all possible event vectors is denoted by V_E and is endowed with the σ-algebra $\mathcal{F}(V_E) = \prod_{e \in E} \mathcal{F}(inst_\perp(e))$.*

Let $\perp_E$ denote the empty event vector *$\perp_E \in V_E$ with $\perp_E(T) = \perp$ for all $e \in E$. We will use the notation $v_1 \triangleright v_2$ for the* overlay *of the event vectors $v_1, v_2 \in V_E$, where $(v_1 \triangleright v_2)(e) = v_2(e)$ if $v_2(e) \neq \perp$, $v_1(e)$ otherwise. Note that $v \triangleright \perp_E = \perp_E \triangleright v = v$.*

In GCL, if the execution of a synchronous component is triggered, then the component transitions into a new state and generates some output event instances as a function of the current state and the actual input event instances. Contrarily to GCL, if a transition occurs, the stochastic synchronous components transition into a random state and generate output event instances randomly. As a result, stochastic synchronous components contain the initial state distribution instead of the initial state, and the transition function assigns a distribution of subsequent states and output event vectors. In stochastic synchronous components, the domain of events is defined by the domain of the transition distribution.

States of stochastic components

The measurable space of potential states might be uncountable. The state of an atomic synchronous stochastic component consist of a given number of a discrete and continuous state variables $s = (z_1, \dots z_k, r_1, \dots r_l) \in S_{Ext} = \mathbf{Z}^k \times \mathbf{R}^l | k, l \in \mathbf{N}$.

Definition 13 (Synchronous stochastic component) *A* synchronous stochastic component *is a tuple $\ominus = (S, \pi^0, I, O, T)$, where*

- *S is the measurable space of potential* states*;*
- *$\pi^0 \in \mathcal{D}(S)$ is the* initial state probability distribution*;*

- *I and O are the finite sets of* input *and* output events*, respectively, with $I \cap O = \emptyset$, while the set of all events is denoted by $E = I \cup U$; and*
- *$T\colon S \times V_I \xrightarrow{\mathrm{m}} \mathcal{D}(S \times V_O)$ is the* transition function*, which determines the (random) next state and vector of output events of the component when executing in a given state with a given input vector. The behavior of the component is deterministic, but probabilistic, i.e., it may depend on a random choice, but there is no nondeterminism within the component.*

Determinism of synchronous components

Deterministic synchronous components, i.e., components with a single initial state $s^0 \in S$ and a deterministic transition function $T_{det}\colon S \times V_I \to S \times V_O$ may be turned into a stochastic component with Dirac distributions, which contain no randomness. Formally, we define $\pi^0 = \delta(s^0)$ and $T(s, v) = \delta(T_{det}(s, v))$.

The definition of the structure of synchronous composite components requires no change from [**?**]. Nevertheless, to make the paper self-contained, we repeat the definition here:

Definition 14 (Synchronous composite component [?]) *A* synchronous composite component *is a tuple $\textcircled{s} = (\mathbf{C}, I, O, \rightleftharpoons)$, where*

- *$\mathbf{C} = \{\ominus_1, \ldots, \ominus_K\}$ is the set of* synchronous stochastic components *that constitute the composite component, such that $\ominus_k = (S_k, \pi^0_k, I_k, O_k, T_k)$ for all $k = 1, \ldots, K$;*
- *$I \subseteq \widehat{I}$ is the set of* exported input events*, where $\widehat{I} = \bigcup_{k=1}^{K} I_k$;*
- *$O \subseteq \widehat{O}$ is the set of* exported input events*, where $\widehat{O} = \bigcup_{k=1}^{K} O_k$;*
- *inputs and outputs are disjoint no two components have common outputs, i.e., $\widehat{I} \cap \widehat{O} = \emptyset$ and $O_k \cap O_\ell = \emptyset$ holds for all $k \neq \ell$; and*
- *$\rightleftharpoons\colon \widehat{I} \setminus I \to \widehat{O}$ is the* channel *function that associates each non-exported input with an output from which it will receive events, i.e., an input is either linked to an output by $\rightleftharpoons$ or is exported. For each $e \in \widehat{I} \setminus I$, we must have $Dom(e) = Dom(\rightleftharpoons(e))$.*

The semantics of synchronous composite components must be adapted to take their probabilistic behavior into account. The initial state distribution of a stochastic composite synchronous component is the product measure of the initial distribution of the subcomponents, where no previous event is raised with one probability. Note that the initial distributions of the subcomponents are independent. The transition function of the composite component is defined in three steps. First, we determine the joint distribution of the transition functions of the subcomponents. Then we define the combine function, which assigns the state of the composite component to the states and output event vectors of the constituent components. Finally, the transition distribution of the composite component is defined with the pushforward distribution of the joint distribution of the next state and output distribution, and the combine function.

Definition 15 (Semantics of synchronous composition) *A synchronous composite component $\textcircled{s} = (\mathbf{C}, I, O, \rightleftharpoons)$ is a synchronous stochastic component $\textcircled{s}\rangle\ominus = (S, \pi^0, I, O, T)$ as follows:*

- *$S = S_1 \times \cdots S_K \times V_{\widehat{O}}$ is the set of potential states, which is formed by all possible combinations of the potential states of the constituent components as well as the last output event vector of every component.*

- π^0 *is the product measure* $\prod_{k=1}^{K} \pi_k^0 \times \delta(\perp_{\widehat{O}})$.

- *To construct* T*, let be the function combine*: $(S_1 \times V_{O_1}) \times \cdots \times (S_K \times V_{O_K})) \overset{\text{m}}{\to} S$ $combine((s_1, v_1), \ldots, (s_K, v_K)) = ((s_1, \ldots, s_K), v_1 \uplus \cdots \uplus v_K)$.
  This function is measurable, because it only rearranges tuple and event vector components. Now let

  $$T(((s_1, \ldots, s_K), v_{\widehat{O}}), v_I) = combine_*(T_1(s_1, v_{I_1}) \times \cdots \times T_K(s_K, v_{I_K})) \times \delta(v_O), \quad (4.1)$$

  where

  - *for all* $k = 1, \ldots, K$ *and* $e \in I_K \cap I$, $v_{I_K}(e) = v_I(e)$;
  - *for all* $k = 1, \ldots, K$ *and* $e \in I_K \setminus I$, $v_{I_K}(e) = v_{\widehat{O}}(\rightleftharpoons(e))$; *and*
  - v_O *is the restriction of* $v_{\widehat{O}}$ *to the exported output events* O.

