UNE IMP EHE CWI QMIPS INR UER ZAR UTO

Quantitative
Modeling
In
Parallel
Systems

Formal Characterisation of Immediate Actions in SPA with Nondeterministic Branching

Holger Hermanns and Michael Rettelbach and Thorsten Weiß
University of Erlangen-Nürnberg, IMMD VII

D W1.T2.2b-4.v1

22 September 1995

Distribution Level: Public

Approved by: U. Herzog

Abstract Stochastic Process Algebras (SPA) are process algebras in which the duration of each activity is given by a random variable. If the stochastic aspect is restricted to Markovian, i.e. exponentially distributed durations, nice algebraic foundations are available. They include a formal semantics and an equational theory for *Markovian bisimulation*, a congruence that can be seen as a stochastic counterpart of strong bisimulation. This paper extends that theory with a stochastic notion of Milner's *observational congruence*. We enrich a basic SPA with *immediate actions* that happen instantaneously if enabled. For the enriched

The QMIPS Project: The QMIPS project is a collaborative research project supported by the CEC as ESPRIT-BRA project no 7269. It is being carried out by the following organisations: CWI (Amsterdam), EHEI (University of Paris V), Imperial College (London), INRIA (Sophia-Antipolis), University of Erlangen, University of Newcastle, University of Torino and University of Zaragoza.

calculus we will derive a sound and complete characterization of *Markovian observational congruence*, a conservative extension of both Markovian bisimulation and observational congruence. The usefulness of immediate actions together with their equational theory will be illustrated by means of an example. Additionally, we will discuss some implementation and modelling issues arising from our results.

1 Introduction

Process algebras — e.g. CSP [12], CCS [16], ACP [2] — have been developed as formal description techniques for systems of processes that run asynchronously in parallel and communicate with each other by synchronously exchanging messages. They have been shown to be well suited for the description and analysis of so-called 'reactive systems' like robot control systems, operating systems, communication protocols, etc.

The investigation of process algebras has led to comprehensive theories extending classical automata theory in several respects: They provide a *compositional description technique* by defining composition operators for building large and complex descriptions from smaller and simpler ones. Most important are the parallel composition operator for describing synchronized execution of concurrent processes and an operator for abstracting from actions that are considered irrelevant. Different descriptions of the same system can be compared by means of *equivalence relations*. Several notions of equivalence with different granularity have been formally defined and investigated. *Algebraic characterizations* of such equivalence relations yield equational laws that can be used for transforming system descriptions into equivalent ones.

In recent years, process algebras have been extended with stochastic variables in order to allow the modular specification of systems that are amenable for performance as well as dependability analysis. This is achieved by specifying durations of activities by random variables.

Following the concept of classical process algebras the system behaviour is described by an abstract language with an underlying state-transition graph representing the exact meaning of the process (*semantic model*). This graph captures the necessary information for functional verification as well as performance evaluation. If only exponentially distributed random variables are allowed, the graph can be easily transformed into a Continuous Time Markov Chain (*CTMC*) that is amenable to straightforward analysis of performance properties. The restriction to exponential distributions has another interesting implication: a powerful algebraic theory allows the comparison of different descriptions by means of equivalences. Algebraic characterizations of such equivalences have also been established. They allow the transformation of system descriptions into equivalent ones [9].

Even though a large class of random variables can be approximated by overlapping exponential distributions, the exclusion of non-exponential distributions is a clear limitation of SPA until now. A first step towards greater expressiveness is the inclusion of *immediate actions*, that happen instantaneously. They are useful to describe certain management and control activities that have no impact on the temporal behaviour of a system. The problem is that the existence of an underlying CTMC can no longer be guaranteed.

Some of the published SPA — like [8, 3] — make use of immediate actions. However, their treatment is not part of the algebraic concept. Both approaches define a formal semantics that defines transition graphs containing immediate transitions. In order to transform this graph into a CTMC an elimination procedure for immediate transitions is performed that is inspired by algorithms used for Generalized Stochastic Petri Nets [1].

This elimination procedure poses problems because it may change important properties of the graph. In general, it is impossible to relate the original graph to the graph that does not contain immediate transitions by means of a meaningful equivalence notion. This implies that compositionality gets lost as soon as the elimination is carried out.

In this paper we present an new, algebraic approach to the treatment of immediate actions in SPA. We define an equivalence relation, *Markovian observational congruence*, that captures the impact of immediate actions on the behaviour of a system formally. This relation turns out to be a congruence and can therefore be applied compositionally. We establish an algebraic characterization of this relation by means of equational laws.

This framework allows the application of immediate actions as a means for a modular specification of complex systems. They can be used to specify management and control activities inside the overall system. Apart from that, it is possible to derive new operators that make internal use of immediate actions. One example might be an operator for sequential composition of processes, where one process is started immediately after another process has terminated. The signaling mechanism that is necessary for this operation can be defined using immediate actions internally.

Besides enhancing modelling convenience, immediate actions do also extend the class of stochastic processes that can be specified. However, we can derive a CTMC without neglecting relevant information if a certain condition holds; as in the above example, immediate actions may only have an internal impact on the system. Otherwise, they are part of the system behaviour and can not be eliminated. Equational laws are our means to eliminate internal immediate actions out of components, without changing relevant properties of the system.

The paper is organized as follows. In the next two sections we will review some fundamental results of classical and Stochastic Process Algebras, respectively. Section 4 combines and extends these results towards a formal characterization of immediate actions. In Section 5 we demonstrate the applicability of this theory by means of a small example. Section 6 discusses implementation and modelling issues arising from our theoretical results. Section 7 concludes with a brief summary.

2 A basic calculus

In this Section, we survey a few basic results related to some notions of equivalence introduced by Park and Milner focussing on a small calculus consisting of action prefix, nondeterministic choice and termination. The presentation follows [16].

Let Act be a set of actions, ranged over by a, b, ..., together with a distinguished action $\tau \in$ Act. Informally τ actions are meant to be (almost) invisible for an external observer. The syntax of the basic calculus $\mathcal{L}$ is defined by the following BNF grammar:

$$P ::= \quad Stop \quad | \quad a.P \quad | \quad P + P$$

Terms generated by this syntax are called *sequential finite processes*, ranged over by P, Q, ... The semantics of this language is defined in an operational style mapping on (finite) labelled transition systems.

Definition 2.1 *With each process $P \in \mathcal{L}$ the semantics associates a labelled transition system ($\mathcal{L}$, P , $\longrightarrow$), where $\longrightarrow \subseteq \mathcal{L} \times \mathrm{Act} \times \mathcal{L}$ is the least relation satisfying the following* *rules:*

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P+Q \xrightarrow{a} Q'}$$

Usually the operational semantics defines labelled transition systems that are considered to be too concrete. For example the semantic model of $a.Stop + Stop$ and $Stop + a.Stop$ are *not* the same because their respective starting states differ syntactically. A successful approach to overcome this drawback relies on regarding equivalence classes of transition systems, that are not distinguishable for an external observer, as semantics of processes.

Definition 2.2 *A symmetric binary relation $\mathcal{B} \subseteq \mathcal{L} \times \mathcal{L}$ is a* strong bisimulation*, if $(P,Q) \in \mathcal{B}$ implies, for all a in* Act*,*

Whenever $P \xrightarrow{a} P'$ then, for some Q', $Q \xrightarrow{a} Q'$ and $(P',Q') \in \mathcal{B}$.

P and Q are strong bisimulation equivalent*, written $P \sim Q$, if $(P,Q) \in \mathcal{B}$ for some strong bisimulation $\mathcal{B}$.*

For many practical purposes, strong bisimulation equivalence is not sufficiently abstract. For this reason, Milner proposed a weaker notion of equivalence, in which the requirement that every a action of one process must be matched by an a action of the other is relaxed, as far as τ actions are concerned.

