(Hybrid) Automata and (Stochastic) Programs

The hybrid automata lattice of a stochastic program

Luca Bortolussi
Dept. of Maths and Computer Science, University of Trieste.
luca@dmi.units.it

Alberto Policriti
Dept. of Maths and Computer Science, University of Udine.
policriti@dimi.uniud.it

Abstract

We define a semantics for stochastic Concurrent Constraint Programming (sCCP), a stochastic process algebra, in terms of stochastic hybrid automata with piecewise deterministic continuous dynamics. To each program we associate a lattice of hybrid models, parameterized with respect to the degree of discreteness left. We study some properties of this lattice, presenting also an alternative semantics in which the degree of discreteness can be dynamically changed.

1 Introduction

Thinking about computation in modern terms—especially when trying to reason on the kind of computation carried out by biological entities—, the following question arises naturally: Which parts of a system should be modelled as continuous and which parts as discrete?

We came to this important issue while studying the possibility to reconcile the mathematical and the computational approaches (in the sense of [FH06]) to Systems Biology [BP10]. More specifically, we have observed that while the mathematical approach to (biological/biochemical) simulation can provide mature and efficient tools formalizing the global laws modelling the dynamics of systems, the computational approach is much more natural when one needs to render a discontinuous mechanisms controlling the system.

Such a reconciliation passes through a formal correspondence established between two formalisms paradigmatic of the two approaches: one more
mathematical (differential equations) and the other computational (programs) [FH06]. In particular, in this paper we try to give a contribution towards an answer to the above question, by showing that a computationally oriented point of view, if not closing the problem, can at least frame it in a reasonable formal setting. What we are proposing here is a technique to associate programs as network of interacting agents and simulation tools with specific characteristics. The special feature of the simulation tools is that they are hybrid, in the sense that both discrete and continuous descriptions of the dynamics of a given system co-exist [Hen96, BL04, Pah09]. There are (at least) two traditions in which such tools have been proposed by (different groups of) the scientific community. One is in the tradition of computer scientists and engineers, interested in mixing an inherently discrete control program with a continuously evolving collection of variables describing (sensing) the environment [Hen96]. The other is the tradition of economists, interested in modelling quantities whose dynamics is piecewise predictable with interleaving events discontinuously modifying their evolution [Dav93]. As a matter of fact, the discontinuities just mentioned are—rather intuitively—more fruitfully presented as stochastically governed, a character that brings us to the special feature of the programs we have in mind to use for our association. We think of stochastic process algebras used to specify networks of communicating agents and we prove that we can ”implement” such stochastic programs by simulation of the associated (stochastic) hybrid system [Dav93, Pah09].

Before commenting on the positive effects of this activity, let us be a bit more precise on our technique. The programming style in which the computational side of our approach is given, is in the tradition of computation as message-based concurrent agents communication. In this tradition programs are nothing but a collection of agents interacting by message-passing, and characterized by a number of states in which they can be found. The stochastic ingredient is introduced by a government of the speed at which interaction can take place. Such speeds are given as rates that are functions of the overall state of the (recorded part of) the system. The formal description of stochastic Concurrent Constraint Programming (sCCP), the language we use, can be found in [Bor06], [BP08b]. As the reader can imagine at this point, the specificities of the language used are no more than technical aspects that, even though important, are not impossible to change.

Starting from a sCCP program we prove that a lattice of hybrid automata can be produced, whose different elements represent the level of discreteness that we are leaving in the ”implementation” by hybrid system simulation of the sCCP program. The extreme points are a fluid-flow approximation [BP09a] consisting of a unique system of differential equations, and a Continuous Time Markov Chain [Nor97] with discrete transitions corresponding to every change of state of every agent on each domain value.
While the fluid-flow approximation is prone to be efficient, as it is completely deterministic, its adherence to the intended semantics of the program is often very questionable [BP09a]. The other extreme has a tendency to produce a large number of states and therefore, even though more precise, is often not the best possible choice. As we prove in Section 8 a very powerful (dynamic) mechanism to direct the choice is available in our setting.

Another positive aspect of the setting we put forward, is a direct consequence of the formalization of the procedure. The correspondence between automata and programs can be, at least in principle, used for studying various properties of the program in correspondence with the continuous/discrete dynamics obtained at the chosen level of precision. Behavioural properties, reachability questions (and, consequently, all the many properties reducible to reachability), and error estimates for simulations, become viable to studies carried out at different degrees of precision for the given stochastic program. Such studies can be interesting, as they can be seen as a variate collection of tests carried out the program—as well as on the (stochastic) parameters it contains—for validation against data.

In the following, we first introduce Piecewise Deterministic Markov Processes [Dav93] (PDMP), a version of Stochastic Hybrid Automata with deterministic continuous evolution that will be the target of sCCP semantics (Section 2). Then, we present Transition-Driven Stochastic Hybrid Automata (TDSHA), a formalism which can be seen as an higher-level language to describe PDMP, rendering in an explicit way the structure of discrete transitions (Section 3). Next, we discuss in detail the relationships between PDMP and TDSHA, showing how to define a PDMP starting from a TDSHA (Section 4). In Section 5, we move towards the core of the paper, introducing stochastic Concurrent Constraint Programming (sCCP). The definition of the hybrid semantics is given in Section 6, for a fixed degree of discreteness. In Section 6.1, we also discuss how to remove stochasticity from discrete transitions, in order to obtain a semantics in terms of Hybrid Automata. Considering all possible choices of degree of discreteness, we obtain a lattice of models, whose definition and main properties are the content of Section 7. Finally, in Section 8 we discuss how to formally define a dynamic partitioning scheme, and in Section 9 we draw final conclusions.

2 Piecewise Deterministic Markov Processes

We first introduce a class of Stochastic Hybrid Automata known as Piecewise Deterministic Markov Processes (PDMP, [Dav93]). They have been introduced in the late 80s as tools to model financial and insurance problems. They basically are Stochastic Hybrid Automata with a continuous dynamics based on ODE and a discrete and stochastic dynamics given by a Markov jump process. Below we recall their definition.
Definition 2.1. A PDMP is a tuple \((Q, X, D, \lambda, R, q_0, x_0)\), such that:

- \(Q\) is a finite set of \textit{discrete states}\(^1\). \(X = \{X_1, \ldots, X_n\}\) is a finite set of real-valued variables. For each \(q \in Q\), let \(D_q \subset \mathbb{R}^n\) be an open set, the continuous domain of mode \(q\). Its boundary is denoted by \(\partial D_q\) and its closure by \(\overline{D}_q\).

- \(q_0\) is the initial mode of the PDMP, while \(x_0\) is the initial value of system variables.

- \(D\) is the \textit{hybrid state space}, defined as the disjoint union of the \(D_q\)'s, namely \(D = \bigcup_{q \in Q} \{q\} \times D_q\). A point \(y \in D\) is thus a pair \(y = (q, x)\), \(x \in D_q\). A subset of \(D\), therefore, is the disjoint union of subsets of the corresponding \(D_q\)'s: \(A = \bigcup_{q \in Q} \{q\} \times A_q\). For each \(D_q\), we consider the Borel \(\sigma\)-algebra \(\mathcal{D}_q\) obtained by restricting the standard Borel \(\sigma\)-algebra of \(\mathbb{R}^n\) to \(D_q\). We also define the \(\sigma\)-algebra \(\mathcal{D} = \{A \in \mathcal{D} | A = \bigcup_{q \in Q} \{q\} \times A_q, A_q \in \mathcal{D}_q\}\) on \(D\).

- To each mode \(q \in Q\) we associate an \textit{autonomous} vector field \(X_q : D_q \rightarrow \mathbb{R}^n\), which is assumed to be \textit{locally Lipschitz continuous}. The flow of such vector field is indicated by \(\phi_q(t, x_0)\), denoting the point reached at time \(t\) starting from \(x_0 \in D_q\) at time 0. We assume that the flow of \(X_q\) is such that solutions do not diverge in finite time.

- \(\lambda : D \rightarrow \mathbb{R}^+\) is the \textit{jump rate} and it gives the hazard of executing a discrete transition. It is required to satisfy the following property:

\[\forall y_0 = (q, x_0) \in D, \exists \epsilon(y_0) > 0 : t \mapsto \lambda(q, \phi_q(t, x_0)) \text{ is integrable in } [0, \epsilon(y_0)].\]  

\([1]\)

- \(R : (D \cup \partial D) \times D \rightarrow [0, 1]\) is the \textit{transition measure} or \textit{reset kernel}. It maps each \(y \in D \cup \partial D\) on a probability measure on \((D, \mathcal{D})\), and it must satisfy:

\[\text{for each } A \in \mathcal{D}, \text{ the function } y \mapsto R(y, A) \text{ is measurable; } \]  

\[R(y, \{y\}) = 0, \text{ for each } y \in D\]  

\([2]\)

The idea of PDMP is that, within each mode \(q\), the process evolves according to the differential equation given by the vector field \(X_q\). While in a mode, the process can jump spontaneously with hazard given by the rate function \(\lambda\). Moreover, a jump will be immediately performed whenever the boundary of the state space of the current mode is hit.

\(^1\)In analogy with hybrid automata jargon, we will also use the term \textit{modes} for discrete states.
In order to formally capture such an evolution, we need to define the sequence of jump times and target states of jumps of the PDMP, given by random variables $T_1, W_1, T_2, W_2, \ldots$. $T_i$ is the time of the $i$-th jump, and $W_i$ is the target state of such a jump. In between times $T_i$ and $T_{i+1}$, the system evolves following the solution of the vector field of the discrete mode where $W_i$ lies, with initial point given by $W_i$. In the following, we let $T_0 = 0$ and $W_0 = (q_0, x_0)$ denote the initial configuration of the PDMP.

We turn now to define the sequence of random variables $(T_i, W_i)$. Let $U_1, U_2, \ldots$ be a sequence of independent random variables uniformly distributed in $(0, 1)$. Define

$$t_*(q, x) = \begin{cases} \inf \{ t > 0 \mid \phi_q(t, x) \in \partial D_q \} \\ \infty, \text{ if no such time exists,} \end{cases}$$

be the hitting time of the boundary $\partial D_q$ starting from $(q, x) \in D$.

We can define the survivor function of the next jump time $T_i$, given that the process re-started at $y = (q, x)$, by

$$F(t, y) = P(T_1 \geq t) = I_{t < t_*(y)} \exp \left( - \int_0^t \lambda(q, \phi_q(s, x)) \, ds \right).$$

This (for example) defines the probability distribution for the first jump time $T_1$, which can be expressed using the uniform random variable $U_1$, by inversion solving for $T_1$ the equation

$$F(T_1, y) = U_1.$$  \[ \text{(4)} \]

Once the time of the first jump has been drawn, we can define the target point $W_1$ as a random variable with distribution $R(y_{T_1} \cdot y_0)$, with $y_{T_1} = \phi_q(T_1, x_0)$, which can be expressed in terms of the uniform random variable $U_2$ by standard inversion methods.

From $W_1$, the process follows the solution of the vector field, until the next jump, determined by the same mechanism presented above. Hence, in general, the time of the $(i + 1)$-th jump is $T_{i+1} = T_i + S_{i+1}$, where $S_{i+1}$ is defined by solving the equation $F(S_{i+1}, W_i) = U_{2i+1}$, while the target point of the jump is obtained by inverting $R(y_{T_{i+1}} \cdot y_0)$. This defines a realization of the stochastic process in the Hilbert cube $[0, 1]^{\omega}$.

A further requirement on PDMP is that, letting $N_t = \sum_k I_{t > T_k}$ be the random variable counting the number of jumps up to time $t$, it holds that $N_t$ is finite with probability 1, i.e.

$$\forall t, \mathbb{E}N_t < \infty.$$  \[ \text{(4)} \]

$\text{I}_\varphi$ stands for the indicator function, whose value is 1 if condition $\varphi$ holds and 0 otherwise.
This condition is enforced to rule out pathologic behaviors like Zeno trajectories. Indeed, problems may arise by the interaction between the reset kernel and the hitting of boundaries, see [Dav93] for further details.

**Example 2.1.** In order to exemplify all the previous notions, we consider a PDMP model of a simple genetic network, in which we have a single gene whose produced protein $P$ acts as a self-repressor. We assume a simple repression model: The binding of a protein to the promoter region of the gene is proportional to the concentration of the protein itself. Unbinding, instead, happens at a constant rate. Both these two events are stochastic. On the other hand, we model production and degradation of the protein $P$ by a simple differential equation. Production happens only if the gene is not repressed, at rate $\alpha$, while degradation happens always at rate $\beta P$.

The PDMP model $(Q, X, D, \lambda, \lambda, q_0, x_0)$ is defined in the following way:

- The set of discrete modes contains one state for the unrepressed gene and one state for the repressed gene: $Q = \{q_{on}, q_{off}\}$. There is one single variable $P$, and the state space is unconstrained in each mode: $D_{q_{on}} = D_{q_{off}} = \mathbb{R}$.
- The vector field is $X_{q_{on}}(P) = \alpha - \beta P$ when the gene produces $P$, and $X_{q_{off}}(P) = -\beta P$ when the gene is repressed.
- The stochastic rate is $\lambda_{q_{on}}(P) = k_b P$ (binding rate) and $\lambda_{q_{off}}(P) = k_u$ (unbinding rate).
- The reset kernel simply makes the PDMP change discrete mode when a stochastic transition fires. Hence, $R((q_{on}, P), \cdot) = (q_{off}, P)$, and $R((q_{off}, P), \cdot) = (q_{on}, P)$.

A simulated trajectory of this model is shown in Figure 1(a).

## 3 Transition-Driven Stochastic Hybrid Automata

In this section we present *Transition-Driven Stochastic Hybrid Automata* (TDSHA), a formalism introduced in [BP09c], [BP09e] as a convenient intermediate layer to map sCCP programs to PDMP or hybrid automata. More specifically, in [BP09e] we introduced a simplified version, which was used as a front-end language for hybrid automata. The full-fledged version on TDSHA appeared, instead, in [BP09c]. The basic idea in definition of TDSHA is to prepare for a manipulation (a.k.a. *merge/split*) of discrete and continuous states.

The hybrid nature of TDSHA manifests itself primarily in the structure of the state space, composed by a (finite) set of *discrete modes* and a finite set of *continuous variables*, as customary for hybrid systems.
The characterizing feature of TDSHA, however, is the emphasis on transitions, which can be of three types:

**Continuous**, describing the continuous dynamics within each mode. Each transition represents a different flow that modifies system’s variables.

**Instantaneous**, describing discrete jumps that happen instantaneously when their activation conditions are satisfied.

**Stochastic**, describing discrete jumps happening stochastically in time, following an exponential probability distribution depending on a rate function. Stochastic and instantaneous transitions can change mode and reset the value of the variables.