Constituent components in a synchronous composite component act independently from each other. The product measure of the constituent transitions in (4.1) corresponds to an independent sample of the next states and output event vectors of the components. The output event vector is delayed until the next event to avoid feedback loops between components formed by the channel function $\rightleftharpoons$.

Similarly to synchronous composite components, the structure of synchronous cascade components can be taken from [**?**] as is:

Definition 16 (Synchronous cascade component) *A* synchronous cascade component *is a tuple* $\copyright = (\mathbf{C}, Exec, I, O, \rightleftharpoons)$*, where*

- $\mathbf{C} = \{\ominus_1, \ldots, \ominus_K\}$ *is the set of* synchronous stochastic components *that constitute the composite component, such that* $\ominus_k = (S_k, \pi_k^0, I_k, O_k, T_k)$ *for all* $k = 1, \ldots, K$*;*

- $Exec \in \mathbf{C}^*$ *is a finite* ordered sequence *(possibly with repetitions) of synchronous components called the* execution sequence *that specifies the order of components to be executed in an execution cycle;*

- $I \subseteq \widehat{I}$ *is the set of* exported input events*, where* $\widehat{I} = \bigcup_{k=1}^{K} I_k$*;*

- $O \subseteq \widehat{O}$ *is the set of* exported input events*, where* $\widehat{O} = \bigcup_{k=1}^{K} O_k$*;*

- *inputs and outputs are disjoint no two components have common outputs, i.e.,* $\widehat{I} \cap \widehat{O} = \emptyset$ *and we have* $O_k \cap O_\ell = \emptyset$ *for all* $k \neq \ell$*; and*

- $\rightleftharpoons\colon \widehat{I} \setminus I \to \widehat{O}$ *is the* channel *function. For each* $e \in \widehat{I} \setminus I$*, we must have* $Dom(e) = Dom(\rightleftharpoons(e))$.

To define the semantics of cascade composition, we define the $lift_i$ and $turn_i$ functions. The $lift_i$ function lifts the transition distribution of the ith constituent component to the level of the composite component if all other subcomponents remain in the same state. The $turn_i$ function calculates the transition distribution of the composite component if only the ith constituent component can change state. The $turn_i$ function assigns the pushforward distribution of the $lift_i$ function and the transition distribution of the ith subcomponent. The transition function of the cascade composite component is defined by binding together the $turn_i$ functions in the order of the execution list.

Definition 17 (Semantics of cascade composition) *A cascade composite component* $ⓒ = (\mathbf{C}, Exec, I, O, \rightleftharpoons)$ *is a synchronous stochastic component* $ⓒ\rangle\ominus = (S, \pi^0, I, O, T)$ *as follows:*

- $S = (S_1 \times \cdots S_K) \times V_{\widehat{O}}$ *is the set of potential states;*
- π^0 *is the product measure* $\prod_{k=1}^{K} \pi_k^0 \times \delta(\perp_{\widehat{O}})$.
- *To calculate* $T(s, v_I)$*, first let* $\mu_0 = \delta(s)$*. We iteratively compute* $\mu_i = \mu_{i-1} \gg\!= turn_i$ *for each* $i = 1, \ldots, |Exec|$*, where* $Exec[i] = \ominus_k$*,*

$$turn_i\left(\left(s_1^{(i-1)}, \ldots, s_k^{(i-1)}, \ldots, s_K^{(i-1)}\right), v_{\widehat{O}}^{(i-1)}\right) = lift_{i*}\left(T_k\left(s_k^{(i-1)}, v_{I_k}^{(i)}\right)\right), \tag{4.2}$$

$$lift_i\left(s_k^{(i)}, v_{O_k}^{(i)}\right) = \left(\left(s_1^{(i-1)}, \ldots, s_{k-1}^{(i-1)}, s_k^{(i)}, s_{k+1}^{(i-1)}, \ldots, s_K^{(i-1)}\right), v_{\widehat{O}}^{(i)}\right), \tag{4.3}$$

  - *for all* $e \in I_k \cap I$, $v_{I_k}^{(i)}(e) = v_I(e))$,
  - *for all* $e \in I_k \setminus I$, $v_{I_k}^{(i)}(e) = v_{\widehat{O}}^{(i-1)}(\rightleftharpoons(e)))$,
  - *for all* $e \in O_k$, $v_{\widehat{O}}^{(i)} = v_{O_k}^{(i)}(e)$*, and*
  - *for all* $e \in \widehat{O} \setminus O_k$, $v_{\widehat{O}}^{(i)} = v_{\widehat{O}}^{(i-1)}(e)$.

  Finally, let $out((s_1, \ldots, s_K), v_{\widehat{O}}) = (((s_1, \ldots, s_K), v_{\widehat{O}}), v_O)$*, where* v_O *is the restriction of* $v_{\widehat{O}}$ *to* O*. Then* $T(s, v_I) = out_*(\mu_{|Exec|})$.

4.1 Synchronous elementary stochastic components

Two types of stochastic elementary components can have synchronous behavior: stochastic sample and stochastic switch.

Definition 18 (Synchronous stochastic sample) *A* synchronous stochastic simple *is a synchronous component* $\ominus_s = (S, \pi^0, I, O, T)$ *with the following restrictions,*

- $S = \{s_0\}$ $\ominus_s$ *has only one state;*
- $\pi^0 \in \delta(s_0)$ *the initial state is always* s_0*;*
- $|I| = |O| = n, n \in \mathbf{N}^+$ *the number of input and output events are the same* $I = \{i_1, \ldots i_n\}$ *and* $O = \{o_1, \ldots o_n\}$*;*
- $Dom(i_i) = Dom(o_i)$*;*
- $T(s_0, v_I) \in \mathcal{D}(\{(s_0, v_O) | (v_I(i_i) = \perp) \rightarrow (v_O(o_i) = \perp)\})$*;*

Stochastic switch

The stochastic switch is just syntactic sugar since any stochastic switch can be replaced by a stochastic sample and a statechart, which sends out the incoming event based on the random sample. The distribution of the stochastic sample must be categorical distribution, and the statechart only has one state. Therefore, we do not define the semantics of stochastic switches separately.