Definition 2.3 *The visibility of an action is filtered by a function* $\widehat{\ }: \mathrm{Act} \to \mathrm{Act} \cup \{\varepsilon\}$, *where $\widehat{a}$ returns a, except for $\widehat{\tau}$. In this case the empty word ε is returned. Furthermore*[1]

$$\begin{aligned} P \overset{a}{\Longrightarrow} P' \quad &:\Longleftrightarrow \\ &\exists P_1, P_2 \in \mathcal{L}:\ P \xrightarrow{\tau}{}^{*} P_1 \xrightarrow{a} P_2 \xrightarrow{\tau}{}^{*} P'. \end{aligned}$$

Definition 2.4 *A symmetric binary relation $\mathcal{B} \subseteq \mathcal{L} \times \mathcal{L}$ is a* weak bisimulation*, if $(P,Q) \in \mathcal{B}$ implies, for all a in* Act*,*

Whenever $P \xrightarrow{a} P'$ then, for some Q', $Q \overset{\widehat{a}}{\Longrightarrow} Q'$ and $(P',Q') \in \mathcal{B}$.

P and Q are weak bisimulation equivalent*, written $P \approx Q$, if $(P,Q) \in \mathcal{B}$ for some weak bisimulation $\mathcal{B}$.*

In order to have a proper notion of equality it is desirable that equivalent processes lead to equivalent behaviour if they are composed with an arbitrary context. This feature, known as the *congruence property*, allows the replacement of components of a large system by equivalent ones without touching the overall behavioural properties.

It turns out that strong bisimulation equivalence is a congruence, whereas weak bisimulation equivalence is not. Weakly bisimulation equivalent processes can lead to different behaviours if both are composed with a third process via a choice operator. Therefore, a variation of this equivalence has been proposed that fixes this problem.

Definition 2.5 *P and Q are observational congruent, written $P \approx^c Q$, if for all a in* Act*,*
(i) Whenever $P \xrightarrow{a} P'$ then, for some Q', $Q \overset{a}{\Longrightarrow} Q'$ and $P' \approx Q'$.
(ii) Whenever $Q \xrightarrow{a} Q'$ then, for some P', $P \overset{a}{\Longrightarrow} P'$ and $P' \approx Q'$.

[1] The relation $\xrightarrow{\tau}{}^{*}$ is the reflexive, transitive closure over $\xrightarrow{\tau}$

The above equivalences compare processes indirectly, by means of their underlying transition systems, i.e. semantic model. Apart from that, it is possible to characterize equality of processes syntactically by means of equational laws. A set of laws forms a *sound* axiomatisation of a congruence if *only* equivalent processes can be equated syntactically. The axiomatisation is *complete*, if *all* semantic equivalences can be proven syntactically. Sound and complete axiomatisations of the congruences $\sim$ and $\approx^c$ are known [16]:

Proposition 2.1 *The following equational laws form a sound and complete axiomatisation of $\sim$ for $\mathcal{L}$.*

$$\begin{array}{ll} \mathrm{A1} & P + Stop = P \\ \mathrm{A2} & P + (Q + R) = (P + Q) + R \\ \mathrm{A3} & P + Q = Q + P \\ \mathrm{A4} & P + P = P \end{array}$$

Proposition 2.2 *The following equational laws together with those of Proposition 2.1 form a sound and complete axiomatisation of $\approx^c$ for $\mathcal{L}$.*

$$\begin{array}{ll} \mathrm{A5} & P + \tau.P = \tau.P \\ \mathrm{A6} & a.(P + \tau.Q) + a.Q = a.(P + \tau.Q) \\ \mathrm{A7} & a.\tau.P = a.P \end{array}$$

3 Stochastic Process Algebras

The first intention of Stochastic Process Algebras has been to extend the concept of classical process algebras towards an integrated consideration of performance aspects [7]. Therefore a random variable is associated with every activity of a process, specifying its duration. This has been proven to be particularly successful if only exponentially distributed random variables are used. In the following Section we will survey the theoretical results for this area [9].

Deviating from the use of a in Section 2, we now assume that $a \in \mathrm{Act}_M$ determines only the *name* of an action, whereas a second parameter $\lambda \in \mathrm{R}^+$, called *rate*, determines the duration until the action is performed instantaneously. Informally, the duration will be exponentially distributed with mean $1/\lambda$. We will use λ, μ, ... to range over rates. Actions belonging to Act_M will be called *Markovian* actions.

The language $\mathcal{L}_M$ is defined by the following BNF grammar:

$$P \ ::= \ Stop \ \mid \ (a, \lambda).P \ \mid \ P + P \ \mid \ P \|_S P \ \mid \ P \backslash a$$

Apart from different prefix operators in $\mathcal{L}_M$ and $\mathcal{L}$ we have added two additional operators, Hiding and Parallel Composition. The new results in this paper will cover all these operators. However, the sublanguage of *sequential finite processes*, not containing Hiding and Parallel Composition, will (again) be sufficiently rich for explaining the key idea of this approach.

It is worth noticing that the nature of the operator "+" changes from nondeterministic choice towards a choice with implicit probabilities. This phenomenon is due to a race condition that is assumed to exist between concurrently enabled activities. For example, the process $(a, \lambda).P + (a, \mu).Q$ will enter the state P with probability $\lambda/(\lambda + \mu)$ whereas Q will be chosen with probability $\mu/(\lambda + \mu)$.

$$\frac{}{(a,\lambda).P \xrightarrow{a,\lambda,\varepsilon} P} \qquad \frac{P \xrightarrow{a,\lambda,w} P'}{P + Q \xrightarrow{a,\lambda,+_l w} P'} \qquad \frac{Q \xrightarrow{a,\lambda,w} Q'}{P + Q \xrightarrow{a,\lambda,+_r w} Q'}$$

$$\frac{P \xrightarrow{a,\lambda,w} P'}{P \parallel_S Q \xrightarrow{a,\lambda,\parallel_l w} P' \parallel_S Q} \; a \notin S \qquad \frac{Q \xrightarrow{a,\lambda,w} Q'}{P \parallel_S Q \xrightarrow{a,\lambda,\parallel_r w} P \parallel_S Q'} \; a \notin S$$

$$\frac{P \xrightarrow{a,\lambda,v} P' \quad Q \xrightarrow{a,\mu,w} Q'}{P \parallel_S Q \xrightarrow{a,\lambda\mu,(v\parallel w)} P' \parallel_S Q'} \; a \in S$$

$$\frac{P \xrightarrow{a,\lambda,w} P'}{P\backslash a \xrightarrow{\tau,\lambda,w} P'\backslash a} \qquad \frac{P \xrightarrow{a,\lambda,w} P'}{P\backslash b \xrightarrow{a,\lambda,w} P'\backslash b} \; a \neq b$$

Figure 1: Semantic rules for $\mathcal{L}_M$

Definition 3.1 *With each process $P \in \mathcal{L}_M$ the semantics associates a labelled transition system $(\mathcal{L}_M\ ,\ P\ ,\ \longrightarrow)$, where $\longrightarrow\ \subseteq \mathcal{L}_M \times (\mathrm{Act}_M \times \mathrm{R}^+ \times Lab) \times \mathcal{L}_M$ is the least relation satisfying the rules in Fig. 1, where $Lab := \{+_l, +_r, \|_l, \|_r, \|, (,)\}^*$ and $\varepsilon \in Lab$ is the empty word.*

This definition maps processes onto transition systems, where transitions are labelled with triples of action, rate and an additional third label for reasons of semantic consistency. This label distinguishes two transitions joining the same states with the same action names and rates, using the concept of *proved transitions* of [5]. For a detailed explanation of this label, as well as a precise motivation of all language operators the reader is invited to consult [8, 6].

