Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 192 (2007) 45–60
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Process Algebra Having Inherent Choice: Revised Semantics for Concurrent Systems[1]

Harald Fecher[2]

Department of Computing
Imperial College
London, UK

Heiko Schmidt[3]

Institut für Informatik
Christian-Albrechts-Universität
Kiel, Germany

Abstract

Process algebras are standard formalisms for compositionally describing systems by the dependencies of their observable synchronous communication. In concurrent systems, parallel composition introduces resolvable nondeterminism, i.e., nondeterminism that will be resolved in later design phases or by the operating system. Sometimes it is also important to express inherent nondeterminism for equal (communication) labels. Here, we give operational and axiomatic semantics to a process algebra having a parallel operator interpreted as concurrent and having a choice operator interpreted as inherent, not only w.r.t. different, but also w.r.t. equal next-step actions. In order to handle the different kinds of nondeterminism, the operational semantics uses μ-automata as underlying semantical model. Soundness and completeness of our axiom system w.r.t. the operational semantics is shown.

Keywords: nondeterminism, process algebra, axiom system, expansion theorem, μ-automata

1 Introduction

Process algebras, see [2] for an overview, are standard formalisms for compositionally describing systems based, e.g., on synchronous communication on an abstract level by the dependencies of their observable communication. They serve as a domain for semantical foundations of programming or modeling languages and are also used as modeling languages [6].

[1] This work is in part financially supported by the DFG project *Refism* (FE 942/1-1)

[2] Email: hfecher@doc.ic.ac.uk

[3] Email: hsc@informatik.uni-kiel.de

doi:10.1016/j.entcs.2007.08.016

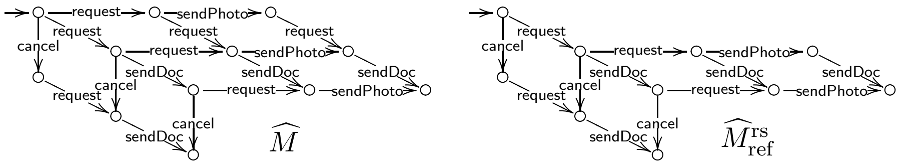

Fig. 1. The transition system $\widehat{M}$ corresponding to the semantics of (1) and a refinement $\widehat{M}^{\mathrm{rs}}_{\mathrm{ref}}$ of $\widehat{M}$ w.r.t. ready simulation

Example 1.1 Suppose there are two processes running concurrently on a single processor computer that both have the possibility to send print jobs to a printer. Prior to sending the data to the printer, any process must gain exclusive access to the printer by synchronizing on an action request. After that, the print job can be sent. In our example, the first process sends a photo (sendPhoto), whereas the second sends a document (sendDoc). Furthermore, the first process can be disrupted by a user via action cancel. This simplified concurrent system (in a parallel environment) can be modeled in process algebra based on synchronous communication by

$$(\mathsf{request.sendPhoto} + \mathsf{cancel}) \| (\mathsf{request.sendDoc}) \tag{1}$$

where the printer and the user that can disrupt belong to the environment. Here, operator $\|$ denotes parallel composition, $a.B$ denotes action prefix, and $+$ denotes the choice operator. The semantics of (1) in terms of transition system is given by $\widehat{M}$ of Figure 1.

In the transition system of the above example two kinds of choice can be distinguished: (i) External choice represented by outgoing edges with different labels. This choice occurs in implementations and remains undecided until in an execution the environment decides which of the possible actions is performed. (ii) Internal (nondeterministic) choice represented by outgoing edges with the same labels. This nondeterminism is resolved by the scheduler [4] of the operating system, since the two processes run on a single processor computer: The scheduler will decide which process may perform its request action, i.e., synchronize to get exclusive access to the printer. This refinement, performed by adding the scheduler, is formally made precise by ready simulation [5], which essentially allows the removal of multiple outgoing edges with the same label, as long as one remains (see Figure 1).

Nondeterminism that will be removed in later design phases or by the operating system is called *resolvable nondeterminism*, whereas nondeterminism that has to remain in implementations and is decided independently for each execution (e.g., by random), is called *inherent nondeterminism*. Consequently, the nondeterminism introduced by abstracting from schedulers in a concurrent setting is resolvable. Note, however, that in standard semantics based on bisimulation the abstraction from schedulers can be regarded as inherent (since transition systems with bisimulation cannot express resolvable nondeterminism). We consider this to be counterintuitive in most cases, because schedulers usually do not show “random” behaviour that is determined independently for each execution.

[4] In a generative rather than reactive setting, different outgoing labels can also express internal choice and thus be resolved by the scheduler. However, this paper only considers reactive systems.

Branching time logics, like the modal μ-calculus [17], are often used for describing properties of process algebras. Unfortunately, they are not preserved under refinement based on ready simulation. Consider, e.g., the statement that there is an immediate request action enabled such that afterwards no document can be sent to the printer. This property is described by the μ-calculus formula

$$\langle\mathsf{request}\rangle\,([\mathsf{sendDoc}]\,\mathit{false}\,). \tag{2}$$

Property (2) holds in $\widehat{M}$ of Figure 1, but not in its refinement $\widehat{M}^{\mathrm{rs}}_{\mathrm{ref}}$ of Figure 1. This illustrates that branching times properties applied to a concurrent setting need to be interpreted w.r.t. sets of schedulers: $\langle a\rangle\,\phi$ requires the existence of a scheduler such that ϕ holds after the execution of a, and $[a]\,\phi$ states that, independent of which scheduler is chosen, ϕ will hold after the execution of a. Consequently, property (2) has to be understood as follows: There is a scheduler for the next step such that request is enabled and its execution (which is deterministic if the scheduler is given) leads to a state where sendDoc is for any scheduler disabled. Since the refinement in Figure 1 specializes schedulers, μ-calculus formula (2) is not preserved.

In Example 1.1, we illustrated that resolvable nondeterminism naturally arises through parallel composition. In the following examples we argue that also inherent nondeterminism, i.e., internal choice that remains in running implementations, occurs in applications:

Example 1.2 In the situation of Example 1.1, assume a faulty channel between the first process and the printer. Then it is possible that a signal request (e.g., encoded as 1) can be turned into a signal cancel (e.g., encoded as 0). This is reflected by the process algebra term

$$(\mathsf{request.sendPhoto} + \mathsf{request} + \mathsf{cancel})\,||\,(\mathsf{request.sendDoc})\,. \tag{3}$$

We get inherent nondeterminism w.r.t. the same label request, because in case of a faulty transmission it has to be handled as an action cancel. The nondeterminism is inherent, because refinement cannot decide on the existence of a fault, this is decided for every execution.

