Tableau Methods for PA-Processes

Richard Mayr

Institut für Informatik, Technische Universität München,
Arcisstr. 21, D-80290 München, Germany;
e-mail: mayrri@informatik.tu-muenchen.de
Phone: +49 89 289-22397
Category (A)

Abstract. PA (Process algebra) is the name that has become common use to denote the algebra with a sequential and parallel operator (without communication), plus recursion. PA-processes are a superset of both Basic Parallel Processes (BPP) [Chr93] and context-free processes (BPA).
We study three problems for PA-processes: The reachability problem, the partial deadlock reachability problem ("Is it possible to reach a state where certain actions are not enabled ?") and the partial livelock reachability problem ("Is it possible to reach a state where certain actions are disabled forever ?"). We present sound and complete tableau systems for these problems and compare them to non-tableau algorithms.

Keywords: tableau systems, temporal logic, process algebras, PA-processes

1 Introduction

The Basic Process Algebra PA is a simple model of infinite state concurrent systems. It has operators for nondeterministic choice, parallel composition and sequential composition. PA-processes and Petri nets are incomparable, meaning that neither model is more expressive than the other one. PA is not a syntactical subset of CCS [Mil89], because CCS does not have an explicit operator for sequential composition. However, as CCS can simulate sequential composition by parallel composition and synchronization, PA is still a weaker model than CCS. PA-processes are a superset of both Basic Parallel Processes (BPP) [Chr93] and context-free processes (BPA).

We study three problems for PA-processes:

1. The reachability problem: "Is a state reachable from the initial state ?"
2. The partial deadlock reachability problem: "Is it possible to reach a state s s.t. no action a in a given set of actions A is enabled ?" (If A is the set of all actions then the state s is a deadlock)
3. The partial livelock reachability problem: "Is it possible to reach a state s s.t. for every state s' reachable from s no action a in a given set of actions A is enabled ?" (If A is the set of all actions then the state s is a deadlock)

We present tableau systems that solve these problems and compare them to non-tableau algorithms, pointing out strengths and weaknesses of tableau techniques. While the reachability problem is NP-complete for PA-processes, the partial deadlock reachability problem and the partial livelock reachability problem only require polynomial time.

In section 2 we define PA-processes. Sections 3, 4 and 5 are about partial deadlock reachability, partial livelock reachability and reachability. The paper closes with a section that summarizes the results, compares tableau systems to iterative algorithms and mentions some related work.

2 Preliminaries

The definition of PA is as follows: Assume a countably infinite set of atomic actions $Act = \{a, b, c, \ldots\}$ and a countably infinite set of process variables $Var = \{X, Y, Z, \ldots\}$. The class of PA expressions is defined by the following abstract syntax

$$E ::= \epsilon \mid X \mid aE \mid E + E \mid E \| E \mid E.E$$

A PA is defined by a family of recursive equations $\{X_i := E_i \mid 1 \leq i \leq n\}$, where the X_i are distinct and the E_i are PA expressions at most containing the variables $\{X_1, \ldots, X_n\}$. We assume that every variable occurrence in the E_i is *guarded*, i.e. appears within the scope of an action prefix, which ensures that PA-processes generate finitely branching transition graphs. This would not be true if unguarded expressions were allowed. For example, the process $X := a + a \| X$ generates an infinitely branching transition graph. For every $a \in Act$ the transition relation $\xrightarrow{a}$ is the least relation satisfying the following inference rules:

$$aE \xrightarrow{a} E \qquad \frac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'} \qquad \frac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'} \qquad \frac{E \xrightarrow{a} E'}{X \xrightarrow{a} E'}(X := E)$$

$$\frac{E \xrightarrow{a} E'}{E \| F \xrightarrow{a} E' \| F} \qquad \frac{F \xrightarrow{a} F'}{E \| F \xrightarrow{a} E \| F'} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F}$$