**Definition 3.1.** A Transition-Driven Stochastic Hybrid Automaton (TDSHA) is a tuple 
\[ T = (Q, X, TC, TD, TS, init, E), \]
where:

- \( Q \) is a finite set of control modes.
- \( X = \{X_1, \ldots, X_n\} \) is a set of real valued system’s variables.
- \( E \) is a set of transition labels or events, which are relevant for synchronization.
- \( TC \) is the set of continuous transitions or flows, whose elements are tuples \( \tau = (\alpha, q, s, f) \), where:
  - \( \alpha \in E \) is the event label associated to the transition and referred to by \( \text{event}[\tau] \).
  - \( q \in Q \) is the discrete mode in which the transition is active. Given a continuous transition \( \tau \), its mode is referred to by \( \text{cmode}[\tau] \).
  - \( s \) is a vector in \( \mathbb{R}^n \), i.e. of size \( |X| \), denoting the stoichiometry or magnitude of the flow, i.e. the intensity of the effect of the transition for each variable. It is referred to by \( \text{stoich}[\tau] \).
  - \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a (sufficiently smooth) function, specifying the functional form of the flow. It is referred to by \( \text{rate}[\tau] \).
- \( TD \) is the set of instantaneous transitions, whose elements are tuples of the form \( \delta = (\alpha, q_1, q_2, \text{guard}, \text{reset}, p) \), where:

---

Notation: the time derivative of \( X_j \) is denoted by \( X'_j \), while the value of \( X_j \) after a change of mode is indicated by \( X'_j \).
\( \alpha \in \mathcal{E} \) is the event label associated to the transition and referred to by \text{event}[\delta].

\( q_1 \) is the \text{exit-mode}, i.e. the mode in which the transition can take place. It is denoted by \text{e}_1[\delta].

\( q_2 \) is the \text{enter-mode}, i.e. the mode targeted by the transition. It is referred to by \text{e}_2[\delta].

\text{guard} is a quantifier-free first-order formula with free variables in \( \mathbf{X} \), representing the closed set \( G_\delta = \{ \mathbf{x} \in \mathbb{R}^n \mid \text{guard}[\mathbf{x}] \} \) in which the transition is active. It is referred to by \text{guard}[\delta].

\text{reset} is a deterministic update of the form \( \mathbf{X}' = r(\mathbf{X}) \). It is identified by \text{reset}[\delta]. Usually, we write the reset as a conjunction of atomic updates of the form \( X'_{i_1} = r_{i_1}(\mathbf{X}), \ldots, X'_{i_k} = r_{i_k}(\mathbf{X}) \).

The identity reset will be denoted in the following by \( \top \).

\( p : \mathbb{R}^n \rightarrow \mathbb{R}^+ \) is the \text{priority}, i.e. a weight function used to resolve non-determinism between two or more active transitions. It is denoted by \text{priority}[\delta].

- \( \mathcal{T}\mathcal{S} \) is the set of \text{stochastic transitions}, whose elements are tuples of the form

\[ \eta = (\alpha, q_1, q_2, \text{guard}, \text{reset}, f), \]

where

- \( q_1, q_2, \text{guard}, \text{reset}, \) and \( \alpha \) are defined in the same way as for transitions in \( \mathcal{T}\mathcal{D} \),
- \( f : \mathbb{R}^n \rightarrow \mathbb{R}^+ \) is the rate function giving the hazard of taking transition \( \eta \). Such function is referred to by \text{rate}[\eta].

- \( \text{init} \in Q \times \mathbb{R}^n \) is a point giving the initial state of the system.

**Remark 3.1.** Both priority and rates introduced in Definition 3.1 make TD-SHA stochastic. Priorities define, at each point, a discrete distribution of a random variable choosing among enabled instantaneous transitions. Rates, on the other hand, define a random race in continuous time, giving the delay for the next spontaneous jump.

**Remark 3.2.** The sets \( G_\delta \) are defined to be closed in order to deal correctly with urgent transitions, i.e. transitions to be taken as soon as available.

**Remark 3.3.** Continuous transitions are undoubtedly the distinguishing feature of TDSHA. Usually, hybrid systems are defined in a monolithic way.

---

\(^4\) Even though there is no real additional difficulty in considering stochastic resets—i.e. in assuming \text{reset} to be a transition measure—we decided to avoid such move for the sake of simplicity. Furthermore, all resets of the fragment of sCCP we will be considering, are of such kind.
by associating a set of differential equations to each mode. In TDSHA, instead, we describe the different flows influencing system’s variables, and then deduce a set of ODE from such information. This permits a fine-grained modelling of the continuous dynamics of the system. This approach has been borrowed from the hybrid process algebra HYPE [?]. [GBHnt].

**Example 3.1.** Consider again the simple genetic network of Example 2.1. We show now how to describe such a network by a TDSHA $\mathcal{T} = (Q, X, \mathcal{T}, \mathcal{X}, \mathcal{Y}, \mathcal{E}, \text{init}, \mathcal{E})$.

- The set of discrete modes, as in the PDMP, is $Q = \{q_{\text{on}}, q_{\text{off}}\}$, while $P$ is the only variable

- There are three continuous transitions: degradation in the two modes and production in the active mode of the gene: $(\text{prod}, q_{\text{on}}, (1, \alpha), (\text{deg}, q_{\text{on}}, (-1), \beta P), \text{and} (\text{deg}, q_{\text{off}}, (-1), \beta P)$.

- There are two stochastic transitions: the binding of the repressor to the gene $(\text{bind}, q_{\text{on}}, q_{\text{off}}, \text{true}, \tau, k_b P)$, and the unbinding $(\text{unbind}, q_{\text{off}}, q_{\text{on}}, \text{true}, \tau, k_u)$.

- $\mathcal{X} = \emptyset$, as there are no instantaneous transitions.

**Product of TDSHA.** We define now a notion of product for TDSHA. Due to the asynchronous nature of sCCP, such a product will be itself asynchronous, i.e. it will essentially ignore event labels. As a preliminary step, we define the embedding of states into pairs of states for continuous, instantaneous and stochastic transitions. Let $\mathcal{T} = (Q, X, \mathcal{T}, \mathcal{X}, \mathcal{Y}, \mathcal{E}, \text{init}, \mathcal{E})$ be a TDSHA, $\tau \in \mathcal{T}$ be a continuous transition, $u$ be a mode in a set $Q'$, $Q' \cap Q = \emptyset$, and $Y$ be a set of variables, with $X \subseteq Y$. The embedding $i_1$ of $\tau$, w.r.t. $u$ and $Y$ is defined as $i_1(\tau, u, Y) \overset{\text{def}}{=} (\text{event}[\tau], (\text{cmode}[\tau], u), \text{stoich}[\tau]^Y, \text{rate}[\tau])$, where $\text{stoich}[\tau]^Y$ is a vector on $Y$ coinciding with $\text{stoich}[\tau]$ for each $X \in X$, and equal to zero for $X \notin X$. Similarly, $i_2(\tau, u, Y) \overset{\text{def}}{=} (\text{event}[\tau], (u, \text{cmode}[\tau]), \text{stoich}[\tau]^Y, \text{rate}[\tau])$.

An analogous notion of embedding can be defined for instantaneous and stochastic transitions. Let $\delta \in \mathcal{X}$ and $\eta \in \mathcal{E}$; we set

$$
\begin{align*}
i_1(\delta, u, Y) & = (\text{event}[\delta], (e_1[\delta], u), (e_2[\delta], u), \text{guard}[\delta], \text{reset}[\delta], \text{priority}[\delta]), \\
i_2(\delta, v, Y) & = (\text{event}[\delta], (v, e_1[\delta]), (v, e_2[\delta]), \text{guard}[\delta], \text{reset}[\delta], \text{priority}[\delta]), \\
i_1(\eta, u, Y) & = (\text{event}[\eta], (e_1[\eta], u), (e_2[\eta], u), \text{guard}[\eta], \text{reset}[\eta], \text{rate}[\eta]), \\
i_2(\eta, v, Y) & = (\text{event}[\eta], (v, e_1[\eta]), (v, e_2[\eta]), \text{guard}[\eta], \text{reset}[\eta], \text{rate}[\eta]).
\end{align*}
$$

**Definition 3.2.** Given two TDSHA $\mathcal{T}_i = (Q_i, X_i, \mathcal{T}, \mathcal{X}, \mathcal{Y}, \mathcal{E}, \text{init}_i, \mathcal{E}_i)$ for $i \in \{1, 2\}$, their product $\mathcal{T} = \mathcal{T}_1 \otimes \mathcal{T}_2$, $\mathcal{T} = (Q, X, \mathcal{T}, \mathcal{X}, \mathcal{Y}, \mathcal{E}, \text{init}, \mathcal{E})$, is defined by:
\[ Q = Q_1 \times Q_2 \text{ and } X = X_1 \cup X_2; \]

\[ \mathcal{T}_C = \{i_1(\tau, v, X) \mid \tau \in \mathcal{T}_{C_1}, v \in Q_2\} \cup \{i_2(\tau, u, X) \mid \tau \in \mathcal{T}_{C_2}, u \in Q_1\}; \]

\[ \mathcal{T}_D = \{i_1(\delta, v, X) \mid \delta \in \mathcal{T}_{D_1}, v \in Q_2\} \cup \{i_2(\delta, u, X) \mid \delta \in \mathcal{T}_{D_2}, u \in Q_1\}; \]

\[ \mathcal{T}_S = \{i_1(\eta, v, X) \mid \eta \in \mathcal{T}_{S_1}, v \in Q_2\} \cup \{i_2(\eta, u, X) \mid \eta \in \mathcal{T}_{S_2}, u \in Q_1\}; \]

\[ \text{init}[(u, v)] = \text{init}_1[u] \land \text{init}_2[v]; \]

\[ \mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2. \]

The previous definition is quite natural, given the absence of synchronization mechanisms for TDSHA, yet we wish to focus the attention of the reader on the effect on flows: for each variable shared between \( T_1 \) and \( T_2 \), the result obtained is the addition of the right-hand sides of the corresponding differential equations.

**Remark 3.4.** All transitions of TDSHA are labelled by events, hence we can also define a product synchronizing transitions labelled by the same event. Introducing a notion of synchronized product, however, requires to specify a scheme to combine rates and priorities of discrete transitions being synchronized. It is difficult to establish a general rule, and different choices are possible. For instance, in the field of stochastic process algebras, there are different combinators used for stochastic rates, like the minimum and the product, see [Hil96] for a deep discussion. Hence, we are essentially assuming the correct choice of a synchronizing product for TDSHA as application-dependent.

**Dynamics of TDSHA.** Informally, we can describe the dynamics of TDSHA in the following way:

- Within each discrete mode \( q \in Q \), the system follows the solution of a system of ODE, constructed combining the effects of the continuous transitions \( \tau \) acting on mode \( q \). Essentially, the ODE for variable \( X_i \) is obtained by adding up the rate of all such \( \tau \) times the \( i \)-th component of the vector \( \text{stoich}[\tau] \):

  \[
  \dot{X} = \sum_{\tau \mid \text{cmode}[\tau] = q} \text{stoich}[\tau] \cdot \text{rate}[\tau] \text{ in mode } q \in Q.
  \]

- Two kinds of discrete jumps are possible: stochastic transitions are fired according to their rate, while instantaneous transitions are fired as soon as their guard becomes true. In both cases, the state of the system is reset according to the policy specified by \( \text{reset} \). Choice among several active stochastic or instantaneous transitions is resolved probabilistically according to their rate or priority, see Remark 3.1.
For instance, if there are two active instantaneous transitions \( \delta_1 \) and \( \delta_2 \) competing for execution, we will choose \( \delta_1 \) with probability 
\[
\frac{\text{priority}[\delta_1]}{(\text{priority}[\delta_1] + \text{priority}[\delta_2])}.
\]

- A trace of the system is therefore a sequence of (random) jumps interleaved by periods of continuous evolution.

If we want to provide a formal definition of the dynamical evolution of TDSHA, we can map them into a well-studied model of Stochastic Hybrid Automata, namely Piecewise Deterministic Markov Processes [Dav93]. This mapping will be presented in detailed in Section 4.

In this sense, TDSHA can be seen as a compositional formalism to model PDMP. Hence, they are related to communicating PDMP [SvdS07]. The main differences are in the way continuous dynamics is defined and in the restriction of TDSHA on deterministic resets (which can be, however, easily removed).

### 3.1 Determinization

We recall now the notion of \textit{determinization} of a stochastic hybrid automaton, a technique that we introduced in [BP08a] and discussed in detail in [BP10]. It essentially is a systematic way to remove stochasticity in favour of deterministic discrete transitions. The basic idea, cast in the setting of TDSHA, is to replace a stochastic transition with an instantaneous one. The instantaneous transition is subject to time constraints such that it fires at the average time of the stochastic transition.

The idea is the following: let \( \eta \in \mathcal{T}\mathcal{S} \) be a stochastic transition with rate \( f = \text{rate}[\eta] \) and guard \( g = \text{guard}[\eta] \). Let \( I_g \) be the indicator function of the guard. The firing time of transition \( \eta \) can be defined observing that we can treat it as a non-homogeneous Poisson process, using an uniform random variable \( U \), and solving the equation

\[
\int_{t_0}^{t} I_g(X(s)) \cdot f(X(s)) ds = -\log U. \tag{5}
\]

If we want to model condition (5) in TDSHA, we need to introduce a new continuous variable \( Z_\eta \), and let \( Z_\eta \) be governed by the differential equation

\[
\dot{Z}_\eta = I_g(X) \cdot f(X). \tag{6}
\]

Then, condition (5) becomes \( Z_\eta = -\log U \). If we want to fire the stochastic transition at its expected time, we can replace the condition \( Z_\eta = -\log U \) by \( Z_\eta = \mathbb{E}[-\log U] = 1 \), see [BP09a] for further mathematical details. This gives rise to an instantaneous transition, which takes implicitly into account time through the variable \( Z_\eta \).

The previous discussion is formalized in the following definition.
Definition 3.3. Let $\mathcal{T} = (Q, X, T\mathcal{C}, T\mathcal{D}, T\mathcal{E}, \text{init}, \mathcal{E})$ be a TDSHA, and let $\eta \in T\mathcal{E}$ be a stochastic transition. The determinization of $\mathcal{T}$ with respect to $\eta$ is the TDSHA $d_\eta(\mathcal{T}) = (Q_\eta, X_\eta, T\mathcal{C}_\eta, T\mathcal{D}_\eta, T\mathcal{E}_\eta, \text{init}_\eta, \mathcal{E}_\eta)$, where:

- $Q_\eta = Q$, $\mathcal{E}_\eta = \mathcal{E}$;
- $X_\eta = X \cup \{Z_\eta\}$;
- $T\mathcal{E}_\eta = T\mathcal{E} \setminus \{\eta\}$;
- $T\mathcal{C}_\eta = T\mathcal{C} \cup \{(\text{event}[\eta], e_1[\eta], 1 Z_\eta, I_{\text{guard}[\eta]} \cdot \text{rate}[\eta])\}$;
- $T\mathcal{D}_\eta = T\mathcal{D} \cup \{(\text{event}[\eta], e_1[\eta], e_2[\eta], \text{guard}[\eta] \land Z_\eta \geq 1, \text{reset}[\eta] \land Z_\eta' = 0, \text{rate}[\eta])\}$
- $\text{init}_\eta = \text{init} \land (Z_\eta = 0)$.

Basically, in the previous definition we just do a few simple things:

- We remove $\eta$ from the set of stochastic transitions.
- We add the variable $Z_\eta$ and we introduce the ODE given by equation (6) for the mode in which transition $\eta$ is active.
- We introduce an instantaneous transition replacing $\eta$, which fires whenever $Z_\eta$ equals one.