Example 1.3 In the situation of Example 1.1, consider request to be an abstract action [25] for more refined labels like requestLowRes and requestHighRes, where the first establishes access to the low resolution printing features of the printer, and the second gains access to the high resolution features. Then, based on the printing capabilities offered by the printer, the usual photo can be sent (sendPhoto) or a low resolution alternative (sendLowResPhoto). This is reflected by the following process algebra term having inherent nondeterminism w.r.t. the same label request:

$$(\mathsf{request.sendPhoto} + \mathsf{request.sendLowResPhoto} + \mathsf{cancel})\,||\,(\mathsf{request.sendDoc})\,. \tag{4}$$

Note that in these scenarios we have both inherent nondeterminism, introduced by the choice operator of the first process, and resolvable nondeterminism, introduced by the unknown scheduler of the parallel composition. In case there is no resolvable nondeterminism, i.e., if we consider concrete systems only with inherent nondeterminism, transition systems together with bisimulation as underlying equivalence notion is an appropriate semantical model. Here, bisimulation equivalence is a specialization of ready simulation for concrete systems, i.e., a ready simulation between concrete systems implies bisimilarity. The μ-calculus yields a suitable logic, since it characterizes transition systems up to bisimulation.

Nevertheless, transition systems are not an appropriate model whenever inherent and resolvable nondeterminism occur in a single setting, like in Examples 1.2 and 1.3. This is because a choice for an underlying equivalence or preorder has to be made: Bisimulation interprets nondeterminism as inherent, ready simulation as resolvable. Furthermore, when using resolvable nondeterminism, we expect a three-valued satisfaction interpretation over the μ-calculus, with the possibility that a formula is neither satisfied nor falsified. For example, we expect that $\langle \mathsf{request} \rangle ((\langle \mathsf{sendDoc} \rangle\, \mathit{true}\,) \vee (\langle \mathsf{sendLowResPhoto} \rangle\, \mathit{true}\,))$ holds in (4), since all its implementations do, but property (2) is "unknown" for (4), since there are implementations that satisfy the property and there are implementations that do not.

Contribution. We give an operational and an axiomatic semantics to a process algebra having a parallel operator interpreted in a concurrent (rather than distributed) setting and a choice operator interpreted as inherent, not only w.r.t. different, but also w.r.t. equal next-step actions. In particular, we adjust the semantics of [3], which has an inherent as well as a resolvable choice operator, to our interpretation of parallel composition, since in their interpretation parallel composition yields inherent nondeterminism. In order to handle the two kinds of nondeterminism adequately, a semantical model with two kinds of transition relations, as in [3] and [14], is used, namely μ-automata [16] with their standard refinement notion and their three-valued satisfaction relation over the μ-calculus. The two transition relations are defined using structural operational semantics. One relation corresponds to the execution of actions, the other corresponds to the removing of the underspecification for the next action execution, which is called concretization. In order to develop an axiomatic semantics, the process algebra is extended by further operators, especially by a choice operator corresponding to resolvable nondeterminism. From our axiom system, we derive an expansion theorem, which expresses the parallel composition operator in terms of choice operators. Soundness and completeness of this axiom system w.r.t. the operational semantics is shown.

2 Syntax

In order not to distract from the technical problems and their solutions, we only present a simple process algebra that does not have recursion, sequential composition, and parallel composition with a synchronization mechanism. Our process algebra consists of action prefix, inherent nondeterminism, parallel composition, and action renaming. Note that a (CSP-based) hiding operator is a special case of a renaming operator, where the action is renamed to the internal action. Here, we follow the philosophy that internal actions are observable. In other words, we do not consider weak equivalence or weak refinement notions. Furthermore, we assume that nondeterminism obtained via "mixed choices" is resolved inside the environment: For example, if the system provides actions a and b and the environment their corresponding counterparts, then the environment decides if a or b is executed.[5] The renaming/hiding operator is also of special interest in our setting, since it introduces nondeterminism, too. For example, the process that provides action a leading to B_1 and that provides action b leading to B_2 does not contain nondeterminism, but after hiding a and b, i.e., renaming a and b to the internal action, a nondeterminism occurs, since now an internal step either leads to B_1 or to B_2. Again, the two interpretations of inherent or resolvable nondeterminism are possible for the renaming operator. But in our setting the nondeterminism obtained through hiding (and therefore implicitly for renaming) should be a resolvable nondeterminism, which is argued as follows: The system has a stimulus for any hidden action. These can be considered to be provided by an additional component continuously providing hidden actions. Then the scheduler of the parallel composition decides, which parallel component executes next. Consequently, hiding (and therefore renaming) yields resolvable nondeterminism with our interpretation of schedulers of parallel components, which assumes that schedulers do not behave randomly.

Before we formally present our process algebra, we introduce some notations. Let $\mathcal{A}ct$ denote the set of all actions, let $|S|$ denote the cardinality of a set S, and let $\mathbb{P}(S)$ denote its power set. Furthermore, $\circ$ denotes relational composition. For a binary relation $\rho \subseteq S \times I$ with subsets $X \subseteq S$ and $Y \subseteq I$ we write $X \circ \rho$ for $\{i \in I \mid \exists x \in X : (x, i) \in \rho\}$ and $\rho \circ Y$ for $\{s \in S \mid \exists y \in Y : (s, y) \in \rho\}$. For a ternary relation $\rightsquigarrow \subseteq S \times \mathcal{A}ct \times I$ we write $s \overset{a}{\rightsquigarrow} i$ for $(s, a, i) \in \rightsquigarrow$, and we write $\overset{a}{\rightsquigarrow}$ for the binary relation $\{(s, i) \mid s \overset{a}{\rightsquigarrow} i\}$, thus $\{s\} \circ \overset{a}{\rightsquigarrow} = \{i \in I \mid s \overset{a}{\rightsquigarrow} i\}$. Furthermore, we write $s \overset{a}{\not\rightsquigarrow}$, iff $\{s\} \circ \overset{a}{\rightsquigarrow} = \emptyset$.

PA, the set of all basis process algebra terms, is generated by

$$B ::= 0 \mid a.B \mid B + B \mid B \| B \mid B\langle a/b \rangle \,,$$

where $a, b \in \mathcal{A}ct$. Process 0 describes a deadlocked process, i.e., no further actions can be executed. We sometimes omit symbol 0 by writing a instead of $a.0$. Process $a.B$ allows the execution of action a resulting in the process B. Inherent choice is described by $B_1 + B_2$, and $B_1 \| B_2$ describes parallel composition. The parallel composition has implicit resolvable nondeterminism, introduced by abstraction from a scheduler favoring one of the two sides. $B\langle a/b \rangle$ describes the process where the execution of a in B becomes the execution of b. All other action execution, including action b, remains unaffected. Note that this renaming process also introduces resolvable nondeterminism as described in the beginning of this section.

3 Operational Semantics

3.1 μ-automata

As underlying semantical model we use μ-automata [16] in the notation of [9], except that we omit fairness constraints and that we do not consider propositions.

[5] Otherwise, the system has to resolve this nondeterminism in which case two communications instead of one have to be modeled: The environment sends its provided actions and the system answers which action it chooses.