Chapter 5

Semantics of stochastic asynchronous components

5.1 Timing of events

As time passes, events may occur randomly in stochastic components. As a result, we shall extend the definition of event instances with occurrence time. Arming event instances model the occurrence of a given event instance at a given time. An arming event instance consists of an event instance and an alarm clock, which determines when the event instance will occur. This alarm clock is defined by the absolute time of occurrence, which is positive real number $t \in \mathbb{R}^+$. We introduce arming vectors to model a given set of arming event instances. Arming vectors define a set of arming event instances.

Definition 19 (Armed event instance) *Let be $ei_t = (t, (e, p))$ an arming event instance. The set of arming instances of an event is $arm(e) = \mathbb{R}^+ \times inst(e)$. The arming instance $(\tau, (e, p)) \in arm(e)$ corresponds to the event e occurring with parameter $p \in Dom(e)$ in exactly τ units of time. To describe situations where an event may not be armed, we define $arm_\perp(e) = arm(e) \uplus \{\perp\}$. We endow these sets with the corresponding product and disjoint union σ-algebras $\mathcal{F}(arm(e))$ and $\mathcal{F}(arm_\perp(e))$, respectively, to form measurable spaces.*

Definition 20 (Arming vector) *An* arming vector *av_E over a finite set of events E is a countable set of arming event instances, which may be empty, where $av_E \subseteq (\prod_{e \in E} arm(e)) \uplus \{\perp\}$. We use Arm_E to denote the set of all arming vectors over E and endow it with the product σ-algebra $\mathcal{F}(Arm_E) = \prod_{e \in E} \mathcal{F}(arm_\perp(e))$ to form a measurable space. We write $\pm_E$ for the* empty arming vector*, where $\pm_E(e) = \perp$ for all $e \in E$.*

5.2 Stochastic asynchronous component

The stochastic asynchronous component contains initial state distribution instead of an initial state, and the transition function returns with a distribution. Contrarily to synchronous components, the initial state distribution also determines the initial events besides the initial state.

Stochastic asynchronous components can generate the arming event instances during transitions. The transition function of the distributions determine both the parameters and the time of the arming instances. The collection of the arming events is called the alert

input of the component. As a result, the component may generate its own input event instances on each transition. In stochastic asynchronous components, two or more events may occur at once. In this case, we use a fixed priority sequence to determine in which order the events will affect the component.

Moreover, if a stochastic asynchronous component transitions, the component cannot generate more than one output event instance. Multiple output events can occur at the same time, if the component generates multiple arming input event instances, when a transition occurs. The time of the new arming instances shall be the actual time. In the semantics there is no dedicated notation for actual time. All timing information is contained by arming vectors.

Definition 21 (Asynchronous component) *An* asynchronous component *is a tuple* $\ominus = (S, \pi^0, I, A, Prio_A, O, T)$*, where*

- S *is the measurable space of potential* states*;*
- $\pi^0 \in \mathcal{D}(S \times Arm_A)$ *is the* initial state and arming vector probability distribution*, which determines the (random) initial state of the component;*
- I *and* O *are the finite sets of* input *and* output events*, respectively, with* $I \cap O = \emptyset$*, while the set of all events is denoted by* $E = I \cup U$*;*
- $A \subseteq I$ *is the set of* alert input events*;*
- $Prio_A \in A^*$ *is the* alert input event priority sequence*, in which every alert input event* $e \in A$ *occurs exactly once. Simultaneous alert input events will be processed according to this ordering.*
- $T \colon S \times inst(I) \overset{\mathrm{m}}{\to} \mathcal{D}(S \times inst_\perp(O) \times Arm_A)$ *is the* transition function*, which determines the (random) next state, the output event (if any), and the alert input events to be armed when executing in a given state with a given input event instance.*

Nondeterminism of asynchronous components

The behavior described by the transition function T of the asynchronous component is deterministic, but probabilistic. This is in contrast with [**?**], where nondeterminism was suggested to resolve issues with concurrency, such as between parallel regions of a statechart. We instead resolve the concurrency by either fixed or random choice.

5.3 Stochastic asynchronous adapter

The asynchronous adapters are similar to adapters in GCL. In SGCL, the adapters contain a priority sequence to determine in which order the outputs of a synchronous component are percieved by its environment. Moreover, the adapters cannot contain clock in SGCL. Periodic execution can be modeled with composition and periodic event sources.

Definition 22 (Asynchronous adapter) *An* asynchronous adapter *for a synchronous component is a tuple* $\boxplus = (\ominus, e_c, Trig, Prio_O)$*, where*

- $\ominus = (S_s, \pi^0_s, I_s, O_s, T_s)$ *is the* wrapped synchronous component*;*

- $e_c \notin I_s$ *is the* control event*, which has no parameters* $Dom(e_c) = \top$*; and*
- $Trig \subseteq I_s \cup \{e_c\}$ *is the set of* trigger events *that cause the component to execute. Other incoming events are buffered until a trigger event is received. By convention, we require that* $e_c \in Trig$*.*
- $Prio_O \in O_s^*$ *is an* output event priority sequence*, in which each output event* $e \in O_s$ *appears exactly once. Simultaneous output events will be emitted according to this ordering. This is in contrast with [***?** *], where a non-deterministic ordering (any permutation of output events is allowed) is used.*

The asynchronous adapters buffer the input and output event instances in queues. Initially, all input and output queues are empty. The adapter state consists of the state of the wrapped component and the input and output buffers. The execution of the wrapped synchronous component can be triggered externally by a dedicated e_c control event or by an input event. In addition, the adapter can be triggered internally by the internal alert event e_o of the adapter. Additionally, we will use the notation $av_{e_o}(v) = \pm_A$ if $v = \bot_O$, otherwise $av_{e_o}(v) = \{e_o \mapsto (0, (e_o, \top))\}$, which schedules the alert input event e_o immediately.