The determinization of a set $\{\eta_1, \ldots, \eta_k\}$ of stochastic transitions can be defined inductively as

$$d_{\eta_1, \ldots, \eta_k}(\mathcal{T}) = d_{\eta_k}(d_{\eta_1, \ldots, \eta_{k-1}}(\mathcal{T}))$$

Example 3.2. Consider the TDSHA of Example 3.1. Its determinization with respect to both two stochastic transitions is the TDSHA constructed from the initial way according to the following rules:

- There are two new variables, $Z_1$ and $Z_2$, which will govern the time of the binding and the unbinding.
- There are two new continuous transitions, $(q_{on}, (0, 1, 0), k_b P, \cdot)$ governing variable $Z_1$, and $(q_{off}, (0, 0, 1), k_u, \cdot)$ governing variable $Z_2$.
- Stochastic transitions are replaced by two instantaneous transitions: $(q_{on}, q_{off}, 1, Z_1 \geq 1, \text{true}, \cdot)$, and the unbinding $(q_{off}, q_{on}, 1, Z_2 \geq 1, \text{true}, \cdot)$.

5The vector $1_{Z_\eta}$ is equal to one in correspondence of variable $Z_\eta$, and it is zero elsewhere.
Figure 1: Simulation of the PDMP/TDSHA of Example 2.1 (Figure 1(a)), and its determinized version (Figure 1(b)). Parameters are such that the dynamics of binding/unbinding is slow compared to production/degradation. As evident in the two plots, the dynamics is similar, with the only difference that in the determinized model, stochastic variability is not present.

In Figure 1, we compare the trajectories of the stochastic TDSHA and of its determinization.

Studying the relationships between $T$ and $d_\eta(T)$ requires a precise mathematical formulation of the dynamics of TDSHA, hence a formal definition of the mapping from TDSHA to PDMP. However, at this stage we are still investigating the relationship between $T$ and $d_\eta(T)$. However, from an empirical point of view, it seems that the qualitative behaviour is preserved in passing from $T$ to $d_\eta(T)$, even in the case in which determinization is carried over all stochastic transitions, at least in case of simple genetic networks. Essentially, the qualitative aspects of the dynamics that are preserved are those connected to the intrinsic discreteness of a system, like the presence of expression peaks in the previous example, caused by the alternation between the active and the repressed states of the gene. Clearly, determinization will loose all features of the dynamics connected with the stochasticity, like peak frequency and duration in Figure 1(a). If noise plays a central role in determining the behaviour of a system, then determinization is not appropriate. However, it can be used to assess whether certain observed effects (like noise-sustained oscillations) are a consequence of noise or just of discreteness. A study in this direction, for a circadian clock model, can be found in [BP09d]. A deeper discussion on this point can be found in [BP10].

4 From TDSHA to PDMP

The mapping of TDSHA into PDMP is quite straightforward, with the exception of the definition of the reset kernel. We start the discussion by
providing the intuition of the mapping and explaining the problem with resets.

Suppose that \( T = (Q, X, \mathcal{E}, \mathfrak{Q}, \mathfrak{T}, \text{init}, \mathcal{E}) \) is a TDSHA, and let \( P = (Q, X, D, \lambda, R, q_0, x_0) \) be its target PDMP.

- Discrete modes and continuous variables are the same both in \( T \) and in \( P \).
- The state space of the PDMP, encoding the invariant region of continuous variables in each discrete mode, is defined from guards of instantaneous transitions in \( \mathfrak{T} \). The basic idea is that the invariant region is the set of points in which no instantaneous transition is active:

\[
D_q = \bigcap_{\delta \in \mathfrak{T} \mid e_1[\delta] = q} G_\delta^c.
\]

Hence, \( D_q \) is the intersection of the complements of the activation sets \( G_\delta \). Note that \( D_q \) is open, because each \( G_\delta \) is closed. This guarantees that, as soon as the guard of an instantaneous transition becomes true, the PDMP hits the boundary \( \partial D_q \) of the current mode \( q \) and is consequently forced to jump.

- The vector field in mode \( q \) is constructed from continuous transitions, by summing up their effects on system variables:

\[
X_q(x) = \sum_{\tau \in \mathcal{E} \mid \text{mode}[\tau] = q} \text{stoich}[\tau] \cdot \text{rate}[\tau](x). \tag{7}
\]

- The rate function \( \lambda \) is defined from the information contained in stochastic transitions of \( \mathfrak{T} \). More specifically, the rate function \( \lambda \) is defined by adding point-wise the rates of all active stochastic transitions:

\[
\lambda(q, x) = \sum_{\eta \in \mathfrak{T} \mid e_1[\eta] = q} \lambda(\eta, q, x), \tag{8}
\]

where \( \lambda(\eta, q, x) = I_{\text{guard}[\eta](x)} \text{rate}[\eta](x) \).

- The reset kernel \( R \) for \( x \in D_q \) is obtained by choosing the reset of each one of the active stochastic transitions in \( x \) with a probability proportional to its rate. If all such resets jump to points in the interior of \( D \) (i.e. to points in which no instantaneous transition is active), then we can define the reset in the following way:

\[
R((q, x), A) = \sum_{\eta \in \mathfrak{T} \mid e_1[\eta] = q} \frac{\lambda(\eta, q, x)}{\lambda(q, x)} d(e_2[\eta], r_\eta(x))(A). \tag{9}
\]
where $A \in D$, the Borel $\sigma$-algebra of $D$, \textbf{reset}[\eta]$ is of the form $X' = r_\eta(X) \in D e_2[\eta]$ and $\delta_{(q,x)}(A)$ is the Dirac measure on the point $(q,x) \in D$, assigning probability 1 to $(q,x)$ and 0 to the rest of the space. If there is a transition $\eta$ such that $(e_2[\eta], r_\eta(x)) \not\in D$, then this definition does not work any more. In fact, in PDMP we cannot reset the systems to points not in $D$.

• The reset kernel $R$ on the boundary $\partial D$ is defined from resets of instantaneous transitions. If more than one transition is active in a point $x \in \partial D_q$, we choose one with probability proportional to the value of the priority function in $x$. Also in this case, we must take care of situations in which an instantaneous transition resets to a point not in $D$.

• The initial point is $(q_0,x_0) = \text{init}$.

Summarizing, the problem with the definition of the reset kernel is that on the one hand, PDMP definition forces the resets to bring the system inside the allowed region $D_q$ of the target mode, on the other hand, transitions in TDSHA are introduced independently of one another. Hence the previous condition cannot, in general, be guaranteed. It may happen, in fact, that in the target state of a transition (either instantaneous or stochastic) other instantaneous transitions are active. Such inconvenience can be dealt with by constructing the reset kernel in such a way that it incorporates the effect of all possible sequences of instantaneous transitions firing at the same time instant.

However, the above solution is viable only if in the TDSHA $T$ we are not required to fire infinite sequences of instantaneous transitions simultaneously. This is crucial, otherwise we may enter into an infinite loop of instantaneous transitions which does not allow the system to evolve. We will call terminating a TDSHA satisfying such property. In this direction we need to tackle two problems:

1. Define the notion of “terminating TDSHA”

2. Given a terminating TDSHA, construct the correct reset kernel.

Subsequently, the issue of discovering whether a TDSHA is terminating or not, should be also addressed.

We will start by considering the first problem, then defining the reset kernel.
Suppose to be in a point $(q,x)$, $x \not\in D_q$, where $x$ is not necessarily a point of the border of $D_q$. Define the set

$$\mathcal{T}(q,x) = \{ \delta \in \mathcal{T} | \ e_1[\delta] = q \land x \in G_\delta \}.$$
Clearly, $x \notin D_q$ implies $\exists \Delta(q, x) \neq \emptyset$.

We will define now the set $\exists(q, x)$ of executable sequences of instantaneous transitions. Given a discrete transition $\delta \in \exists(q, x)$, we indicate with $r_\delta(x)$ the point to which $x$ is mapped by $\text{reset}[\delta]$. Moreover, let $\cdot$ denote the concatenation of sequences, defined for sequences and set of sequences in the standard way.

**Definition 4.1.** The set $\exists(q, x)$ of executable sequences of instantaneous transitions from the point $(q, x)$ is defined by

$$\exists(q, x) = \bigcup_{\delta \in \exists(q, x)} \Delta \cdot \exists(e_2[\delta], r_\delta(x)).$$

(10)

Note that this recursive definition may generate infinite sequences of transitions, as the following example shows:

**Example 4.1.** Consider a TDSHA with one mode $q$ and one variable $X$ and two instantaneous transitions, $\exists(q, x) = \{\delta_1, \delta_2\}$, with $G_{\delta_1} = [1, 2]$, $G_{\delta_2} = [2, \infty[$, $r(\delta_1, x) = x + 1$, and $r(\delta_2, x) = x - 1$. Then $\exists(q, 1)$ contains the single sequence of infinite length $\delta_1 \delta_2 \delta_1 \delta_2 \ldots$

As stated above, we consider such behaviours as pathological and we will rule them out by definition. In the following, let $\exists = \bigcup_{(q, x) \in D} \exists(q, x)$.

**Definition 4.2.** The set $\exists(q, x)$ of instantaneous transitions is *terminating* if the following two conditions hold:

$$|\exists| < \infty$$

$$\forall \alpha \in \exists, |\alpha| < \infty.$$

The definition above is the constraint we will impose on $\exists(q, x)$ for the mapping to PDMP to work.

If $|\exists| < \infty$, then obviously $|\exists(q, x)| < \infty$, hence we can define a probability measure $p$ on it as follows:

1. for $\delta \in \exists(q, x)$, $p(\delta) = \frac{\text{priority}[\delta]}{\sum_{\delta \in \exists(q, x)} \text{priority}[\delta]}$, where $\text{priority}[\exists(q, x)] = \sum_{\delta \in \exists(q, x)} \text{priority}[\delta]$;

2. for $\alpha = \delta \cdot \alpha'$, $p(\alpha) = p(\delta)p(\alpha').$

**Proposition 4.1.** $p$ is a probability measure on $\exists(q, x)$.

**Proof.** Clearly, $p(\alpha) \geq 0$, hence we just need to prove that $\sum_{\alpha \in \exists(q, x)} p(\alpha) = 1$. We can proceed by induction on the length of the longest sequence in $\exists(q, x)$. The base case of induction regards the situation in which the longest sequence has length one. In such case, the thesis follows from the fact that $\sum_{\delta \in \exists(q, x)} p(\delta) = 1$. For the inductive step, observe that by condition (10),
Hence, kernel \( R \) reached by \( e \) set of instantaneous transitions \( 6 \) and suppose \( \eta \) let \( \) region, we immediately apply a sequence of discrete transitions. Formally, the hybrid state space. The idea is simple: if we jump out of the allowed

Therefore, each possible sequence \( \alpha \) of discrete transitions is chosen according to its probability \( p(\alpha) \).

We still need to deal with jumps of stochastic transitions leading out of the hybrid state space. The idea is simple: if we jump out of the allowed region, we immediately apply a sequence of discrete transitions. Formally, let \( \eta \in \Xi(q, x) \), where \( \Xi(q, x) = \{ \eta \in \Xi | e_1[\eta] = q \wedge \text{guard}[\eta](x) \} \), and suppose \( r(\eta, x) \notin D \). Let \( e_2[\eta] = q_0 \) and \( r(\eta, x) = x_\eta \), and consider \( \Xi(q, x) \neq \emptyset \) (as \( x_\eta \notin D_{q_0} \)). We can now simply modify the tentative reset kernel \( R((q, x), A) \) of equation (9) by replacing \( \delta_q, x_\eta(A) \) with \( R((q, x), A) \). Hence,

\[
R((q, x), A) = \sum_{\eta \in \Xi(q, x)} p(\alpha) \delta_{e_2[\eta]}(\bar{r}(\alpha, x))(A).
\]

This may raise a problem of representability of such a kernel. In fact, if the sets \( \Xi(q, x) \) are different in each point, then we may need to compute an infinite number of sequences of instantaneous transitions in order to write down explicitly the definition of \( R \), which is clearly infeasible. The condition of terminating instantaneous transitions prevents this, assuming

\[\text{Remark 4.1. The reset kernel is defined in each point as depending on } \Xi(q, x). \text{ This may raise a problem of representability of such a kernel. In fact, if the sets } \Xi(q, x) \text{ are different in each point, then we may need to compute an infinite number of sequences of instantaneous transitions in order to write down explicitly the definition of } R, \text{ which is clearly infeasible. The condition of terminating instantaneous transitions prevents this, assuming} \]

\[\text{Note that we are defining } R \text{ not just in } \partial D_{q_0} \text{ but on the entire } D_{q_0}.\]
a finite global number of possible different sequences. As we will see below, this guarantees the possibility of partitioning the state space in a finite number of sets, such that the possible sequences are the same in each point of the set.

**Remark 4.2.** In the previous discussion we implicitly assumed that the PDMP derived at the end of this process satisfies the conditions (1)-(4) of PDMP. To be completely precise we can require that the rate and reset functions of TDSHA are sufficiently smooth, which is generally the case. The most critical condition is condition (4), which can be violated by subtle interactions among instantaneous transitions.

If a TDSHA does not have instantaneous transitions, a sufficient condition for (4) to hold is that rate functions are bounded [Dav93]. Standard rate functions used in systems biology, however, are not bounded in $\mathbb{R}^n$ (think of a mass action rate). One way to enforce such condition is letting the system evolve in a bounded domain, sufficiently large that points outside it have no biological or physical significance. Technically, we can do this by introducing suitable instantaneous transitions in the TDSHA. First, we need to introduce a new mode $\Delta \in Q$ in which the rate and the vector field are identically zero. Then the new instantaneous transitions will make the system jump from the boundary of the bounded domain into this new state $\Delta$, with identity reset. Stated otherwise, we stop all trajectories leaving the allowed domain. Note that such trick is standard in PDMP [Dav93], although generally used to stop the evolution at a finite time horizon (in this setting $\Delta$ is known as cemetery point).

For a TDSHA generated in the context of sCCP programs, we will show that boundedness of continuous domains is again a sufficient condition for (4) to hold.

All this discussion can be condensed in the following

**Definition 4.3.** Let $\mathcal{T} = (Q, X, T\mathcal{C}, T\mathcal{D}, T\mathcal{E}, \text{init, } \mathcal{E})$ be a TDSHA with terminating instantaneous transitions. The PDMP $\mathcal{P}_\mathcal{T} = (Q, X, D, \lambda, R, q_0, x_0)$ associated with $\mathcal{T}$ is defined according to the following conditions:

1. The vector field $X_q$ is defined according to equation (7).
2. The rate function $\lambda(q, x)$, for $x \in D_q$, is defined by equation (8).
3. The reset kernel for a point $x \in D_q$ is defined according to equation (12), while for a point $x \in \partial D_q$, it is given by equation (11).
4. $(q_0, x_0) = \text{init}$.

We turn now to study the general problem of checking whether a set of instantaneous transitions is terminating or not. Let $\mathcal{H} = Q \times \mathbb{R}^n$, and consider the instantaneous transitions $T\mathcal{D}$ of a TDSHA $\mathcal{T}$ with $n$ variables.
Checking if $\mathfrak{T}$ is terminating means to find out whether, starting from a generic point $(q, x)$ and applying instantaneous transitions, we always reach a quiescent point after a finite amount of steps, i.e., a point in which no transition in $\mathfrak{T}$ is active. Moreover, we need to show that the set of different executable sequences of transitions in $\mathfrak{T}$ is finite. Unfortunately, such a test is undecidable.