Definition 3.1 [μ-automata] A *μ-automaton* M w.r.t. $\mathcal{A}ct$ is a tuple $(S, \widetilde{S}, s^{\mathrm{i}}, \rightrightarrows, \longmapsto)$ such that $(s \in)S$ is the set of OR-states, $(\tilde{s} \in)\widetilde{S}$ the set of BRANCH-states (disjoint from S), $s^{\mathrm{i}} \in S$ its initial element, $\rightrightarrows \subseteq S \times \widetilde{S}$ the OR-transition relation, and $\longmapsto \subseteq \widetilde{S} \times \mathcal{A}ct \times S$ the BRANCH-transition relation.

The BRANCH-states do not contain underspecification for the next action execution, whereas OR-states can be underspecified in that sense. This underspecification is resolved via the OR-transition relation (for this reason also called *concretization relation*), which is made precise by the standard refinement notion of μ-automata:

Definition 3.2 [μ-refinement] A relation $R \subseteq (S_1 \times S_2) \cup (\widetilde{S}_1 \times \widetilde{S}_2)$ is a *μ-refinement* between two μ-automata M_1 and M_2 if $(s_1^{\mathrm{i}}, s_2^{\mathrm{i}}) \in R$ and

- $\forall (s_1, s_2) \in R, \tilde{s}_1 \in (\{s_1\} \circ \rightrightarrows) : \exists \tilde{s}_2 \in (\{s_2\} \circ \rightrightarrows) : (\tilde{s}_1, \tilde{s}_2) \in R$,
- $\forall (\tilde{s}_1, \tilde{s}_2) \in R, a \in \mathcal{A}ct, s_1 \in (\{\tilde{s}_1\} \circ \overset{a}{\longmapsto}) : \exists s_2 \in (\{\tilde{s}_2\} \circ \overset{a}{\longmapsto}) : (s_1, s_2) \in R$, and
- $\forall (\tilde{s}_1, \tilde{s}_2) \in R, a \in \mathcal{A}ct, s_2 \in (\{\tilde{s}_2\} \circ \overset{a}{\longmapsto}) : \exists s_1 \in (\{\tilde{s}_1\} \circ \overset{a}{\longmapsto}) : (s_1, s_2) \in R$.

M_1 *μ-refines* M_2 if there exists a μ-refinement R between M_1 and M_2.

Labeled transition systems are straightforwardly embedded into μ-automata by using OR-states having exactly one outgoing transition. The restriction of μ-refinement onto these systems coincides with bisimulation. As commonly known, μ-refinement yields a partial order. Furthermore, μ-automata come with a three-valued satisfaction relation over the μ-calculus, which is preserved under refinement.

3.2 Operational semantics rules

In order to define the operational semantics we use two different kinds of expressions: one where underspecification for the next step is allowed (PA, corresponding to the OR-states) and one where it is not (PA^{con}, corresponding to the BRANCH-states), i.e., where the resolvable nondeterminism for the next execution is resolved. Then additionally to the step transition relation ($\longmapsto$, corresponding to BRANCH-transition relation) from PA^{con} to PA, a concretization relation ($\rightrightarrows$, corresponding to the OR-transition relation) from PA to PA^{con}, which resolves the resolvable nondeterminism for the next execution, is used. Formally, PA^{con} denotes the set of all process algebra terms generated by

$$P ::= [0] \mid [a.\widehat{B}] \mid P + P \mid P|\rangle_{\widehat{B},A,\widetilde{B}} P \mid P\langle a/b, v \rangle\,,$$

where $a, b \in \mathcal{A}ct$, $A \subseteq \mathcal{A}ct$, $\widehat{B}, \widetilde{B} \in$ PA, and $v \in \{\mathbf{d}, \mathbf{s}\}$. The intuition of the operators is similar to the one of the operators given in Section 2, except that here the scheduler of the parallel composition and of the renaming operator is determined for the next step. First, we explain the intuition of parallel composition $P_1|\rangle_{B_1,A,B_2} P_2$, at first neglecting B_1 and B_2, which will be explained later, and using the notation $P_1|\rangle_A P_2$ instead. Here, A specifies a scheduler, which is not necessary in standard semantics based on ready simulation, since there schedulers are adequately handled via the ready simulation. However, the scheduler needs to be modeled if both resolvable nondeterminism and inherent choice should appear in one setting. The scheduler information is interpreted as follows: The right side is favored in $P_1|\rangle_A P_2$ for actions from A, whereas the left side is favored for actions from $\mathcal{A}ct \setminus A$. This favoring concerns only the next step, i.e., after the execution of an action, any scheduling is allowed again. This is even the case if an action in parallel to $P_1|\rangle_A P_2$ is executed, e.g., if P_3 executes action a leading to B_3 in $(P_1|\rangle_A P_2)|\rangle_A P_3$ the resulting process is, roughly speaking, $(P_1\|P_2)\|B_3$, where all next step scheduling is removed. Note that this approach is more appropriate than the approach where the partial scheduler is kept (in which case $(P_1|\rangle_A P_2)\|B_3$ would be the result), since the scheduler is global and therefore can depend on any past execution.[6] Furthermore, associativity of $\|$ would be lost in the alternative approach, which is illustrated later in Example 3.5. In order to model the undoing of the scheduler information efficiently, the parallel composition stores the original processes of its components (here B_1 and B_2) and replaces the non-executing component by its stored original one, where no scheduler information is present. This will be clarified by the transition rules.

The scheduler information of the next execution v is added to the renaming operator: The execution of the action corresponding to the **s**ource label a of the renaming, which becomes b, is favored in $B\langle a/b, \mathbf{s}\rangle$, whereas the execution of the action corresponding to the **d**estination label b is favored in $B\langle a/b, \mathbf{d}\rangle$. Here, favoring means that for $B\langle a/b, \mathbf{d}\rangle$, process B may only execute a (which will be renamed to b) if B cannot execute b and analogously for $B\langle a/b, \mathbf{s}\rangle$, where a is favored. Again, this scheduling of the renaming operator only applies for the next action execution, i.e., after the execution of any action, possibly different from b, the current favoring is removed.