Definition 23 (Semantics of asynchronous adapters) *An asynchronous adapter* $\oplus\!\!\!\!\!-\!\!\!\!\!\ominus = (\ominus, e_c, Trig, Prio_O)$ *(where* $\ominus = (S_s, \pi_s^0, I_s, O_s, T_s)$ *is the wrapped synchronous component) is an asynchronous component* $\oplus\!\!\!\!\!-\!\!\!\!\!\ominus\rangle\!\ominus = (S, \pi^0, I, A, Prio_A, O, T)$*:*

- $S = S_s \times V_I \times V_O$ *is the set of potential states* $(s_s, v_I, v_O) \in S$ *formed by pairs of wrapped synchronous component states and buffered input and output event vectors.*
- $\pi^0 = (\pi_s^0 \times \delta(\bot_I) \times \delta(\bot_O)) \times \delta(\pm_\emptyset)$ *is the initial state and arming vector probability distribution, which is the initial state probability distribution of the wrapped synchronous component with no buffered events or armed input events affixed.*
- $I = I_s \uplus \{e_c, e_o\}$ *is the set of wrapped and control inputs, where* $e_o \notin I_s$ *is the internal* output alert event *with* $Dom(e_o) = \top$*.*
- $A = \{e_o\}$ *and* $Prio_A = \langle e_o \rangle$*, because only the output alert event will be used to trigger timing-dependent behavior.*
- $O = O_s$ *is the set of wrapped outputs.*
- *The transition function* $T((s_s, v_I, v_O), (e_I, p_I))$ *is defined as follows:*
  - *First, if* $e_I \in \{e_c, e_o\}$*, let* $v_I' = v_I$*. Otherwise, let* $v_I' = v_I \triangleright \{e_I \mapsto (e_I, p_I)\}$*.*
  - *If* $e_I = e_o$*, then we may output a single output event from our buffer* v_O*.*
    - *If* $v_O = \bot_O$*, we set* $T((s_s, v_I, v_O), (e_I, p_I)) = \delta((s_s, v_I', v_O), \bot, \pm_A)$*, since there is no output event to flush from the buffer.*
    - *Otherwise, let* e_p *be the first event in* $Prio_O$ *such that* $v_O(e_p) \neq \bot$ *and let* $v_O'(e_p) = \bot$ *while* $v_O'(e) = v_O(e)$ *for all* $e \in O \setminus \{e_p\}$*.*

$$T((s_s, v_I, v_O), (e_I, p_I)) = \delta((s_s, v_I', v_O'), v_O(e_p), av_{e_o}(v_O')). \tag{5.1}$$

    Thus, we flush the event e_p *with the highest priority from the buffer and schedule the next output event to be flushed immediately if there is any.*
  - *If* $e_I \notin Trig$*, then let* $T((s_s, v_I, v_O), (e_I, p_I)) = \delta((s_s, v_I', v_O), \bot, \pm_A)$*, adding the input event instance to the buffer for further processing.*

– *If $e_I \in Trig$, then we may let the wrapped synchronous component process the buffered events. Consider the function lift*: $S_s \times V_O \xrightarrow{\text{m}} S \times inst_\perp(O) \times Arm_A$ *defined as*

$$lift(s'_s, v'_O) = ((s'_s, \perp_I, v_O \triangleright v'_O), \perp, av_{e_o}(v_O \triangleright v'_O)). \tag{5.2}$$

Then $T((s_s, v_I, v_O), (e_I, p_I)) = lift_*(T_s(s_s, v'_I))$.

Limits for queue size

In GCL and SGCL, one can specify an upper limit for the queue size. Finite queue size is essential for analysis methods, which explore the whole state space of the model. Without a queue size limit, the state space would be infinite. In IoT applications, it is hard to give an upper limit to the queue size. Using simulation-based analysis methods such as Monte-Carlo algorithms do not need finite queue length if the probability of infinite queue length is zero.

5.4 Stochastic asynchronous composite component

In SGCL, we extended the definition of asynchronous composite components with a $Prio_{\mathbf{C}}$ priority sequence, which resolves potential nondeterminism in a composite asynchronous component. Nondeterminism may appear in the model if a channel connects an output event of a subcomponent to the input events of two or more subcomponents. The priority sequence defines in which order the asynchronous components can react to events.

Definition 24 (Asynchronous composite component) *An* asynchronous composite component *is a tuple* $ⓐ = (\mathbf{C}, Prio_{\mathbf{C}}, I, O, \rightleftharpoons)$*, where*

- $\mathbf{C} = \{⊜_1, \ldots, ⊜_K\}$ *is the set of* asynchronous stochastic components *that constitute the composite component, such that* $⊜_k = (S_k, \pi^0_k, I_k, A_k, Prio_{A_k}, O_k, T_k)$ *for all* $k = 1, \ldots, K$*;*
- $Prio_{\mathbf{C}}$ *is the* component priority sequence*, where each component* $⊜_k \in \mathbf{C}$ *occurs exactly once;*
- $I \subseteq \widehat{I}$ *is the set of* exported input events*, where* $\widehat{I} = \bigcup_{k=1}^{K} I_k$*;*
- *we have* $A \subseteq I$*, where* $A = \bigcup_{k=1}^{K} A_k$ *is the set of all alert input events;*
- $O \subseteq \widehat{O}$ *is the set of* exported output events*, where* $\widehat{O} = \bigcup_{k=1}^{K} O_k$*;*
- *inputs and outputs are disjoint and no two components have common alert inputs or outputs, i.e.,* $\widehat{I} \cap \widehat{O} = \emptyset$ *and we have* $A_k \cap A_\ell = O_k \cap O_\ell = \emptyset$ *for all* $k \neq \ell$*; and*
- $\rightleftharpoons \subseteq \widehat{O} \times \widehat{I}$ *is the* channel *relation that connects arbitrary outputs to arbitrary inputs. For each* $o \rightleftharpoons i$*, we must have* $Dom(o) = Dom(i)$*. To ensure that the behavior of the composite component remains deterministic, we require that there is no constituent asynchronous component* $⊜_k$ *with* $i_1, i_2 \in I_k$ *for all* $o \in \widehat{O}$*,* $o \rightleftharpoons i_1$*, and* $o \rightleftharpoons i_2$ *with* $i_1 \neq i_2$*, i.e., a single output event is routed to at most one input event of any constituent asynchronous component.*

In stochastic asynchronous components, input queues contain the input event instances of the subcomponents until the events are processed. The queues are defined as a sequence of event instances. The state of an asynchronous composite component consists of the states of the subcomponents and their input queues..