Alternatively, PA-processes can be represented by a state described by a term of the form

$$G ::= \epsilon \mid X \mid G_1.G_2 \mid G_1 \| G_2$$

and a set of rules Δ of the form $X \xrightarrow{a} G$ whose application to states must respect sequential composition. This is described by the following inference rules:

$$X \xrightarrow{a} G \quad \text{if } (X \xrightarrow{a} G) \in \Delta$$

$$\frac{E \xrightarrow{a} E'}{E \| F \xrightarrow{a} E' \| F} \qquad \frac{F \xrightarrow{a} F'}{E \| F \xrightarrow{a} E \| F'} \qquad \frac{E \xrightarrow{a} E'}{E.F \xrightarrow{a} E'.F}$$

We assume w.r. that for every variable X there is at least one rule $X \xrightarrow{a} t$. The property $\exists t'.\ t \xrightarrow{a} t'$ will also be denoted by $t \xrightarrow{a}$. Sequences of actions will be denoted by σ.

Basic Parallel Processes (BPP) are the subset of PA-processes without sequential composition, while context-free processes (BPA) are the subset of PA-processes without parallel composition.

While PA-processes and Petri nets are incomparable, there is a one-to-one correspondence between BPPs and a class of labelled Petri nets, the so called *communication-free nets* [Esp96]. In these nets every transition has exactly one input place with an arc labelled by 1. The translation of a BPP algebra into a communication-free net goes as follows: Introduce a place for each process variable and a transition for each transition rule. For a rule $X \xrightarrow{a} Y_1^{m_1} \| \ldots \| Y_n^{m_n}$ introduce a transition t labelled by a, an arc labelled by 1 leading from place X to t and arcs labelled by m_i leading from t to places Y_i. The other direction is analogous.

3 Partial Deadlock Reachability

Partial deadlock reachability problem:
Instance: A labelled transition system (LTS) with initial state r_0 and a set of atomic actions A.
Question: Is it possible to reach a state r s.t. $\not\exists a \in A.\ r \xrightarrow{a}$?

In the special case of $A = Act$ this is the problem if a state of deadlock is reachable.

For general Petri nets the partial deadlock reachability problem is equivalent to the problem of deciding if a state is reachable where certain places are unmarked. This problem has the same complexity as the reachability problem. So the partial deadlock reachability problem which is decidable, but at least EXPSPACE-hard [May84].

The situation is different for Basic Parallel Processes (BPP). While the reachability problem for BPP is NP-complete [Esp95], the partial deadlock reachability problem can be decided in polynomial time. It suffices to prove the property for communication-free nets, as they are equivalent to BPPs.

Now we define A-traps, a generalization of traps for labelled Petri nets. Although they can be defined for general Petri nets, they are only useful for labelled communication-free nets.

Definition 1. Let $N = (S, T, W, L, Act)$ be a labelled communication-free Petri net with places S, transitions T, a weight function W and a labeling function $L : T \rightarrow Act$, where Act is a set of atomic actions. Let $A \subseteq Act$ be a set of actions. An *A-trap* $U \subseteq S$ is a trap of N s.t. $\forall s \in U\ \exists t \in (s\cdot).\ L(t) \in A$.

Lemma 2. *The partial deadlock reachability problem for BPP is decidable in polynomial time.*

Proof Consider the corresponding communication-free net N with initial marking Σ_0. Let M be the maximal A-trap in N. It is possible to reach a state s.t. no action in A is enabled iff M is unmarked by Σ_0.

The maximal A-trap can be computed in polynomial time. Therefore the partial deadlock reachability problem can be decided in polynomial time. □

Later we will show that this result even holds for PA-processes by using a completely different (non-Petri net) technique.

Now we present a sound and complete tableau system that solves the partial deadlock reachability problem for PA-processes. First we define some predicates:

Definition 3. Let $A \subseteq Act$ be a set of actions. Sequences of actions are denoted by σ.

$$S_A(t) :\iff \exists \sigma, t'.\ t \xrightarrow{\sigma} t'.\ t' \neq \epsilon \wedge \ \not\exists a \in A.\ t' \xrightarrow{a} \tag{1}$$

$$\mathcal{E}(t) :\iff \exists t \xrightarrow{\sigma} \epsilon \tag{2}$$

$$D_A(t) :\iff \exists \sigma, t'.\ t \xrightarrow{\sigma} t'.\ \not\exists a \in A.\ t' \xrightarrow{a} \tag{3}$$

It is clear that $D_A(t) \iff S_A(t) \vee \mathcal{E}(t)$.