The concretization relation $\rightrightarrows$ is given in the upper section of Table 1, where the underspecification of the next step is resolved, and the step transition relation $\longmapsto$ is presented in the lower section of Table 1, where actions are executed resulting in processes having underspecification for the next step executions. We give some comments on $\rightrightarrows$: The resolution of next-step-underspecification has to take place in every subpart that can potentially make the next execution, consequently resolution does not take place for B in $a.B$. In the parallel composition, however, the next-step-underspecification is resolved by choosing an arbitrary scheduler. The parallel composition operator stores the original processes such that it can efficiently undo the concretization. In the renaming operator, the next-step-underspecification is resolved by either favoring the action corresponding to the source label ($\mathbf{s}$) or favoring the action corresponding to the destination label ($\mathbf{d}$) of the renaming. We proceed with some comments on $\longmapsto$: The rules for $[a.B]$ and $P_1 + P_2$ are standard. The left side of the parallel composition $P_1|\rangle_{B_3,A,B_4} P_2$ can execute a if (i) the left side is favored for a by the scheduler ($a \notin A$) or (ii) the right side does not provide a ($P_2 \not\xrightarrow{a}$). Symmetric constraints hold for the execution of a on the right side of $P_1|\rangle_{B_3,A,B_4} P_2$. As already mentioned before, the next-step-underspecification reso-

[6] The approach where the partial scheduler is kept makes only sense if each parallel component has its own scheduler and there is an additional global scheduler which decides which of the parallel components is favored.

$$\frac{}{0 \rightrightarrows [0]} \qquad \frac{}{a.B \rightrightarrows [a.B]} \qquad \frac{B_1 \rightrightarrows P_1 \quad B_2 \rightrightarrows P_2}{B_1 + B_2 \rightrightarrows P_1 + P_2}$$

$$\frac{B_1 \rightrightarrows P_1 \quad B_2 \rightrightarrows P_2 \quad A \subseteq \mathcal{A}ct}{B_1 \| B_2 \rightrightarrows P_1 |\rangle_{B_1,A,B_2} P_2} \qquad \frac{B \rightrightarrows P \quad v \in \{\mathbf{d},\mathbf{s}\}}{B\langle a/b\rangle \rightrightarrows P\langle a/b,v\rangle}$$

$$\frac{}{[a.B] \xmapsto{a} B} \qquad \frac{i \in \{1,2\} \quad P_i \xmapsto{a} B}{P_1 + P_2 \xmapsto{a} B}$$

$$\frac{P_1 \xmapsto{a} B_1 \quad a \in A \Rightarrow P_2 \not\xmapsto{a}}{\begin{array}{c} P_1 |\rangle_{B_3,A,B_4} P_2 \xmapsto{a} B_1 \| B_4 \\ P_2 |\rangle_{B_4,\mathcal{A}ct\setminus A,B_3} P_1 \xmapsto{a} B_4 \| B_1 \end{array}} \qquad \frac{P \xmapsto{c} B \quad c \notin \{a,b\}}{P\langle a/b,v\rangle \xmapsto{c} B\langle a/b\rangle}$$

$$\frac{P \xmapsto{b} B \quad v = \mathbf{s} \Rightarrow P \not\xmapsto{a}}{P\langle a/b,v\rangle \xmapsto{b} B\langle a/b\rangle} \qquad \frac{P \xmapsto{a} B \quad v = \mathbf{d} \Rightarrow P \not\xmapsto{b}}{P\langle a/b,v\rangle \xmapsto{b} B\langle a/b\rangle}$$

Table 1
Upper section: Resolving of the next-step-underspecification via relation $\rightrightarrows \subseteq \mathrm{PA} \times \mathrm{PA}^{con}$. Lower section: Action executions via $\longmapsto \subseteq \mathrm{PA}^{con} \times \mathcal{A}ct \times \mathrm{PA}$.

lution has to be undone for the parallel component that did not make the execution. Therefore, process B_3, resp. B_4 replaces the non-executed side. In $P\langle a/b,v\rangle$ an action c different from a and b can be executed leading to $B\langle a/b\rangle$, whenever P can execute c leading to B. This is stated in the first rule. Furthermore, $P\langle a/b,v\rangle$ can execute b, leading to a process $B\langle a/b\rangle$, where B can be obtained after executing b in P, whenever v favors the destination label ($v = \mathbf{d}$) or no action a is provided by P. Similarly, $P\langle a/b,v\rangle$ can execute b, leading to a process $B\langle a/b\rangle$, where B can be obtained after executing a in P, whenever v favors the source label ($v = \mathbf{s}$) or no action b is provided by P. In all these three cases, the scheduler is removed after the execution. Note that by definition, the target processes of $\longmapsto$ do not contain any scheduling information.

Definition 3.3 [Operational semantics] The operational semantics of a process algebra term $B \in \mathrm{PA}$ is the μ-automaton $(\mathrm{PA}, \mathrm{PA}^{con}, B, \rightrightarrows, \longmapsto)$, where $\rightrightarrows$ and $\longmapsto$ are given in Table 1. We say that a process algebra term B from PA *refines* another one B', written $B \leq B'$, if the operational semantics of B μ-refines the operational semantics of B'. Furthermore, B is *refinement equivalent* to B', written $B \equiv B'$, if B refines B' and B' refines B.

Example 3.4 The operational semantics for $(a + a.b)\|a$ is illustrated in Figure 2.

Example 3.5 We illustrate that associativity of the parallel composition does not hold, if the resolution of underspecification in the parallel composition rule of Table 1 is not undone, i.e., if the original process does not replace the current process in case of non-execution. Under this assumption, $\tilde{B}_1 = a.b\|(a\|a.c)$ does not refine $\tilde{B}_2 = (a.b\|a)\|a.c$: By definition $\tilde{B}_1 \rightrightarrows [a.b] |\rangle_{\mathcal{A}ct}([a] |\rangle_{\mathcal{A}ct} [a.c])$. Since action c has to be possible afterwards, this process can only be adequately matched by $\tilde{B}_2$ via a process that is refinement equivalent to $([a.b] |\rangle_{\{a\}} [a]) |\rangle_{\{a\}} [a.c]$ or to $([a.b] |\rangle_{\emptyset} [a]) |\rangle_{\{a\}} [a.c]$.

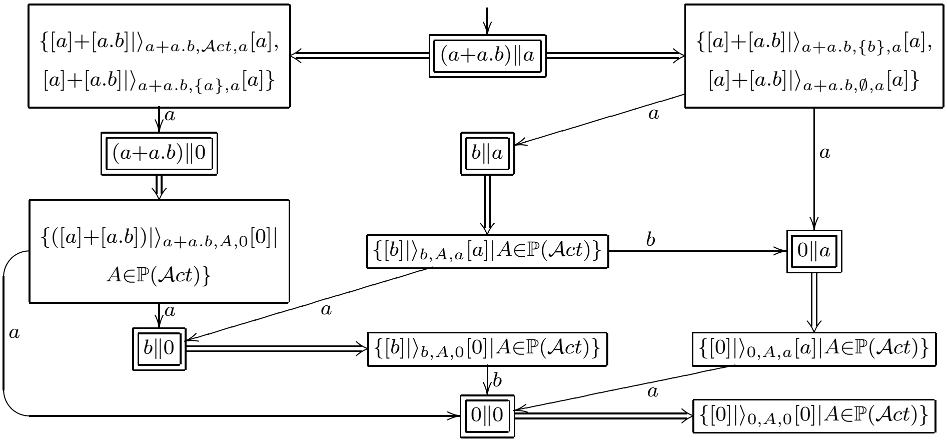
$\{[a]+[a.b]|\rangle_{a+a.b,\mathcal{A}ct,a}[a],$
$[a]+[a.b]|\rangle_{a+a.b,\{a\},a}[a]\}$
$(a+a.b)\|a$
$\{[a]+[a.b]|\rangle_{a+a.b,\{b\},a}[a],$
$[a]+[a.b]|\rangle_{a+a.b,\emptyset,a}[a]\}$
a
a
$(a+a.b)\|0$
$b\|a$
a
$\{([a]+[a.b])|\rangle_{a+a.b,A,0}[0]|$
$A\in\mathbb{P}(\mathcal{A}ct)\}$
$\{[b]|\rangle_{b,A,a}[a]|A\in\mathbb{P}(\mathcal{A}ct)\}$
b
$0\|a$
a
a
a
$b\|0$
$\{[b]|\rangle_{b,A,0}[0]|A\in\mathbb{P}(\mathcal{A}ct)\}$
$\{[0]|\rangle_{0,A,a}[a]|A\in\mathbb{P}(\mathcal{A}ct)\}$
b
a
$0\|0$
$\{[0]|\rangle_{0,A,0}[0]|A\in\mathbb{P}(\mathcal{A}ct)\}$