The transition function of the composite component is determined by the sequential transitioning of its subcomponents. Only one subcomponent can activate transition at once. Each subcomponent executes state transition the following way. Firstly, the subcomponent gets the last event instance from its input queue. Then the subcomponent transitions into a new state, and generates some new event instances.

The output event instances are redirected to the input buffers of other subcomponents via the channels. Event instances in the input queues are generated within the composite component. Consequently, these events are defined as alarm input events. The component priority sequence defines which component will transition first if multiple queues have events. As a result, the nondeterminism of the asynchronous components can be resolved.

Definition 25 (Semantics of asynchronous composition) *For every asynchronous composite component* $ⓐ = (\mathbf{C}, Prio_{\mathbf{C}}, I, O, \rightleftharpoons)$ *we may construct an asynchronous component* $ⓐ\rangle\!\ominus = (S, \pi^0, I \cup \{e_s\}, A, Prio_A, O, T)$ *as follows:*

- $S = (S_1 \times \cdots \times S_K) \times (inst(I_1)^* \times \cdots \times inst(I_K)^*)$ *is the product state space of the constituent asynchronous components and their corresponding input event queues.*
- *Consider the function combine*: $\prod_{k=1}^{K}(S_k \times Arm_{A_k}) \xrightarrow{\mathrm{m}} S \times Arm_A$, *where*

$$combine((s_1, av_1), \ldots, (s_K, av_K)) = (((s_1, \ldots, s_K), (\langle\rangle, \ldots, \langle\rangle)), av_1 \uplus \cdots \uplus av_K). \tag{5.3}$$

  Then $\pi_0 = combine_*(\pi_1^0 \times \cdots \times \pi_K^0)$.
- $A = \{e_s\} \uplus A_1 \uplus \cdots \uplus A_k$ *is the set of all alert input events, where* e_s *is the* step alert event *with* $Dom(e_s) = \top$.
- *Let* $Prio_{\mathbf{C}} = \langle \ominus_1, \ldots, \ominus_K \rangle$. *Then* $Prio_A = \langle e_s \rangle +\!\!+ Prio_{A_1} +\!\!+ \cdots +\!\!+ Prio_{A_K}$ *is the alert input priority sequence, which contains the alert input priority sequences of the constituent asynchronous components in the order of their component priority.*
- *The transition function* $T(((s_1, \ldots, s_K), (q_1, \ldots, q_K)), (e_I, p_I))$ *is defined as follows:*
  - *If* $e_I = e_s$, *then we may execute one of the constituent asynchronous components for a single step, possibly scheduling the execution of the next step immediately afterwards.*
    - *If* $q_1 = \ldots = q_K = \langle\rangle$, *then there is no queued input event to process. Thus, we may set* $T(s, (e_I, p_I)) = \delta(s, \bot, \pm_A)$.
    - *Otherwise, let* k *be the index of the component* $\ominus_k$ *occurring earliest in* $Prio_{\mathbf{C}}$ *such that* $q_k \neq \langle\rangle$ *and let* $q_k = \langle (e, p) \rangle +\!\!+ q_k'$. *For all other* $i \neq k$, *let* $q_i' = q_i$. *Let us consider the function lift*: $S_k \times inst_\bot(O_k) \times Arm_{A_k} \xrightarrow{\mathrm{m}} S \times inst_\bot(O) \times Arm_A$,

$$lift(s_k', ei_k, av_k) = (((s_1, \ldots, s_{k-1}, s_k', s_{k+1}, \ldots, s_K), (q_1'', \ldots, q_K'')), ei, av). \tag{5.4}$$

      In the definition above, if $ei_k = \bot$, *then* $q_i'' = q_i'$ *for all* $i = 1, \ldots, K$ *and* $ei = \bot$. *If* $ei_k = (e', p')$, *then let* $q_i'' = \langle (e'', p') \rangle +\!\!+ q_i'$ *for all components* $\ominus_i \in \mathbf{C}$ *and events* $e' \rightleftharpoons e''$ *such that* $e'' \in I_i$. *For all other components*

$\ominus_j \in \mathbf{C}$, *let* $q''_j = q'_j$ *instead. Moreover, let* $ei = (e', p')$ *if* $e' \in O$, $\perp$ *otherwise. Lastly, if* $q''_1 = \ldots = q''_K = \langle\rangle$, *let* $av = av_K$, *else let* $av = av_K \cup \{e_s \mapsto (t, (e_s, \top))\}$, *where* $t \in \mathbf{R}^+$ *is the actual time. Thus we obtain* $T(s, (e_I, p_I)) = lift_*(T_k(s_k, (e, p)))$.

– *Otherwise, we add the input event* (e_I, p_I) *to the respective queue and schedule an execution step immediately afterwards. Formally, let* $q'_k = q_k +\!\!+ \langle (e_I, p_I) \rangle$ *for the component* $\ominus_k \in \mathbf{C}$ *such that* $e_I \in I_k$ *and* $q'_i = q_i$ *for all other components* $\ominus_i \in \mathbf{C}$. *Then we may set*

$$T(s, (e_I, p_I)) = \delta(((s_1, \ldots, s_K), (q'_1, \ldots, q'_K)), \perp, \{e_s \mapsto (0, (e_s, \top))\}). \quad (5.5)$$

5.5 Semantics of elementary stochastic components

The set of elementary components is a subset of stochastic components, where we made the following restrictions the component must have only one state, and all transitions are loops. There are four types of elementary stochastic components: stochastic event source, periodic event source, delay, and sample. The event filters and the stochastic rules are just syntactic sugar to facilitate the definition of the transition function of stochastic elementary components.