**Proposition 4.2.** Checking if a set of instantaneous transitions $\mathfrak{T}$ of a TDSHA $\mathcal{T}$ is terminating is undecidable.

**Proof.** The proof proceeds by encoding Unlimited Register Machines, URM [Cut80], a Turing-complete formalism. Registers are described by variables and in each program we only use a finite number of registers. Discrete modes correspond to the different instructions of a program, which are also finite. Finally, the different instructions of a URM can be encoded by transitions in $\mathfrak{T}$ in the following way:

- Reset of register $m$ to zero: $(\text{res}_m, q_i, q_{i+1}, \text{true}, x'_m = 0, 1)$;
- Increment of register $m$: $(\text{inc}_m, q_i, q_{i+1}, \text{true}, x'_m = x_m + 1, 1)$;
- Copy of register $m_1$ to register $m_2$: $(\text{copy}_{m_1, m_2}, q_i, q_{i+1}, \text{true}, x'_{m_2} = x_{m_1}, 1)$;
- Conditional jump to instruction $j$ if registers $m_1$ and $m_2$ are equal: $(\text{jump}_{m_1, m_2, j}, q_i, q_{j}, x_{m_1} = x_{m_2}, \top, 1)$ and $(\text{nojump}_{m_1, m_2, j}, q_i, q_{i+1}, x_{m_1} \neq x_{m_2}, \top, 1)$.

Quiescent states, in this setting, are terminal states of the URM. If we could check whether a set $\mathfrak{T}$ constructed from an URM is terminating or not, we would be able to prove whether computations of our URM halt or not (for each initial condition). ■

Actually, the termination problem for TDSHA is semi-decidable. In fact, it is equivalent to the existence of a finite quotient by bisimulation, which itself must be acyclic. First, we recall now some notions that we will use in the following.

A *transition system* is a tuple $T = (X, \Sigma, \rightarrow)$, where $X$ is the state space, $\Sigma$ is a finite alphabet, and $\rightarrow \subseteq X \times \Sigma \times X$ is the transition relation. Let $R \subseteq X$ and $\text{Pre}_\sigma(R) = \{ x \in X \mid \exists y \in R, x \rightarrow_\sigma y \}$ be the set of predecessors of a region $R$ by $\rightarrow_\sigma$ transitions, $\sigma \in \Sigma$. An equivalence relation $\sim \subseteq X \times X$ is a bisimulation if and only if the predecessor of each equivalence class is the union of equivalence classes. Alternatively, a bisimulation can be defined in the usual way: $x \sim y$ if and only if

1. if $x \rightarrow_\sigma x'$, then there exists $y'$ such that $y \rightarrow_\sigma y'$ and $x' \sim y'$;

2. if $y \sim y'$, then there exists $x'$ such that $x \rightarrow_\sigma x'$ and $x' \sim y'$.
2. if \( y \rightarrow_{\sigma} y' \), then there exists \( x' \) such that \( x \rightarrow_{\sigma} x' \) and \( x' \sim y' \).

A transition system \((X, \Sigma, \rightarrow)\) can be easily constructed from \( \mathcal{H} \) and \( \mathcal{T}\Omega \):

- \( X = \mathcal{H} \)
- \( \Sigma = \{ \eta \in \mathcal{T}\Omega \} \)
- \( \rightarrow = \{(q_1, x), \eta, (q_2, y) \mid \eta = (q_1, q_2, p_\eta, G_\eta, r_\eta, \alpha), x \in G_\eta, y = r_\eta(x)\} \)

**Theorem 4.1.** The set \( \mathcal{T}\Omega \) of instantaneous transitions is terminating if and only if the following two conditions hold:

1. There exists a finite bisimulation \( \sim \) for the associated transition system \((X, \Sigma, \rightarrow)\)
2. The graph of the quotient transition system \((X/ \sim, \Sigma, \rightarrow / \sim)\) is an acyclic graph.

**Proof.** One direction of the implication is obvious: if the two conditions hold, then the number of different sequences of instantaneous transitions corresponds to the number of different paths in the graph, hence it is finite, as the graph is acyclic. For the other direction, by assuming that \( \mathcal{T} \) is finite, we can construct a finite bisimulation relation in the following way: let \( y_1, y_2 \in \mathcal{H}, y_1 \sim y_2 \) if and only if \( \mathcal{T}(y_1) = \mathcal{T}(y_2) \). Clearly \( \sim \) is a finite equivalence relation, as its possible classes are no more than the subsets of \( \mathcal{T} \). It is easy to check that \( \sim \) is also a bisimulation, as two points related by \( \sim \) can take exactly the same transitions and necessarily reach two bisimilar points. Finally, the graph of the bisimulation needs to be acyclic otherwise we would obtain infinite sequences of instantaneous transitions. \( \blacksquare \)

The previous theorem tells us that checking if a set of instantaneous transitions is terminating is the same as computing a quotient by bisimulation and checking the acyclicity of the corresponding graph. As finding out if a transition system has a finite quotient by bisimulation is semi-decidable, so is checking termination.

**Remark 4.3.** On the ground of the results presented in the final part of this section, one could argue that it could have been better to enforce decidability of termination on TDSHAs \( a-priori \) by definition. The reason why we decided not to proceed in such a way is twofold. On the one hand, a general definition simplifies the definition of the hybrid semantics of \textit{sCCP}. This is, ultimately, a consequence of the fact that the discrete control that can be included in an \textit{sCCP} program finds a natural counterpart in the discrete component of the hybrid system and, therefore, should not constrained in any way. On the other hand, the ability to “(discretely) compute on boundaries” is the specific feature of hybrid systems, marking the difference with
the PDMP formalism by enhancing their expressing power. Moreover, for TDSHA obtained from sCCP programs, we will be able to prove that instantaneous transitions are indeed terminating, hence we do not have to check such property with algorithmic means (cf. Theorems 6.1, 4.1, and 8.1 below).

5 Stochastic Concurrent Constraint Programming

In this section we present a simplified version of stochastic Concurrent Constraint Programming (sCCP [Bor06]), a stochastic extension of CCP [Sar93], together with its CTMC and ODE-based semantics [BP09a].

A (simplified) sCCP program consists of a set of agents interacting via a shared store, containing a finite set of variables $X = \{X_1, \ldots, X_n\}$, taking values in a countable domain $\text{Dom}(X_i)$, usually a subset of the real numbers —although more structured numerical domains are possible. A configuration $c$ of the store is simply a valuation of the variables $X$. The basic action $\pi$ executable by agents is a guarded update of some variables, of the form $g(X) \rightarrow u(X, X')$, where $g(X)$ is a quantifier-free first order formula, with atomic formulae given by inequality predicates on variables $X$, and $u(X, X')$ is a conjunction of predicates on $X, X'$ of the form $X' = f(X)$ ($X'$ denotes variables of $X$ after the update), for some function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$. If $f(X) = X + k$, with $k$ a vector of constants, we talk of constant updates. In addition, the language has all the basic constructs of process algebras: non-deterministic choice, parallel composition, and recursive calls.

The characteristic feature of sCCP is the fact that each action $\pi$ is given a stochastic duration by associating to it an exponentially distributed random variable, whose rate depends on the state of the system through a function $\lambda : \mathbb{R}^n \rightarrow \mathbb{R}^+$. Stochastic actions are denoted by $[\pi]$.

**Definition 5.1.** A sCCP program is a tuple $A = (A, D, X, \text{init}(X))$, where

1. The initial agent $A$ and the set of definitions $D$ are defined according to the following grammar:

   \[
   D = \emptyset \mid D \cup D \mid \{C \overset{\text{def}}{=} M\} \\
   \pi = [g(X) \rightarrow u(X, X')]_{\lambda(X)} \\
   M = \pi.C \mid M + M \\
   A = C \mid A \parallel A
   \]

2. $X$ is the set of variables of the store (with global scope);

---

7The notion of entailment of CCP (see [Sar93]) takes here a very simple form: a guard $g(X)$ is entailed by a configuration $c$, $c \vdash g(X)$, if $g(c)$ is true.
3. \( \text{init}(X) \) is a predicate on \( X \) of the form \( X = x_0 \), assigning an initial value to store variables.

All agents definable in \( \text{sCCP} \) are of the form \( C \overset{\text{def}}{=} M \), where \( C \) is the agent’s name (used for recursion) and \( M \) is a summation, i.e. a choice among different actions. Note that all agents in \( \mathcal{D} \),\(^8\) are sequential, i.e. they do not contain any occurrence of the parallel operator, whose usage is restricted at the upper level of the network. Indeed, a \( \text{sCCP} \) program is the parallel composition of sequential agents. Moreover, all variables are global, hence no parameter is passed in recursive calls. These restrictions do not reduce the expressive power of the language, see Remark 5.3.

Remark 5.1. \( \text{sCCP} \) has been used as a modelling language for biological systems in [BP08b], where models of biochemical and genetic networks have been addressed. The modelling style is predominantly reaction-centric: agents represent reactions and store variables maintain the numerousness of the different reactants. However, an entity-centric approach can be adopted when the modeller wants to distinguishably model some entities as agents. The typical example is an explicit model of genes, suggested by the fact that they are present in a single copy in a cell and they have more than one internal state (depending on which transcription factors are bounded). Hence, genes are usually described in \( \text{sCCP} \) as multi state agents, see Example 5.1 below. For a more detailed discussion on modelling styles, we refer the reader to [BP08b].

For biochemical and genetic networks, the simplified language presented here suffices. However, more advanced reasoning capabilities of the constraint store can be exploited to model more complex situations, like those arising due to combinatorial explosion in formation of protein complexes or those involving spatial features. We tackled these problems with \( \text{sCCP} \) in [BFP07] [BP09f].

Example 5.1. We will illustrate the concepts introduced from now on by means of an example coming from biological systems. Specifically, we consider a simple model of a (procaryotic) genetic regulatory network with a single gene, expressing a protein acting, after dimerisation, as a repressor of its own production. We assume a cooperative repression: two dimers are required to bind to the promoter region of the gene. The \( \text{sCCP} \) model is given by \( \mathcal{A} = (A, \mathcal{D}, X, \text{init}) \), where the variables are \( X = \{X_p, X_{p2}\} \), storing the quantity of the protein \( p \) and of its dimer \( p2 \) and the components in \( \mathcal{D} \) are (* stands for true):

\[ \text{In the following, with a slight abuse of notation, we sometimes write } C \in \mathcal{D} \text{ for } C \overset{\text{def}}{=} M \in \mathcal{D}. \]
The initial network $A$ is $\text{gene}_0 \parallel \text{deg} \parallel \text{dimer}$ with initial values of the store variables are given by

$$\text{init}(X_p, X_{p2}) = (X_p = 0) \land (X_{p2} = 0).$$

Notice: there is no need to introduce agents for proteins or dimers, as the quantity of these objects needs only to be measured by stream variables. The repression mechanism is represented by a gene unable of expressing a protein whenever in state $\text{gene}_2$. We did not decrement $X_{p2}$ before entering states $\text{gene}_1$ and $\text{gene}_2$. This can be seen as an abstract description of the repression mechanism.\(^9\)

Remark 5.2. The pros and cons of using sCCP as a modelling language for biological systems are discussed in detail in [BP08b]. Basically, sCCP combines on one side the logical simplicity of process algebras and on the other side the computational power of constraints. As a matter of fact, the constraint store can be more general than that used in this paper, whereby more complex information (like spatiality) can be managed just by a simple programming activity. Further work is needed, however, to export the techniques developed here to a more general version of the store.

Operational semantics The operational semantics of sCCP is given by a transition relation that generates a labelled transition system (LTS), from which a Continuous Time Markov Chain (CTMC [Nor97]) can be inferred, see [Bor06], [BP08b] for further details. We sketch briefly the mechanism:

- A congruence relation $\equiv$ is defined on the set of all possible agents, making it a monoid with respect to stochastic summation $+$ and also a monoid with respect to parallel composition $\parallel$. The set of agents modulo $\equiv$ is indicated by $\mathfrak{A}$.

- A configuration of the system is a pair $\langle A, x \rangle \in \mathfrak{A} \times \mathbb{R}^n$, with $A$ a parallel agent (modulo $\equiv$) and $x \in \mathbb{R}^n$ a valuation of the system variables.

\(^9\)We abstracted repression following [BCP06]. A model in which binding and unbinding is treated explicitly can be easily defined, however it does not add relevant features to the example.
\[ \begin{align*}
\text{((R1))} &\quad \frac{\mathbf{x} \vdash g(\mathbf{x}); \quad \mathbf{x}, \mathbf{x}' \vdash u(\mathbf{x}, \mathbf{x}'); \quad \lambda(\mathbf{x}) > 0}{\langle [g(\mathbf{X}) \rightarrow u(\mathbf{X}, \mathbf{X}')] \lambda(\mathbf{X}), C + M, \mathbf{x} \rangle \xrightarrow{\lambda(\mathbf{x})} \langle C, \mathbf{x}' \rangle} \\
\text{((R2))} &\quad \frac{\langle M, \mathbf{x} \rangle \xrightarrow{\lambda} \langle C', \mathbf{x}' \rangle, C \overset{\text{def}}{=} M}{\langle C, \mathbf{x} \rangle \xrightarrow{\lambda} \langle C', \mathbf{x}' \rangle} \\
\text{((R3))} &\quad \frac{\langle A_1, \mathbf{x} \rangle \xrightarrow{\lambda} \langle A_1', \mathbf{x}' \rangle}{\langle A_1 \parallel A_2, \mathbf{x} \rangle \xrightarrow{\lambda} \langle A_1', \mathbf{x}' \rangle} \\
\end{align*} \]

Table 1: Structural Operational Semantics for the restricted version of sCCP presented here. The first rule deals with basic actions, the second rule with the application of agents’ definitions, and the third with parallel composition. The number attached as label in the transition relation represents the rate of the transition. Note also how basic actions are active only if several conditions are satisfied simultaneously: the guard of the transition must be true in the current state of the store, the rate of the transition must be strictly positive, and the constraint store must be updated consistently.

- The transition relation \( \xrightarrow{\subseteq} \subseteq (\mathcal{A} \times \mathbb{R}^n) \times \mathbb{R}^+ \times (\mathcal{A} \times \mathbb{R}^n) \), defining the LTS, is given by Table 1 and commented in the caption therein. The label of the relation records the actual rate of the transition.

- From the (reachable portion of the) LTS we obtain a Continuous Time Markov Chain (CTMC), described through the infinitesimal generator matrix (cf. [Nor97]), in the following way: the rate of going from state \( s_1 \) to state \( s_2 \) is obtained by summing the rates of all edges of LTS connecting \( s_1 \) to \( s_2 \).

Reduced Transition Systems sCCP sequential agents can be seen as automata synchronizing on store variables and they can be conveniently represented as labelled graphs, called in [BP09a] Reduced Transition Systems (RTS). Before giving a formal definition, we need few additional concepts:

**Definition 5.2.** Let \( \mathcal{A} = (A, \mathcal{D}, \mathbf{X}, \text{init}(\mathbf{X})) \) be a sCCP program and let \( C \in \mathcal{D}, C \overset{\text{def}}{=} M = \sum_{i=1}^{k} \pi_i . C_i \), be a sequential agent.