Fig. 2. The operational semantics for $(a + a.b)\|a$, where $\mathcal{A}ct = \{a, b\}$. OR-states of the μ-automaton have double-lined frames, whereas BRANCH-states have single-lined frames. OR-transitions are drawn as double-line arrows, whereas BRANCH-transitions are drawn as labeled single-line arrows. A state described by a set stands for a set of states, described by the elements of the set and having the same incoming and outgoing transitions as the state labeled with the set.

Thus after the execution of a we would need that $a.b\|(a\|c)$ refines $([a.b]\,|\rangle_{\{a\}}[a])\|\,[c]$ or refines $([a.b]\,|\rangle_{\emptyset}[a])\|\,[c]$. Furthermore, $a.b\|(a\|c)$ can be concretized such that either b is possible after the execution of a or no b is possible after the execution of a. But only one of these concretizations is possible in $([a.b]\,|\rangle_{\{a\}}[a])\|\,[c]$ and in $([a.b]\,|\rangle_{\emptyset}[a])\|\,[c]$. Hence, $\tilde{B}_1$ does not refine $\tilde{B}_2$.

Example 3.6 Process $a.(B_1\|(a.B_2))$ refines $(a.B_1)\|(a.B_2)$, but not vice versa. This illustrates that refinement over PA does not yield an equivalence relation.

Theorem 3.7 *Refinement is preserved under all process algebra operators, i.e., if* $B_1 \leq B_1' \wedge B_2 \leq B_2'$ *then* $a.B_1 \leq a.B_1' \;\wedge\; B_1 + B_2 \leq B_1' + B_2' \;\wedge\; B_1\|B_2 \leq B_1'\|B_2' \;\wedge\; B_1\langle a/b\rangle \leq B_1'\langle a/b\rangle$.

Note that the straightforward extension of $\leq$ to PA^{con} is also preserved under all process algebra operators for PA^{con}.

4 Axiomatic Semantics

In this section, we present a sound and complete axiom system for the refinement over PA.

4.1 Process algebra extension

In order to define the axioms, further terms are introduced: The most interesting one is the resolvable nondeterminism operator $\oplus$, which is also of interest for modeling by itself. The intuition of resolvable nondeterminism $B_1 \oplus B_2$ is that either B_1 or B_2 is implemented, but not both. Thus, $B_1 + B_2$ is in general not an allowed implementation.

Further added terms are: (i) A next step restriction process $B \setminus a$, where B may not execute action a as its next step. Note that action a may be executed

$$\frac{i \in \{1,2\} \qquad B_i \rightrightarrows P}{B_1 \oplus B_2 \rightrightarrows P} \qquad\qquad \frac{B_2 \rightrightarrows P_2}{c.B_1 >_a B_2 \rightrightarrows c.B_1 >_a P_2}$$

$$\frac{B \rightrightarrows P}{B \setminus a \rightrightarrows P \setminus a} \qquad \frac{B_1 \rightrightarrows P_1 \qquad B_2 \rightrightarrows P_2}{B_1|\rangle_{B_3,A,B_4}B_2 \rightrightarrows P_1|\rangle_{B_3,A,B_4}P_2} \qquad \frac{B \rightrightarrows P}{B\langle a/b, v\rangle \rightrightarrows P\langle a/b, v\rangle}$$

$$\frac{P_2 \not\xmapsto{a}}{c.B_1 >_a P_2 \xmapsto{c} B_1} \qquad\qquad \frac{P \xmapsto{a} B \qquad b \neq a}{P \setminus b \xmapsto{a} B}$$

Table 2
Additional transition rules for the extended process algebra.

if an action different from a is executed before. (ii) A conditional prefix term $c.B_1 >_a B_2$, which is equivalent to $c.B_1$ whenever B_2 cannot execute action a in its next step and it is equivalent to 0 if B_2 can execute action a in its next step. (iii) A parallel composition $B_1|\rangle_{B_3,A,B_4}B_2$ where the scheduler information for the next execution and the replacing processes for non-execution are already given. (iv) A renaming operator $B\langle a/b, v\rangle$, where the scheduler information for the next execution is already present. Also some counterparts of these new expressions are added to the process algebra terms where the next-step-underspecification is resolved. Formally, we define $\widetilde{\mathrm{PA}}$ to be the set of all process algebra terms generated by

$$B ::= 0 \mid a.B \mid B + B \mid B\|B \mid B\langle a/b\rangle \mid B \oplus B \mid B \setminus a \mid c.B >_a B \mid B|\rangle_{B,A,B}B \mid \\ B\langle a/b, v\rangle$$

and we define $\widetilde{\mathrm{PA}}^{con}$ to be the set of all process algebra terms generated by

$$P ::= [0] \mid [a.\widehat{B}] \mid P + P \mid P|\rangle_{\widehat{B},A,\widetilde{B}}P \mid P\langle a/b, v\rangle \mid P \setminus a \mid c.\widehat{B} >_a P$$

where $a, b, c \in \mathcal{A}ct$, $A \subseteq \mathcal{A}ct$, $\widehat{B}, \widetilde{B} \in \widetilde{\mathrm{PA}}$ and $v \in \{\mathbf{d}, \mathbf{s}\}$. The previous transition rules (Table 1) are extended by those from Table 2. We give some comments on $\rightrightarrows$: the resolvable nondeterminism $\oplus$ is resolved by choosing either the right or the left side and resolving this term. In $B_1|\rangle_{B_3,A,B_4}B_2$ only B_1 and B_2 have to be resolved, since no next step analysis takes place in term B_3 and B_4. By the same argument, only B_2 has to be resolved in $c.B_1 >_a B_2$. Now some comments on $\longmapsto$: In $c.B_1 >_a P_2$ a c-step to B_1 is possible iff the right hand side cannot execute a. In $P \setminus b$ all executions of P that differ from b can take place as next step, in which case $\setminus b$ is removed, since this restriction only holds for the next step execution.

Remark 4.1 For terms of PA, after every $\rightrightarrows$-step the same set of provided actions is obtained. This does not hold for terms of $\widetilde{\mathrm{PA}}$, as, e.g., illustrated by $a \oplus 0$.