5.5.1 Stochastic event source

The event source components raises their output events only once at a random time. The event source has no real input events only alarm inputs. Each output event has its own alarm input event. The output events have no parameters. The initial distribution defines when the output events of the event source will occur. The transitions function generates the corresponding output event for each alarm input event with a Dirac distribution. The transition function generates no arming input event.

Definition 26 (Stochastic event source) *Stochastic event source is a stochastic asynchronous component* $\ominus_{es} = (S, \pi^0, I, A, Prio_A, O, T)$.

- $S = s_0$ *has only one state,*
- $I = A$, *all input events of a stochastic event source are alert input events,*
- *the number of output, input and alert input events are the same* $|A| = |O| = n, n \in \mathbb{N}^+$ *and let be* $A = a_1, a_2, \ldots a_n$ *and* $O = o_1, o_2 \ldots o_n$,
- $Dom(o_i) = Dom(a_i) = \top$ *for all* $i = 1, 2, \ldots n$,
- $\pi^0 \in \mathcal{D}(\{(s_0, \{(t_1, (a_1, \top)), \ldots (t_n, (a_n, \top)))\})|t_1, \ldots t_n \in \mathbf{R}^+\}$,
- $T(s_0, (t, (a_i, \top))) = \delta(s_0, (o_i, \top), \pm_A)$

5.5.2 Stochastic periodic event source

The definition of the periodic event source is the same as the event source except for the transition function. The transition function of periodic event sources generates other arming input event instances. If an instance of an arming input event triggers a transition, then the transition function will generate another instance of the given arming input event.

The occurrence time distribution of the new instance is defined with stochastic rules. Note that the definition of periodic event sources enables time-dependent distributions.

Definition 27 (Stochastic periodic event source) *Stochastic periodic event source is a stochastic asynchronous component* $\ominus_{pes} = (S, \pi^0, I, A, Prio_A, O, T)$.

- $S = s_0$ *has only one state,*
- $I = A$*, all input events of a stochastic event source are alert input events,*
- *the number of output, input and alert input events are the same* $|A| = |O| = n, n \in \mathbb{N}^+$ *and let be* $A = a_1, a_2, \ldots a_n$ *and* $O = o_1, o_2 \ldots o_n$*,*
- $Dom(o_i) = Dom(a_i) = \top$ *for all* $i = 1, 2, \ldots n$*,*
- $\pi^0 \in \mathcal{D}(\{(s_0, \{(t_1, (a_1, \top)), \ldots (t_n, (a_n, \top)))\})|t_1, \ldots t_n \in \mathbf{R}^+\}$*,*
- $T(s_0, (t, (a_i, \top))) \in \mathcal{D}(\{(s_0, (o_i, \top), \{(t + t_i, (a_i, \top))\})|t_i \in \mathbf{R}^+\})$ *thus the transition distribution is an element of the set of distributions where the generated output and alert input events are deterministic only the occurrence time is stochastic*

5.5.3 Stochastic delay

If a delay component receives an incoming event instance, it generates an output event instance a random time later with the same parameters as the incoming event. Each direct (not alarm) input event of a stochastic delay has its own alarm input and output events. If an direct input event instance occurs, the stochastic delay will generate an instance of the corresponding alarm input event with the same parameters. The distribution of the alarm event instance is defined by a stochastic rule. The instance of the alarm input event will generate an instance of the corresponding output event with the same parameters with the Dirac distribution.

Definition 28 (Stochastic delay) *Stochastic periodic event source is a stochastic asynchronous component* $\ominus_d = (S, \pi^0, I, A, Prio_A, O, T)$.

- $S = s_0$ *has only one state,*
- $|I| = 2 \cdot |O| = 2 \cdot |A|$*, each input event has its own alert input and output events* $|A| = |O| = n, n \in \mathbb{N}^+$ *and let be* $A = \{a_1, a_2, \ldots a_n\}$*,* $O = \{o_1, o_2 \ldots o_n\}$ *and*$I = \{i_1, \ldots i_n, a_1, \ldots a_n\}$*,*
- $Dom(o_i) = Dom(a_i) = Dom(i_i)$ *for all* $i = 1, 2, \ldots n$ *the parameters domains of the corresponding input, alarm input and output events are the same,*
- $\pi^0 = \delta(s_0, \pm)$ *has no initial arming events,*
- $T(s_0, (t, (i_i, p_i))) \in \mathcal{D}(\{(s_0, \perp, \{(t + t_i, (a_i, p_i))\})|t_i \in \mathbf{R}^+\})$
- $T(s_0, (t, (a_i, p_i))) = \delta(s_0, (o_i, p_i), \pm_A)$

5.5.4 Stochastic sample

If a sample component receives an input event instance, it generates an output event immediately with random parameters. Each input event of a stochastic sample has its own output event. If an instance of an input event occurs the stochastic sample will generate an instance of the corresponding output event immediately. The parameters distributions of the output events are defined with stochastic rules. The stochastic sample has no alarm input events.

Definition 29 (Stochastic sample) *Stochastic periodic event source is a stochastic asynchronous component* $\circledS_s = (S, \pi^0, I, A, Prio_A, O, T)$.

- $S = s_0$ *has only one state,*
- $A = \emptyset$ *has no arming input event*
- $|I| = |O|$*, each input event has its own output events* $|I| = |O| = n, n \in \mathbb{N}^+$ *and let be* $O = \{o_1, o_2 \dots o_n\}$ *and*$I = \{i_1, \dots i_n, a_1, \dots a_n\}$*,*
- $Dom(o_i) = Dom(i_i)$ *for all* $i = 1, 2, \dots n$ *the parameters domains of the corresponding input and output events are the same,*
- $\pi^0 = \delta(s_0, \pm_A)$ *has no initial arming events,*
- $T(s_0, (t, (i_i, p_{in,i})) \in \mathcal{D}(\{(s_0, (o_i, p_{out,i}), \pm_A) | p_{out,i} \in Dom(i_i)\})$

Chapter 6

Semantics of runs

Now we are ready to describe the probabilistic semantics of stochastic Gamma models. First, composite components are translated into their corresponding atomic components according to Definitions 15, 17, and 25. Synchronous components are converted into asynchronous components according to Definition 23 by constructing an asynchronous adapter. Thus, it is sufficient to describe the stochastic semantics of atomic asynchronous components.

