2010 Seventh International Conference on the Quantitative Evaluation of Systems

Hybrid semantics for PEPA

Luca Bortolussi*, Vashti Galpin†, Jane Hillston†, Mirco Tribastone‡
*Dipartimento Matematica ed Informatica, Università degli Studi di Trieste
luca@dmi.units.it
†Laboratory for Foundations of Computer Science, School of Informatics, University of Edinburgh
{Vashti.Galpin,Jane.Hillston}@ed.ac.uk
‡Institut für Informatik, Ludwig-Maximilians-Universität München
tribastone@pst.ifi.lmu.de

***Abstract*—In order to circumvent the problem of state-space explosion of large-scale Markovian models, the stochastic process algebra PEPA has been given a fluid semantics based on ordinary differential equations, treating all entities as continuous. However, low numbers of instances and/or relatively slow dynamics may make such approximation too coarse for some parts of the system. To deal with such situations, we propose an hybrid semantics lying between these two extremes, treating parts of the system as discrete and stochastic and others as continuous and deterministic. The underlying mathematical object for the quantitative evaluation is a stochastic hybrid automaton. A case study of a client/server system with breakdowns and repairs is used to discuss the accuracy and the cost of this hybrid analysis.**

I. Introduction

Stochastic process algebra, such as PEPA [9], EMPA [1] and IMC [8], are extensively used for modelling and quantitative analysis of systems. Their semantics is given in terms of Continuous Time Markov Chains (CTMCs). However, all these tools suffer from the problem of state space explosion, namely the combinatorial growth of the size of state space, which hinders their practical use. Furthermore, stochastic simulation can also be impractical in many cases.

An alternative technique for analyzing PEPA is fluid-flow approximation [10], which is based on the principle of describing in a continuous fashion the discrete entities in play. A set of Ordinary Differential Equations (ODEs) is associated with a PEPA model, whose solutions are related to the average behavior of the CTMCs. Furthermore, in the limit of infinitely large populations, the trajectories of CTMCs are indistinguishable from those of ODEs [17].

However, a wholly fluid approximation is not always justifiable. This is the case for systems which have some components present in few copies or subject to a slow dynamics; describing them with continuous variables may lead to poor results. Examples include client-server interactions with one or few servers, and, in a different context, description of genetic networks [4].

In these cases, an intermediate approach can be more appealing: just a portion of the system is made continuous, while the rest is kept discrete. This generates hybrid models, with a dynamics combining continuous deterministic evolution with discrete stochastic jumps. This hybrid scheme typically should behave better than ODEs and be more efficient than stochastic simulation, which is the only practicable approach for analyzing CTMCs with large state spaces. Our aim is to obtain a hybrid approximation for large systems that are not amenable to analysis using CTMC semantics rather than to provide a new hybrid formalism.

In this paper, we define a hybrid semantics for PEPA, adapting to this context the ideas introduced in [3]. The starting point is the parametric Markovian semantics of PEPA [17]. The target formalism, combining ODEs and stochastic jumps, is that of Piecewise Deterministic Markov Processes (PDMPs) [6]. However, working with this formalism is difficult, and we find it easier to manipulate a higher level description of PDMPs, called Transition-Driven Stochastic Hybrid Automata (TDSHA) [3], describing the system as a collection of discrete transitions (corresponding to instantaneous or stochastic jumps) and continuous transitions (representing flows acting on system variables). In particular, it is easier to define a notion of product for TDSHA than for PDMPs, which here is tailored to match the PEPA notion of multi-way synchronization. Equipped with these notions, we can define the hybrid semantics in a compositional way, first associating a TDSHA with each PEPA component, and then composing these TDSHA by product.

The first step in this process is the partition of PEPA actions into those amenable to continuous approximation and those to be kept discrete. This choice is crucial, as it influences the accuracy of the hybrid approximation and the efficiency of simulation. At this stage, it is left to the modeler, although we discuss some heuristics that can guide the choice.

In the paper we also discuss a case study modelling a simple client/server interaction, analyzing experimentally the quality and the performance of the hybrid semantics.

Structure of the paper: Section II introduces PEPA and the parametric semantics of PEPA [17]. Section III introduces TDSHA and the synchronization product, while Section IV contains the definition of the hybrid semantics. Finally, Section V discusses the case study, Section VI covers related work and Section VII draws conclusions and discusses future work.

II. PEPA

This section gives a brief overview of PEPA and its fluid-flow approximation. The most relevant notions will be informally introduced through a case study. The reader is referred to [9], [17] for a formal account.

978-0-7695-4188-4/10 $26.00 © 2010 IEEE
DOI 10.1109/QEST.2010.31

$$S_w \stackrel{def}{=} (request, r_{rp}).S_l + (break, r_{br}).S_b$$
$$S_l \stackrel{def}{=} (log, r_{lg}).S_w$$
$$S_b \stackrel{def}{=} (fix, r_{fx}).S_w$$
$$U_r \stackrel{def}{=} (request, r_{rq}).U_t$$
$$U_t \stackrel{def}{=} (think, r_{th}).U_r$$
$$System \stackrel{def}{=} S_w \underset{\{request\}}{\bowtie} U_r[N]$$

Figure 1. A client-server system with breakdowns and repairs.

A. Overview of PEPA

The PEPA model in Figure 1 will be used throughout this paper for illustrative purposes. The model describes the interaction of clients with servers that may break down. A server is modelled as a *sequential component*, i.e., a component that may perform activities in sequence. An activity is denoted by (α, r), where α is the action type and r denotes the parameter of an exponential distribution associated with the duration the activity. Sequential components evolve through a set of *local states* (or *derivatives*). For instance, the local states of the server are S_w, S_l and S_b. A *choice*, denoted by the symbol $+$, indicates that a sequential component enables two or more activities. A race condition determines which of these will be executed. The choice in the local state S_w means that the server may occasionally fail while serving a request. A client is also modelled as a component evolving through the states U_r and U_t. The former enables an activity with the same action type as S_w.

Shared action types indicate synchronisation between PEPA components. The synchronisation behaviour is defined in the *model component* (i.e., $System$ in the case study). The expression $S_w \underset{\{request\}}{\bowtie} U_r$ implies that the two sequential components synchronise on all the activities whose types are in the action set in the cooperation operator. Synchronisation alters the rate of execution of the activity, which will be the minimum of the two individual rates (e.g., $\min(r_{rp}, r_{rq})$ in the example). The process array $U_r[N]$ is a shorthand notation for a composition of N components U_r over empty cooperation sets. This captures that users can execute independently from one another, but each user must synchronise with the server to perform the $request$ activity.

Another operator (not illustrated in the example) is *hiding*. For a set of actions L, any actions will be replaced with the silent action, τ. This permits actions viewed as internal to a system to be hidden from observers. In the example, we could choose to hide server logging by specifying $(S_w \underset{\{request\}}{\bowtie} U_r[N])/\{log\}$.

PEPA has a two-level syntax with S a sequential component and P a model component

$$S ::= (\alpha, r).S \mid S + S \mid C_s \quad P ::= P \underset{L}{\bowtie} P \mid P/L \mid C$$

where C names a model or sequential component, C_s names a sequential component, α is an action from the action set $\mathcal{A}$, r a positive real and $L \subseteq \mathcal{A}$. Structured operational semantics give rise to the derivative set $ds(P)$ and the derivative graph $D(P)$ for a PEPA model P. The graph can be interpreted as a CTMC (continuous time Markov chain) [9].

B. Population-based semantics

When interpreted with the standard Markovian semantics [9], PEPA exhibits rapid state-space growth as a function of the number of sequential components in the model. The population-based semantics in this section admits a differential-equation interpretation of a PEPA model which reduces dramatically the cost of the analysis.