As in the case of $\mathcal{L}$, this operational definition produces different transition systems for processes that exhibit the same behaviour, like $P+Q$ and $Q+P$. Moreover, the fact that the minimum of two exponential distributions is again an exponential distribution, given by the sum of the two rates, has an interesting implication. It can be concluded that the stochastic behaviour of processes like $(a, \lambda).Stop + (a, \mu).Stop$ and $(a, \lambda + \mu).Stop$ is not distinguishable. This observation has initiated the definition of equivalence relations based on the *probabilistic bisimulation* of Larsen and Skou [15].

First, we define the function γ, that computes the sum of the rates of all transitions labelled with action a, emanating from P, and leading to any process in a given destination set.

Definition 3.2 *$\gamma : \mathcal{L}_M \times \mathrm{Act}_M \times 2^{\mathcal{L}_M} \longrightarrow I\!R^+$ is defined as follows:*

$$\gamma(P, a, B) := \sum_{(\lambda, w) \in I} \lambda \ ,$$

where $I := \{(\lambda, w) \mid P \xrightarrow{a,\lambda,w} P' \wedge P' \in B\}$.

Definition 3.3 *An equivalence relation $\mathcal{B} \subseteq \mathcal{L}_M \times \mathcal{L}_M$ is a* Markovian bisimulation, *if $(P, Q) \in \mathcal{B}$ implies for arbitrary actions $a \in \mathrm{Act}_M$ and arbitrary equivalence classes*[2] *$C \in \mathcal{L}_M/_{\mathcal{B}}$ that $\gamma(P, a, C) = \gamma(Q, a, C)$.*

Two processes P and Q are Markovian bisimulation equivalent, *written $P \sim_M Q$, if $(P, Q) \in \mathcal{B}$ for some Markovian bisimulation $\mathcal{B}$.*

This notion of equivalence is the stochastic counterpart of strong bisimulation equivalence. It has been proven to be a congruence with respect to the operators of $\mathcal{L}_M$. Additionally, a sound and complete axiomatisation for (a superset) of $\mathcal{L}_M$ was presented in [9]. This axiomatisation allows an intuitive comparison of strong bisimulation for $\mathcal{L}$ with Markovian bisimulation for $\mathcal{L}_M$:

Proposition 3.1 *The following equational laws form a sound and complete axiomatisation of $\sim_M$ for sequential finite processes of $\mathcal{L}_M$.*

A0	$(a, \lambda).P + (a, \mu).P = (a, \lambda + \mu).P$
A1	$P + \mathit{Stop} = P$
A2	$P + (Q + R) = (P + Q) + R$
A3	$P + Q = Q + P$

In Proposition 3.1 the law A0 embodies the summation property of the minimum of exponential distributions mentioned above. A0 has replaced A4 of Proposition 2.1, the idempotence of Choice in the classical calculus. All other laws remain unchanged.

Following the concept described in Section 2 the question arises whether a counterpart of observational congruence can also be defined. Unfortunately, this turns out to be impossible. Relaxing the observability of invisible actions (named τ) does not work in the stochastic setting, because only the name of the action is invisible. The stochastic behaviour of an action (τ, λ) remains observable for an observer. Consider a process $(a, \mu).(\tau, \lambda).\mathit{Stop}$. As the sequence of two exponentially distributed phases is no longer exponentially distributed, it is impossible to find a rate ν such that $(a, \nu).\mathit{Stop}$ is equivalent to the above process with respect to their random distributions.

Relaxing the observability of sequences of invisible actions such that only mean durations are observable is one idea to tackle this problem. *Weak Isomorphism* is a notion of equivalence that allows the amalgamation of certain τ sequences without affecting the integrity of steady state measures [11]. Although weak isomorphism is not a congruence it is preserved by some operators and can therefore be applied compositionally in appropriate circumstances.

[2] If $\mathcal{S}$ is an equivalence relation over $\mathcal{X}$ we denote by $\mathcal{X}/\mathcal{S}$ the set of equivalence classes of $\mathcal{X}$ induced by $\mathcal{S}$.

4 Immediate actions

$$
\begin{array}{lll|lll}
\dfrac{}{(a,\lambda).P \xrightarrow{a,\lambda,\varepsilon} P} & & & \dfrac{}{\underline{a}.P \overset{\underline{a}}{\dashrightarrow} P} & & \\[2ex]
\dfrac{P \xrightarrow{a,\lambda,w} P'}{P+Q \xrightarrow{a,\lambda,+_l w} P'} & & & \dfrac{P \overset{\underline{a}}{\dashrightarrow} P'}{P+Q \overset{\underline{a}}{\dashrightarrow} P'} & & \\[2ex]
\dfrac{Q \xrightarrow{a,\lambda,w} Q'}{P+Q \xrightarrow{a,\lambda,+_r w} Q'} & & & \dfrac{Q \overset{\underline{a}}{\dashrightarrow} Q'}{P+Q \overset{\underline{a}}{\dashrightarrow} Q'} & & \\[2ex]
\dfrac{P \xrightarrow{a,\lambda,w} P'}{P \parallel_S Q \xrightarrow{a,\lambda,\parallel_l w} P' \parallel_S Q} & & a \notin S & \dfrac{P \overset{\underline{a}}{\dashrightarrow} P'}{P \parallel_S Q \overset{\underline{a}}{\dashrightarrow} P' \parallel_S Q} & & \underline{a} \notin S \\[2ex]
\dfrac{Q \xrightarrow{a,\lambda,w} Q'}{P \parallel_S Q \xrightarrow{a,\lambda,\parallel_r w} P \parallel_S Q'} & & a \notin S & \dfrac{Q \overset{\underline{a}}{\dashrightarrow} Q'}{P \parallel_S Q \overset{\underline{a}}{\dashrightarrow} P \parallel_S Q'} & & \underline{a} \notin S \\[2ex]
\dfrac{P \xrightarrow{a,\lambda,v} P' \quad Q \xrightarrow{a,\mu,w} Q'}{P \parallel_S Q \xrightarrow{a,\lambda\mu,(v\parallel w)} P' \parallel_S Q'} & & a \in S & \dfrac{P \overset{\underline{a}}{\dashrightarrow} P' \quad Q \overset{\underline{a}}{\dashrightarrow} Q'}{P \parallel_S Q \overset{\underline{a}}{\dashrightarrow} P' \parallel_S Q'} & & \underline{a} \in S \\[2ex]
\dfrac{P \xrightarrow{a,\lambda,w} P'}{P \backslash a \xrightarrow{\tau,\lambda,w} P' \backslash a} & & & \dfrac{P \overset{\underline{a}}{\dashrightarrow} P'}{P \backslash \underline{a} \overset{\underline{\tau}}{\dashrightarrow} P' \backslash \underline{a}} & & \\[2ex]
\dfrac{P \xrightarrow{a,\lambda,w} P'}{P \backslash b \xrightarrow{a,\lambda,w} P' \backslash b} & & a \neq b & \dfrac{P \overset{\underline{a}}{\dashrightarrow} P'}{P \backslash \underline{b} \overset{\underline{a}}{\dashrightarrow} P' \backslash \underline{b}} & & \underline{a} \neq \underline{b}
\end{array}
$$

Figure 2: Semantic rules for $\mathcal{L}_{IM}$

In this Section we will enrich the language $\mathcal{L}_M$ with immediate actions. This implies a semantic model that can not be directly transformed into a Continuous Time Markov Chain, because it may contain immediate transitions. The (partial) solution we develop here is an algebraic one. We develop an equational theory that allows the elimination of *invisible immediate* actions, under certain restrictions, without changing any relevant property of the system.