6.1 Extended state of an asynchronous component

At a given time, the behavior of an asynchronous component is determined by its actual state and all the alarm clocks generated by the previous transitions. The state of a stochastic asynchronous component does not contain all information about the component since the alarm clocks of the arming vectors contain the timing-related behavior. As a result, a new kind of extended state has to be introduced, which contains all information about a given component at a given time.

Definition 30 (Extended State) *If we let a component run, then the actual state of the component at given time consist of the state) and the arming vector, which a contains all the arming and output event instances generated by the component. The extended state* $es = (s, av_A, ei) \in S \times Arm_A \times inst(O)$ *of a stochastic asynchronous component* $\circledcirc = (S, \pi^0, I, A, Prio_A, O, T)$ *consists of the state of the component and the arming vector of the alert input events. Let be the set of extended states* $S_{Ext} = S \times Arm_A \times inst(O)$.

Extended state and transitions

If the extended state is $es_K = (s_K, av_{S\,K}, ei_{out\,K})$
and a $(s_K, ei_{in\,K}) \to (s_{K+1}, av_{K+1}, ei_{out\,K+1})$ transition occurs,
then the extended state will change to: $es_{K+1} = (s_{K+1}, av_{S\,K} \uplus av_{K+1} \setminus \{ei_K\}, ei_{K+1})$. Let be $\mathcal{F}(S_{Ext})$ the σ-algebra of extended states and let be $F_{es} \in \mathcal{F}(S_{Ext})$ a measurable set of extended states.

6.2 Earliest event in a stochastic component

To get the earliest arming event instance of an arming vector in a stochastic asynchronous component, we define the $min^{\ominus}(Arm_A)$. This function searches for the arming event instance, which has the earliest occurrence time. If multiple arming event instances occur at a minimal time, then the function chooses the arming event instance with the highest priority according to the priority sequence of the component. The elementary stochastic components are designed in a way to prevent that two or more arming event instance generated at the same time. In case of composite stochastic models the component priority sequence guarantees the determinism.

Definition 31 (Earliest arming event instance) *Let be the earliest arming event instance of an arming vector in a stochastic asynchronous component* $min^{\ominus}(Arm_A) = (t_{min}, (e_{min}, p_{min})) \in Arm_A$*, where* $\forall ei_t = (t, (e, p)) \in Arm_A$ *either:*

- $t_{min} < t$ *or*
- $t_{min} < tande_{min}$ *is before than* e *in* $Prio_A$

6.3 Paths of asynchronous components

As a stochastic asynchronous component runs, it changes states and generates events randomly. As time passes, the number of state changes and event instances can be indefinitely large. The distribution of the runs is determined by the transition function and initial distribution of the component. Each run is determined by the timed sequence of state changes and event occurrences. All of this information is stored in the extended states. Similarly to PRISM, we use paths to model the random runs. A path contains a sequence of extended states. Note that in contrast to the paths of continuous Markov-chains and stochastic timed automata, a path of a stochastic asynchronous component does not have to contain dedicated time delays. The timing information is stored within the arming vectors within the extended states. The time of a transition can be calculated using the $min^{\ominus}$ function. The alarm clock of $min^{\ominus}(es_K)$ determine the time of the Kth transition within the path.

Definition 32 (Path of a component) *Let be path of a stochastic asynchronous component the sequence of extended states:*

$\rho = \langle es_1, es_2, \ldots, es_K \rangle \in {S_{Ext}}^K$

and the set paths of a stochastic asynchronous component:

$Path^{\ominus} = {S_{Ext}}^K$

Infinite cycles in asynchronous components

In composite asynchronous components, one can define such behavior, which generates an infinite number of events at a given time. This behavior may occur if the instance of an alert event can cause (with one probability) another instance instantaneously in an infinite cycle. Infinite events in finite time cannot be simulated and cannot occur in real systems. As a result, the ability of a component to generate infinite events within finite time is a semantic error.

If a stochastic component runs indefinitely, then the set of possible runs might be continuously infinite. The set of paths of a given component might contain infinite pieces of infinitely long paths. We introduce the cylinder sets of paths to define sigma-algebras on the set of paths. The cylinder sets can be interpreted as the intervals of paths.

Definition 33 (Cylinder sets of a path) *For path $\rho = \langle F_{es\,1}, \ldots, F_{es\,K}\rangle \subset {S_{Ext}}^K$ with $K \leq L$ extended states, the $Cyl_L(\rho)$* cylinder set *is the set of (finite or infinite) traces with the prefix ρ if $K = L$, or just the singleton set $\{\rho\}$ if $K < L$.*

Let be $\mathcal{F}(Path^{\ominus})$ be the smallest sigma algebra on $Path^{\ominus}$ which contains all cylinder sets Cyl.

Transition function as conditional distributions

Similarly to Markov chains, the transition function can be written as a conditional distribution. In this form, the transition function is the conditional distribution of the next extended state of the component, assuming the specific extended state of the component:

$$D_{\ominus}(es_{K+1}|es_K) = next_*(av_K)(T(s_K, min^{\ominus}(av_K))),$$
$$D_{\rho}(\langle es_1, \ldots es_K, es_{K+1}\rangle|\langle es_1, \ldots es_K\rangle) = next_*(av_K)(T(s_K, min^{\ominus}(av_K)))$$

where *next* assigns the next extended state to an outcome of a transition function and the arming vector of the previous extended state:

$$es_{K+1} = (s_{K+1}, av_{K+1}, ei_{K+1}),$$
$$es_K = (s_K, av_K, ei_K),$$
$$next\colon Arm \times S \times Arm \times inst(O) \overset{\mathrm{m}}{\to} S_{Ext},$$
$$next(av_K)(s_T, av_T, ei_T) = (s_T, av_K \uplus av_T, ei_T)$$