The operational semantics of terms in $\widetilde{\mathrm{PA}}$ and refinement (equivalence) over $\widetilde{\mathrm{PA}}$ are defined as in Definition 3.3 except that the extended concretization and step transition relations are used.

4.2 Equations

Before we present the axiom system, we discuss some of the (standard) axioms that do not hold in general, i.e., where refinement equivalence cannot be guaranteed,

$$\frac{}{\mathrm{nuf}(0)} \qquad \frac{}{\mathrm{nuf}(a.B_1)} \qquad \frac{\mathrm{nuf}(B_1) \qquad \mathrm{nuf}(B_2)}{\mathrm{nuf}(B_1 + B_2)} \qquad \frac{\mathrm{nuf}(B)}{\mathrm{nuf}(B \setminus a)}$$

$$\frac{\mathrm{nuf}(B_2)}{\mathrm{nuf}(c.B_1 >_a B_2)} \qquad \frac{\mathrm{nuf}(B_1) \qquad \mathrm{nuf}(B_2)}{\mathrm{nuf}(B_1 |\rangle_{B_3,A,B_4} B_2)} \qquad \frac{\mathrm{nuf}(B)}{\mathrm{nuf}(B\langle a/b, v\rangle)}$$

Table 3
Definition of predicate nuf.

which is denoted by $\not\simeq$:

$B+B \not\simeq B$, since the left hand side allows more kinds of resolution of underspecification. For example, $(a \oplus b) + (a \oplus b)$ can be resolved to $[a] + [b]$, which provides a as well as b. On the other hand, $a \oplus b$ cannot be resolved such that both actions are provided. By similar arguments,

$$(B_1 \| B_2) + (B_1 \| B_3) \not\simeq B_1 \| (B_2 + B_3), \tag{5}$$

since the underspecification in B_1 can possibly be resolved in two different ways such that it cannot be matched by a single resolution of B_1. More precisely, after $\rightrightarrows$, it is possible that the left hand side of (5) can provide more actions than the right hand side. In case B_1 is next-step-underspecification-free, e.g., if B_1 is of form $\sum_{i \in I} a_i.B_i$ for some I, a_i, and B_i, we do not have this problem. Predicate nuf on $\widetilde{\mathrm{PA}}$, which is formally defined in Table 3, collects those **n**ext step **u**nderspecification **f**ree processes. Resolution of next-step-underspecification yields a unique term if nuf holds:

Lemma 4.2 *Suppose $B \in \widetilde{\mathrm{PA}}$ and $\mathcal{A}ct \neq \emptyset$. If* $\mathrm{nuf}(B)$*, then B has a unique target w.r.t. $\rightrightarrows$, i.e.,* $\mathrm{nuf}(B) \Rightarrow |\{B\} \circ \rightrightarrows| = 1$.

Nevertheless, the processes of (5) are not equivalent even if $\mathrm{nuf}(B_1)$ holds, since the resolvable nondeterminism of the parallel composition can be resolved in different ways. For example, after applying $\rightrightarrows$ on $(a.b) \| (a.c + 0)$ there is exactly one transition t labeled with a and either b or c is possible afterwards. On the other hand a $\rightrightarrows$ step from $((a.b) \| (a.c)) + ((a.b) \| 0)$ exists where two transitions t_1, t_2 labeled with a are possible such that b is possible after t_1 and c is possible after t_2. We remedy this problem by resolving the resolvable nondeterminism of the parallel operator before the parallel composition is expanded. This motivates why we introduced the term $B_1 |\rangle_{B_3,A,B_4} B_2$ already in $\widetilde{\mathrm{PA}}$.

In Table 4 axioms for the refinement relation $\preceq$ over $\widetilde{\mathrm{PA}}$ are presented, where we assume for simplicity that $\mathcal{A}ct$ is finite.

Theorem 4.3 *The axioms from Table 4 are sound and complete, i.e., $\forall B_1, B_2 \in \widetilde{\mathrm{PA}} : B_1 \preceq B_2 \iff B_1 \leq B_2$.*

Example 4.4 By using Axiom A32, we get $a.(b \oplus c) \preceq (a.(b \oplus c)) \oplus (a.b)$. Furthermore, by using Axiom A32 and then Axiom A3 we get $(a.(b \oplus c)) \oplus (a.b) \preceq (a.(b \oplus c)) \oplus (a.(b \oplus c)) \simeq a.(b \oplus c)$. Thus we have shown that $a.(b \oplus c) \simeq (a.(b \oplus c)) \oplus (a.b)$. Note that this cannot be shown by using only Axioms A1 – A31, since the necessary removal of $\oplus$ becomes impossible. This illustrates that Axiom A32 is also essential for the completeness of our Axiom system w.r.t. $\equiv$.

$$\begin{array}{ll}
(A1) & B_1 \oplus B_2 \simeq B_2 \oplus B_1 \\
(A2) & B_1 \oplus (B_2 \oplus B_3) \simeq (B_1 \oplus B_2) \oplus B_3 \\
(A3) & B \oplus B \simeq B \\
(A4) & B_1 + B_2 \simeq B_2 + B_1 \\
(A5) & B_1 + (B_2 + B_3) \simeq (B_1 + B_2) + B_3 \\
(A6) & a.B + a.B \simeq a.B \\
(A7) & B + 0 \simeq B \\
(A8) & B_1 + (B_2 \oplus B_3) \simeq (B_1 + B_2) \oplus (B_1 + B_3)
\end{array}$$

$$\begin{array}{ll}
(A9) & B_1 \| B_2 \simeq \bigoplus_{A \subseteq \mathcal{A}ct} (B_1 |\rangle_{B_1,A,B_2} B_2) \\
(A10) & B_1 |\rangle_{B_4,A,B_5} B_2 \simeq B_2 |\rangle_{B_5,\mathcal{A}ct \setminus A,B_4} B_1 \\
(A11) & B_1 |\rangle_{B_4,A,B_5} (B_2 \oplus B_3) \simeq (B_1 |\rangle_{B_4,A,B_5} B_2) \oplus (B_1 |\rangle_{B_4,A,B_5} B_3) \\
(A12) & B_1 |\rangle_{B_4,A,B_5} (B_2 + a.B_3) \simeq ((B_1 \setminus a) |\rangle_{B_4,A,B_5} B_2) + a.(B_4 \| B_3) \quad \text{if } a \in A \\
(A13) & B_1 |\rangle_{B_4,A,B_5} (B_2 + a.B_3) \simeq (B_1 |\rangle_{B_4,A,B_5} B_2) + (a.(B_4 \| B_3) >_a B_1) \\
 & \hfill \text{if } a \notin A \wedge \mathrm{nuf}(B_1) \\
(A14) & 0 \,|\rangle_{B_4,A,B_5} 0 \simeq 0
\end{array}$$