We introduce the set of immediate actions Act_I, ranged over by $\underline{a}$, $\underline{b}$, $\ldots$, disjoint from Act_M the set of *Markovian* actions known from Section 3. The union of Act_I and Act_M will be denoted Act_{IM}, ranged over by α, β, $\ldots$. We allow synchronization of immediate actions as well as Markovian actions (i.e. $S \subseteq \mathrm{Act}_{IM}$), but *not* between Markovian and immediate actions. The language $\mathcal{L}_{IM}$ is defined by the following BNF grammar:

$$
\begin{array}{lllllll}
P & ::= & \mathit{Stop} & \mid & (a,\lambda).P & \mid & \underline{a}.P \quad \mid \\
 & & P+P & \mid & P \| _S P & \mid & P \backslash \alpha
\end{array}
$$

In this calculus the behaviour of the choice operator is twofold. Choices between Markovian actions still imply branching probabilities, whereas choices between immediate action are truly *non-deterministic*. In addition, if both types of actions are involved in a choice, the immediate action is prioritised, because the probability that the Markovian duration finishes before the immediate one is zero.

Definition 4.1 *With each process $P \in \mathcal{L}_{IM}$ the semantics associates a labelled transition systems* $(\mathcal{L}_{IM}, P, \longrightarrow, \dashrightarrow)$, *where* $\longrightarrow \subseteq \mathcal{L}_{IM} \times \mathrm{Act}_M \times \mathrm{R}^+ \times Lab \times \mathcal{L}_{IM}$ *and* $\dashrightarrow \subseteq \mathcal{L}_{IM} \times \mathrm{Act}_I \times \mathcal{L}_{IM}$ *are the least relations satisfying the rules in Fig. 2, where* $Lab := \{+_l, +_r, \|_l, \|_r, \|, (,)\}^*$ *and* $\varepsilon \in Lab$ *is the empty word.*

The key idea for a proper treatment of the two action types is to adopt the concept of observational congruence for immediate actions, whereas Markovian bisimulation remains the basic notion of equivalence for Markovian actions. This intuition will be formalized in Definition 4.2. Of course, the interrelation between both types of actions has to be captured as well.

4.1 Weak Markovian Bisimulation

Let us consider a process like $(a, \lambda).P + \underline{\tau}.Q$. The priority of the immediate action implies that an arbitrary *Markovian* left part (like $(b, \mu).R$) of the choice does not affect the behaviour of the choice. More generally: Markovian bisimulation must only be checked for processes that do not possess an immediate $\underline{\tau}$ transition. These processes belong to $\Upsilon := \{P \mid \nexists P' : P \overset{\tau}{\dashrightarrow} P'\}$. In Section 4.3 we will see why it is necessary to postulate this priority only for invisible immediate $\underline{\tau}$ instead of arbitrary immediate $\underline{a}$.

Definition 4.2 *An equivalence relation $\mathcal{B} \subseteq \mathcal{L}_{IM} \times \mathcal{L}_{IM}$ is a* weak Markovian bisimulation, *if $(P, Q) \in \mathcal{B}$ implies for all equivalence classes $C \in \mathcal{L}_{IM}/\mathcal{B}$, and for all processes $P' \in \mathcal{L}_{IM}$:*
(i) For all immediate actions $\underline{a} \in \mathrm{Act}_I$ it holds that whenever $P \overset{a}{\dashrightarrow} P'$ then, for some $Q' \in \mathcal{L}_{IM}$, $Q \overset{\widehat{a}}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{B}$, where $\Longrightarrow$ is defined analogous to $\Longrightarrow$ (Def. 2.3).
(ii) For all Markovian actions $a \in \mathrm{Act}_M$ it holds that whenever $P \in \Upsilon$ then, for some $\overline{Q} \in \Upsilon$, $Q \overset{\tau}{\dashrightarrow}{}^* \overline{Q}$ and $\gamma(P, a, C) = \gamma(\overline{Q}, a, C)$.

Markovian observational equivalent, *written $P \approx_M Q$, if $(P, Q) \in \mathcal{B}$ for some weak Markovian bisimulation $\mathcal{B}$, in other words:*

$$\approx_M := \bigcup \{ \mathcal{B} \mid \mathcal{B} \textit{ is a weak Markovian bisimulation} \}$$

Weak Markovian bisimulation merges weak bisimulation for immediate actions *(i)* with Markovian bisimulation for Markovian actions *(ii)*. Because of the priority of $\underline{\tau}$ the latter need not be checked if $P \notin \Upsilon$. Furthermore, if $P \in \Upsilon$, it is sufficient that Q can invisibly and immediately descend to a process $\overline{Q} \in \Upsilon$ that is Markovian bisimulation equivalent to P.

Proposition 4.1 $\approx_M$ *is the largest weak Markovian bisimulation.*

For the proof of this proposition, as well as all the others in this section, the interested reader is referred to [19].

In order to allow the replacement of subsystems by equivalent ones it is necessary that our equivalence satisfies the congruence property for all operators of the language. As with weak bisimulation equivalence in the classical calculus, $\approx_M$ is respected by all language operators *except* the choice operator:

Proposition 4.2 *Let $P \in \mathcal{L}_{IM}$ and $Q \in \mathcal{L}_{IM}$. $P \approx_M Q$ implies*

$$\begin{array}{lll} (1) & \forall a \in \mathrm{Act}_M\, \forall \lambda \in \mathrm{R}^+ : & (a,\lambda).P \approx_M (a,\lambda).Q \\ (2) & \forall \underline{a} \in \mathrm{Act}_I : & \underline{a}.P \approx_M \underline{a}.Q \\ (3) & \forall R \in \mathcal{L}_{IM}\, \forall S \subseteq \mathrm{Act}_{IM} : & P\|_S R \approx_M Q\|_S R \\ (4) & \forall \alpha \in \mathrm{Act}_{IM} : & P\backslash\alpha \approx_M Q\backslash\alpha \end{array}$$

In order to illustrate that $\approx_M$ in not a congruence with respect to choice we consider the following counterexample. By Definition of $\approx_M$ it is obvious that

$$\underline{\tau}.(a,\lambda).Stop \;\approx_M\; (a,\lambda).Stop$$

holds. Supposing that $\approx_M$ is a congruence with respect to choice, we can conclude that

$$\underbrace{\underline{\tau}.(a,\lambda).Stop + (b,\mu).Stop}_{P} \;\approx_M\; \underbrace{(a,\lambda).Stop + (b,\mu).Stop}_{Q}$$

must also hold. In P the transition labelled (b,μ) is not considered in Definition 4.2(i) because $P \notin \Upsilon$. Thus we have

$$P \;\approx_M\; \underline{\tau}.(a,\lambda).Stop \;\approx_M\; (a,\lambda).Stop,$$

which leads to, using $P \approx_M Q$:

$$\begin{array}{l} (a,\lambda).Stop \;\approx_M\; \underline{\tau}.(a,\lambda).Stop \;\approx_M \\ \underbrace{\underline{\tau}.(a,\lambda).Stop + (b,\mu).Stop}_{P} \;\approx_M\; \underbrace{(a,\lambda).Stop + (b,\mu).Stop}_{Q} \end{array}$$

But obviously the processes $(a,\lambda).Stop$ and $(a,\lambda).Stop + (b,\mu).Stop$ are not equivalent. Thus the assumed congruence property turns out to be false. Following Milner, we therefore define an equivalence in order to overcome this difficulty.

4.2 Markovian Observational Congruence

Definition 4.3 *P and Q are Markovian observational congruent, written $P \approx^c_M Q$, if for all Markovian actions $a \in \mathrm{Act}_M$, for all immediate actions $\underline{a} \in \mathrm{Act}_I$, for all equivalence classes $C \in \mathcal{L}_{IM}/\approx_M$, and for all processes $P', Q' \in \mathcal{L}_{IM}$:*
(i) Whenever $P \in \Upsilon$, then $\gamma(P,a,C) = \gamma(Q,a,C)$.
(ii) Whenever $Q \in \Upsilon$, then $\gamma(Q,a,C) = \gamma(P,a,C)$.
(iii) Whenever $P \overset{\underline{a}}{\dashrightarrow} P'$, then for some $Q' \in \mathcal{L}_{IM}$, $Q \overset{\underline{a}}{\Longrightarrow} Q'$ and $P' \approx_M Q'$.
(iv) Whenever $Q \overset{\underline{a}}{\dashrightarrow} Q'$, then for some $P' \in \mathcal{L}_{IM}$, $P \overset{\underline{a}}{\Longrightarrow} P'$ and $P' \approx_M Q'$.