Reduced Context and Numerical Vector Form: We consider the *reduced context* of the original model, i.e., a model component which disregards the multiplicities of its sequential components. The reduced context of the example, denoted by $red(System)$, is

$$red(System) = S_w \underset{\{request\}}{\bowtie} U_r. \tag{1}$$

The ODE of a PEPA model is obtained from a symbolic representation of a CTMC whose state descriptor is in the *numerical vector form* (NVF) The NVF is a vector, denoted by ξ, of non-negative integers in which each element gives the number of components (still discrete at this stage) in the system that are in a particular local state. The local states of interest are those of the sequential components appearing in the reduced context. For example, $\xi = (1, 0, 0, N, 0)$ gives the initial state of the model if one associates $\xi_1, \xi_2, \xi_3, \xi_4, \xi_5$ with the population counts of the local states S_w, S_l, S_b, U_r, and U_t, respectively. In general, let d be the number of sequential components in the reduced context (e.g., $d = 2$ in the example). Each sequential component will be denoted by $\mathcal{C}_i$, $i = 1, \ldots, d$. Each local derivative of $\mathcal{C}_i$ will be denoted by $C_{i,j}$, $j = 1, \ldots, |ds(\mathcal{C}_i)|$

Parametric Derivation Graph: According to the semantics of PEPA, one can write the transition $S_w \xrightarrow{(request, r_{rp})} S_l$, indicating that one component S_w can perform an action $request$ at rate r_{rp} and subsequently behaves as S_l. From this transition it is possible to infer one that generalises as follows:

$$S_w \xrightarrow{(request, r_{rp}\xi_1)}_{\star} S_l. \tag{2}$$

This says that if there are ξ_1 components in state S_w, one of these may perform a $request$ at a rate which is equal to the individual rate r_{rp} multiplied by the number of components in that state. The component S_l is said to belong to the *parametric derivation set* of S_w, written $S_l \in ds^\star(S_w)$. (For a sequential component $ds^\star(S) = ds(S)$.) The star symbol used here and in (2) highlights that these relations are different from their counterparts in the Markovian semantics of PEPA. Specifically, the rates are now functions of the state descriptor in the NVF instead of positive reals. An operational semantics—not shown here due to space constraints—defines the behaviour of

the model compositionally [17]. For instance, the similar transition for U_r

$$U_r \xrightarrow{(request, r_{rq}\xi_4)}_\star U_t \tag{3}$$

can be composed with (2) to collect the behaviour for the reduced context (1):

$$S_w \underset{\{request\}}{\bowtie} U_r \xrightarrow{\left(request, \min(r_{rp}\xi_1, r_{rq}\xi_4)\right)}_\star S_l \underset{\{request\}}{\bowtie} U_t \tag{4}$$

which gives the rate at which one of the ξ_1 components in state S_w and one of the ξ_4 components in state U_r synchronise and subsequently become S_l and U_t, respectively. Again, one can write $S_l \underset{\{request\}}{\bowtie} U_t \in ds^\star(S_w \underset{\{request\}}{\bowtie} U_r)$. This transition follows from the following semantic rule:

$$\frac{E \xrightarrow{(\alpha, r_1(\xi))}_\star E' \qquad F \xrightarrow{(\alpha, r_2(\xi))}_\star F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha, r(\xi))}_\star E' \underset{L}{\bowtie} F'}, \ \alpha \in L,$$

$$r(\xi) = \frac{r_1(\xi)}{r^\star_\alpha(E,\xi)} \frac{r_2(\xi)}{r^\star_\alpha(F,\xi)} \min\left(r^\star_\alpha(E,\xi), r^\star_\alpha(F,\xi)\right)$$

where $r^\star_\alpha(E,\xi)$ denotes the *parametric apparent rate* of α in component E, computed by summing the (symbolic) rates at which the derivatives of E may perform α-activities. Transitions of the kind (2–4) induce a *parametric derivation graph* whose nodes are PEPA components and arcs give the action type and the parametric rate at which an activity may be performed. The parametric derivation graph whose nodes are elements of $ds^\star(red(System))$ gives the overall system's behaviour in terms of a symbolic CTMC.

Generating Functions and ODE model: With each transition of this graph is associated a *generating function* $\varphi_\alpha(\xi, l)$ where α is the action type and l is a *jump vector* of integers which records the discrete changes in the population levels of the components when the transition is taken. The vector l is computed as the difference between the two *indicator vectors*, denoted by $ind(\cdot)$, of the components involved in the transition. An element of the indicator vector is one if the corresponding local state appears in the component. For instance, the generating function of (4) is

$$\varphi_{request}(\xi, l) = \min(r_{rp}\xi_1, r_{rq}\xi_4), \tag{5}$$

where

$$\begin{aligned} l &= ind(S_l \underset{\{request\}}{\bowtie} U_t) - ind(S_w \underset{\{request\}}{\bowtie} U_r) \\ &= (0,1,0,0,1) - (1,0,0,1,0) = (-1,1,0,-1,1) \end{aligned}$$

The symbolic generating functions are sufficient to define an ODE model. Let $x_i(t)$ be the continuous real-valued function which approximates the time-course evolution of ξ_i. The ODE defining $x_i(t)$ is constructed as

$$\frac{dx_i(t)}{dt} = \sum_{l,\alpha} l_i \varphi_\alpha(x(t), l)$$

For instance, Equation (5) contributes the summand $\min\left(r_{rp}x_1(t), r_{rq}x_4(t)\right)$ to the expression for $dx_1(t)/dt$ and $dx_4(t)/dt$ and its negation to $dx_2(t)/dt$ and $dx_5(t)/dt$. Let $x(t)$ be the vector with elements $x_i(t)$, the ODE model will be compactly written as $dx(t)/dt = \sum_{l,\alpha} l\varphi_\alpha(x(t), l)$.

We have now seen the two extremes of PEPA semantics, one leading to a CTMC interpretation and the other to an ODE interpretation. Both are suitable under some sets of circumstances and unsuitable in others. However, the circumstances can have disadvantages for both approaches with respect to different aspects of the system. By moving to a hybrid approach, we can mitigate these disadvantages by choosing part of the system to be continuous and deterministic and other parts to be stochastic and discrete.

In the next section, we introduce the underlying semantic model that we will use to express the hybrid semantics. Transition-driven stochastic hybrid automata (TDSHA) embody both continuous and discrete stochastic behaviour (as well as instantaneous) and hence are suitable for the task.

III. Transition-driven Stochastic Hybrid Automata

Transition-Driven Stochastic Hybrid Automata (TDSHA) are a formalism introduced in [3] and they define a subclass of Piecewise Deterministic Markov Processes, (PDMPs) [6], [5], a well-studied class of Stochastic Hybrid Automata combining differential equations and Markovian jumps, developed for financial modelling. TDSHA emphasize transitions, which can be either discrete or continuous.

We will define the hybrid semantics of PEPA in terms of TDSHA, as they are more manageable than PDMPs. For this task, TDSHA will be enhanced with a synchronisation product, based on standard PEPA semantics [9], and a notion of hiding.

Definition III.1. A Transition-Driven Stochastic Hybrid Automaton (TDSHA) is a tuple $\mathcal{T} = (Q, \mathbf{X}, \mathcal{TC}, \mathcal{TS}, init, \mathcal{E})$, where:

- Q is a finite set of *control modes* and $\mathbf{X} = \{X_1, \ldots, X_n\}$ is a set of real valued *system variables*.[1] $Q \times \mathbb{R}^n$ is the hybrid state space.
- $\mathcal{E}$ is a finite set of event or action names, labelling transitions.
- $\mathcal{TC}$ is the *multiset* of *continuous transitions or flows*, whose elements η are 4-tuples $\eta = (q, \mathbf{s}, f, e)$, where $q \in Q$, $\mathbf{s}$ is a vector in $\mathbb{R}^n$ of size $|\mathbf{X}|$, $f : \mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz continuous function, and $e \in \mathcal{E}$. The elements of a tuple η are also written q_η, $\mathbf{s}_\eta$, f_η, e_η, respectively.
- $\mathcal{TS}$ is the *multiset* of *stochastic transitions*, whose elements are tuples of the form $\eta = (q_1, q_2, g, r, f, e)$, where q_1 is the *exit-mode*, q_2 is the *entry-mode* and $e \in \mathcal{E}$. Moreover, g is a first-order formula with free variables from $\mathbf{X}$, and r is the reset, a conjunction of formulae of the form $X' = \rho(\mathbf{X})$ for some variables in $\mathbf{X}$, with $\rho : \mathbb{R}^n \to \mathbb{R}$. Variables not appearing in

[1] The value of X_j after a change of mode is denoted by X'_j.

r are unmodified and the formula $true$ corresponds to the identity reset. Finally, $f : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is the locally Lipschitz continuous rate function giving the instantaneous probability of taking transition η. The elements of a tuple η are indicated by q_1^η, q_2^η, g_η, r_η, f_η, e_η, respectively.