1. The derivative set \( \text{Der}(C) \) of \( C \) is defined recursively as \( \text{Der}(C) = \{ C \} \cup \bigcup_i \text{Der}(C_i) \).

2. The action set \( \text{action}(C) \) of \( C \) is \( \text{action}(C) = \{ \pi_1, \ldots, \pi_k \} \). The definition can be extended to sets of agents \( S = \{ C_1, \ldots, C_n \} \) by letting \( \text{action}(S) = \bigcup_{i=1}^{n} \text{action}(C_i) \).

In our running example 5.1, we have, for instance, \( \text{Der}(\text{gene}_0) = \{ \text{gene}_0, \text{gene}_1, \text{gene}_2 \} \) and \( \text{action}(\text{dinner}) = \{ \pi_1 = [ \ast \rightarrow X_p' = X_p - 2 \land X_p' = X_p + 1] \_k . X_p(X_p - 1)/2, \pi_2 = [ \ast \rightarrow X_p' = X_p + 2 \land X_p' = X_p - 1] \_k . X_p(2) \}. \)
In the previous definition, we consider each action as a distinct object, hence two sets \( \text{action}(C) \) and \( \text{action}(C') \) are always disjoint, for \( C \neq C' \). If \( C \overset{\text{def}}{=} \pi.C' + M, \) then \( \text{exit}[\pi] = C \) and \( \text{enter}[\pi] = C' \), while \( \text{guard}[\pi], \text{update}[\pi] \), and \( \text{rate}[\pi] \) denote respectively the guard, update and rate function of the action \( \pi \). A special class of \text{sCCP} agents is the following:

**Definition 5.3.** Let \( \mathcal{A} = (A, \mathcal{D}, X, \text{init}(X)) \) be a \text{sCCP} program. \( \mathcal{A} \) is \textit{simple} if and only if for all agents \( C_i \) and \( C_j \) in parallel in \( A \), \( \text{Der}(C_i) \cap \text{Der}(C_j) = \emptyset \) (i.e., there are no two copies of the same agent in \( A \)).

**Remark 5.3.** The syntactic restrictions introduced for \text{sCCP} seem rather strong. Indeed, the limitations on the use of parallel operator and even the requirement of being simple are only apparent restrictions. In fact, every \text{sCCP} program with agents containing parallel operators and possibly present in several copies can be transformed in an equivalent simple \text{sCCP} program, where “equivalent” means that the resulting CTMC is isomorphic. The idea is to use additional store variables to count the number of copies of different agents in parallel in the system. Something similar is done in the following to give the definition of the ODE-based semantics.

We can now define the Reduced Transition System:

**Definition 5.4.** Let \( \mathcal{A} = (A, \mathcal{D}, X, \text{init}(X)) \) be a \text{sCCP} program. The \textit{reduced transition system} for an agent \( C \in \mathcal{D} \) is the labelled multi-graph \( \text{RTS}(C) = (S(C), E(C), \ell) \) defined by:

1. \( S(C) = \text{Der}(C) \);
2. \( E(C) = \{(\text{exit}[\pi], \text{enter}[\pi]) \mid \pi \in \text{action}(C)\} \);
3. \( \ell(e) = (\text{guard}[\pi], \text{update}[\pi], \text{rate}[\pi]), \) where \( \pi \) is the action defining the edge \( e \in E(C) \).

With slight abuse of notation, in the following we write \( \text{rate}[e], e \in E(C) \), to denote the rate function of the action \( \pi \) defining \( e \). The same caveat applies to \( \text{guard} \) and \( \text{update} \).

In Figure 2, we show the RTS for the three agents gene\(_0\), deg, and dimer defined in Example 5.1.

**Extended \text{sCCP} programs.** Before going any further, we need a technical trick that simplifies the following presentation. Essentially, we want to introduce one new variable \( P_C \) for each component \( C \in \mathcal{D} \) of a \text{sCCP} program. These variables will count the number of copies of \( C \) present in parallel in the system at a certain time. To take into account the effects of transitions on agents, we modify updates and rate functions.

**Definition 5.5.** Let \( \mathcal{A} = (A, \mathcal{D}, X, \text{init}(X)) \) be a \text{sCCP} program. The \textit{extension} \( \mathcal{A}^+ = (A^+, \mathcal{D}^+, Y, \text{init}^+(Y)) \) of \( \mathcal{A} \) is defined by:
Figure 2: Reduced Transition Systems for the three agents gene₀, deg, and dimer defined in Example 5.1. Each edge is labelled by its rate function and by the guard and the update (depicted in the guard → update notation).

1. \( Y = X \cup P \), where \( P = \{ P_C \mid C \in D \} \).

2. Each \( C \overset{D}{=} M \in D \), with \( M = \sum_{i=1}^{k} \pi_i, C_i \), is replaced by \( C^+ \overset{D}{=} M^+ \), \( M^+ = \sum_{i=1}^{k} \pi^+_i, C^+_i \).

3. If \( \pi_i = [g_i(X) \rightarrow u_i(X, X')]_{\lambda_i(X)} \), then \( \pi^+_i = [g_i(X) \rightarrow u_i(Y, Y')^+\lambda^+_i(Y)] \), where
   - \( u_i(Y, Y')^+ = u_i(X, X') \land P'^+_C = P_C - 1 \land P'^+_C_i = P_{C_i} + 1 \) if \( C \neq C_i \),
   - \( u_i(Y, Y')^+ = u_i(X, X') \) otherwise;
   - \( \lambda^+_i(Y) = P_C \cdot \lambda_i(X) \)

4. If \( A = C_1 \parallel \ldots \parallel C_n \), then \( A^+ = C^+_1 \parallel \ldots \parallel C^+_n \)

5. \( \text{init}^+(Y) = \text{init}(X) \land \bigwedge_{C \in D} (P_C = \#(C, A)) \), with \( \#(C, A) \) denoting the number of copies of \( C \) in \( A \).

Taking the extension of a simple \( s\text{CCP} \) program does not modify the CTMC associated to it by the operational semantics.

**Proposition 5.1.** Let \( A = (A, D, X, \text{init}(X)) \) be a simple \( s\text{CCP} \) program and \( A^+ = (A^+, D^+, Y, \text{init}^+(Y)) \) be its extension. The LTS of \( A \) and \( A^+ \) are isomorphic as labelled multi-graphs.

**Proof.** Consider an \( A^+ \)-configuration \( \langle A^+, (x, p) \rangle \in \mathfrak{A} \times \mathbb{R}^{n+m} \). We call it well-defined if and only if \( p_C = \#(C, A^+) \) for \( C \in D^+ \). We define now the
two mappings involved in the isomorphism. Let \( \psi((A, x)) = (A^+, (x, p)) \), with \( p_C = \#(C, A^+) \) for \( C \in \mathcal{D}^+ \), and \( \psi^{-1}((A^+, (x, p))) = (A, x) \). It is easy to see that \( \psi \) and \( \psi^{-1} \) form a bijection between \( A \)-states and well-defined \( A^+ \)-states.

We show now that action \( \pi \) is executable in state \( (A, x) \) with rate \( \lambda \) if and only if action \( \pi^+ \) is executable in state \( \psi((A, x)) \) with the same rate \( \lambda \). Let \( \pi \) be an action executable in \( (A, x) \), say \( \pi \in \text{action}(C) \), then \( C \) must be one if the agents in parallel in \( A \), hence \( p_C = \#(C, A) = 1 \), as \( A \) is simple. Now, if preconditions of guards and updates of the rule of Table 1 are satisfied by \( \pi \), then so are for \( \pi^+ \) (as straightforwardly checked), and \( \text{rate}[\pi^+](x, p) = p_C \cdot \text{rate}[\pi](x) = \text{rate}[\pi](x) \). The reverse direction from \( \pi^+ \) to \( \pi \) follows analogously. \( \blacksquare \)

This means that for simple sCCP programs \( A \) we can work indistinguishably with \( A \) or with their extension \( A^+ \). This property is no more true for non-simple agents, as \( P_C \)-variables can take values greater than one, hence \( \text{rate}[\pi] \) is generally different from \( \text{rate}[\pi^+] \). However, we can obtain an extended model strongly bisimilar to \( A \) essentially by removing duplicates of the same processes.

**ODE-based semantics** In [BP09a] we defined a fluid-flow approximation of sCCP. The method works by approximating as continuous the system’s variables \( X \) and considering the effect of each transition on those variables. Two additional restrictions on sCCP programs are required: all updates must be of the constant type and the program must be simple (cf. Definition 5.3). The constraints on the constant updates introduce a restriction in the expressive power of the language, yet they allow us to interpret sCCP-actions as continuous fluxes. Hence, these assumptions will be considered in force throughout the rest of the paper.

The method proceeds in the following way:

1. Start from a simple sCCP program \( A = (A, \mathcal{D}, X, \text{init}(X)) \) and construct its extension \( A^+ = (A^+, \mathcal{D}^+, Y, \text{init}^+(Y)) \).

2. For each \( \text{RTS}(C) \) and each edge \( e \in E(C) \), define the stoichiometric vector \( \nu_e \) as a vector with elements indexed by \( Y = X \cup P \). The entry \( \nu_e[X] \), for a variable \( X \in Y \), contains the net variation on \( X \) induced by \( \text{update}[e] \) (for instance, if \( X' = X + 2 \) is the (only) conjunct of \( \text{update}(e) \) on \( X \), then \( \nu_e[X] = 2 \)). Note that for variables \( P_C \in P \) \( \nu_e[P_C] = 0 \) for all \( C \), unless \( e = (C', C'') \), \( C' \neq C'' \), in which case \( \nu_e[P_{C''}] = -1 \) and \( \nu_e[P_{C''}] = 1 \).

In addition, we consider also the rate of edge \( e \in E(C) \) \( \phi_e \), defined by \( \phi_e = \text{rate}[e] \cdot I(\text{guard}[e]) = \text{rate}[e] \cdot P_{C(e)} \cdot I(\text{guard}[e]), \) where \( I(\cdot) \) is the indicator function (any function turning logical values into 0/1) and \( P_{C(e)} \) is the variable associated to the exit state of \( e \).
3. The differential equations for agent $C$ are $\dot{Y} = \Phi_C(Y)$, with $\Phi_C = \sum_{e \in E(C)} \nu_e \phi_e$. If $A = C_1 \parallel \ldots \parallel C_n$, then the differential equations for $A$ are $\dot{Y} = \Phi_A(Y) = \Phi_{C_1}(Y) + \ldots + \Phi_{C_n}(Y)$.

Going back to Example 5.1, we obtain, for instance, the following differential equation for the dimer protein $X_{p2}$:

$$\dot{X}_{p2} = P_{dimer} \left( k_2 \frac{1}{2} X_p (X_p - 1) - k_{-2} X_{p2} \right),$$

where $P_{dimer}$ is the variable associated to component dimer, which can be ignored as it is constantly equal to 1.

6 From sCCP to TDSHA

In this section we define a semantics for sCCP in terms of TDSHA. The basic idea is to partition all possible transitions executable by an sCCP agent into two classes: those remaining discrete and those to be approximated as continuous. Discrete actions are further divided into those that remain stochastic and those that are replaced by their determinization. Different partitions’ schemata correspond to different TDSHA. By parametrising upon such schemata, we will obtain a lattice of different TDSHA models.

Note that this approach generalizes both [BP09e] and [BP09c], as we allow stochastic and non-stochastic discrete transitions within the same framework. Indeed, this semantics captures all possible intermediate models ranging from stochastic programs to their fluid-flow approximation and passing through stochastic and non-stochastic hybrid automata.

First of all, we define a mapping from a sCCP program to a TDSHA where all discrete transitions are stochastic, parametrising only on the degree of continuity. No instantaneous transitions will be introduced at this stage. Then, we will generalize such mapping by allowing also non-stochastic discrete transitions. This second step will be discussed in Section 6.1.

Finally, in Section 6.2, we will focus on the issue of choosing a specific degree of continuity, i.e. we discuss possible heuristics to partition transitions.

The mapping from sCCP to TDSHA with just stochastic discrete transitions proceeds in two steps. First we convert into TSHAs each sequential component of a sCCP program, then all these TSHAs are combined using the product construction.

Given a sCCP program $A = (A, D, X, \text{init}(X))$, first of all we construct its extended version $A^+ = (A^+, D^+, Y, \text{init}^+(Y))$, see Definition 5.5. Let now $C \in D^+$ be one of the components of the initial network $A$, and let $RTS(C) = (S(C), E(C), \ell)$ be its RTS.
A specific continuous/discrete scheme of approximation is formalized by the choice of a boolean vector \( \kappa \in \{0, 1\}^m, m = |E(C)| \), indexed by edges in \( E(C) \): for \( e \in E(C) \), \( \kappa[e] = 1 \) stands for a continuous approximation of the transition, while \( \kappa[e] = 0 \) implies that the transition will remain discrete.

Let \( E(\kappa, C) = \{ e \in E(C) | \kappa[e] = 1 \} \) and \( E(\neg \kappa, C) = \{ e \in E(C) | \kappa[e] = 0 \} \).

In order to guarantee that the vector field constructed from continuous transitions is sufficiently regular, we introduce the notion of continuously approximable action.

**Definition 6.1.** An action \( \pi \in E(C) \) of a component \( C \) of the initial network of the extended sCCP program \( A^+ \) is continuously approximable if and only if \( \text{rate}(\pi) \) is differentiable and \( \text{rate}(\pi)[X] = 0 \) whenever \( \text{guard}(\pi)[X] \) is false.\(^{10}\)

A vector \( \kappa \) is consistent if and only if \( \kappa[e] = 1 \) only for edges \( e \) that are continuously approximable.

In the following, we suppose to work with consistent \( \kappa \)'s.

At this point we are ready to introduce the basic components of our target TDSHA. In order to illustrate the mapping, we will exemplify it by using the self-regulatory network of the Example 5.1.

**Discrete Modes.** The modes of the TDSHA will be essentially the states \( S(C) \) of \( \text{RTS}(C) \). However, continuous transitions cannot change mode, hence we need to consider as equivalent those states that can be reached by a path of continuous edges. Actually, a distinction among those collapsed states will be maintained through the dependence on state variables \( P \), added in the extension \( A^+ \), as explained below.

**Definition 6.2.** Let \( \text{RTS}(C) \) be the RTS of agent \( C \), let \( E(\kappa, C) \subseteq E(C) \) be the subset of continuous transitions, and let \( s_1, s_2 \in S(C) \).

\( s_1 \sim_\kappa s_2 \) if and only if there is a path of edges in \( E(\kappa, C) \) from \( s_1 \) to \( s_2 \) in the non-oriented graph \( G_\kappa(C) = (S(C), E(\kappa, C)) \).

Clearly \( \sim_\kappa \), being a reachability relation in a non-oriented graph, is an equivalence relation, hence we can define the set \( S_\kappa(C) = S(C)/\sim_\kappa \) of RTS-states modulo \( \sim_\kappa \).

For each edge \( e \in E(\kappa, C) \), we define the stoichiometric vector \( \nu_e \) as an \( |Y| \)-vector, \( Y = X \cup \{ P_C | C \in D \} \), such that \( \nu_e[X] = h \) if and only if variable \( X \) is updated by transition \( e \) according to the formula \( X' = X + h \).