We define a probability measure on the set of paths iteratively. We use a generalized form of Kolmogorov's extension theorem since we can construct iteratively probability measures on infinitely long sequences of random phenomena using Kolmogorov's Extension Theorem. Thus we can define a probability measure the following way: firstly, We specify the probability measure for the one-long sequences; secondly, we assume that we defined the probability measure for the set of random variable sequences with a given length; finally, we define a probability measure for the set of one longer variable sequences. In the case of stochastic components, the random phenomena will be the distribution of extended states. The initial distribution π^0_{es} of a stochastic component specifies the initial extended state of the component at the beginning of each run. As transitions occur, then the state distribution changes as well. We use distribution binding to define the probability measure on the set of one longer paths. The bind operator determines the state distribution after the occurrence of a transition. If π^L_{es} denotes the state distribution after L number of transitions, then $\pi^{L+1}_{es} = \pi^L_{es} \gg\!= D_{\ominus}$ denotes the state distribution if a new transition occurs.

Definition 34 (Probability measure on a finite path) *Probability measure of an $L \in \mathbf{N}$ long path:$Pr^{\ominus}_{\rho} : \mathcal{F}(Path^{\ominus}_L) \overset{\mathrm{m}}{\to} \mathbf{R}^+$ where*

- *if L=0 then $Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_0)) = \pi^0$*
- *We assume that we defined the probability measure $Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_L))$ for all L long paths*
- *let be the probability measure for the sigma algebra of the cylinder set of all L long paths $\rho_L \in Path^{\ominus}_L$ $Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_{L+1}))$ for all $L+1$ long paths $\rho_{L+1} \in Path^{\ominus}_{L+1}$, where:*

– $es_K = (s_K, av_K)$, $(\tau_{K+1}, ei_{K+1}) = min^{\ominus}(av_K)$:
The time of the next event transition is determined by the previous extended state by getting the earliest arming event instance in the Kth arming vector.
– $Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_{L+1})) = Pr^{\ominus}_{\rho}(\mathcal{F}(\rho_L)) \gg= D_{\rho}$:
The probability measure of the L long path and the stochastic transition distribution of the next extended state are binded together.

Asynchronous components as probabilistic programs

The semantics of stochastic asynchronous components is defined similarly to probabilistic programs. There is a strong analogy between the semantics of asynchronous stochastic components and probabilistic programs. In the semantics of asynchronous components, the extended state is analogous to the set of variable name-value pairs. The stochastic transition function is analogous to the expressions of probabilistic programming languages. Finally, both the effect of sequential state transitions and the concatenation of probabilistic program blocks are defined with the distribution binding. As a result, we can construct a probabilistic program for each asynchronous component, which has the same semantics and represents the same behavior.

6.4 Traces of asynchronous components

In practical applications, we want to analyze only the output events of a given component. As a result, we introduce traces which are sequence output event instances and the time interval between them.

Definition 35 (Traces of asynchronous components) *A* trace *of the asynchronous component $\ominus = (S, \pi^0, I, A, Prio_A, O, T)$ is a finite or infinite sequence $\langle \tau_1, ei_1, \tau_2, ei_2, \ldots \rangle$ of alternating* delays τ_i *and* event instances $ei_i \in inst_{\perp}(O)$. *We let $Trace^{\ominus}(O)$ denote the set of all (finite or infinite) traces and let $Trace^{\ominus}_{\leq L}(O)$ denote set of a traces with at most L event instances.*

Definition 36 (Cylinder sets of finite traces) *For a finite trace $tr = \langle \tau_1, ei_1, \ldots, \tau_K, ei_K \rangle \in \left(\mathbb{R}^{\geq 0} \times inst_{\perp}(O)\right)^K$ with $K \leq L$ event instances, the $Cyl_L(tr) = \langle I_{\tau 1}, F_{e1}, \ldots I_{\tau L}, F_{eL} \rangle$* cylinder set *is the set of (finite or infinite) traces with the prefix tr if $K = L$, or just the singleton set $\{tr\}$ if $K < L$.*

The traces can be derived directly from paths by leaving out the arming vector and the state from the extended states. Thus, we define the *toTrace* function the following way:

Definition 37 (Path to trace transformation) *Let be the toTrace function set of traces: $toTrace : Path^{\ominus} \overset{\text{m}}{\to} Trace^{\ominus}$,where:*
$toTrace(\langle (s_0, av_0, ei_0), \ldots, (s_K, av_K, ei_K) \rangle) = \langle \tau_1, ei_1, \ldots, \tau_K, ei_K \rangle$ *and*
$(\tau_{i+1}, \_) = min^{\ominus}(av_i)$

We define a probability measure on traces using the pushforward distribution of the probability measure of paths and the *toTrace* function.

Definition 38 (Probability measure of traces) *Let be the probability measure of the sigma algebra of cylinder set of traces: $Pr^{\ominus}_{tr} : \mathcal{F}(Trace^{\ominus}) \overset{\text{m}}{\to} \mathbf{R}^+$,where: $Pr^{\ominus}_{tr} = toTrace_*(Pr^{\ominus}_{\rho})$*

6.5 Continuous Stochastic Logic

Continuous stochastic logic is a defined with time-dependent logic expressions in stochastic models. For instance, continuous stochastic logic is used together with Markov-chains, stochastic Petri-nets and stochastic timed automata. Continuous stochastic logic expressions can specify a given set of paths that satisfy or violate a given condition. Atomic expression specifies a Boolean condition on the state space of the stochastic model. Logic and temporal logic expressions can combine atomic expressions and other expressions. In addition, one can use stochastic expressions to specify the probability of paths that satisfy a given expression or the expected time until a given condition is true. Finally, continuous stochastic logic can define conditional probabilities and expected values. Continuous stochastic logic can be applied to SGCL models. The atomic expressions of the continuous stochastic logic assign a Boolean value to the output events in the traces. In the analysis components, template sentences define continuous stochastic logic expressions in a simplified way. SGCL template sentences support only a subset of continuous stochastic logic expressions. Template sentences can specify time-limited conditions on the SGCL models.