The nodes in the tableau will be sequences of formulae, that are interpreted conjunctively. Such sequences will be denoted by Γ. The branches of the proof tree are interpreted disjunctively. So the tableau will be successful iff there is a successful leaf.

In order to avoid carrying around unnecessary formulae we use the following "cleanup-procedure" after each rule application: If a formula Φ occurs more than once in a sequence Γ, then delete all occurrences but the first. The rules for the construction of the tableau are as follows:

$$\text{D}\ \ \frac{D_A(t), \Gamma}{S_A(t), \Gamma \quad \mathcal{E}(t), \Gamma}$$

$$\text{SD}\ \ \frac{S_A(t_1.t_2), \Gamma}{S_A(t_1), \Gamma \quad \mathcal{E}(t_1), S_A(t_2), \Gamma}$$

$$\text{PD}\ \ \frac{S_A(t_1 \| t_2), \Gamma}{S_A(t_1), D_A(t_2), \Gamma \quad D_A(t_1), S_A(t_2), \Gamma}$$

$$\text{ESD}\ \ \frac{\mathcal{E}(t_1.t_2), \Gamma}{\mathcal{E}(t_1), \mathcal{E}(t_2), \Gamma}$$

$$\text{EPD}\ \ \frac{\mathcal{E}(t_1 \| t_2), \Gamma}{\mathcal{E}(t_1), \mathcal{E}(t_2), \Gamma}$$

$$\text{E}\ \ \frac{S_A(X), \Gamma}{\Gamma} \quad \text{if } \not\exists a \in A.\ X \xrightarrow{a}$$

$$\text{Step1}\ \ \frac{S_A(X), \Gamma}{S_A(t_1), \Gamma \quad \ldots \quad S_A(t_k), \Gamma} \quad \text{where } X \rightarrow t_i \text{ and } \exists a \in A.\ X \xrightarrow{a}$$

$$\text{Step2}\ \ \frac{\mathcal{E}(X), \Gamma}{\mathcal{E}(t_1), \Gamma \quad \ldots \quad \mathcal{E}(t_k), \Gamma} \quad \text{where } X \xrightarrow{a} t_i$$

$$\text{EE}\ \ \frac{\mathcal{E}(\epsilon), \Gamma}{\Gamma}$$

Note that the applicable rule and the result of the rule application is uniquely determined by the sequent.

Proposition 4. *For every valid instance of a rule, the antecedent is true if and only if one of its consequents is true.*

Definition 5. (Termination conditions) A node n in the tableau consisting of a sequence of formulae Γ is a terminal node if either

1. Γ is empty
2. On the path from the root-node to the node n, there is a node n' with a sequence Γ' s.t. $\Gamma' \subseteq \Gamma$. (Every formula in Γ' also occurs in Γ).

Terminals of type 1 are successful, while terminals of type 2 are unsuccessful.

Lemma 6. *The tableau is finite.*

Proof There are only finitely many subterms of the root-term and only finitely many subterms of the terms t on the right hand side of rules $X \xrightarrow{a} t$. So there are only finitely many different sequents in the tableau, because of the "cleanup"-condition. So no branch in the tableau can have infinite length, due to termination condition 2. As the tableau is finitely branching the result follows. □

Theorem 7. *For every PA-process t and every set of actions $A \subseteq Act$, $D_A(t)$ holds iff there is a successful tableau with root $D_A(t)$.*