- $init$ is a point $(q_0, \mathbf{x_0}) \in Q \times \mathbb{R}^n$, the initial state of the system.

The version of TDSHA introduced above differs from the one presented in [3] because the class of instantaneous discrete transitions is omitted. Such transitions, in fact, are not used in the definition of the hybrid semantics of PEPA, and we wanted to avoid the introduction of additional technical details that may distract the reader. Furthermore, in [3] a different product of TDSHA was used.

A. Dynamics of TDSHA

A TDSHA has two types of transitions. Continuous transitions represent flows. For each $\eta \in \mathcal{TC}$, $\mathbf{s}_\eta$ and f_η describe the flow: its effect on variable X_i is given by $\mathbf{s}_{\eta,i} \cdot f_\eta(\mathbf{X})$. Stochastic transitions, instead, happen at a specific rate describing an exponential distribution, and can change system variables according to their reset function, depending on the point at which the jump occurred. The dynamics of TDSHA can be defined similarly to PDMPs [6] in the following way.

Continuous transition: Within each mode $q \in Q$, the system follows the solution of a set of ODEs, constructed by combining the effects of the continuous transitions acting on q. More precisely, setting $\mathcal{TC}(q) = \{\eta \in \mathcal{TC} \mid q_\eta = q\}$, we define the ODEs in mode $q \in Q$ as

$$\frac{d\mathbf{X}}{dt} = \sum_{\eta \in \mathcal{TC}(q)} \mathbf{s}_\eta f_\eta(\mathbf{X}), \tag{6}$$

whose solution, starting from point $\mathbf{x_0}$ at time t_0, is denoted by $\phi_q(t, t_0, \mathbf{x_0})$. Existence and uniqueness of the solution is guaranteed by Lipschitz continuity of f_η.

Stochastic transitions: These are fired according to their rate. When this happens, the state of the system is reset according to the specified policy. Choice among several active transitions is performed probabilistically. Let $\mathcal{TS}(q, \mathbf{x}) = \{\eta \in \mathcal{TS} \mid q_1^\eta = q, g_\eta(\mathbf{x}) = true\}$ be the set of stochastic transitions active in $(q, \mathbf{x})$. We define the *global rate* in $(q, \mathbf{x})$ as $\lambda(q, \mathbf{x}) = \sum_{\eta \in \mathcal{TS}(q,\mathbf{x})} f_\eta(\mathbf{x})$. The *reset measure* R associates a distribution on the hybrid state space with each point $(q, \mathbf{x})$. $R(q, \mathbf{x}, \cdot)$ is a finite measure: from $(q, \mathbf{x})$ we reach point $(q_2^\eta, r_\eta(\mathbf{x}))$ with probability $f_\eta(\mathbf{x})/\lambda(q, \mathbf{x})$, for each $\eta \in \mathcal{TS}(q, \mathbf{x})$.

Traces: We can now define traces starting from the initial point $(q_0, \mathbf{x_0})$. We will use two sequences U_i^1, U_j^2 of independent random variables, uniformly distributed in $(0, 1)$. From $(q_0, \mathbf{x_0})$, the TDSHA follows the solution of the ODEs $\phi_{q_0}(t, 0, \mathbf{x_0})$ until the first stochastic jump. Its firing time T_1 has survival function $\mathbb{P}(T_1 > t) = e^{-\Lambda(q_0,t)}$, where the cumulative rate up to time t is $\Lambda(q_0, t) = \int_0^t \lambda(q_0, \phi_{q_0}(t, 0, \mathbf{x_0}))$ (the integral is defined due to Lipschitz continuity of all f_η). The firing time T_1 is defined by solving the equation $e^{-\Lambda(q_0,T_1)} = U_1^1$ [6]. The execution of a stochastic transition can change the current mode or the value of system variables, and the variable U_1^2 is used to select the target point $(q_1, \mathbf{x_1})$ from the (finite) distribution $R(q_0, \phi_{q_0}(T_1, 0, \mathbf{x_0}), \cdot)$. Then, the system restarts from such a point, following the flow $\phi_{q_1}(t, T_1, \mathbf{x_1})$ until the next stochastic jump. This process can be repeated to to provide a single trace.

Simulation: The definition of a trace gives a method to simulate an execution of a TDSHA and $\Lambda(q, t)$ can be computed by coupling $d\Lambda(q, t)/dt = \lambda(q, \mathbf{X}(t))$ to the system (6).

B. Composition of TDSHA

Since our mapping will be compositional, we need to compose two TDSHA. There are a number of ways to do this but here we reflect PEPA semantics, specifically rate calculation. We start by defining the apparent rate of action α in a mode q. Given a particular set of values for the variables, we consider all transitions from the mode and the (functional) rates at which they can happen.

Definition III.2. Let $\mathcal{T} = (Q, \mathbf{X}, \mathcal{TC}, \mathcal{TS}, init, \mathcal{E})$ be a TDSHA. The *apparent rate* of $\alpha \in \mathcal{E}$ at point $(q, \mathbf{x}) \in Q \times \mathbb{R}^n$ is defined as

$$\begin{aligned} \mathsf{r}_\alpha(q, \mathbf{x}) &= \sum \{f_\eta(\mathbf{x}) \mid \eta \in \mathcal{TC}(q), e_\eta = \alpha\} \\ &+ \sum \{f_\eta(\mathbf{x}) \mid \eta \in \mathcal{TS}(q, \mathbf{x}), e_\eta = \alpha\} \end{aligned}$$

Next, we define the synchronisation of two TDSHA with respect to a set L of actions. We assume disjointness of variables to simplify the overall treatment and this property will be satisfied by the definition of the hybrid semantics for PEPA. Also, we assume that the set $\mathcal{E}$ is partitioned into continuous and stochastic actions so that it is there are no synchronisations across types of transitions.

The actions in L are those which are synchronised on and need to be treated differently from the other actions. Given two continuous transitions $\eta_1 = (q_1, \mathbf{s}_1, f_1, \alpha)$ and $\eta_2 = (q_2, \mathbf{s}_2, f_2, \alpha)$ for TDSHA $\mathcal{T}_1$ and $\mathcal{T}_2$ with disjoint variables, the rate of their synchronisation, in the style of PEPA parametric semantics, is $\mathsf{rs}(\eta_1, \eta_2) =$

$$\frac{f_1(\mathbf{X_1})}{\mathsf{r}_\alpha(q_1, \mathbf{X_1})} \frac{f_2(\mathbf{X_2})}{\mathsf{r}_\alpha(q_2, \mathbf{X_2})} \min\{\mathsf{r}_\alpha(q_1, \mathbf{X_1}), \mathsf{r}_\alpha(q_2, \mathbf{X_2})\}.$$

The transition corresponding to their synchronisation is then $((q_1, q_2), \mathbf{s}_1 \oplus \mathbf{s}_2, \mathsf{rs}(\eta_1, \eta_2), \alpha)$, where $\mathbf{s}_1 \oplus \mathbf{s}_2$ is the *direct sum* of the two vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ (which can be thought here simply as concatenation of vectors).

Stochastic transitions are treated similarly. Guards and resets for synchronised actions are the conjunction of guards and resets of the two transitions.