**Example.** Consider the gene component of Example 5.1. Its RTS, shown in Figure 2, has three states, corresponding to the three components gene\(_0\), gene\(_1\), and gene\(_2\), with state variables denoted by \( P_0 \), \( P_1 \), and \( P_2 \), respectively. The RTS has also 6 transitions, indexed by \( e_1, \ldots, e_6 \). Consider

\(^{10}\) Guards of continuously approximable \( \pi \) are, in fact, redundant.
the \( \kappa \) vector equal to \((1,0,0,1,1,1)\): edges \( e_1, e_4, e_5, e_6 \) will be approximated as continuous, while the other two remain discrete. The relation \( \sim_\kappa \) has a quotient state space containing two classes: \( S_1 = \{ \text{gene}_0 \} \) and \( S_2 = \{ \text{gene}_1, \text{gene}_2 \} \). Such a partitioning of the gene’s states can be seen as a way to render a slower dynamics for the binding/unbinding mechanism of the first repressor, to be compared to a faster dynamics of the second copy of the repressor.

**Continuous flow.** The continuous evolution for TDSHA is constructed easily: To each edge \( e \in E(\kappa, C) \) looping on \([s]\) we associate the continuous transition 
\[ (\cdot, [s], \nu_e, \text{rate}[e]), \]
with stoichiometric vector \( \nu_e \) and rate function \( \text{rate}[e] \).

Hence, 
\[ \mathcal{T}_C = \{(\cdot, \text{exit}(e)], \nu_e, \text{rate}[e]) | e \in E(\kappa, C)\}. \]

**Example 6.1.** Continuing the previous example, we have that the set of continuous transitions for the TDSHA associated to the gene component is:
\[ \mathcal{T}_C = \{(\text{[gene}_0], (1, 0, 0, 0, 0)^T, k_{p1} \cdot P_0), ([\text{gene}_1], (1, 0, 0, 0, 0)^T, k_{p2} \cdot P_1),
([\text{gene}_1], (0, 0, 0, -1, 1)^T, k_{b2} \cdot X_{p2} \cdot P_1), ([\text{gene}_2], (0, 0, 0, 1, -1)^T, k_{u2} \cdot P_2)\}. \]

**Stochastic transitions.** Stochastic transitions are defined in a very simple way, as guards and rates are basically copied from the original sCCP edge. The only technicality is the definition of the reset.

Consider the state counting variables \( P = \{ P_C \mid C \in D \} \). They can assume values less than or equal to one, as the initial program is simple. Moreover, they range in the whole real-valued interval \([0,1]\) whenever we are in a clustered state \([s]\) collapsing \( s_1, \ldots, s_k \) of RTS(C). In this case, the state variables \( P_{s_1}, \ldots, P_{s_k} \) must sum exactly to 1, their value representing the likelihood of being in state \( s_1, \ldots, s_k \) while within the cluster \([s]\), respectively.

In order to deal with state clusters correctly, we have to ensure that when a state \([s]\) is left, all its state variables are set to zero. Moreover, if a discrete transition looping in \([s]\) takes place, then the variable of its target state \( s_i \) must be set to 1, while all other variables of \([s]\) are to be reset to 0. To enforce this, consider an sCCP edge connecting states \( s_1 \) and \( s_2 \), with

\[ \text{update}[e] \overset{\text{def}}{=} X' = f(X) \land P_{s_1}' = P_{s_1} - 1 \land P_{s_2}' = P_{s_2} + 1, \]

and define the function \( f_P \) on \( P \) which is 1 on the component corresponding to \( P_{s_2} \) and zero elsewhere. In this way, \( P' = f_P(P) \) implements the correct updating policy. Let now \( \bar{f} \) combine \( f \) and \( f_P \): \( \bar{f} \left( \begin{array}{c} X \\ P \end{array} \right) = \left( \begin{array}{c} f(X) \\ f_P(P) \end{array} \right) \).

\[11\text{Transition names are irrelevant for this construction, hence we omit them.}\]
Putting everything together, we have that the stochastic transition associated with \( e \in E(\neg \kappa, C) \) with \( e = (s_1, s_2) \) is
\[
(\cdot, [s_1], [s_2], \text{guard}[e], Y' = \bar{f}(Y), \text{rate}[e]) \in \mathcal{T}. 
\]

We can now collect all our considerations into the following definition.

**Definition 6.3.** Let \( \mathcal{A} = (A, \mathcal{D}, X, init_0) \) be a simple sCCP program and \( \mathcal{A}^+ = (A^+, \mathcal{D}^+, Y, init^+_0) \) be its extended version. Let \( C \) be a sequential component in parallel in \( \mathcal{A}^+ \), with \( RTS(C) = (S(C), E(C), \ell) \). Fix a boolean vector \( \kappa \in \{0, 1\}^m, m = |E(C)| \). The Transition-Driven Stochastic Hybrid Automaton associated with \( C \) with respect to \( \kappa \) is \( T(C, \kappa) = (Q, Y, \mathcal{T}, \mathcal{D}, \mathcal{G}, init) \), where

- \( Q = S_{\kappa}(C) = S(C)/\sim_{\kappa} \);
- \( \mathcal{T} = \{ (\cdot, [\text{exit}(e)], \nu_{Y,e}, \text{rate}[e]) \mid e \in E(\kappa, C) \} \);
- \( \mathcal{D} = \emptyset \);
- \( \mathcal{G} = \{ (\cdot, [s_1], [s_2], \text{guard}[e], Y' = \bar{f}(Y), \text{rate}[e]) \mid e = (s_1, s_2) \in E(\neg \kappa, C) \} \);
- \( init = init_0 \).

**Example.** From the previous definition it is easy to generate the TDSHA relative to our running example above, in which \( \kappa = (1, 0, 0, 1, 1, 1) \). Once we have the TDSHA, we can generate the corresponding PDMP, which is shown in Figure 3.

Definition 6.3 gives a way to associate a TDSHA with a sequential agent of a sCCP program. In order to define the TDSHA for the whole program, we will use the product construction.

**Definition 6.4.** Let \( \mathcal{A} = (A, \mathcal{D}, X, init_0) \) be a simple sCCP program and \( \mathcal{A}^+ = (A^+, \mathcal{D}^+, Y, init^+_0) \) be its extended version, with \( A^+ = C_1 \parallel \cdots \parallel C_n \). Fix a boolean vector \( \kappa_i \) for each sequential agent \( C_i \). The Transition-Driven Hybrid Automaton for the sCCP program \( \mathcal{A} \), with respect to \( \kappa = \kappa_1 \oplus \cdots \oplus \kappa_n \) is
\[
\mathcal{T}(\mathcal{A}, \kappa) = \mathcal{T}(C_1, \kappa_1) \otimes \cdots \otimes \mathcal{T}(C_n, \kappa_n).
\]

**Example.** Consider again the sCCP model of Example 5.1. It has three components: gene, deg and dimer, with 6, 1, and 2 edges respectively. We consider three vectors \( \kappa_1 = (1, 0, 0, 1, 1, 2), \kappa_2 = (1), \) and \( \kappa_3 = (1, 1) \). The
Figure 3: PDMP associated with the gene component of Example 5.1. Transitions approximated continuously determine a set of ODE, while discrete transitions are stochastic and are depicted here as edges of a graph (showing rates, guards and resets). The ODEs are obtained from continuous transitions according to the recipe of Section 3. We chose to display TDSHA in this format as it is similar to the classic representation of hybrid automata. Notice that, considering only the gene component, the transition from the first to the second state can never happen, as it requires $X_{p2} > 0$, but $\dot{X}_{p2} = 0$ and $X_{p2}(0) = 0$. However, this deadlock will be overcome by considering also the dimer component, which defined the dynamics of the the dimer, cf. Figure 5.

Figure 4: PDMP obtained from the product of TDSHA associated with the three components of the sCCP program of Example 5.1. See also the caption of Figure 3.

product TDSHA of these three components generates the PDMP depicted
in Figure 4.

If we want to construct a PDMP from the TDSHA associated to a sCCP program, such a TDSHA $T(A, \kappa)$ must have terminating instantaneous transitions, cf. Definition 4.2. In the mapping we put forward in Definition 6.4, this property holds trivially, as $T \varnothing = \emptyset$. Hence we have the following:

**Proposition 6.1.** The TDSHA $T(A, \kappa)$ associated to a sCCP program $A$ has terminating instantaneous transitions.

### 6.1 Adding non-stochastic discrete transitions

We turn now to generalize the previous mapping by allowing also non-stochastic discrete transitions. We will follow the approach of [BP08a], [BP10], [BP09e], where we replace stochastic transitions with their determinization, as defined in Section 3.1.

Given a sCCP program $A$, we focus on a component $C$ in parallel in the initial network of the extended version $A^+$ of $A$, and we construct $RTS(C) = (S(C), E(C))$. Consider a consistent $|E(C)|$-vector $\kappa$, defining a partition of the actions of $C$ into discrete and continuous. In order to partition discrete actions into stochastic and non-stochastic, we introduce another $|E(C)|$-vector $\omega$, such that $\omega[e] = 1$ implies that the edge $e$ is to be treated as non-stochastic. In order to interface correctly $\omega$ with $\kappa$, $\omega$ must have

$$\kappa[e] = 1 \Rightarrow \omega[e] = 1.$$  

From now on, we refer to pairs of $|E(C)|$-vectors $\kappa$, $\omega$ satisfying the above condition, as a *partitioning scheme* for the component $C$. Hence, we can define the set of edges to be treated as discrete and deterministic as $E(\omega, C) = \{ e \in E(C) | \kappa[e] = 0 \land \omega[e] = 1 \}$.

The definition of the TDSHA associated to component $C$ with respect to the partitioning scheme $(\kappa, \omega)$ can be given quite simply using the determinization operator $d(\cdot)$ of Definition 3.3.

**Definition 6.5.** Let $C$ be a component of the initial network of the extension $A^+ = (A^+, D^+, Y, init^+(Y))$ of the sCCP program $A = (A, D, X, init(X))$, and let $(\kappa, \omega)$ be a partitioning scheme for $C$. The TDSHA associated to $C$ with respect to the partitioning scheme $(\kappa, \omega)$ is

$$T(C, \kappa, \omega) = d_{\eta_{e_1}, \ldots, \eta_{e_k}}(T(C, \kappa)),$$

where $\eta_{e_1}, \ldots, \eta_{e_k}$ are the stochastic transitions in $T(C, \kappa)$ corresponding to edges $\{e_1, \ldots, e_k\} = E(\omega, C)$.

\footnote{With $\oplus$ we denote the direct sum of vectors which can be thought of as their concatenation.}
Figure 5: PDMP obtained from the Example 5.1 with respect to the partitioning scheme presented in Section 6.1. With respect to Figure 4, we can observe the presence of the two variables $Z_i$ used to fire the deterministic transitions at the average time of their corresponding stochastic $sCCP$-edge.

**Definition 6.6.** Let $A = (A, D, X, init(X))$ be a simple $sCCP$ program and $A^+ = (A^+, D^+, Y, init^+(Y))$ be its extended version, with $A^+ = C_1 \parallel \ldots \parallel C_n$. Fix a partitioning scheme $(\kappa_i, \omega_i)$ for each component $C_i$. The Transition-Driven Hybrid Automaton for the $sCCP$ program $A$, with respect to $\kappa = \kappa_1 \oplus \cdots \oplus \kappa_n$ and $\omega = \omega_1 \oplus \cdots \oplus \omega_n$ is

$$\mathcal{T}(A, \kappa, \omega) = \mathcal{T}(C_1, \kappa_1, \omega_1) \otimes \cdots \otimes \mathcal{T}(C_n, \kappa_n, \omega_n).$$

**Example.** Going back to our running example 5.1, we consider the TDSHA obtained by the following partitioning scheme for the gene component:

$$\kappa = (1, 0, 0, 1, 1, 1) \quad \omega = (1, 1, 1, 1, 1, 1),$$

i.e. such that the two discrete transitions are deterministic. For the other two components, deg and dimer, the partitioning scheme sets every transition as continuous, i.e. $\kappa = \omega = 1$. The PDMP obtained is shown in Figure 5. Actually, this is a hybrid automaton, as there is no stochastic transition. Note the two “clock-like” variables $Z_2$ and $Z_3$, governing the activation of the two transitions changing mode.

We turn now to prove that the TDSHA with mixed continuous, stochastic, and deterministic transitions has terminating instantaneous transitions. Consider an instantaneous transition $\delta \in \mathcal{T}\mathfrak{D}$. When such a transition fires, it must hold that $Z_\delta \geq 1$. After its execution, $Z_\delta$ is reset to zero. In order for $Z_\delta$ to reach again the value 1, current time $t$ must advance to $t + \Delta t$, for a $\Delta t > 0$, hence transition $\delta$ can fire at most once at each time instant.
This property is true for all instantaneous transitions in $\Sigma$, therefore the possibility of having infinite sequences of transitions of $\Sigma$ in the same time instant is ruled out.

**Theorem 6.1.** The TDSHA $T(A, \kappa, \omega)$, associated to a $sCCP$ program $A$, has terminating instantaneous transitions.

**Example 6.2.** We consider again the $sCCP$ program of Example 5.1, showing plots comparing the different dynamics exhibited by four semantics, for the same parameter's value. The four models we take into account are the CTMC generated by the standard SOS semantics, the ODE obtained by the fluid-flow approximation, the PDMP obtained with the partitioning scheme of Section 6 and shown in Figure 4 and the hybrid automaton shown in Figure 5 with respect to the partitioning scheme given in this section. Plots of the dynamics are shown and compared in Figure 6. As we can see, the behaviour of the stochastic and non-stochastic PDMP are similar to the original CTMC, despite a reduction in the noise. The fluid-flow model, instead, treating as continuous all the states of the gene, fails to capture the oscillatory pattern evident in the other three cases.

**Remark 6.1.** In the presentation of the mapping from $sCCP$ to TDSHA we ignored the conditions that must be satisfied by the PDMP associated to the TDSHA. Conditions (1) and (2) hold if the functions defining rates and resets are sufficiently smooth.

Condition (3) is satisfied by construction (each $sCCP$ action alters the state of the system, either the store or the internal state of an agent).

The most delicate one is condition (4), which requires that the number of jumps is finite on average in any time instant. In order to prove such condition, we need to enforce the boundedness of the rate $\lambda$. This guarantees that the number of stochastic jumps is finite on average. In fact, if $\lambda \leq M$, then on average the number of jumps up to time $T$ is less than or equal to $M \cdot T$. In order to satisfy such condition, we can require that the domain of continuous variables is bounded. This can be enforced by the trick discussed in Remark 4.2. If the rate function is continuous (which we assume it is) on a bounded domain, then it is also bounded.