Proof

1. If there is a successful tableau, then it has a leaf marked by the empty sequence. This leaf is certainly true. By Proposition 4 the root-formula $D_A(t)$ must be true.
2. If $D_A(t)/S_A(t)/\mathcal{E}(t)$ holds then there is a sequence σ of finite length s.t. $t \xrightarrow{\sigma} t'$ and $\not\exists a \in A.\ t' \xrightarrow{a}$, with $t' \neq \epsilon$ or $t' = \epsilon$, respectively. Choose such a sequence σ of minimal length and let $n := length(\sigma)$. We prove by induction on lexicographically ordered pairs $(n, size(t))$ that if $D_A(t)/S_A(t)/\mathcal{E}(t)$ is true, then there is a successful tableau with root $D_A(t)/S_A(t)/\mathcal{E}(t)$.
   (a) If $n = 0$ then $t = \epsilon$ or $t = (X.t_1)||t_2$ s.t. $D_A(t_2)$ and $\not\exists a \in A.X \xrightarrow{a}$. ($t_1$ and t_2 may be ϵ) If $t \neq \epsilon$, then by the D-rule we get $S_A(t)$ and by the PD-rule $S_A(X.t_1), D_A(t_2)$. The SD-rule yields $S_A(X), D_A(t_2)$ and by the E-rule we get $D_A(t_2)$. By induction hypothesis there is a correct tableau for $D_A(t_2)$. If $t = \epsilon$, then the D- and EE-rules yield a successful tableau.
   (b) If $n > 0$ then apply the tableau-rules. By Proposition 4 the rules are valid, so least one of the children is true. Also either the terms are smaller or (in case of the Step-rules) the new sequence σ' is shorter. Thus by induction hypothesis there is a successful tableau. This construction can't be stopped by termination condition 2, because this would contradict the minimality of the length of σ.

□

The tableau built by these rules is very redundant and can contain branches of exponential length. However, it is still possible to find a polynomial algorithm for the problem.

Theorem 8. *The partial deadlock reachability problem for PA-processes t is decidable in polynomial time.*

Proof

1. First we decide $\mathcal{E}(X)$ for all variables X. This is done by a marking algorithm. First mark all variables X s.t. there is a rule $X \xrightarrow{a} \epsilon$. If there is a rule $Y \xrightarrow{a} t$ s.t. Y is unmarked so far and all variables in t are marked then mark Y. Do this until no new variable can be marked. This requires polynomial time.
2. Now decide $S_A(X)$ for all variables X. First mark all variables X s.t. $\not\exists a \in A. X \xrightarrow{a}$.
   For all rules $Y \xrightarrow{a} t_1.t_2$ or $Y \xrightarrow{a} t_1 \| t_2$ s.t. Y is unmarked use the rules D, SD, PD and E and the already acquired knowledge about $\mathcal{E}(Z)$ and $S_A(Z)$ to prove $S_A(Y)$. In addition to the normal failure cases this fails when we encounter unmarked variables. Repeat this until all attempts for all rules fail. This takes polynomial time.
3. As we know the values of $\mathcal{E}(X)$ and $S_A(X)$ for every variable X we can decide $D_A(t)$ in polynomial time with the help of the rules D, SD, PD and E.

□

4 Partial Livelock Reachability

Partial livelock reachability problem:
Instance: A labelled transition system (LTS) with initial state r_0 and a set of atomic actions A.
Question: Is is possible to reach a state r s.t. no state r' that is reachable from r enables any action in A ?

In the special case of $A = Act$ this is the problem if a state of deadlock is reachable.

Now we present a sound and complete tableau system that solves the partial livelock reachability problem for PA-processes. First we define some predicates:

Definition 9. Let $A \subseteq Act$ be a set of actions.

$$\begin{aligned}
\mathcal{E}(t) &:\Longleftrightarrow \exists t \xrightarrow{\sigma} \epsilon \\
En_A(t) &:\Longleftrightarrow \exists a \in A.\ t \xrightarrow{a} \\
N_A(t) &:\Longleftrightarrow \not\exists t \xrightarrow{\sigma} t'.\ En_A(t') \\
R_A(t) &:\Longleftrightarrow \exists t \xrightarrow{\sigma} t'.\ N_A(t') \\
RI_A(t) &:\Longleftrightarrow \exists t \xrightarrow{\sigma} t'.N_A(t') \wedge \neg\mathcal{E}(t')
\end{aligned}$$