Unsynchronised transitions from a mode (q_1, q_2) are simply all transitions of q_1 and of q_2.

Definition III.3. Let $\mathcal{T}_i = (Q_i, \mathbf{X}_i, \mathcal{TC}_i, \mathcal{TS}_i, init_i, \mathcal{E}_i)$ for $i = 1, 2$ be two TDSHA. Their *L-synchronised product*[2] with respect to the set of actions $L \subseteq \mathcal{E}$, $\mathcal{T}_1 \underset{L}{\bowtie} \mathcal{T}_2 = (Q, \mathbf{X}, \mathcal{TC}, \mathcal{TS}, init, \mathcal{E})$, is defined by

[2]We overload the cooperation operator for notational convenience.

- $Q = Q_1 \times Q_2$;
- $\mathbf{X} = \mathbf{X_1} \cup \mathbf{X_2}$;
- $\mathcal{TC} = \mathcal{TC}_s \cup \mathcal{TC}_u$, where

$$\mathcal{TC}_s = \big\{((q_1,q_2), \mathbf{s}_1 \oplus \mathbf{s}_2, \mathsf{rs}(\eta_1,\eta_2), \alpha) \mid \alpha \in L, \eta_i = (q_i, \mathbf{s}_i, f_i, \alpha) \in \mathcal{TC}_i, i = 1,2\big\},$$

$$\mathcal{TC}_u = \big\{((q_1,q_2), \mathbf{s}_1 \oplus \mathbf{0}, f_1, \alpha) \mid \alpha \notin L, (q_1, \mathbf{s}_1, f_1, \alpha) \in \mathcal{TC}_1, q_2 \in Q_2\big\} \cup \big\{((q_1,q_2), \mathbf{0} \oplus \mathbf{s}_2, f_2, \alpha) \mid \alpha \notin L, (q_2, \mathbf{s}_2, f_2, \alpha) \in \mathcal{TC}_2, q_1 \in Q_1\big\},$$

- $\mathcal{TS} = \mathcal{TS}_s \cup \mathcal{TS}_u$, where

$$\mathcal{TS}_s = \big\{((q_1,q_2),(q_1',q_2'), g_1 \wedge g_2, r_1 \wedge r_2, \mathsf{rs}(\eta_1,\eta_2), \alpha) \mid \alpha \in L, \eta_i = (q_i, q_i', f_i, g_i, r_i, \alpha) \in \mathcal{TS}_i, i = 1,2\big\},$$

$$\mathcal{TS}_u = \big\{((q_1,q_2),(q_1',q_2), g_1, r_1, f_1, \alpha) \mid \alpha \notin L, (q_1, q_1', g_1, r_1, f_1, \alpha) \in \mathcal{TS}_1, q_2 \in Q_2\big\} \cup \big\{((q_1,q_2),(q_1,q_2'), g_2, r_2, f_2, \alpha) \mid \alpha \notin L, (q_2, q_2', g_2, r_2, f_2, \alpha) \in \mathcal{TS}_2, q_1 \in Q_1\big\}$$

- $init = ((q_1,q_2), \mathbf{x}_\mathbf{0}^1 \oplus \mathbf{x}_\mathbf{0}^2)$ for $init_1 = (q_1, \mathbf{x}_\mathbf{0}^1)$ and $init_2 = (q_2, \mathbf{x}_\mathbf{0}^2)$.

The use of this definition will be illustrated with an example when we consider the mapping of PEPA.

We now define hiding applied to a TDSHA. In PEPA, hiding by a set of actions results in the actions in the set effectively being replaced by a special, distinct action, namely the *silent action* τ. We assume that synchronisation on τ cannot occur, so it can be used in both continuous and stochastic transitions.

Definition III.4. Let $\mathcal{T} = (Q, \mathbf{X}, \mathcal{TC}, \mathcal{TS}, init, \mathcal{E})$ be a TDSHA and $L \subseteq \mathcal{E}$. The hiding of L in $\mathcal{T}$, denoted by $\mathcal{T}/L$, is obtained by replacing all occurrences of actions in L in the transitions of $\mathcal{T}$ by the silent action τ.

IV. Mapping PEPA to TDSHA

We now have the definitions for a semantics of PEPA in terms of TDSHA. The semantic construction proceeds in two steps, in line with [4], [3]. First, each PEPA sequential component is translated into a TDSHA then these automata are combined together. We prove some properties of the hybrid semantics and consider various choices made in the mapping as well modelling decisions.

Clearly, some parts of the PEPA model should remain discrete and some should be continuous. To this end, some sequential derivatives will be counted in a discrete fashion (as in the integral NVF for the Markovian semantics) and the remainder will be treated as continuous (as in the fluid-flow semantics).

From a modelling point of view, we expect to have criteria to decide how to partition the sequential components and their derivatives, and we discuss this after the mapping. Technically, we identify some actions as continuous and some as discrete. Actions are the basic behavioral units in PEPA models, and we decide if their effect on the model is fluid or discrete. An action (apart from τ) is treated uniformly in the model, in the sense that it is always discrete or always continuous in the hybrid semantics. This assumption simplifies the overall presentation, as we do not need to specify the synchronisation of a fluid action with a discrete one.

Therefore we assume a partition of the set of actions $\mathcal{A}$ into discrete actions $\mathcal{A}_d$ and continuous actions $\mathcal{A}_c$

Example IV.1. Returning to the example presented in Section II, Consider

$$System \stackrel{def}{=} S_w[2] \underset{\{request\}}{\bowtie} U_r[100]$$

where there are two servers and 100 clients. The servers can perform the actions $\{request, log, break, fix\}$ and the clients $\{request, think\}$. Due to the nature of the interaction between servers and clients, and the large number of clients, we will make the actions $\{request, log, think\}$ continuous and the rest will be discrete. We will use Sys to denote $red(System)$, hence $Sys = S_w \underset{\{request\}}{\bowtie} U_r$.

Considering the whole system, we can then partition the derivative set in two sets. To do this formally, a few definitions are required.

Definition IV.1. Given a reduced PEPA component P, let the *set of sequential local states* of P be defined as

$$seq(P) = \{D \mid D \in ds(C), C \text{ is a sequential component in } P,\ P \text{ reduced}\}.$$

Definition IV.2. For a sequential PEPA component C, let $\chi(C) = pre(C) \cup post(C)$ for

$$pre(C) = \{\alpha \mid C' \xrightarrow{(\alpha, r(\xi))}_\star C \text{ and } C' \neq C\}$$
$$post(C) = \{\alpha \mid C \xrightarrow{(\alpha, r(\xi))}_\star C' \text{ and } C' \neq C\}$$

Discrete local states are those for which both entry and exit actions are discrete. They can also be involved in continuous actions, as far as these actions do not change the local state. For instance, a component like $C = (\alpha, r).C + (\beta, s).C1$, with $\alpha \in \mathcal{A}_c$ and $\beta \in \mathcal{A}_d$, can be treated as discrete, because α does not change C. All other local states are treated as continuous. Hence $seq(P)$ is partitioned into continuous local states P_c and discrete local states P_d as given in the following definition.

Definition IV.3. Given a reduced PEPA model P then

$$P_c = \{C \mid C \in seq(P), \chi(C) \cap \mathcal{A}_c \neq \emptyset\}$$
$$P_d = \{C \mid C \in seq(P), \chi(C) \subseteq \mathcal{A}_d\}$$

Example IV.2. For the example, $Sys_c = \{S_w, S_l, U_r, U_t\}$ and $Sys_d = \{S_b\}$ illustrating that the derivatives of a component can be treated differently.

We next move on considering the mapping of a sequential component to a TDSHA.

A. Mapping of a sequential component

First, we will motivate the mapping and then define it formally. Given a sequential PEPA component, let its derivative set be $\{D_1, \ldots, D_p, C_1, \ldots, C_m\}$ where the D_i are the discrete local states and the C_j are the continuous local states. The TDSHA will have a set of variables

$\mathbf{X}_C = \{X_1^C, \ldots, X_m^C\}$ which track the values for the continuous local states, namely $(X_1^C, \ldots, X_m^C) \in \mathbb{R}^m_{\geq 0}$.