Proposition 4.3 $\approx^c_M$ *is an equivalence relation.*

Lemma 4.1 *Let* $P \in \mathcal{L}_{IM}$ *and* $Q \in \mathcal{L}_{IM}$.

$$P \approx^c_M Q \Longrightarrow P \approx_M Q$$

The converse of this Lemma is not true. A counterexample is: *Stop* $\approx_M$ $\underline{\tau}$.*Stop*, but *Stop* $\not\approx^c_M$ $\underline{\tau}$.*Stop*.

The difference between $\approx_M$ and $\approx^c_M$ is the treatment of $\underline{\tau}$ transitions exiting the starting state. If $P \approx^c_M Q$ holds, then either both or neither of them possesses an initial $\underline{\tau}$-transition. In contrast, $\approx_M$ relaxes the observation of initial $\underline{\tau}$ transitions. This difference is clarified in the next Proposition.

Proposition 4.4 *Let* $P \in \mathcal{L}_{IM}$ *and* $Q \in \mathcal{L}_{IM}$.

$$\begin{array}{ll} \text{(a)} & P \approx_M Q \iff \\ & P \approx^c_M Q \;\vee\; P \approx^c_M \underline{\tau}.Q \;\vee\; \underline{\tau}.P \approx^c_M Q \\ \text{(b)} & P \approx_M Q \wedge P \in \Upsilon \wedge Q \in \Upsilon \Longrightarrow P \approx^c_M Q \end{array}$$

In contrast to $\approx_M$, Markovian observational congruence turns out to be a proper notion of equality, because it is preserved by all algebraic contexts, including the choice operator.

Proposition 4.5 $\approx^c_M$ *is a congruence with respect to all operators of* $\mathcal{L}_{IM}$, *i.e.* $P \approx^c_M Q$ *implies:*

$$\begin{array}{lll} (1) & \forall a \in \mathrm{Act}_M \, \forall \lambda \in \mathrm{R}^+ : & (a,\lambda).P \approx^c_M (a,\lambda).Q \\ (2) & \forall \underline{a} \in \mathrm{Act}_I : & \underline{a}.P \approx^c_M \underline{a}.Q \\ (3) & \forall R \in \mathcal{L}_{IM} : & P + R \approx^c_M Q + R \\ (4) & \forall \alpha \in \mathrm{Act}_{IM} : & P \backslash \alpha \approx^c_M Q \backslash \alpha \\ (5) & \forall R \in \mathcal{L}_{IM} \, \forall S \subseteq \mathrm{Act}_{IM} : & P \|_S R \approx^c_M Q \|_S R \end{array}$$

4.3 Equational theory for Markovian Observational Congruence

Hitherto we have studied congruence properties of Markovian observational equivalence and congruence. We shall now focus on a sound and complete set of equational laws for $\approx^c_M$ that makes it possible to derive equality of two descriptions on the syntactic level. As a first step we will investigate the subclass of sequential finite processes, referred to as $\mathcal{L}^{seq}_{IM}$, which will subsequently be extended towards finite processes, i.e. full $\mathcal{L}_{IM}$.

It will be shown in the sequel that the laws of Figure 3 form a sound and complete axiomatisation of $\approx^c_M$ for sequential finite processes. These laws make clear that Markovian observational congruence is a conservative extension of both Markovian bisimulation equivalence, for the Markovian part of the language, and observational congruence, for the immediate part. Remark that A0 to A4 are exactly the laws for Markovian bisimulation equivalence, whereas A1 to A7 are essentially the laws for observational congruence, only A4 is typically phrased as $P + P = P$. It is easy to show that these two versions of *idempotence* are interchangeable in the classical calculus. However, in our calculus the usual law is *not*

A0	$(a,\lambda).P + (a,\mu).P = (a,\lambda+\mu).P$	A1	$P + \mathit{Stop} = P$
A2	$P + (Q + R) = (P + Q) + R$	A3	$P + Q = Q + P$
A4	$\underline{a}.P + \underline{a}.P = \underline{a}.P$	A5	$P + \underline{\tau}.P = \underline{\tau}.P$
A6	$\underline{a}.(P + \underline{\tau}.Q) + \underline{a}.Q = \underline{a}.(P + \underline{\tau}.Q)$	A7	$\underline{a}.\underline{\tau}.P = \underline{a}.P$
A8	$(a,\lambda).\underline{\tau}.P = (a,\lambda).P$	A9	$(a,\lambda).P + \underline{\tau}.Q = \underline{\tau}.Q$

Figure 3: Axioms of $\approx^c_M$ for sequential finite processes

valid because this would contradict our intention for the Markovian part of the language, captured by A0.

The laws A8 and A9 relate Markovian to immediate actions, where A8 is a simple adaptation of A7. The treatment of both types of actions appearing in a choice context is captured by A9. The immediate action has priority over the timed one, if the first is invisible. The invisibility condition assures that the immediate action will happen under every circumstance. Otherwise, if Definition 4.2 were relaxed such that also $(a,\lambda).P + \underline{b}.Q = \underline{b}.Q$ ($\widetilde{\text{A9}}$) holds, the occurrence of $\underline{b}$ can be prevented by forcing synchronization with a $\underline{b}$ partner that does not exist. In this case the right part can perform an (a,λ) transition whereas the left one can not. Even though synchronization is not expressible with sequential finite processes, Def. 4.2 only implies A9 but not $\widetilde{\text{A9}}$ in order to allow an extension towards finite processes. For sequential (or unsynchronized) finite processes we can define a coarser congruence than $\approx^c_M$ where A9 is replaced by $\widetilde{\text{A9}}$, see [19].

Proposition 4.6 *The laws presented in Figure 3 are sound with respect to* $\approx^c_M$.

In order to show that the above set of laws $\mathcal{A} := \{\text{A0},\ldots,\text{A9}\}$ is complete, we first define a normal form and show that each sequential finite process can be transformed into a normal form by application of the laws. Afterwards we show that two normal forms can be transformed into each other if they are Markovian observational congruent, from which the completeness result follows as Corollary 4.1. We use $\mathcal{M} \vdash \phi$ to denote derivability of ϕ by application of equational laws in $\mathcal{M}$. Since the choice operator is commutative and associative, we abbreviate $P_{k_1} + P_{k_2} \ldots + P_{k_n}$ by $\sum_{k\in\{k_1\ldots k_n\}} P_k$.

Definition 4.4 *A process P is in* normal form, *if:*

$$P \equiv \mathit{Stop} \;\vee\; P \equiv \sum_{i\in I} (a_i,\lambda_i).P_i + \sum_{j\in J} \underline{b}_j.P_j$$

under the following conditions

1. *every* P_i *and* P_j *is in normal form.*
2. $a_n = a_m \wedge n \neq m \implies P_m \not\mathrel{\hat{=}} P_n$
3. $\underline{b}_k = \underline{b}_l \wedge k \neq l \implies P_k \not\mathrel{\hat{=}} P_l$
4. $\forall P_k \forall R \in \mathcal{L}_{IM}^{seq} : P_k \not\mathrel{\hat{=}} \underline{\tau}.R$
5. $\left(\exists j \in J : \underline{b}_j = \underline{\tau}\right) \implies I = \emptyset$
6. $P \overset{\underline{b}}{\Longrightarrow} P' \overset{\underline{\tau}}{\dashrightarrow} \widetilde{P} \;\vee\; P \overset{\underline{\tau}}{\dashrightarrow} P' \overset{\underline{b}}{\Longrightarrow} \widetilde{P} \implies$
$\nexists P_j : P \overset{\underline{b}}{\dashrightarrow} P_j \wedge P_j \mathrel{\hat{=}} \widetilde{P}$

where the relation $\hat{=}$ *is defined by:*

$$P \mathrel{\hat{=}} Q :\iff \{\mathrm{A2}, \mathrm{A3}\} \vdash P = Q$$

.