Intuitively $N_A(t)$ means that t will never be able to do any action from A, and $R_A(t)$ means that a state t' is reachable from t where all actions in A are disabled forever. $RI_A(t)$ means that a state t' is reachable from t that has no terminating computation and where all actions in A are disabled forever. This can also be expressed in temporal logic: Let $A := \{a_1, \ldots, a_n\}$.

$$N_A(t) = t \models \Box(\neg a_1 \wedge \ldots \wedge \neg a_n)$$
$$R_A(t) = t \models \Diamond\Box(\neg a_1 \wedge \ldots \wedge \neg a_n)$$

The partial livelock reachability problem is to check if some PA-process t satisfies $R_A(t)$.

The branches in the tableau are interpreted conjunctively, so the tableau is successful if all branches are successful. A branch is successful if it ends with a successful leaf. As the rules for the construction of the tableau are nondeterministic, the tableau is not unique. However, $R_A(t)$ holds iff there is at least one successful tableau with root $R_A(t)$.

$$\text{RPD}\ \frac{R_A(t_1 \| t_2)}{R_A(t_1) \quad R_A(t_2)}$$

$$\text{RSD 1}\ \frac{R_A(t_1.t_2)}{R_A(t_2)}\ (\mathcal{E}(t_1))$$

$$\text{RSD 2}\ \frac{R_A(t_1.t_2)}{RI_A(t_1)}$$

$$\text{Found}\ \frac{R_A(X)}{N_A(X)}$$

$$\text{Step R}\ \frac{R_A(X)}{R_A(t_i)}\ (X \to t_i)$$

$$\text{RIPD 1}\ \frac{RI_A(t_1 \| t_2)}{RI_A(t_1) \quad R_A(t_2)}$$

$$\text{RIPD 2}\ \frac{RI_A(t_1 \| t_2)}{R_A(t_1) \quad RI_A(t_2)}$$

$$\text{RISD 1}\ \frac{RI_A(t_1.t_2)}{RI_A(t_1)}$$

$$\text{RISD 2}\ \frac{RI_A(t_1.t_2)}{RI_A(t_2)}\ (\mathcal{E}(t_1))$$

$$\text{Found RI}\ \frac{RI_A(X)}{N_A(X)}\ (\neg\mathcal{E}(X))$$

$$\text{Step RI}\ \frac{RI_A(X)}{RI_A(t_i)}\ (X \to t_i)$$

$$\text{Step N}\ \frac{N_A(X)}{N_A(t_1) \quad \ldots \quad N_A(t_n)}\ (\neg En_A(X),\ X \to t_i, i = 1, \ldots, n)$$

$$\text{NPD}\ \frac{N_A(t_1 \| t_2)}{N_A(t_1) \quad N_A(t_2)}$$

$$\text{NSD 1} \quad \frac{N_A(t_1.t_2)}{N_A(t_1)} \; (\neg\mathcal{E}(t_1))$$

$$\text{NSD 2} \quad \frac{N_A(t_1.t_2)}{N_A(t_1) \quad N_A(t_2)}$$

Definition 10. A node in the tableau is a terminal node if the formula is either

1. $N_A(\epsilon)$
2. $N_A(t)$ and a node with the same formula occurred earlier in the branch.
3. $R_A(\epsilon)$
4. $R_A(t)$ and a node with the same formula occurred earlier in the branch.
5. $RI_A(\epsilon)$
6. $RI_A(t)$ and a node with the same formula occurred earlier in the branch.
7. $N_A(X)$ and $En_A(X)$ is true.

Terminals 1,2,3 are successful, while the terminals of types 4,5,6,7 are unsuccessful.