The counts of the discrete local states can also be expressed as a vector of the form $(d_1, \ldots, d_p) \in \mathbb{N}^p$ but this vector will determine the modes in the TDSHA, and hence variables for these local states are not necessary.

PEPA is conservative in the sense that fresh components cannot be created, hence a single copy of a sequential component can only be in one of its local states at a point in time. With n copies of the components, we expect

$$\sum_{i=1}^{p} d_i + \sum_{j=1}^{m} X_j^C = n.$$

Defining $s = \sum_{j=1}^{m} X_j^C$, it is clear s must be in $\mathbb{N}$. The modes can then be defined as

$$Q_C^n = \{(d_1, \ldots, d_p, s) \in \mathbb{N}^{p+1} \mid \sum_{i=1}^{p} d_i + s = n\}.$$

Example IV.3. The derivative set of the sequential component S_w is $\{S_w,\ S_l,\ S_b\}$ with S_w and S_l continuous and S_b discrete. Since $n = 2$ the modes are $Q^2_{S_w} = \{(2,0), (1,1), (0,2)\}$ representing when both servers are broken, one is broken and none are broken, respectively. The variables are $\mathbf{X}_{S_w} = \{X^{S_w}, X^{S_l}\}$ which capture the proportion of servers able to react to requests and the proportion of servers logging, respectively.

In order to define continuous transitions $\mathcal{TC}_C$, namely those for which $\alpha \in \mathcal{A}_c$, we have to consider two subcases. First, we deal with those actions modifying continuous local states: they are defined in the same way in every mode (although they have no effect in some modes).

$$T_1 = \{(q, \mathbf{1}_j^C - \mathbf{1}_i^C, rX_i^C, \alpha) \mid C_i \xrightarrow{(\alpha, rX_i^C)}_\star C_j, q \in Q_C^n\}$$

We include a tuple for every distinct derivation tree of the parametric transition to take into account the valid PEPA sequential components such as $C \stackrel{def}{=} (\alpha, r).C_1 + (\alpha, r).C_1$, hence T_1 is a multiset. The function in the transition is rX_i^C and this is in line with the discussion of rates given after Equation 2. The indicator vector of local state C_j, $\mathbf{1}_j^C$, is defined as a vector of length $|ds(C)|$ equal to one in the position corresponding to variable X_j^C, and zero elsewhere.

The second case deals with continuous transitions looping on the same local state, and it depends explicitly on discrete modes.

$$T_2 = \{((\ldots, d_i, \ldots, s), \mathbf{0}, rd_i, \alpha) \mid C_i \xrightarrow{(\alpha, rd_i)} C_i\}$$

Example IV.4. The continuous transitions are as follows.

$$\begin{aligned} T_1 &= \{(q, (-1, 1), r_{rp} X^{S_w}, \mathit{request}) \mid q \in Q^2_{S_w}\} \\ &\cup \{(q, (1, -1), r_{lg} X^{S_l}, \mathit{log}) \mid q \in Q^2_{S_w}\} \end{aligned}$$

We have the following flows in all modes.

$$\begin{aligned} dX^{S_w}/dt &= -r_{rp} X^{S_w} + r_{lg} X^{S_l} \\ dX^{S_l}/dt &= r_{rp} X^{S_w} - r_{lg} X^{S_l} \end{aligned}$$

Note that in mode $(2,0)$, both servers are broken, hence $X^{S_w} + X^{S_l} = 0$ and neither transition can have an effect. Also note that $T_2 = \emptyset$.

When $\alpha \in \mathcal{A}_d$, the transitions are stochastic. There are four cases to consider to obtain $\mathcal{TS}_C$ distinguishing whether the source and target local states are discrete or continuous. In each case, as in the continuous case, we ensure there is a tuple for each distinct derivation tree of a parametric transition, again giving a multiset. In the case where both local states in a transition are discrete, we have

$$\begin{aligned} T_{DD} = \{&((\ldots, d_i, \ldots, d_j, \ldots, s), \\ &(\ldots, d_i - 1, \ldots, d_j + 1, \ldots, s), \\ &\mathit{true}, \mathit{true}, rd_i, \alpha) \mid D_i \xrightarrow{(\alpha, rd_i)}_\star D_j\}. \end{aligned}$$

This is the simplest case and simply involves decreasing the count of one discrete local state and increasing the count of another. The function in the transition is defined with respect to the mode definition.

Next if the source is discrete and the target is continuous, the following transitions are obtained.

$$\begin{aligned} T_{DC} = \{&((\ldots, d_i, \ldots, s), (\ldots, d_i - 1, \ldots, s + 1), \\ &\mathit{true}, (X_j^C)' = X_j^C + 1, rd_i, \alpha) \mid D_i \xrightarrow{(\alpha, rd_i)}_\star C_j\} \end{aligned}$$

In this case, the count of the discrete local state is decreased, s is increased and the continuous variable associated with the continuous local state is increased.

Example IV.5. In the example, $T_{DD} = \emptyset$ and T_{DC} is

$$\begin{aligned} \{&((2,0), (1,1), \mathit{true}, (X^{S_w})' = X^{S_w} + 1, 2r_{fx}, \mathit{fix}), \\ &((1,1), (0,2), \mathit{true}, (X^{S_w})' = X^{S_w} + 1, r_{fx}, \mathit{fix})\} \end{aligned}$$

The last two cases involve removing amounts from the continuous variables. It is not correct to simply remove an amount of one from the source local state variable X, as it can have a value of less than one and it would be assigned a negative value. The problem is that continuous flows distribute the mass among many variables that depend on the variable X, either directly or indirectly. In order to ensure non-negativity, we will reduce proportionally all such variables.

We first define the following relation on $\mathbf{X}$.

Definition IV.4. Let $\mathcal{R}$ be the reflexive, symmetric and transitive closure of $\{(X_i^C, X_j^C) \mid C_i \xrightarrow{(\alpha, rX_i^C)}_\star C_j\}$.

This relation pairs variables whose associated local states are connected by a path of continuous transitions, forwards or backwards.

Definition IV.5. Given the derivative set $\{C_1, \ldots, C_m\}$ of a sequential PEPA component C, the *set of variables (continuously) dependent on* X_i^C is

$$\gamma(X_i^C) = \{X_j^C \mid (X_i^C, X_j^C) \in \mathcal{R}\}$$

The total sum of variables in $\gamma(X_i^C)$ is denoted by

$$\Gamma(X_i^C) = \sum_{Y \in \gamma(X_i^C)} Y.$$

$$
\begin{array}{rcl}
\mathbf{X}_{System} & = & \{X^{S_w}, X^{S_l}, X^{U_r}, X^{U_t}\} \qquad init_{System} \;=\; \{q_0, (2,0,100,0)\} \qquad \mathcal{E}_{System} \;=\; \mathcal{A} \\
Q_{System} & = & \{q_0, q_1, q_2\} \;\text{ with }\; q_0 = (0,2,100),\; q_1 = (1,1,100),\; q_2 = (2,0,100) \\
\mathcal{TC}_{System} & = & \{(q_0, (-1,1,-1,1), f, request),\; (q_0, (1,-1,0,0), r_{lg} X^{S_l}, log),\; (q_0, (0,0,1,-1), r_{th} X^{U_t}, think), \\
 & & \;(q_1, (1,-1,1,-1), f, request),\; (q_1, (1,-1,0,0), r_{lg} X^{S_l}, log),\; (q_1, (0,0,1,-1), r_{th} X^{U_t}, think)\} \\
 & & \text{where } f(\mathbf{X}_{System}) = \min\{r_{rp} X^{S_w}, r_{rq} X^{U_r}\} \\
\mathcal{TS}_{System} & = & \{(q_2, q_1, true, T, 2r_{fx}, fix),\; (q_1, q_0, true, T, r_{fx}, fix),\; (q_1, q_2, true, R, r_{br} X^{S_w}, break), \\
 & & \;(q_0, q_1, true, R, r_{br} X^{S_w}, break)\} \;\text{ where } R \text{ is defined as before and } T = (X^{S_w})' = X^{S_w} + 1
\end{array}
$$