In order to ensure that also the number of instantaneous transition fired is bounded, we can observe that the boundedness of the domain implies that also the vector field $\mathcal{X}$ is bounded, say $\mathcal{X}_q[Z_q] \leq K$ for the equation of $Z_q$, hence the time required for the variable $Z_q$ to go from 0 to 1 is no less than $1/K$. Hence, at time $t$ such an instantaneous transition can have fired at most $Kt$ times.
Figure 6: Comparison of simulations of Example 5.1 with respect to four different semantics. (6(a)) Simulation of the standard CTMC model. (6(b)) Simulation of the ODE model associated by fluid flow approximation. (6(c)) Simulation of the PDMP of Section 6. (6(d)) Simulation of the Hybrid Automaton of Section 6.1. Parameters used in the simulations are \( k_{b_1} = 0.0001, \ k_{u_1} = 0.005, \ k_{b_2} = 1, \ k_{u_2} = 1, \ k_{p_1} = 1.0, \ k_{p_2} = 0.2, \ k_d = 0.01, \ k_x = 0.01, \ k_{-x} = 1 \). Stochastic simulations have been generated by the software tool Dizzy [CG]. Hybrid simulations have been generated by a software tool written by the authors, available on request. Plots have been drawn with R.
6.2 Partitioning of transitions

In the previous sections we showed how to construct the hybrid semantics for sCCP, given a partitioning of transitions. A crucial problem is clearly how to define such a partitioning scheme. Such a question is still open and has received a lot of attention in the field of biochemical reactions, see for instance [Pah09, HR05, GP06]. We are currently investigating it in our setting. In the following, we will just sketch some possible criteria, focussing only on the partitioning of transitions between continuous and discrete/stochastic ones. We can consider two classes of heuristics, one agent-based and the other action-based.

Agent-based heuristics. One criterion to partition transitions into discrete and continuous can be related to agents or “species”. More specifically, we can consider the set $Y = X \cup P$ of system and state variables as the set of species, and partition it in two subsets: $Y_D$, the set of discrete species, and $Y_C$, the set of continuous species. Given such subdivision, we can choose as discrete actions all those sCCP-edges such that their reset modifies the value of at least one discrete species.

As for the partition of species, one possible rule can be to define as discrete all states of sCCP agents, i.e. all variables of $P$, plus all variables of $X$ which usually have a “small” value during the dynamic evolution of the system. This last property can be connected with the identification of conservation laws\footnote{A variable $X$ is conserved if it belongs to a subset $X' \subset X$ such that $\sum_{Y \in X'} Y(t) = K$ for all time instant $t$. Therefore, each $X \in X'$ is such that $X \in [0, K]$. If $K$ is small, then each variable in $X'$ should be treated as discrete.}, see also [BP09b], or with the fact that the (expected) value of $X$ remains always small. Note that the use of conservation laws would automatically imply the discreteness of variables of $P$, because variables in $P$ can be partitioned in conserved subsets with total mass one, cf. foonote 13.

Action-based heuristics. Another approach to partition sCCP-edges is to separate actions into discrete and continuous according to their speed. Slow actions, i.e. actions which have a small rate and happen rarely, can be kept discrete, while fast actions, i.e. actions which are very frequent, may be approximated as continuous. This approach is based on the fact that fast actions should reach equilibrium in between the happening of two slow actions. The rates may be computed using bounds on the value of variables, so to guarantee that fast and slow actions will remain so during all the temporal evolution. Then, the partition can be obtained by choosing a threshold and setting as fast all actions whose rate is above such threshold. With such an approach, we may declare as continuous even transitions connecting two discrete modes of the RTS, if they are fast enough. The drawback is that
rates depend on variables, hence it may be difficult to assess a-priori whether an action is fast or slow. Dynamic partitioning schemes may improve such an approach, see Section 8 and [HR05, GP06].

**Remark 6.2.** We remark here that our framework allows a discrete and a continuous transition to modify the same variable, which can undergo a continuous evolution interrupted by discrete, discontinuous jumps.

### 7 Lattice of TDSHA

Definition 6.6 associates a TDSHA with a sCCP agent for a fixed partitioning scheme \((\kappa, \omega)\) of transitions. \(\kappa\) first partitions transitions into discrete and continuous, then \(\omega\) divides discrete transitions into stochastic and deterministic.

Clearly, different choices of \(\kappa\) and \(\omega\) correspond to different TDSHAs, all related to the CTMC-based semantics of the original sCCP program but with a different degree of approximation.

All these different TDSHAs can be arranged into a lattice according to the following pre-order:

**Definition 7.1.** Let \(A\) be a sCCP agent, then \(\mathcal{T}(A, \kappa_1, \omega_1) \subseteq \mathcal{T}(A, \kappa_2, \omega_2)\) if and only if \(\kappa_1[e] = 1 \Rightarrow \kappa_2[e] = 1\) and \(\omega_1[e] = 1 \Rightarrow \omega_2[e] = 1\), for each transition \(e \in E(A) = E(C_1) \cup \ldots \cup E(C_n)\) (i.e. \(\kappa_1 \leq \kappa_2\) and \(\omega_1 \leq \omega_2\)), where \(A = C_1 \parallel \ldots \parallel C_n\) is the initial agent of \(A\).

The bottom element of this lattice is obtained for \(\kappa \equiv 0\), \(\omega \equiv 0\) while the top element is obtained for \(\omega \equiv 1\) and \(\kappa[e] = 1\) if and only if \(e\) is continuously approximable. We remind to the reader that transitions not continuously approximable must be kept discrete.

The two “extreme” choices correspond to two particularly important TDSHAs, as shown in the following propositions.

**Proposition 7.1.** Let \(A\) be a sCCP program. The TDSHA \(\mathcal{T}(A, 0, 0)\) is the CTMC associated with \(A\) by its standard semantics.

**Proof.** In \(\mathcal{T}(A, 0, 0)\) all transitions are stochastic, and discrete mode are in bijection with agents’ states. State variables change only when a stochastic transition fires, hence they are constant between two consecutive firings. The proposition follows because, by construction, there is a bijective correspondence between active actions of the sCCP program and active stochastic transitions of the TDSHA. \(\blacksquare\)

**Remark 7.1.** Consider the definition of rate \(\lambda\) and reset \(R\) of the PDMP associated to the TDSHA \(\mathcal{T}(A, 0, 0)\). \(\lambda\) defines the probability distribution of the time \(T\) of the next firing, while \(R\) gives a probability distribution on the next active stochastic transition/sCCP edge \(E\) to be executed. Due
to independence of $E$ and $T$, we can take their product to obtain the joint distribution of $T$ and $E$. This joint distribution is the one at the basis of the Gillespie's algorithm [Gil77] applied to the CTMC-based semantics of sCCP, cf. [BP08b] for further details.

**Proposition 7.2.** Let $A$ be a sCCP program with initial agent $A = C_1 \parallel \ldots \parallel C_n$. If $e$ is continuously approximable for each $e \in E(A) = E(C_1) \cup \ldots \cup E(C_n)$, $\mathcal{T}(A, 1, 1)$ coincides with the system of ODEs associated with $A$ by its fluid-flow approximation (see the end of Section 5).

**Proof.** $\mathcal{T}(A, 1, 1)$ has one single mode and no discrete edges, hence all transition are continuous. In this case, it is easy to see that, for each component $C$, $\chi_q^C = \Phi_C$ and

$$
\chi_q^{C_1 || C_2} = \chi_q^{C_1} + \chi_q^{C_2} = \Phi_{C_1} + \Phi_{C_2} = \Phi_{C_1 || C_2},
$$

where $\chi^C$ is the vector field of the PDMP associated to the TDSHA $\mathcal{T}(A, 1, 1)$ ($q$ is its only mode) and $\Phi_C$ is defined at the end of Section 5.

In between these two extremes, intermediate TDSHAs characterized by a mix of continuous, stochastic, and discrete deterministic transitions, are available. A particular TDSHA is obtained by choosing to have all transitions discrete and deterministic, i.e. for $\kappa \equiv 0$, $\omega \equiv 1$. In this case, we obtain a (multi-rate) timed automaton [AD94], [HKPV95].

We recall that a multi-rate timed automaton is a tuple $(Q, q_0, E, C, r, Inv)$, where $Q$ is the set of modes of the automaton, $q_0$ is the initial mode, $C$ is the set of clocks, i.e. variables $x$ taking values in the positive reals and evolving according to the differential equation $\dot{x} = k$, and $r : C \times Q \to \mathbb{R}^+$ is a function assigning a rate to each clock in each mode. Furthermore, $Inv : Q \to \mathcal{B}(C)$ assigns to each mode an invariant condition, i.e. a clock constraint in $\mathcal{B}(C)$, the set of propositional formulae whose atomic propositions are of the form $x \sim r$ or $x - y \sim r$, $x, y \in C$, $r \in \mathbb{R}^+$, $\sim \in \{<, \leq, =, \geq, >\}$. Invariants force the automaton to take an edge when they are violated. Finally, edges in $E$ are tuples $(q_1, g, a, r, q_2)$, where $q_1 \in Q$ is the exit mode, $q_2 \in Q$ is the target mode, $a \in \Sigma$ is a letter of a given alphabet, $g \in \mathcal{B}(C)$ is the guard of the edge, which must be true for the edge to be taken, and $r \subseteq C$ is a subset of clocks that are reset to 0 when the edge is crossed.

**Proposition 7.3.** Let $A$ be a sCCP program. The TDSHA $\mathcal{T}(A, 0, 1)$ is equivalent to a multi-rate timed-automaton.

**Proof.** The proposition holds for the TDSHA associated to a sequential component $C$, whose timed automaton $(Q, q_0, E, C, r, Inv)$ is defined in the following way:
• The set $Q$ of modes of the automaton is $Q = \{Q \times \prod_{X \in \mathbb{X}} \text{Dom}(X)\}$, where the domain of a variable $X$ is its domain in the sCCP program, a finite or countable set, cf. Section 5. Hence, modes of the timed automaton record also the value of system variables.

• $q_0 = \text{init}$.

• $C = \{Z_\delta \mid \delta \in \Sigma \mathbb{D}\}$, i.e. there is one clock for each instantaneous edge of the TDSHA.

• $r(Z_\delta, (q, x)) = \text{rate}[\delta](x)I_{\alpha_1[\delta]=q}$, i.e. the rate of the clock $Z_\delta$ in a state $(q, x) \in Q$ of the timed automaton is the value of its rate function calculated on $x$, but only if the component $q$ of the state $(q, x)$ corresponding to discrete modes of the TDSHA is the exit mode $e_1[\delta]$ of the instantaneous edge $\delta$.

• $\text{Inv}(q, x) = \bigwedge_{\delta \in \Sigma \mathbb{D}} Z_\delta < 1$, i.e. the invariant condition in each mode requires each clock variable to be less than 1.

• To each instantaneous transition $\delta = (\alpha, q_1, q_2, \text{guard} \wedge Z_\delta \geq 1, \text{reset}, p)$ we associate the following edges of the timed automaton: $\{(q_1, x), Z_\delta \geq 1, \alpha, \{Z_\delta\}, (q_2, \text{reset}(x)) \mid \text{guard}(x) \text{ is true}\}$, i.e. an edge going from $(q_1, x)$ to $(q_2, \text{reset}(x))$, with guard $Z_\delta \geq 1$, label $\alpha$, and resetting to zero only the clock $Z_\delta$.

The previous mapping works because:

1. all transitions of $\mathcal{T}(C, 0, 1)$ are instantaneous;

2. all system variables $\mathbb{X}$ are governed by the ODEs $\dot{X} = 0$, hence they can be modified only by resets of discrete transitions;

3. between two discrete transitions, the value of variables $\mathbb{X}$ is constant, hence the ODEs for $Z_\varepsilon$ are of the form $\dot{Z_\varepsilon} = \text{const}$ and each $Z_\varepsilon$ is a skewed clock.

The proposition is clearly preserved by TDSHA product. □

Remark 7.2. A central problem in the definition of the hybrid semantics for sCCP is to evaluate how “good” is a specific partitioning scheme. In other words, we need to be able to compare different points of the lattice. In general, we will choose a reference model and compare different hybrid semantics with this one. The choice falls on the original CTMC-based semantics, which is also in line with previous work on hybrid simulation strategies for biochemical systems [HR05]. In order to study the goodness of the approximation, we consider three different strategies: compare different simulation runs, prove analytical theorems, or use a logic-based framework.
The first approach is the classical one, and consists in comparing an ensemble of simulation runs of the CTMC with an ensemble of runs of the hybrid model (or a single run, if no stochasticity is left). The drawback of this approach is that it is computationally very expensive. The good aspect is that it just requires a simulation algorithm.

The second approach, i.e. finding theorems, is much more difficult. In addition, theorems usually deal with asymptotic comparison, hence they can be applied only to models where continuous variables have large values. We have proved results in this direction [Bor10], by tailoring Kurtz’s theorem [Kur70] to our setting. Finally, the third approach is a research line we suggested in [BP10], and essentially consists in defining some qualitative features of interest of the dynamics and express them in a suitable temporal logic. Then, we compare two models with respect to such set of formulae: two models are equivalent if and only if they satisfy the same subset of the original set of formulae.

Whatever the choice, we will not investigate further in this paper the relationships among all the different models associated to a single sCCP program in terms of their semantics. Here we are mainly concerned in presenting the framework from a syntactic perspective.

8 Dynamic Partitioning of Transitions

In the previous sections we have defined a mapping from sCCP to TDSHA fixing the level of discreteness and continuity. This choice, however, can be difficult to perform \textit{a priori}, as one does not know if the system will evolve to a state where a different approximation is more accurate. This is particularly true when one deals with biological systems. In this case, reactions involving large populations of molecular species or having high rates, may be treated as continuous. However, such conditions depend on the state of the system and may change during its evolution. Indeed, there has been a growing attention on hybrid simulation strategies in systems biology, cf. [Pah09].

In order to have dynamic switching, we can extend the discrete modes, introducing states for each admissible partitioning scheme \(\kappa\) and \(\omega\). New discrete transitions need to be added as well, changing the value of \(\kappa\) and \(\omega\) according to some user defined conditions.

Intuitively, for each transition \(e\) the state space must be partitioned in three—possibly empty—regions: one where the edge \(e\) is treated as a continuous transition, one when the edge \(e\) is converted to a stochastic transition, and the last one where the edge \(e\) becomes an instantaneous transitions.

We will define such regions by logical formulae. If \(\phi = \phi(X)\) is a formula in first-order logic interpreted on \(\mathbb{R}\) with \(n\) free variables, we denote the region of \(\mathbb{R}^n\) it defines by \(R_\phi = \{x \in \mathbb{R}^n \mid \phi(x) \text{ true}\}\). Given two formulae \(\phi\) and
ψ, we say they are separated if and only if $\overline{R_\phi} \cap \overline{R_\psi} = \emptyset$, i.e. if the closure of the regions defined by ψ and φ are disjoint.

We consider four formulae:

1. $\text{cont}[e](X)$, encoding the condition to change edge $e$ from discrete to continuous;

2. $\text{disc}[e](X)$, encoding the condition to change edge $e$ from continuous to discrete.

3. $\text{stoch}[e](X)$, expressing when to change the edge $e$ from deterministic to stochastic.

4. $\text{det}[e](X)$, expressing when to change the edge $e$ from stochastic to deterministic.

We suppose that the formulae $\text{cont}[e]$, $\text{disc}[e]$ are separated, as well as $\text{stoch}[e]$ and $\text{det}[e]$. This guarantees, for instance, that, if we change the edge $e$ from continuous to discrete, it cannot change back immediately (i.e. before time elapses) to continuous.