$$\begin{array}{ll}
(A15) & B\langle a/b \rangle \simeq B\langle a/b, \mathbf{d} \rangle \oplus B\langle a/b, \mathbf{s} \rangle \\
(A16) & (B_1 \oplus B_2)\langle a/b, v \rangle \simeq B_1\langle a/b, v \rangle \oplus B_2\langle a/b, v \rangle \\
(A17) & (B_1 + c.B_2)\langle a/b, v \rangle \simeq B_1\langle a/b, v \rangle + c.(B_2\langle a/b \rangle) \quad \text{if } c \notin \{a, b\} \\
(A18) & (B_1 + a.B_2)\langle a/b, \mathbf{d} \rangle \simeq B_1\langle a/b, \mathbf{d} \rangle + (b.(B_2\langle a/b \rangle) >_b B_1) \quad \text{if } \mathrm{nuf}(B_1) \\
(A19) & (B_1 + b.B_2)\langle a/b, \mathbf{d} \rangle \simeq (B_1 \setminus a)\langle a/b, \mathbf{d} \rangle + b.(B_2\langle a/b \rangle) \\
(A20) & (B_1 + a.B_2)\langle a/b, \mathbf{s} \rangle \simeq (B_1 \setminus b)\langle a/b, \mathbf{s} \rangle + b.(B_2\langle a/b \rangle) \\
(A21) & (B_1 + b.B_2)\langle a/b, \mathbf{s} \rangle \simeq B_1\langle a/b, \mathbf{s} \rangle + (b.(B_2\langle a/b \rangle) >_a B_1) \quad \text{if } \mathrm{nuf}(B_1) \\
(A22) & 0\langle a/b, v \rangle \simeq 0
\end{array}$$

$$\begin{array}{ll}
(A23) & (B_1 \oplus B_2) \setminus a \simeq (B_1 \setminus a) \oplus (B_2 \setminus a) \\
(A24) & (B_1 + B_2) \setminus a \simeq (B_1 \setminus a) + (B_2 \setminus a) \\
(A25) & 0 \setminus a \simeq 0 \\
(A26) & (a.B) \setminus a \simeq 0 \\
(A27) & (b.B) \setminus a \simeq b.B \quad \text{if } b \neq a
\end{array}$$

$$\begin{array}{ll}
(A28) & c.B_1 >_a (B_2 \oplus B_3) \simeq (c.B_1 >_a B_2) \oplus (c.B_1 >_a B_3) \\
(A29) & c.B_1 >_a (B_2 + a.B_3) \simeq 0 \\
(A30) & c.B_1 >_a (B_2 + b.B_3) \simeq c.B_1 >_a B_2 \quad \text{if } b \neq a \\
(A31) & c.B >_a 0 \simeq c.B
\end{array}$$

$$\begin{array}{ll}
(A32) & B_1 \preceq B_1 \oplus B_2
\end{array}$$

Table 4

Axioms for $\preceq$, where $\simeq = \preceq \cap \succeq$. Here, $\bigoplus_{i \in \{j\}} B_i$ is defined to be B_j and for I finite, $\bigoplus_{i \in I} B_i$ is defined to be $B_{i'} \oplus (\bigoplus_{i \in I \setminus \{i'\}} B_i)$ where $i' \in I$ (by the associativity and commutativity of $\oplus$ the definition is independent of the chosen i'). Note that we assume that $\mathcal{A}ct$ is finite. Otherwise, more complex formulas have to be used in order to remain in $\widetilde{\mathrm{PA}}$.

How the parallel composition can be expressed in terms of nondeterminism, known as the expansion theorem [23,4], is one of the most important rules. In our setting the expansion theorem is more complicated than usually, since resolvable as well as inherent nondeterminism has to be handled. It is directly derived from our axiom system and illustrated in the following, where $\sum_{j=0}^{-1} B_j$ yields 0:

Theorem 4.5 (Expansion theorem)

$$B\|B' \equiv \bigoplus_{A\subseteq \mathcal{A}ct} \bigoplus_{i=0}^{n} \bigoplus_{i'=0}^{n'} \Big(\sum_{j\in J_{(A,i,i')}} a_{i,j}.(B_{i,j}\|B') + \sum_{j'\in J'_{(A,i,i')}} a'_{i',j'}.(B\|B'_{i',j'}) \Big)$$

$$\begin{aligned} \textit{where} \quad & B = \textstyle\bigoplus_{i=0}^{n} \sum_{j=0}^{m_{(i)}-1} a_{i,j}.B_{i,j}, \qquad\qquad B' = \bigoplus_{i=0}^{n'} \sum_{j=0}^{m'_{(i)}-1} a'_{i,j}.B'_{i,j}, \\ & J_{(A,i,i')} = \{j < m_{(i)} \mid a_{i,j} \in A \Rightarrow \forall j' < m'_{(i')} : a_{i,j} \neq a'_{i',j'}\}, \\ & J'_{(A,i,i')} = \{j' < m'_{(i')} \mid a'_{i',j'} \notin A \Rightarrow \forall j < m_{(i)} : a_{i,j} \neq a'_{i',j'}\}. \end{aligned}$$

Here, (i) the scheduler ($A \subseteq \mathcal{A}ct$) is determined by a resolution of resolvable nondeterminism, (ii) any combination of resolutions of both sides is considered, and (iii) the complete inherent nondeterminism of a component for a is taken if this component is favored by the scheduler or if the current resolution of the other component cannot provide a.

5 Related Work

An overview on algebraic approaches to nondeterminism is given in [27]. Nondeterminism is often interpreted as angelic (chosen positively w.r.t. to a desired property) or demonic (chosen by an adversary). In [22], both nondeterminism interpretations are modeled in a single setting. The angelic/demonic view is orthogonal to our view, leading to the following four interpretations: (i) The task of resolving resolvable, angelic nondeterminism is a *satisfiability* check, i.e., an abstraction has an implementation satisfying the desired property if there is an angelic resolution. (ii) On the other hand, the task of resolving resolvable, demonic nondeterminism is a *satisfaction* check on the abstract level: the property holds if a demonic resolution satisfies it. (iii) Inherent, angelic nondeterminism ensures the existence of a step such that the desired property holds, whereas (iv) inherent, demonic nondeterminism ensures that all possible next steps satisfy the property. In alternating-time temporal logic [1], interpretations (iii) and (iv) are generalized to multiple actors.

In [3], both our choice operators (inherent and resolvable) are used in a process algebra and an operational semantics in terms of μ-automata as well as an axiomatic semantics is given. The difference to our work consists in the semantics of the parallel operator, since the semantics of the parallel operator in [3] is based on inherent instead of resolvable scheduling choice, i.e., schedulers behave “randomly” also at the concrete level, which is in most applications, like schedulers in operating systems, not the case. If the scheduler can prefer different parallel components per action, we cannot apply the usual approach to split the parallel composition by using the *left merge* operator, as it is also made in [3]. Therefore our axiom system becomes more complicated by using additional operators and by using predicate nuf. A further choice operator is presented in [3]. The interpretation of this choice operator $\oplus'$ is similar to our resolvable choice operator $\oplus$, except that the resolving can be moved outwards, e.g., $a.(b \oplus' c)$ is equivalent to $a.b \oplus' a.c$, which is not the case for $\oplus$. The choice operator $\oplus'$ has the same expressiveness as $\oplus$ w.r.t. the describable sets of concrete systems, i.e., replacing $\oplus'$ by $\oplus$, or vice versa, does not change the set of concrete systems that refine the corresponding expressions. A difference arises in the refinement relation between different abstract levels, i.e., the refinement relation between non-concrete systems is different.