Figure 2. Example of TDSHA synchronisation

Notice that $Y - Y/\Gamma(X_i^C) \geq 0$ for all $Y \in \gamma(X_i^C)$ if $\Gamma(X_i^C) \geq 1$, and

$$\sum_{Y \in \gamma(X_i^C)} Y/\Gamma(X_i^C) = 1.$$

The remaining transition definitions are now given.

$$
\begin{array}{rcl}
T_{CD} & = & \{((\ldots, d_j, \ldots, s), (\ldots, d_j + 1, \ldots, s - 1), \\
 & & \quad true, R_i, rX_i^C, \alpha) \mid C_i \xrightarrow{(\alpha, rX_i^C)}_\star D_j\}
\end{array}
$$

where the reset is defined by

$$R_i = \bigwedge \{X' = X - X/\Gamma(X_i^C) \mid X \in \Gamma(X_i^C)\}.$$

In this case, mass is proportionally reduced from the continuous variables associated with the local states reachable by continuous transitions from the source transition, and the appropriate discrete local state is increased by one.

It is possible to have both continuous transitions and stochastic transitions between continuous local states, so

$$
\begin{array}{rcl}
T_{CC} & = & \{((d_1, \ldots, d_p, s), (d_1, \ldots, d_p, s), \\
 & & \quad true, R_i' \wedge R_{i,j}'', rX_i^C, \alpha) \mid C_i \xrightarrow{(\alpha, rX_i^C)}_\star C_j\}
\end{array}
$$

where

$$R_i' = \bigwedge \{X' = X - X/\Gamma(X_i^C) \mid X \in \gamma(X_i^C) \wedge X \neq X_i^C\}$$

$$R_{i,j}'' = \begin{cases} (X_j^C)' = X_j^C + 1 & X_C^j \notin \gamma(X_i^C) \\ (X_j^C)' = X_j^C - X_j^C/\Gamma(X_i^C) + 1 & X_C^j \in \gamma(X_i^C). \end{cases}$$

The choice of removing mass proportionally from all variables in $\gamma(X_i^C)$ is not the only possibility. Another approach would be to take as much from a variables as possible (until an amount of one is taken or it becomes zero) and then take more from variables that depend on X_i^C through continuous transitions until an amount of one is reached, starting first with those that directly depend on X_i^C and then those that are indirectly dependent. Our approach, however, captures better the idea that the continuous transitions immediately start to distribute mass around the variables in $\gamma(X_i^C)$. This can be viewed as nearly decomposability [13] where we assume that the fast portions of the state space approach steady state between slow transitions.

Example IV.6. For the server, T_{CC} is empty, and

$$
\begin{array}{rcl}
T_{CD} & = & \{((1,1), (2,0), true, R, r_{br} X^{S_w}, break), \\
 & & \quad ((0,2), (1,1), true, R, r_{br} X^{S_w}, break)\}
\end{array}
$$

where $\gamma(X^{S_w}) = \{X^{S_w}, X^{S_l}\}$ and

$$
\begin{array}{rcl}
R & = & (X^{S_w})' = X^{S_w} - X^{S_w}/(X^{S_w} + X^{S_l}) \\
 & \wedge & (X^{S_l})' = X^{S_l} - X^{S_l}/(X^{S_w} + X^{S_l})
\end{array}
$$

For all stochastic transitions, the guard is $true$. A guard is unnecessary because the only condition required is that there is no transition if the local state is zero. This condition is guaranteed because the rate will be zero if the local state is zero. We can now define the mapping.

Definition IV.6. Let $C[n]$ be a component in a PEPA model P. $\mathcal{T}_C^n = (Q_C^n, \mathbf{X}_C, \mathcal{TC}_C, \mathcal{TS}_C, init_C, \mathcal{E}_C)$ is the TDSHA associated with $C[n]$, with respect to the partitioning $\mathcal{A}_c, \mathcal{A}_d$ of actions and is defined by

- Q_C^n and $\mathbf{X}_C$ as described above
- $\mathcal{TC}_S = T_1 \cup T_2$
- $\mathcal{TS}_C = T_{DD} \cup T_{DC} \cup T_{CD} \cup T_{CC}$
- $\mathcal{E}_C = \mathcal{A}$
- $init_C$ is extracted from P.

Example IV.7. All the elements of the TDSHA for S_w have been defined except for the initial state which is $((0,2), X^{S_w} = 2 \wedge X^{S_l} = 0)$ or $((0,2), (2,0))$ meaning that both servers start in the state S_w.

The TDSHA for U_r is much simpler. It has a single mode $q' = (100)$, variables X^{U_r}, X^{U_t}, no stochastic transitions, initial state $(q', (100,0))$, $\mathcal{E}_{U_r} = \mathcal{A}$ and

$$
\begin{array}{rcl}
\mathcal{TC}_{U_r} & = & \{(q', (-1,1), r_{rq} X^{U_r}, request), \\
 & & \quad (q', (1,-1), r_{th} X^{U_t}, think)\}
\end{array}
$$

B. Mapping of a model

It is now straightforward to define the semantics of a PEPA model P recursively.

Definition IV.7. Let P be a PEPA model. Then $\mathcal{T}_P$ is the TDSHA defined recursively by

- $\mathcal{T}_P = \mathcal{T}_C^n$ for $P = C[n]$,
- $\mathcal{T}_{P/L} = \mathcal{T}_P / L$,
- $\mathcal{T}_{P_1 \underset{L}{\bowtie} P_2} = \mathcal{T}_{P_1} \underset{L}{\bowtie} \mathcal{T}_{P_2}$.

Example IV.8. See Figure 2 for $\mathcal{T}_{System} = \mathcal{T}_{S_w}^2 \underset{L}{\bowtie} \mathcal{T}_{U_r}^{100}$.

C. Properties of the hybrid semantics

Proposition IV.1. *Let P be a PEPA model and let $\mathcal{T}_P$ be its associated TDSHA for a partition $(\mathcal{A}_c, \mathcal{A}_d)$ of actions. Assuming a non-negative integral initial vector, then for each sequential component model C of P with multiplicity N_C, for all $t \geq 0$ and state (q, x) at time t,*

1) $(\sum_{i=1}^{p} d_i^C) + (\sum_{i=1}^{m} X_i^C) = N_C$ *where* d_i^C *are the values for* C *in* q,
2) *for each* d_i^C *in* q, $d_i^C \in \{0, 1 \ldots, N_C\}$,
3) *for each* X_i^C, $\Gamma(X_i^C) \in \{0, 1, \ldots, N_C\}$,
4) *for each* X_i^C, $X_i^C \in [0, N_C]$.

Proof sketch: 1) This is true in the initial state and all stochastic transitions remove a mass of 1 from one or more variables (since $\sum_{Y \in \gamma(X)} Y/\Gamma(X) = 1$ in the case of continuous local states) and increase another variable by 1, preserving the total. Continuous transitions leave $\sum_{i=1}^{m} X_i^C$ unchanged due to terms that cancel in the ODEs. 2) If $d_i^C = N_c$ then $d_j^C = s^C = 0$ for $j \neq i$ by (1) and no transition can increase it. If $d_i^C = 0$, the only transition that can reduce it has rate $r d_i^C$ and cannot fire. Transitions increment and decrement d_i^C by one. 3) This is true initially. $\Gamma(X_i^C)$ is conserved by continuous transitions due to terms that cancel. Stochastic transitions increment and decrement by one. 4) Similarly to (2), if $X_i^C = N_C$ it cannot be increased. If $X_i^C > 0$ then $\Gamma(X_i^C) \geq 1$ by (3), hence $X_i^C \geq X_i^C/\Gamma(X_i^C)$ and $(X_i^C)' \geq 0$. If $X_i^C = 0$ then $(X_i^C)' = 0$. ■

Lemma IV.1. *For a PEPA model* P, $r_\alpha^\star(red(P), \mathbf{x}) = \mathsf{r}_\alpha(q_P, \mathbf{x})$ *where* q_P *is the initial mode of* $\mathcal{T}_P$.