Lemma 4.2 *For every sequential finite process* P *there is a process* $\underline{P}$ *in normal form, with* $\mathcal{A} \vdash P = \underline{P}$.

Proposition 4.7 *If* $\underline{P}$ *and* $\underline{Q}$ *are in normal form, then the following is true:*

$$\underline{P} \approx_M^c \underline{Q} \implies \underline{P} \mathrel{\hat{=}} \underline{Q}$$

The proof is by induction on the sum of depths of nested prefixes in P and Q.

Corollary 4.1 *The set* $\mathcal{A}$ *forms a sound and complete axiomatisation of* $\approx_M^c$ *for sequential finite processes, i.e.* $\forall P, Q \in \mathcal{L}_{IM}^{seq} : P \approx_M^c Q \implies \mathcal{A} \vdash P = Q$.

Proof: The soundness was shown in Proposition 4.6. Let $P \approx_M^c Q$. By Lemma 4.2, we can find normal forms $\underline{P}$ and $\underline{Q}$ with $P = \underline{P}$ and $Q = \underline{Q}$. Because of the soundness of the axioms we get $\underline{P} \approx_M^c P \approx_M^c Q \approx_M^c \underline{Q}$, thus $\underline{P} \approx_M^c \underline{Q}$.

By Proposition 4.7 we derive $\underline{P} \mathrel{\hat{=}} \underline{Q}$, which can be weakened to $\underline{P} = \underline{Q}$. Thus we conclude $P = Q$ from $P = \underline{P} = \underline{Q} = Q$. □

Now we consider adding parallel composition and hiding to the language, reaching full $\mathcal{L}_{IM}$. Figure 4 shows additional laws, giving a sound and complete theory for this extended language together with those of Figure 3. It is easily confirmed that these additional laws respect Markovian observational congruence, i.e. they are sound. The remainder of this section deals with completeness for finite processes.

Lemma 4.3

(a) *For every finite process* P *there exists a finite process* P' *not containing any parallel operator, such that* $P = P'$.

(b) *For every finite process* P *there exists a process* P' *not involving hiding, such that* $P = P'$.

Corollary 4.2 *The axioms* A0–A16 *form a complete axiomatization of* $\approx_M^c$ *for finite processes, i.e.* $\forall P, Q \in \mathcal{L}_{IM} : P \approx_M^c Q \implies P = Q$.

$$
\begin{array}{lrcl}
\text{A10} & P\|_S Q = & & \sum\limits_{\substack{a_i \in \mathrm{Act}_M \\ a_i \notin S}} (a_i,\lambda_i).(P_i\|_S Q) + \sum\limits_{\substack{b_j \in \mathrm{Act}_M \\ b_j \notin S}} (b_j,\mu_j).(P\|_S Q_j) + \\
 & & & \sum\limits_{\substack{\underline{a}_i \in \mathrm{Act}_I \\ \underline{a}_i \notin S}} \underline{a}_i.(P_i\|_S Q) + \sum\limits_{\substack{\underline{b}_j \in \mathrm{Act}_I \\ \underline{b}_j \notin S}} \underline{b}_j.(P\|_S Q_j) + \\
 & & & \sum\limits_{\substack{a_i, b_j \in S \setminus \mathrm{Act}_I \\ a_i = b_j}} (a_i,\lambda_i\mu_j).(P_i\|_S Q_j) + \\
 & & & \sum\limits_{\substack{\underline{a}_i, \underline{b}_j \in S \cap \mathrm{Act}_I \\ a_i = b_j}} \underline{a}_i.(P_i\|_S Q_j) \\
\text{A11} & Stop\backslash\alpha & = & Stop \\
\text{A12} & (P+Q)\backslash\alpha & = & P\backslash\alpha + Q\backslash\alpha \\
\text{A13} & ((a,\lambda).P)\backslash a & = & (\tau,\lambda).P\backslash a \\
\text{A14} & ((b,\lambda).P)\backslash a & = & (b,\lambda).P\backslash a \qquad \text{if } a \neq b \\
\text{A15} & (\underline{a}.P)\backslash\underline{a} & = & \underline{\tau}.P\backslash\underline{a} \\
\text{A16} & (\underline{b}.P)\backslash\underline{a} & = & \underline{b}.P\backslash\underline{a} \qquad \text{if } \underline{a} \neq \underline{b}
\end{array}
$$

Figure 4: Additional axioms of $\approx^c_M$ for Parallel Composition and Hiding

Proof: By Lemma 4.3 hiding, as well as parallel operators, in P and Q can be eliminated leading to equivalent sequential finite processes $\underline{P}$ and $\underline{Q}$ that are captured by Corollary 4.1. □

Even though the above set of axioms is rich enough to ensure the completeness result, it is not very convenient to work with. Various additional laws can however be derived that are used very often when comparing processes, for example:

L1 $P + \underline{\tau}.(P+Q) \;=\; \underline{\tau}.(P+Q)$

L2 $P\|_{\{\}}Stop \;=\; P$

L3 $P\|_S Q \;=\; Q\|_S P$

L4 $(P\|_S Q)\|_S R \;=\; P\|_S(Q\|_S R)$

L5 $(P\|_S Q)\backslash\alpha \;=\; (P\backslash\alpha)\|_S(Q\backslash\alpha)$, if $\alpha \notin S$

5 An example

In order to illustrate the achieved equational theory we will demonstrate its usefulness by means of an example. In this example we will use recursive defining equations that are not part of the language $\mathcal{L}_{IM}$. However, the language can be easily extended with a *recursion* operator allowing the description of those equations. The algebraic theory is not touched by this extension, apart from the completeness result. Since we add another operator a *complete* axiomatisation relies on additional laws that still have to be established.

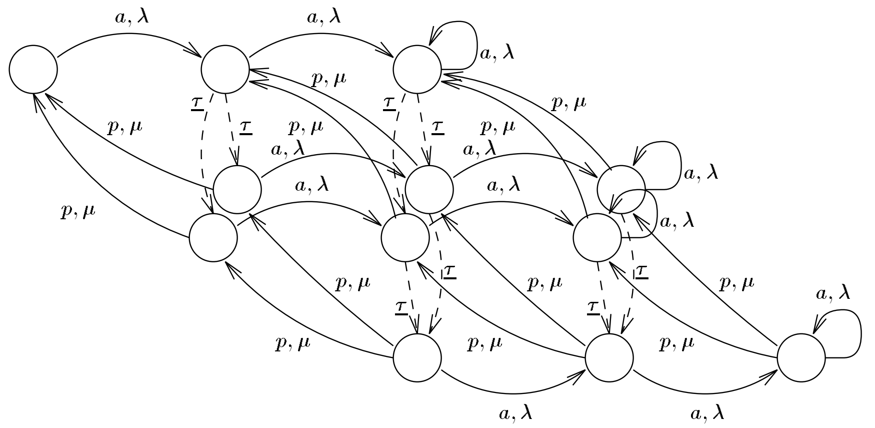

Figure 5: Semantic model of the M/M/2/2 specification

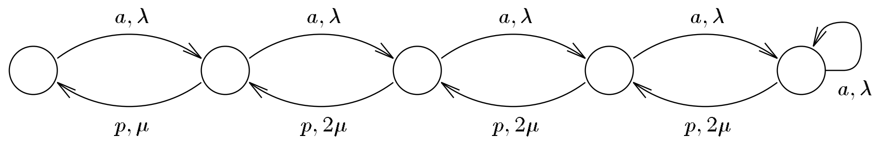

Figure 6: Equivalent model of the M/M/2/2 specification

The example we consider here is a modular description of an M/M/n/k queueing system incorporating immediate actions. The example has been investigated from the model construction point of view in [6]. The use of immediate actions turned out to be crucial for the modularity of the specification. However, these can be eliminated leading to a strict Markovian model that can be transformed into a CTMC. In order to keep the presentation compact we choose $n = k = 2$.