Proposition 11. *If all consequents of an instance of a rule are true and the side conditions are satisfied then the antecedent is true.*

Lemma 12. *If a formula F in the tableau is true and the node is not a terminal node, then there is an instance of a rule with antecedent F and satisfied side conditions s.t. all consequents are true.*

Proof

1. If F is of the form $R_A(t_1.t_2)$, then there are two cases:
   (a) If $\neg\mathcal{E}(t_1)$ then $t_1.t_2$ is equivalent to t_1, and $RI_A(t_1)$. So the rule is RSD 2.
   (b) If $\mathcal{E}(t_1)$ then there are two cases:
      i. $t_1 \longrightarrow \epsilon$ and $t_2 \longrightarrow t_2'$ with $N_A(t_2')$. So we have $R_A(t_2)$ and the rule is RSD 1.
      ii. $t_1.t_2 \longrightarrow t_1'.t_2$ with $N_A(t_1'.t_2)$. If $\mathcal{E}(t_1')$ then $N_A(t_2)$ and thus $R_A(t_2)$ and the rule is RSD 1. If $\neg\mathcal{E}(t_1')$ then $t_1'.t_2$ is equivalent to t_1' and $RI(t_1)$ and the rule is RSD 2.
2. If F is of the form $R_A(X)$ then either we have already reached a state with $N_A(X)$ (rule Found) or we can reach one from one of the successor states t_i that are reachable from X in one step (rule Step R).
3. If F is of the form $RI_A(t_1 \| t_2)$ then $t_1 \xrightarrow{\sigma_1} t_1'$, $t_2 \xrightarrow{\sigma_2} t_2'$ s.t. $N_A(t_1')$, $N_A(t_2')$ and at least one of t_1', t_2' has no terminating computation. These two cases are covered in rules RIPD 1 and RIPD 2.
4. For the case of $F = RI_A(t_1.t_2)$ the argument is equivalent to the one in case 1a.
5. If $F = RI_A(X)$ then either we have already found the correct X s.t. $N_A(X)$ and X can never terminate $(\neg\mathcal{E}(X))$, or we can find it from at least one of the successor states t_i. The first case is rule Found RI, and the second case is rule Step RI.

6. If $F = N_A(t_1.t_2)$ then there are two cases.
   (a) If $\neg\mathcal{E}(t_1)$ then $t_1.t_2$ is equivalent to t_1 and thus $N_A(t_1)$. The rule is NSD 1.
   (b) If $\mathcal{E}(t_1)$ then both $N_A(t_1)$ and $N_A(t_2)$ must hold. The rule is NSD 2.

All the other rules RPD, Step N and NPD are sound and complete. □

Lemma 13. *All possible tableaux are finite.*

Proof directly by definition of the rules and termination conditions 2,4 and 6. □

A node in the tableau is called true iff the formula at this node holds (see Def. 9).

Lemma 14. *If $N_A(t)$ holds then there is a successful tableau with root $N_A(t)$.*

Proof If a node is true then by Lemma 12 and Lemma 13 we can (by choosing the right rules) construct a finite tableau s.t. every node is true. If the root node is $N_A(t)$ then every node in the tableau is of the form $N_A(t')$ for some t'. No branch can terminate with termination condition 7, because every node is true. So every branch must end with a terminal node of type 1 or 2 and is successful. □

Lemma 15. *If $N_A(t)$ doesn't hold then every tableau with root $N_A(t)$ has an unsuccessful branch.*

Proof If $N_A(t)$ doesn't hold then there is a sequence σ s.t. $t \xrightarrow{\sigma} t'$ and $\exists a \in A.\ t' \xrightarrow{a}$. Choose such a sequence of minimal length. By induction on the length of σ we can prove the existence of a branch ending with a terminal of type 7. In the construction we always choose the rule and its consequent according to σ. The construction cannot terminate by condition 1, because every node in the branch is false and it cannot terminate by condition 2, because σ has minimal length. □

Lemma 16. *Let t be a PA-term and $A \subseteq Act$. If $R_A(t)$ ($RI_A(t)$) holds then there is a successful tableau with root $R_A(t)$ ($RI_A(t)$).*

Proof There is a sequence σ s.t. $t \xrightarrow{\sigma} t'$ and $N_A(t')$ (and additionally $\neg\mathcal{E}(t')$ if $RI_A(t)$). Choose such a sequence of minimal length. We prove by induction on lexicographically ordered pairs $(length(\sigma), size(t))$ that there is a successful tableau with root $R_A(t)$ ($RI_A(t)$).