Proof sketch: By induction on the structure of P. ■

Next, we consider the two extremes that are possible for partitioning the action set: all continuous or all stochastic, and show their equivalence with two PEPA semantics.

Theorem IV.1. *Let* P *be a PEPA model and* $\mathcal{T}_P$ *its associated TDSHA for the fully continuous partition of actions* $\mathcal{A}_c = \mathcal{A}$. *Then,* $\mathcal{T}_P$ *is equivalent to the fluid-flow approximation of* P, *in the sense that the ODEs of the TDSHA coincides with the ODEs of the PEPA model.*

Proof sketch: We define a multiset of tuples $\mathcal{G}^\star(P)$ from the parametric transitions from P where the multiplicity is determined by the number of distinct derivations for a transitions as defined in [17]. The generating functions and hence ODEs can be expressed in terms of $\mathcal{G}^\star(P)$. We demonstrate a bijection between $\mathcal{G}^\star(P)$ and $\mathcal{TC}_P$ by structural induction taking into account that there is effectively only one mode. The bijection leads to the conclusion that the ODEs are the same. ■

Theorem IV.2. *Let* P *be a PEPA model and* $\mathcal{T}_P$ *its associated TDSHA for the fully discrete partition of actions* $\mathcal{A}_d = \mathcal{A}$. *Then* $\mathcal{T}_P$ *is a CTMC isomorphic to the population-based CTMC of* P.

Proof sketch: We show that the entries of the generator matrices for the two CTMCs are the same, and that there is a bjiection between states. The proof proceeds by induction on P. ■

D. Discussion

The partitioning of PEPA actions into discrete and continuous is a crucial choice which influences the performance and the quality of the hybrid approximation, and which is currently left to the modeller. We provide now some heuristics that can guide this choice.

Multiplicity: Components with small numbers fit better with discrete stochastic modelling, and those with large numbers, continuous deterministic modelling, hence actions can be partitioned by this criterion.

Rates: Actions that are fast have small changes in small periods of time hence they should be continuous. The opposite applies to slow actions, and hence they are suitable to be stochastic.

Efficiency of modelling: If an action being stochastic leads to a state space that is too large or a simulation that does not complete, it might be better treated as continuous.

Furthermore, note that the different choices for $\mathcal{A}_c$ and $\mathcal{A}_d$ lead to a lattice of models, with $\mathcal{A} = \mathcal{A}_c$ and $\mathcal{A} = \mathcal{A}_d$ as the extremal points of the lattice. This is a formal framework in which we can investigate the effect of specific partitioning on the model behavior. Intuitively, moving in the lattice by increasing the level of discreteness should result in models with an higher accuracy (relative to the original CTMC semantics) but less efficiency. A precise formalization of this idea is under investigation, and requires a formal notion of error. One aim of this analysis is to develop techniques to automatically suggest a (set of) "good" partitions to the modeller, obtaining the best trade-off between accuracy and efficiency. Another possibility can be the definition of dynamic partitioning policies, similarly to [3], although a formal treatment of error in this case seems more difficult.

Finally, it is possible to define a hybrid semantics in which there is only one mode, and both discrete and continuous component counts are represented by continuous variables. This semantics is equivalent to the one defined here. However, we chose the latter as it is more intuitive. The single mode semantics may give executional advantages, as it describes discrete modes implicitly, thus avoiding one source of combinatorial complexity.

V. Numerical Validation

We employ the case study presented in Section II to study the accuracy of the TDSHA and the cost of the analysis with respect to the fully stochastic simulation (in the following, referred to as simply the *stochastic simulation*) of the CTMC which underlies the PEPA model.

A. Performance Metrics

The validation concerns the steady-state equilibrium distribution of the number of servers that are broken, denoted by $\overline{X}^{S_b}$. Specifically, we are interested in the following metrics: $\mathbb{P}(\overline{X}^{S_b} = i)$, $i = 0, 1$. Notice that for $i = 0$ this corresponds to evaluating the steady-state probability that all servers are available. Although this information may be valuable in practical applications, it cannot be evaluated with differential analysis, which only gives an estimate for the expectation of the aforementioned distribution, i.e., $\sum_i i\, \mathbb{P}(\overline{X}^{S_b} = i)$.

Table I
VALIDATION: APPROXIMATION ERRORS AND RUNTIME COMPARISON.

Parameters	Errors		Runtimes (s)		
$(N_c, N_s, scale)$	$\overline{X}^{S_b}=0$	$\overline{X}^{S_b}=1$	H	S	S/H
(10, 2, 0.1)	1.82%	3.58%	267	3	1.2E-2
(100, 2, 0.1)	0.67%	1.35%	1099	38	3.5E-2
(300, 2, 0.1)	0.70%	3.42%	529	69	1.3E-1
(10, 6, 0.1)	6.44%	0.52%	352	3	7.0E-3
(100, 6, 0.1)	2.35%	0.89%	566	18	3.1E-2
(300, 6, 0.1)	2.82%	1.53%	317	25	8.0E-2
(10, 2, 10.0)	0.54%	0.96%	547	253	4.6E-1
(100, 2, 10.0)	0.08%	0.21%	827	2618	3.2E+0
(300, 2, 10.0)	0.80%	3.20%	252	5092	2.0E+1
(10, 6, 10.0)	2.49%	2.64%	485	154	3.1E-1
(100, 6, 10.0)	3.86%	1.39%	623	1298	2.1E+0
(300, 6, 10.0)	1.30%	1.14%	876	5112	5.8E+0
(10, 2, 100.0)	0.13%	0.35%	204	3186	1.6E+1
(100, 2, 100.0)	0.35%	1.24%	589	20344	3.4E+1
(300, 2, 100.0)	0.01%	0.06%	438	51682	1.2E+2
(10, 6, 100.0)	2.19%	0.96%	217	1100	5.1E+0
(100, 6, 100.0)	2.14%	1.81%	301	13207	4.4E+1
(300, 6, 100.0)	0.09%	3.98%	592	39956	6.7E+1

B. Set-up

We generated eighteen instances of the model by varying rate values and the population sizes of the clients and the servers, respectively denoted by N_c and N_s. The rates were of kind: $r_{rp} = scale \times 1000$, $r_{rq} = scale \times 100$, $r_{lg} = scale \times 2000$, $r_{th} = scale \times 10$, $r_{br} = 0.01$, $r_{fx} = 0.05$, where the factor $scale$ was varied across the instances. The rates of the discrete actions were kept fixed in order to study the behaviour of the hybrid simulation with respect to varying intensities of the actions which are modelled deterministically. In all cases the models where chosen such that their CTMCs could be solved numerically for the equilibrium distribution using the tools available in the *PEPA Eclipse Plug-in* (a Java implementation of a modelling environment for PEPA [16]). Therefore, the accuracy of the TDSHA could be gauged by calculating the relative percentage error of the average hybrid estimate with respect to the numerical solution.

Instead, the runtime of the hybrid simulation was compared against stochastic simulation. The cost of the numerical solution of the CTMC was not considered because we are interested in analysing large-scale models where explicit enumeration of the state space is not feasible. All simulations were conducted using the method of batch means (e.g., [14]). To eliminate as many sources of bias as possible, the hybrid simulator and the stochastic one used the same configuration parameters. The simulations were set up using the same batch length and were terminated when the 95% confidence intervals of all performance metrics were within 5% of their simulated averages.