The queueing system is built out of a few modules: an arrival process A, a queue Q and a server $Multi$ consisting of two processors P. The arrival process is modeled as a Poisson stream with rate λ in the usual way.

$$A := (a, \lambda).A$$

The queue waits for jobs. This passivity is expressed by the rate 1, the neutral element of multiplication. This implies that the rate of a synchronized action, a, is only determined by the synchronization partner. If the queue is not empty, it can alternatively deliver ($\underline{d}$) a job to either of the servers. If it is full, incoming jobs are ignored.

$$\begin{aligned} Q_0 &:= (a,1).Q_1 \\ Q_1 &:= (a,1).Q_2 + \underline{d}.Q_0 \\ Q_2 &:= (a,1).Q_2 + \underline{d}.Q_1 \end{aligned}$$

The two servers are working independently. If possible, a server takes a job out of the queue ($\underline{d}$) without spending time. Afterwards it processes the job with rate μ, (p, μ).

$$Multi := P \parallel_{\emptyset} P \qquad P := \underline{d}.P_w \qquad P_w := (p, \mu).P$$

The whole system consists of the arrival stream, queue and multiserver appropriately synchronized. From the environment, the immediate action $\underline{d}$ is not visible.

$$Sys := (A \parallel_{\{a\}} Q_0 \parallel_{\{\underline{d}\}} Multi)\backslash\{\underline{d}\}$$

The semantic model obtained via the operational semantic rules is depicted in Figure 5. One may argue that this model is far away from the model usually expected (Figure 6). However, both belong to the same equivalence class of Markovian observational congruence. More precisely, Figure 6 represents a normal form of the system Sys in the sense of Definition 4.4.

To establish this property formally we apply the laws A0 – A16 iteratively. The normal form is constructed step by step out of the original specification. At first, the starting state is expanded (A10), afterwards the symmetries and priorities can be exploited by means of other axioms and laws.

$$\begin{aligned}
Sys := & (A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
\stackrel{\text{A10}}{=} & (a,\lambda).(A\|_{\{a\}}(Q_1\|_{\{\underline{d}\}}(P\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
\stackrel{\text{A10}}{=} & (a,\lambda).(\quad (a,\lambda).(A\|_{\{a\}}(Q_2\|_{\{\underline{d}\}}(P\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P\|_{\emptyset}P_w)))\backslash\{\underline{d}\} \\
& \qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad) \\
\stackrel{\text{L3}}{=} & (a,\lambda).(\quad (a,\lambda).(A\|_{\{a\}}(Q_2\|_{\{\underline{d}\}}(P\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad) \\
\stackrel{\text{A4}}{=} & (a,\lambda).(\quad (a,\lambda).(A\|_{\{a\}}(Q_2\|_{\{\underline{d}\}}(P\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
& \qquad) \\
\stackrel{\text{A9}}{=} & (a,\lambda).\underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P)))\backslash\{\underline{d}\} \\
\stackrel{\text{A7}}{=} & (a,\lambda).\underbrace{(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_{\emptyset}P))\backslash\{\underline{d}\}}_{Sub_0}
\end{aligned}$$

After reducing the expanded starting term, a subterm Sub_0 remains that presents no obvious target for further transformation. Again, this subterm can be reduced after exploring its initial behaviour using A10.

$$
\begin{aligned}
Sub_0 := &(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_\emptyset P))\backslash\{\underline{d}\}\\
\stackrel{\text{A10}}{=} &(a,\lambda).(A\|_{\{a\}}(Q_1\|_{\{\underline{d}\}}(P_w\|_\emptyset P)))\backslash\{\underline{d}\}\\
&+(p,\mu).(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P\|_\emptyset P)))\backslash\{\underline{d}\}\\
\stackrel{\text{A3}}{=} &(p,\mu).Sys\\
&+(a,\lambda).(A\|_{\{a\}}(Q_1\|_{\{\underline{d}\}}(P_w\|_\emptyset P)))\backslash\{\underline{d}\}\\
\stackrel{\text{A10}}{=} &(p,\mu).Sys\\
&+(a,\lambda).(\quad (a,\lambda).(A\|_{\{a\}}(Q_2\|_{\{\underline{d}\}}(P_w\|_\emptyset P)))\backslash\{\underline{d}\}\\
&\qquad\qquad + \quad \underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_\emptyset P_w)))\backslash\{\underline{d}\}\\
&\qquad\quad)\\
\stackrel{\text{A9}}{=} &(p,\mu).Sys\\
&+(a,\lambda).\underline{\tau}.(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_\emptyset P_w)))\backslash\{\underline{d}\}\\
\stackrel{\text{A7}}{=} &(p,\mu).Sys\\
&+(a,\lambda).\underbrace{(A\|_{\{a\}}(Q_0\|_{\{\underline{d}\}}(P_w\|_\emptyset P_w)))\backslash\{\underline{d}\}}_{Sub_1}
\end{aligned}
$$

As a result of the transformation of Sub_0 a new subterm Sub_1 remains to be transformed. In general more than one new subterm can arise out of the such a step that has to be explored. But since we consider a system with a finite state space, the number of subterms is limited.

The construction of the normal form terminates, if the set of subterms to be explored is empty. The result is a compact description of the transition system that is sketched in Figure 6:

$$
\begin{aligned}
Sys &= (a,\lambda).Sub_0\\
Sub_0 &= (p,\mu).Sys + (a,\lambda).Sub_1\\
Sub_1 &= (p,2\mu).Sub_0 + (a,\lambda).Sub_2\\
Sub_2 &= (a,\lambda).Sub_3 + (p,2\mu).Sub_1\\
Sub_3 &= (a,\lambda).Sub_3 + (p,2\mu).Sub_2
\end{aligned}
$$

6 Discussion

In this section we will consider two interesting issues arising from the formal treatment of immediate actions. First, we will discuss how the sketched reduction strategy can be implemented efficiently for arbitrary specifications. Then we will turn our attention to the implications for the modeller who wants to specify systems in a modular fashion.

6.1 Implementation issues

In the above example we have transformed a highly modular specification into the most compact representation that contains all the necessary information for functional as well as performance analysis. Although the result is not surprising in our specific example, the main advantage is that this transformation is not restricted to simple cases. The algebraic theory does not only cover typical Markov models like M/M/n/k or MMPP$^{[m]}$/E$_2$/n/k, it has been developed in order to facilitate the modular specification of arbitrary systems that might not have a typical Markov Model underlying it. Independent of the modellers knowledge of Markov theory, the formal derivation of the underlying CTMC can be performed automatically by means of operational semantic rules and equivalence reduction.

There are two main ways to implement this formal derivation, that differ in the attack point for equivalence reduction. The reduction can take place by transforming the specification syntactically, as in our example. This strategy relies mainly on term rewriting by applying the equational laws. Such an algorithm would produce a compact normal form as output. Afterwards this output is interpreted by means of the operational semantic rules. The concrete implementation of the term rewriting algorithm is currently under development. It is based on similar work for the calculus of Section 2 [13].