**Remark 8.1.** One possibility to define such formulae is to consider two (sufficiently regular, usually continuous) functions $f^c_e, f^s_e : \mathbb{R}^n \to \mathbb{R}$. Function $f^c_e$ can be used to discriminate between continuous and discrete regions for edge $e$ by looking at its sign. More specifically, we define, for a fixed, small $\varepsilon > 0$.

1. $\text{cont}[e](x) := f^c_e(x) \geq \varepsilon$;

2. $\text{disc}[e](x) := f^c_e(x) \leq -\varepsilon$.

Using $\varepsilon$ instead of 0 guarantees that $\text{cont}[e]$ and $\text{disc}[e]$ are separated, hence avoids pathological situations of infinite sequences of instantaneous transitions changing edge $e$ back and forth from continuous to discrete in the same time instant. A similar mechanism can be used to define $\text{stoch}[e]$ and $\text{det}[e]$.

Now, suppose $e$ is continuous. If the current trajectory enters in a region of the state space in which $\text{disc}[e]$ becomes true, then we must trigger an instantaneous transition in order to move from $\kappa_1[e] = 1$ to $\kappa_2[e] = 0$. All the variables must remain unchanged. It may also happen that this transition triggers a change in the value of $\omega[e]$, from deterministic to stochastic, but this will be implemented by another instantaneous transition.

However, in this scenario it may happen that the new relation $\sim_{\kappa_2}$ splits in two the current mode $[s] \in S_{\kappa_1}(C)$, say $[s]_{\kappa_1} = [s_1]_{\kappa_2} \cup [s_2]_{\kappa_2}$. In this case, we need to introduce two instantaneous transitions, one going to $[s_1]$ and the other to $[s_2]$. Now, consider the value of the state variables of $[s]$, $P_{[s]} = \sum_{s_i \in [s]} P_{s_i}$. It can be proved that $P_{[s]} = 1$. Moreover, $P_{[s]} = P_{[s_1]} + P_{[s_2]}$ but, clearly, it is not necessarily the case that the two quantities on the
right hand side of the equality are equal. This means that the system may “prefer” to move to states in \([s_1]\) than to those in \([s_2]\). This situation is correctly modelled using priorities, i.e. weighting transition to \([s_i]\) by \(P_{[s_i]}\) and re-normalizing variables in \([s_1]\) and \([s_2]\) to maintain the property \(P_{[s]} = 1\) for each \([s] \in S_\kappa(C)\).

We now give a formal definition for this construction, following a similar strategy as in Section 6: first we construct TDSHA for sequential components, then we apply the product construction to combine the local constructions. In order to fix the notation, consider the TDSHA \(T(C, \kappa, \omega) = (Q, X, T_C, T_D, T_S, init, \mathcal{E})\) associated with a component \(C\), with respect to a fixed partitioning scheme \(\kappa, \omega\). With \(Q_{\kappa,\omega}\) we indicate the set \(Q_{\kappa,\omega} = \{(\kappa,\omega) \mid [s] \in Q\}\). Moreover, \(T_C, T_D, T_S\) denote the sets \(T_C, T_D, T_S\), respectively, with states in \(Q\) replaced by the corresponding states (equivalence classes) in \(Q_{\kappa,\omega}\). A similar rule applies to \(init_{\kappa,\omega}\).

The dynamic change of an edge \(e\) from deterministic to stochastic, instead, presents less difficulties, as we do not have to deal with state splitting or merging. In this case, however, we enable such transitions only for discrete edges.

**Definition 8.1.** Let \(A = (A, D, X, init_0)\) be a simple sCCCP program and \(A^+ = (A^+, D^+, Y, init^+_0)\) be its extended version. Let \(C\) be a sequential agent in parallel in \(A^+\), with \(RTS(C) = (S(C), E(C), \ell)\) and \(|E(C)| = m\). Moreover, let \(cont[e], disc[e], stoch[e], det[e], e \in E(C)\) be defined as above. The TDSHA with dynamic partitioning associated with \(C\) is \(T(C, cont, disc, stoch, det) = (Q, X, T_C, T_D, T_S, init, \mathcal{E})\), with:

1. \(Q = \bigcup_{\kappa,\omega \in \{0,1\}^m, \omega \geq \kappa} Q_{\kappa,\omega};\)
2. \(T_C = \bigcup_{\kappa,\omega \in \{0,1\}^m, \omega \geq \kappa} T_{C,\kappa,\omega};\)
3. \(T_S = \bigcup_{\kappa,\omega \in \{0,1\}^m, \omega \geq \kappa} T_{S,\kappa,\omega};\)
4. \(T_D = \bigcup_{\kappa,\omega \in \{0,1\}^m, \omega \geq \kappa} T_{D,\kappa,\omega} \cup \bigcup_{\ell=1}^{\kappa_1} T_{D,\kappa,0,\ell} \cup \bigcup_{\ell=1}^{\kappa_2} T_{D,\kappa,\ell} \cup \bigcup_{\ell=1}^{\kappa_1} T_{D,0,\ell} \cup \bigcup_{\ell=1}^{\kappa_2} T_{D,1,\ell},\) where

\[
T_{D,\ell} = \left\{ \left( \alpha^e_{\ell}, [s_1]_{\kappa_1,\omega}, [s_2]_{\kappa_2,\omega}, disc[e], Y' = g(Y), P_{[s_2]} \right) \right\} \quad e \in E(C),
\]

\[
\kappa_1[e] = 1, \kappa_2[e] = 0, \kappa_1[e'] = \kappa_2[e'] \quad \text{for} \ e \neq e', [s_1]_{\kappa_1} \cap [s_2]_{\kappa_2} \neq \emptyset
\]

where \(g\) assigns value \(P_{[s_2]}^{e'}\) for \(s' \in [s_2]_{\kappa_2}\) 0 to any other \(P_s\), and it is the identity on \(X\). \(\alpha^e_{\ell}\) is a special event name reserved for the change of the continuity status of edge \(e\). Moreover

\[
T_{D,0,\ell} = \left\{ \left( \alpha^e_{\ell}, [s]_{\kappa_1,\omega}, [s]_{\kappa_2,\omega}, cont[e], Y' = Y, 1 \right) \right\} \quad e \in E(C),
\]

where \(\kappa_1[e] = 0, \kappa_2[e] = 1\).
\[ \kappa_1[e] = 0, \kappa_2[e] = 1, \omega_2[e] = 1, \kappa_1[e'] = \kappa_2[e'] \land \omega_1[e'] = \omega_2[e'] \text{ for } e' \neq e \}; \\
\Xi_1 = \left\{ \left( \alpha_e^\ast, [s], \kappa_1[e], \omega_1[e], \omega_2[e], \omega_2[e'] \right) \bigg| e \in E(C), \kappa[e] = 0, \omega_1[e] = 1, \omega_2[e] = 0, \omega_1[e'] = \omega_2[e'] \text{ for } e' \neq e \right\}; \\
\Xi_0 = \left\{ \left( \alpha_e^\ast, [s], \kappa_1[e], \omega_1[e], \omega_2[e], \omega_2[e'] \right) \bigg| e \in E(C), \kappa[e] = 0, \omega_1[e] = 0, \omega_2[e] = 1, \omega_1[e'] = \omega_2[e'] \text{ for } e' \neq e \right\}; \\
5. \ \text{init} = \text{init}^+; \\
6. \ \mathcal{E} = \mathcal{E}_0 \cup \{ \alpha_e^\ast, \alpha_e^\ast | e \in E(C) \}, \text{ where } \mathcal{E}_0 \text{ is the set of events associated to component } C \text{ by the static mapping of Definition 6.5.} \\

\textbf{Definition 8.2.} \textit{Let } \mathcal{A} = (A, D, X, \text{init}) \text{ be a simple } s\text{CCP program and } \mathcal{A}^+ = (A^+, D^+, Y, \text{init}^+) \text{ be its extended version, with } A^+ = C_1 \parallel \cdots \parallel C_n. \text{ Moreover, fix predicates } cont_j[e], disc_j[e], stoch_j[e], det_j[e] \text{ for each sequential agent } C_j \text{ of } A^+, \text{ according to Definition 8.1. The Transition-Driven Stochastic Hybrid Automata with dynamic partitioning for the } s\text{CCP program } \mathcal{A}, \text{ with respect to the dynamic partitioning scheme } cont = cont_1 \oplus \cdots \oplus cont_n, disc = disc_1 \oplus \cdots \oplus disc_n, stoch = stoch_1 \oplus \cdots \oplus stoch_n, det = det_1 \oplus \cdots \oplus det_n, \text{ is} \\
\mathcal{T}(\mathcal{A}, cont, disc, stoch, det) = \mathcal{T}(C_1, cont_1, disc_1, stoch_1, det_1) \otimes \cdots \otimes \mathcal{T}(C_n, cont_n, disc_n, stoch_n, det_n). \\

\textbf{Remark 8.2 (On the fly simulation).} \textit{In Definition 8.2, the resulting TD-\text{SHA has a number of modes exponential in the number of transitions that sequential agents can perform. This combinatorial explosion rules out the possibility of generating all the modes together. However, if we restrict to simulation, this is not a real issue, as we need to record only the current mode: the target mode of a transition can be generated on the fly as soon as the transition has been taken, given the knowledge of RTS.} \\

Consider the TD\text{SHA constructed according to a dynamic partitioning scheme, according to Definition 8.2. Also in this case, it is easy to prove that the set of instantaneous transitions is terminating. Consider an } s\text{CCP action } e, \text{ initially treated as discrete. This action will be moved in the set of continuously-approximated actions as soon as a certain function } f^e_e(x)
reaches the value $\varepsilon$ from below. The function depends only on sCCP store variables $X$, which are not reset by the instantaneous transition. Hence, soon after having switched $e$ from discrete to continuous, $f^e_c(x)$ will still be equal to $\varepsilon$. Now, the condition for bringing $e$ back in the discrete set is $f^e_c(x) \leq -\varepsilon$, which is obviously false after the switching. The same principle applies to the switching between stochastic and deterministic discrete transitions. We have also to consider the case in which, in the same instant of the switching of edge $e$, there are instantaneous transitions associated to discrete and deterministic sCCP-edges ready to fire (let’s call them timed transitions). If they fire, then they can reset the value of variables and potentially activate other switchings. In any case, as each timed transition can fire at most once, the number of instantaneous transitions executable in the same time instant is bounded, hence also in this case it is not possible to execute an infinite sequence of instantaneous transitions.

**Theorem 8.1.** Let $A = (A, D, X, \text{init})$ be an sCCP program and let $T(C, \text{cont}, \text{disc}, \text{stoch}, \text{det}) = (Q, X, \mathcal{T}C, \mathcal{T}D, \mathcal{T}S, \text{init}, E)$ be the TDSHA with dynamic partitioning scheme given by predicates cont, disc, stoch, and det. Then its set of discrete transitions $\mathcal{T}D$ is terminating.

### 8.1 Hybrid Simulation Strategies as Dynamic Partitioning Schemes

The many hybrid simulation algorithms proposed in literature [HR02, Neo04, HR05, SK05, KMS04, GCPS06, ACT+05], at a first approximation, differ in two aspects: the kind of continuous dynamics (it can be based on ODE or SDE) and the rules for partitioning reactions into continuous and discrete (usually called fast and slow). More specifically, the partitioning can be static (done at the beginning of the simulation) or dynamic (i.e. recomputed at run-time). In particular, in all such partitioning schemes a transition is either continuous or discrete and stochastic, hence discrete and deterministic transitions are not taken into account. Indeed, hybrid simulation algorithms proposed can be more complex, for instance combining several approximate stochastic simulation algorithms or partitioning transitions in more than two sets, cf. [Pah09] for further details.

Conditions for separating fast and slow reactions are usually twofold:

1. the **number of molecules** of species involved in the reaction must all be bigger than a given threshold. Usually, a fast reaction $j$ must satisfy a condition like $x_i \geq K|\nu_{i,j}|$ for all species $i$ involved in $j$, where $\nu$ is the stoichiometric matrix.

2. the **rate function** of fast reactions must be reasonably bigger than that of slow reactions. Usually, the following constraint is enforced [SK05]:
\( \lambda_j(\mathbf{x}) \Delta t \geq \Lambda \), which ensures that reaction \( j \) fires many times during the time step \( \Delta t \). In [GCPS06] a different partition strategy imposes that rates of fast reactions are \( \Lambda \) times faster than the fastest slow reaction, so as to guarantee a separation of time scales.

The dynamical policies sketched above can be easily accounted for in our setting. First of all, we need to start from a \textbf{sCCP} model of a biochemical network [BP08b], in which reactions are modelled by action capabilities of agents. Then, applying the framework of this paper, we associate a TDSHA with such a model, together with a suitable policy for dynamic partitioning of transitions. All we have to do is define a function \( f_e \) for each \textbf{sCCP} transition \( e \), such that \( f_e(\mathbf{x}) > 0 \) when the associated reaction can be considered fast and \( f_e(\mathbf{x}) < 0 \) when it is slow.

As an example, consider a partition strategy based only on the size of populations, like the one adopted in [Neo04]. In this case, the function \( f^c_e \) for transition \( e \) can be the following\(^{14}\):

\[
    f_e(\mathbf{x}) = \min\{x_i - K||\nu_e[x_i]|| \mid \nu_e[x_i] \neq 0\},
\]

where \( \nu_e[\cdot] \) is the stoichiometry of action \( e \), constructed as in Section 6, and \( K \) is a constant (that can be tuned for the specific system). Of course, more complex policies can be introduced by suitably modifying the functions \( f_e \).

9 Conclusions

In this paper we presented a formal framework in which we associate a lattice of hybrid models to a \textbf{sCCP} program. Such a semantics, either in it static or in its dynamic version (with respect to partitioning) allows first of all the execution of stochastic programs with hybrid simulation strategies. However, the formal setting opens up further possibilities: establishing theoretical properties and relationships among different points in the lattice (asymptotic behaviour, upper bounds), and using formal tools like reachability analysis, abstraction, and model checking. These are further directions of research we will pursue in the future. Currently, we are working on an implementation of the framework.

We conclude with a philosophical observation. Consider a biological / biochemical scenario, say a collection of molecules, which is modelled by a stochastic program, say an \textbf{sCCP} program. In a sense, the most sensible hardware to implement such a program could be considered the very same collection of molecules the program is simulating. Assuming an incomplete ability to set up stochastic parameters to reach a level of precision such as

\(^{14}\)The function \( f^c_e \) can be trivially set to the constant \( 1 + \varepsilon \), or to every function which makes edge \( e \) always stochastic, whenever it is discrete.
the one that would be provided by this ultra-advanced hardware, is therefore rather natural. This observation leads us to two considerations: if, on the one hand, is unsurprising to assume an incomplete ability to set up stochastic parameters for an sCCP program, on the other hand, the introduction of some form of abstraction in the model, like localized laws representing the (deterministic) dynamics of regions of the system, can be a true advance and improvement in the simulation abilities of the end result.

In other words, the hybrid automata used to simulate the sCCP program can provide a genuine improvement on the original code. This is true especially in those situations in which the control (discrete) aspects of the systems are correctly captured within the program code, while the stochastic parameters, can be undetermined.

Acknowledgements

We want to thank Alberto Casagrande for the useful discussions. We thank also FIRB-LIBi project for financial support.

References