If a node is true then by Lemma 12 and Lemma 13 we can (by choosing the right rules) construct a finite tableau s.t. every node is true.

1. If $size(t) = 0$ then $length(\sigma) = 0$ and $t = \epsilon$. In this case $RI_A(t)$ doesn't hold. If $R_A(t)$ holds, then we already have a successful tableau, because $R_A(\epsilon)$ is a successful terminal of type 3.
2. (a) If $size(t) = 1$ then $t = X$, where X is a variable. There are two cases:
      i. If $length(\sigma) = 0$ then apply the rule Found (Found RI) and $N_A(t)$ is true. By Lemma 14 there is a successful tableau.

   ii. If $length(\sigma) > 0$ then apply the right instance of the rule Step R (Step RI) s.t. the consequent is true. The size of the new term may be larger, but the length of the rest of the sequence is shorter, so the induction hypothesis applies.

(b) If $size(t) >= 2$ then t is of the form $t_1 || t_2$ or $t_1.t_2$. Use the right rule from RPD, RSD 1, RSD 2 (RIPD 1, RIPD 2, RISD 1, RISD 2). The new term is smaller and the sequence is certainly not longer. Thus it is possible to apply the induction hypothesis.

This construction cannot be stopped by termination condition 4 or 6, because σ is of minimal length. □

Lemma 17. *Let t be a PA-term and $A \subseteq Act$. If $R_A(t)$ ($RI_A(t)$) doesn't hold then all tableaux with root $R_A(t)$ ($RI_A(t)$) are unsuccessful.*

Proof It suffices to show that there is at least one unsuccessful branch. If a node is false then by Proposition 11 at least one child node must be false. So there is a branch where every node is false. There are two cases:

1. If every node in the branch is of the form $R_A(t')$ or $RI_A(t')$ then the branch must end with an unsuccessful terminal of type 4 or 6.
2. If a node of the form $N_A(t')$ occurs in the branch, then by Lemma 15 the tableau has an unsuccessful branch.

□

Theorem 18. *Let t be a PA-term and $A \subseteq Act$. Then $R_A(t)$ holds iff there is a successful tableau with root $R_A(t)$.*

Proof By Lemma 16 and Lemma 17. □

The complexity of the algorithm obtained from the tableau isn't even in NP, but in PSPACE. On the other hand it is possible to find a polynomial time algorithm that decides $R_A(t)$. This algorithm proceeds in four steps:

1. Decide $\mathcal{E}(X)$ for all variables X.
2. Decide $N_A(X)$ for all variables X using the previously collected information.
3. Decide $R_A(X)$ and $RI_A(X)$ for all X using the previously collected information.
4. Decide $R_A(t)$ using the previously collected information.

The first part uses the algorithm described in the proof of Theorem 8. It requires polynomial time.

For the second part define $P_A(t) := \neg N_A(t)$. Now use a marking algorithm to decide $P_A(X)$ for every X.

Start Mark all variables X s.t. $\exists a \in A. X \xrightarrow{a}$.

Step For every unmarked X and every rule $X \to t$ do if $P'_A(t)$ then mark X. P'_A is defined by

$$P'_A(t_1 \| t_2) = P'_A(t_1) \vee P'_A(t_2)$$
$$P'_A(t_1.t_2) = P'_A(t_1) \vee (\mathcal{E}(t_1) \wedge P'_A(t_2))$$
$$P'_A(X) = \text{if } X \text{ is marked then true else false}$$

Repeat Step until no new variable is marked.

$N_A(X)$ is true iff X is not marked. This algorithm also requires polynomial time.