To apply the operational semantics first, is the starting step of the other strategy. Then, reduction takes place on the transition graph as in [8, 3]. For this purpose, efficient graph algorithms exist that can be adopted [14]. The major drawback of this approach is that the whole transition graph is often far too large to be stored even though it could be dramatically reduced once it has been stored. In fact, this strategy does not make use of the compositional properties of the underlying theory. As we have seen, the theory allows the replacement of components by reduced ones with equivalent behaviour. Therefore, an intermediate strategy, inspired by [11, 18], could be to produce transition graphs componentwise, and reduce each of the graphs. Then, the whole graph is constructed stepwise and also reduced in every step. This strategy makes substantial use of the congruence property of Markovian Observational congruence, like the above term rewriting approach.

6.2 Modelling issues

The algebraic foundations of immediate actions make it possible to enrich our language with two constructs taken from LOTOS, namely *process enabling* and *disabling*. The former construct, denoted by $P \gg Q$, models the sequential execution of two processes P and Q. From the modelling point of view it is of great value to express sequential execution of processes directly. Typical situations where standard SPA fail to permit simple specifications are *fork-and-join*-structures. Here, the process enabling operator will allow one to model these structures in an intuitive way. A rough example is a specification like

$$Fork \gg (P\|_S P\|_S P) \gg Join,$$

where the whole process is split up into three concurrent instances of P after executing the subprocess $Fork$. The process $Join$ is started after successful termination of all the instances of P.

By means of the disabling operator P [> Q, the preemption of process P by Q is modelled. The process P can perform arbitrary actions as long as Q, typically representing a timer, does not perform any action. But as soon as Q executes an action, P is stopped and control is handed over to process Q. This operator allows us to split the specification of timers from the description of other components.

The reason why we need immediate actions first before adding these operators to our language lies in the semantics of these operators [4]. Both make use of internal control activities that are meant to have no visible impact on the behaviour of the system. Immediate, internal actions allow us to model those control activities in a proper way.

Another construct that might be of interest for the modeller is a *temporal abstraction* mechanism. Similar to the effect of the (functional) hiding operator that internalizes the functional impact of certain actions, it seems interesting to introduce a *temporal* hiding operator. This operator would abstract away the temporal impact of certain Markovian actions by turning them into immediate ones. Such an operator would allow the modeller to delay the decision which activities of a component are considered to take no time. This decision can be influenced by the specification of other components.

The most comfortable way would be to specify a threshold value up to which the temporal impact of Markovian actions are considered to be negligible. Actions with a rate above this value would be considered to be immediate ones. However, we believe that such an operator is not readily available. A simple example might illustrate the problems. If the threshold value is μ, with $\lambda < \mu < 2\lambda$, then $(a, 2\lambda)$ is regarded as being negligible, i.e. immediate, whereas (a, λ) is not. This raises the question whether $(a, \lambda).P + (a, \lambda).P$ performs an immediate or a Markovian action. Both solutions pose problems. Just as functional hiding, we would like to apply temporal hiding to each of the components (cf. law A12), which means that each instance of (a, λ) is considered to be below the threshold. On the other hand, one of our first results has been that $(a, \lambda).P + (a, \lambda).P$ and $(a, 2\lambda).P$ should behave in the same manner (cf. law A0) , forcing us to choose an immediate action.

As an alternative, it might be possible to provide an operator that turns Markovian actions with a given *name* a into immediate ones $\underline{a}$, independent from the associated rate. This seems to be feasible. Taking into consideration that immediate actions must also be invisible in order to be eliminated successfully, a combination of functional and temporal abstraction might also be of interest.

7 Conclusion

In this paper we have extended the algebraic theory of SPA towards a formal treatment of immediate actions. Starting from classical process algebraic considerations we have defined the circumstances under which immediate actions have a visible impact on the behaviour of a system. For this purpose we defined an equivalence relation, Markovian observational congruence and discussed various properties of this relation. In particular, a sound and complete axiomatisation for finite processes has been developed that permits a compositional elimination strategy for immediate actions that are hidden from the environment.

As we have seen, this type of immediate actions is particularly useful to modularize specifications. In particular, it is the base for two operators that we are planning to include in the language. These will allow the user to specify disruption by means of timeouts and sequential composition of processes in a natural way.

In general, the equivalence notion introduced in this paper has one drawback, similar to classical observational congruence: It is not possible to eliminate invisible and immediate branching that leads to different states that are not Markovian observational equivalent.

Thus, for processes like $\underline{\tau}.(a,\lambda).Stop + \underline{\tau}.(b,\mu).Stop$ it is impossible to find a Markovian observational congruent process that does not contain any $\underline{\tau}$ action.

One may argue that such a system is not fully specified because the choice between the two branches is non-deterministic in the true sense of the word. The behaviour of such a choice can neither be determined by the environment via synchronization, nor is the behaviour determined by the specification itself, e.g. by means of branching probabilities. This lack of expressiveness initiated the development of probabilistic choice mechanisms as a proper means to specify and evaluate such behaviour [17]. In that approach, branching probabilities are an explicit part of the syntactic description of invisible and immediate actions.

Acknowledgements

The authors would like to thank the anonymous referees as well as Jane Hillston and Stephen Gilmore for their very helpful comments on previous versions of the paper.

References

[1] M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems*. MIT Press, 1986.

[2] J. Baeten and P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Computer Science*. Cambridge University Press, 1990.

[3] M. Bernardo, L. Donatiello, and R. Gorrieri. MPA: A Stochastic Process Algebra. Technical report UBLCS-94-10, Laboratory for Computer Science, University of Bologna, 1994.

[4] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. In P.H.J. van Eijk, C.A. Vissers, and M. Diaz, editors, *The Formal Description Technique LOTOS*, pages 23–73, Amsterdam, 1989. North-Holland.

[5] G. Boudol and I. Castellani. A Non-Interleaving Semantics for CCS Based on Proved Transitions. *Fundamenta Informaticae*, XI:433–452, 1988.

[6] N. Götz. *Stochastische Prozeßalgebren — Integration von funktionalem Entwurf und Leistungsbewertung Verteilter Systeme*, volume 27 (1) of *Arbeitsberichte des IMMD*. Dissertation, Universität Erlangen, Mai 1994.

[7] N. Götz, U. Herzog, and M. Rettelbach. TIPP — A Language for Timed Processes and Performance Evaluation. Technical Report 4/92, Universität Erlangen–Nürnberg, IMMD VII, 1992.

[8] N. Götz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *Proc. of the 16th Int'l Symposium on Computer Performance Modelling, Measurement and Evaluation, PERFORMANCE '93*. Springer, 1993. LNCS 729.

[9] H. Hermanns and M. Rettelbach. Syntax, Semantics, Equivalences, and Axioms for MTIPP. In Herzog and Rettelbach [10].

[10] U. Herzog and M. Rettelbach, editors. *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Martensstr. 3, D-91085 Erlangen, November 1994. University of Erlangen-Nürnberg, IMMD.

[11] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, Department of Computer Science, University of Edinburgh, 1994.

[12] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[13] P. Inverardi and M. Nesi. Deciding Observational Congruence of finite state CCS expressions by rewriting. *Theoretical Computer Science*, 139:315–354, 1995.

[14] P. Kanellakis and S. Smolka. CCS Expressions, Finite State Processes, and Three Problems of Equivalence. *Information and Computation*, 86:43–68, 1990.

[15] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Proc. of 16th ACM Symposium on Principles of Programming Languages*, pages 344–352, 1989.

[16] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[17] M. Rettelbach. Probabilistic Branching in Markovian Process Algebras. In *Proceedings of the 3rd Workshop on Process Algebra and Performance Modelling*, 1995. this issue.

[18] M. Rettelbach and M. Siegle. Compositional Minimal Semantics for the Stochastic Process Algebra TIPP. In Herzog and Rettelbach [10].

[19] T. Weiß. *Congruence Investigations for the Integration of Immediate Actions into Stochastic Process Algebras*. Internal Study, University Erlangen-Nürnberg, 1995. (in German).