The stochastic simulation of the CTMC was performed using the *PEPA Eclipse Plug-in*. The hybrid simulator was implemented in MATLAB [15]. This choice was motivated by the presence of rates in the model that are separated by some orders of magnitude. This lead to *stiff* ODE initial value problems [12], for which convenient solvers are not available in the *PEPA Eclipse Plug-in*. In this study, MATLAB was considered to be a reasonable framework for the development of our prototype. In particular, all the results reported in this section were obtained using the solver `ode15s`. Under these conditions, the differences in the runtimes between the hybrid simulator and stochastic one are also due to the different languages in which the tools are implemented. Importantly, we found that this set-up is rather conservative with regards to the speed-up obtained by hybrid simulation. Our experience with these tools—not entirely reported here due to space constraints—showed that an implementation in MATLAB of our stochastic simulator is about two orders of magnitude slower on average and that a similar speed-up in Java can be observed when comparing the numerical integrators for non-stiff solvers. Therefore, it is not unreasonable to expect some noticeable improvement of the hybrid simulation runtimes with a careful implementation of an ODE solver for stiff problems on the Java platform.

C. Results

The numerical results for the accuracy and the runtimes of the hybrid simulation are reported in Table I. The hybrid estimates agree very well with the numerical solution of the CTMC, with errors less than 7% in all cases and an average of 2.70%. A tighter termination criterion (i.e., 95% confidence intervals within 3% of the mean, which increased runtimes by 30%) for the hybrid simulation resulted in an average approximation error of 0.98%.

As for the comparison of the runtimes, this validation data set highlighted two distinct cases. In all model instances with $scale = 0.1$ stochastic simulation (column labelled with S) is faster than hybrid simulation (label H), in some cases by as many as three orders of magnitude. A different behaviour is observed for larger values of $scale$. For $scale = 10$, the model with parameters $N_c = 10$ and $N_s = 2$ shows comparable runtimes, however for larger population sizes of the clients hybrid simulation becomes more and more convenient. As $scale$ is further increased to 100, the speed-up of hybrid simulation is consistently better for any population size of the model's components.

These results may be interpreted as follows. Smaller values of $scale$ give rises to models in which the rates of the (discrete) activities of breakdown and repair are close to the other (continuous) activities in the system. This leads to relatively more frequent stochastic jumps, that is, to relatively shorter time periods during which the system evolves continuously. This increased stochasticity of the process is such that the stochastic simulation is faster, given the simplicity of the algorithm with respect to the ODE numerical integrator. Instead, when the discrete activity rates are separated further from the continuous ones, the system evolves continuously over a longer time period, during which the integration becomes more convenient than stochastic simulation. This is also confirmed by the fact that, for fixed $scale$ and N_s, the speed-up of hybrid simulation increases with N_c. This is because the actual client population sizes have little impact on the cost of the ODE integration. However the impact on the cost of stochastic simulation is quite remarkable, because it affects the effective rates of execution in the system—recall that the transition rates of the CTMC are in the form $r\,\mathbf{X}$, with $r \geq 0$, thus the larger $|\mathbf{X}|$ the faster the transition rate and the higher the frequency of stochastic events.

VI. Related work

In [3], [4], the authors define a hybrid semantics for stochastic Concurrent Constraint Programming (sCCP), associating to sCCP programs (stochastic) hybrid automata with a fixed [4] or a variable degree of discreteness [3]. Although the spirit of this work is similar, there are considerable differences between PEPA and sCCP, introducing non-trivial distinctions in the *definition* of the hybrid semantics, mainly in the update mechanism of discrete transitions. This was necessary to guarantee the preservation of conservation properties of PEPA models (cf. Proposition IV.1). A hybrid semantics has also previously been provided for Fluid Stochastic Petri Nets [18], [7]. The basic difference is that this formalism describes directly a hybrid system, while our semantics is derived from a PEPA model which has an original semantics in terms of a CTMC. In this context, different partitions of actions into discrete ones and continuous ones generate different hybrid systems for the *same* PEPA model. Furthermore, these hybrid systems can be seen as approximations of the original CTMC. In this sense, our work is also related to the field of hybrid simulation algorithms [11].

VII. Conclusions

We presented a semantics for PEPA in terms of stochastic hybrid automata, in which some local states are approximated as continuous, while others are kept discrete. Such partition policies can be obtained by choosing which actions are to be considered continuous and deterministic and which are to be treated as discrete and stochastic. By inducing a partition on the set of actions, the modeller can have access to models with varying degrees of stochasticity. At the extremes of this lattice we recover the fully stochastic semantics, when all actions are kept discrete, and its deterministic differential approximation, when all actions are made continuous. This approach does not impose constraints on the structure of the PEPA models amenable to this interpretation. Owing to this generality, it is suitable for implementation as a software tool, which is the focus of ongoing work.

From a practical standpoint, the main advantage of hybrid analysis is that it may provide accurate estimates of distributions in the original model (which cannot be computed with the differential model) orders of magnitude faster than full stochastic simulation. The case study discussed in this paper provided hints as to where the hybrid approach is particularly convenient.

We have not yet considered error bounds and this is an item of future research, both theoretically and through further case studies. It is clear that the choice of action partition will affect these bounds.

Another direction for future work is related to convergence properties. For the fluid approximation, it can be proved that, increasing the multiplicity of initial populations proportionally with a size parameter n, the (normalized) sequence of CTMCs converges to the solution of the fluid ODE [11]. We would like to establish a similar result for the hybrid context, along the lines of [2]. The main difficulties lie in dealing consistently with the new update mechanisms of discrete transitions. Additionally, the relationship between the fluid and hybrid semantics in terms of quality of approximation is under investigation.

Acknowledgements: This work has been partially supported by the EU-funded project SENSORIA, IST-2005-016004. Jane Hillston is supported by the EPSRC ARF EP/C543696/01. Vashti Galpin is supported by the EPSRC SIGNAL Project EP/E031439/1. Luca Bortolussi is supported by GNCS and FIRB LIBi. Part of this research was done while Mirco Tribastone was in the School of Informatics at the University of Edinburgh. We thank the reviewers for their helpful comments.

References

[1] M. Bernardo and R. Gorrieri. A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. *Theoretical Computer Science*, 202:1–54, 1998.

[2] L. Bortolussi. Limit behavior of the hybrid approximation of stochastic process algebras. In *Proceedings of ASMTA 2010*, 2010.

[3] L. Bortolussi and A. Policriti. Hybrid semantics of stochastic programs with dynamic reconfiguration. In *Proceedings of CompMod 2009*, 2009.

[4] L. Bortolussi and A. Policriti. Hybrid dynamics of stochastic programs. *Theoretical Computer Science*, 411:2052–2077, 2010.

[5] C.G. Cassandras and J. Lygeros, editors. *Stochastic Hybrid Systems*. CRC Press, 2007.

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

[7] M. Gribaudo and M. Telek. Fluid models in performance analysis. In M. Bernardo and J. Hillston, editors, *Formal Methods for Performance Evaluation (SFM 2007)*, LNCS 4486, pages 271–317. Springer, 2007.

[8] H. Hermanns. *Interactive Markov Chains*. Springer-Verlag, 2002.

[9] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[10] J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST 05)*, 2005.

[11] J. Pahle. Biochemical simulations: stochastic, approximate stochastic and hybrid approaches. *Briefings in Bioinformatics*, 10(1):53–64, 2009.

[12] L.F. Shampine and M.W. Reichelt. The Matlab ODE suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997.

[13] H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[14] W.J. Stewart. *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press, 2009.

[15] The Mathworks. Matlab 7.6.0, 2008.

[16] M. Tribastone, A. Duguid, and S. Gilmore. The PEPA Eclipse Plug-in. *Performance Evaluation Review*, 36(4):28–33, March 2009.

[17] M. Tribastone, S. Gilmore, and J. Hillston. Scalable differential analysis of a process algebra model. To appear in *IEEE Transactions on Software Engineering*.

[18] B. Tuffin, D.S. Chen, and K.S. Trivedi. Comparison of hybrid systems and fluid stochastic petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, 11:77–95, 2001.