In [26] a process algebra having our resolvable choice operator together with a choice operator that is only resolvable w.r.t. the same action, which harmonizes with ready simulation, is presented. Thus full inherent nondeterminism is not handled there. The semantics is given as possible worlds, i.e., sets of concrete systems. No parallel operator is considered at all. This process algebra is extended by recursion in [21], but still no parallel composition is considered. Different kinds of choice operators are considered in hybrid systems, see, e.g., [7], but those choice operators concentrate on underspecification (i.e., resolvable nondeterminism) resulting from time aspects.

Other abstract models that can express inherent as well as resolvable nondeterminism are, e.g., (disjunctive) modal transition systems [18,19], hypermixed Kripke structures [11], generalized Kripke modal transition systems [24], modal automata [9] and ν-automata [12]. These models are often used as abstract models for transition systems in order to improve verification of full branching time properties, as in [13,10].

In some settings the scheduler can be restricted by further constraints, like priority or fairness assumptions. They can be interpreted as a refinement of an abstract level where no constraint on the scheduler is enforced. This allowance to describe a more detailed scheduler is orthogonal to the problem of handling the interaction of inherent nondeterminism with the resolvable nondeterminism of an underspecified scheduler.

In probabilistic process algebras, see [20] for an overview, choice operators are extended with a distribution determining the way the different sides are favored. Randomized choice operators are special kinds of inherent nondeterminism. Nevertheless, it is important to examine inherent choice operators, which do not contain a distribution, for their own, since sometimes the distribution is not known and sometimes it does not exist at all (cf. Examples 1.2 and 1.3). Note that approaches trying to embed pure inherent nondeterminism into probabilistic settings lead to unnecessarily complex models and thus unnecessarily increase the cost of verification.

6 Conclusion

We have presented the structural operational semantics of a process algebra that handles choice operators corresponding to inherent nondeterminism as well as resolvable nondeterminism obtained by abstraction from the scheduling of the parallel composition or the renaming/hiding operator. In particular, μ-automata, which have two kinds of transition relations (one for action execution and one for resolution of resolvable nondeterminism), are used as underlying model for the structural operational semantics. In order to avoid any restriction on schedulers, the resolution of the resolvable nondeterminism has to be made undone if it was not affected by the previous execution step. A sound and complete axiom system has been presented, where in particular a choice operator representing resolvable nondeterminism is used. Note that the increased complexity of our semantics, compared to the standard semantics, is unavoidable if μ-calculus formulas should be preserved under refinement.

Our operational semantics can be straightforwardly adapted to process algebra having parallel composition with a CSP-based synchronization [15], sequential composition, or recursion. The only problem arises for unguarded recursion, i.e., if there exists a variable that is bound without being behind an action prefix: There the definition of the concretization relation $\rightrightarrows$ yields problems, since infinite derivation trees can be generated. This is not very critical, because usually process algebras without unguarded recursion are sufficient for applications.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002.

[2] J. C. M. Baeten. A brief history of process algebra. *Theor. Comput. Sci.*, 335(2-3):131–146, 2005.

[3] J. C. M. Baeten and J. A. Bergstra. Process algebra with partial choice. In B. Jonsson and J. Parrow, editors, *CONCUR*, volume 836 of *LNCS*, pages 465–480. Springer, 1994.

[4] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37:77–121, 1985.

[5] B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[6] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.

[7] S. Bornot and J. Sifakis. On the composition of hybrid systems. In T. A. Henzinger and S. Sastry, editors, *HSCC*, volume 1386 of *LNCS*, pages 49–63. Springer, 1998.

[8] R. Cousot, editor. *Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI 2005, Paris, France, January 17-19, 2005, Proceedings*, volume 3385 of *LNCS*. Springer, 2005.

[9] D. Dams and K. S. Namjoshi. Automata as abstractions. In Cousot [8], pages 216–232.

[10] L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *LICS*, pages 170–179. IEEE Computer Society Press, 2004.

[11] H. Fecher and M. Huth. Ranked predicate abstraction for branching time: Complete, incremental, and precise. In S. Graf and W. Zhang, editors, *ATVA*, volume 4218 of *LNCS*, pages 322–336. Springer, 2006.

[12] H. Fecher, H. Schmidt, and J. Schönborn. Refinement sensitive formal semantics of state machines having inherent nondeterminism. In *MODELS*, 2007. Available at `http://www.informatik.uni-kiel.de/~hsc/models07.pdf`, submitted for publication.

[13] O. Grumberg, M. Lange, M. Leucker, and S. Shoham. *Don't know* in the μ-calculus. In Cousot [8], pages 233–249.

[14] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proc. 11th IEEE Real-Time Systems Symposium (RTSS)*, Orlando, Fl., January 1990.

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

[16] D. Janin and I. Walukiewicz. Automata for the modal mu-calculus and related results. In J. Wiedermann and P. Hájek, editors, *Mathematical Foundations of Computer Science*, volume 969 of *LNCS*, pages 552–562. Springer, 1995.

[17] D. Kozen. Results on the propositional μ-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983.

[18] K. G. Larsen and B. Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE Computer Society Press, 1988.

[19] K. G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *LICS*, pages 108–117. IEEE Computer Society Press, 1990.

[20] N. López and M. Núñez. An overview of probabilistic process algebras and their equivalences. In C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems*, volume 2925 of *LNCS*, pages 89–123. Springer, 2004.

[21] M. E. Majster-Cederbaum. Underspecification for a simple process algebra of recursive processes. *Theor. Comput. Sci.*, 266:935–950, 2001.

[22] C. E. Martin, S. A. Curtis, and I. Rewitzky. Modelling nondeterminism. In D. Kozen and C. Shankland, editors, *MPC*, volume 3125 of *LNCS*, pages 228–251. Springer, 2004.

[23] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[24] S. Shoham and O. Grumberg. 3-valued abstraction: More precision at less cost. In *LICS*, pages 399–410. IEEE Computer Society Press, 2006.

[25] B. Thomsen. An extended bisimulation induced by a preorder on actions. Master's thesis, Aalborg University Centre, 1987.

[26] S. Veglioni and R. De Nicola. Possible worlds for process algebras. In D. Sangiorgi and R. de Simone, editors, *CONCUR*, volume 1466 of *LNCS*, pages 179–193. Springer, 1998.

[27] M. Walicki and S. Meldal. Algebraic approaches to nondeterminism: An overview. *ACM Comput. Surv.*, 29(1):30–81, 1997.