The third part is a marking algorithm that uses two different markings R_A and RI_A. If a variable X is marked by R_A (RI_A) then it means that it is already known that $R_A(X)$ ($RI_A(X)$) is true.

Start If $N_A(X)$ then mark X with R_A. If $N_A(X)$ and $\neg\mathcal{E}(X)$ then mark X with RI_A.

Step For every X and every rule $X \to t$ do: If $R'_A(t)$ then mark X with R_A. If $RI'_A(t)$ then mark X with RI_A. The functions R'_A and RI'_A are defined by

$$R'_A(t_1 \| t_2) = R'_A(t_1) \wedge R'_A(t_2)$$
$$R'_A(t_1.t_2) = RI'_A(t_1) \vee (\mathcal{E}(t_1) \wedge R'_A(t_2))$$
$$R'_A(X) = \text{if } X \text{ is marked by } R_A \text{ then true else false}$$

$$RI'_A(t_1 \| t_2) = RI'_A(t_1) \wedge R'_A(t_2) \vee R'_A(t_1) \wedge RI'_A(t_2)$$
$$RI'_A(t_1.t_2) = RI'_A(t_1) \vee (\mathcal{E}(t_1) \wedge RI'_A(t_2))$$
$$RI'_A(X) = \text{if } X \text{ is marked by } RI_A \text{ then true else false}$$

Repeat Step until no new variable is marked.

In the last step just use the rules for the functions used in the third step and the previously collected information to decide $R_A(t)$. This also takes polynomial time.

Theorem 19. *Let t be a PA term and $A \subseteq Act$. It is decidable in polynomial time if $R_A(t)$.*

5 The Reachability Problem for PA

We present two algorithms that decide the reachability problem for PA-processes. The first one does a simple bounded search, while the second one is tableau-based and works by decomposition.

First we define the size of an instance of the problem:

Definition 20. For convenience of notation we also write $t_1 \succ t_2$ for $\exists \sigma.\ t_1 \xrightarrow{\sigma} t_2$. An instance of the reachability problem for PA-processes consists of two processes t_1, t_2 and a set of rules Δ of the form $X \xrightarrow{a} t$. The question is if $t_1 \succ t_2$. The size of the instance is $size(t_1) + size(t_2) + \sum_{(X \xrightarrow{a} t) \in \Delta} size(t)$.

Note that this definition is different from the one for BPPs. There terms of the form $Y^n = Y\|\ldots\|Y$ are allowed whose size is $log(n)$, while for PA the size of this term would be n. Even for this weaker definition of the size, the reachability problem is NP-hard [Esp95] (even for BPPs).

Lemma 21. *The reachability problem for PA is in PSPACE.*

Proof Let n be the size of the instance of the problem. It can be shown that if a state t can be reached from the initial state t_0, then it can be reached via a path s.t. the size of every intermediate state t' is bounded by a constant $c \leq O(n^2)$. Details can be found in [May96b]. $\square$

The argument of Lemma 21 is somewhat crude, because it does not take the internal structure of PA-processes into account. By using this structure we can derive a more accurate complexity bound.

Now we present the sound and complete tableau system that solves the reachability problem for PA-processes. The nodes in the tableau will be sequences of formulae, that are interpreted conjunctively. Such sequences will be denoted by Γ. The branches of the proof tree are interpreted disjunctively. So the tableau is successful iff there is a successful leaf. To avoid carrying around unnecessary formulae we use the following "cleanup-procedure" after each rule application: If a formula Φ occurs more than once in a sequence Γ, then delete all occurrences but the first.

$$\text{SP} \quad \frac{t_1.t_2 \succ t_3\|t_4, \Gamma}{t_1 \succ \epsilon, t_2 \succ t_3\|t_4, \Gamma}$$

$$\text{PS} \quad \frac{t_1\|t_2 \succ t_3.t_4, \Gamma}{t_1 \succ t_3.t_4, t_2 \succ \epsilon, \Gamma \quad t_1 \succ \epsilon, t_2 \succ t_3.t_4, \Gamma}$$

$$\text{SS} \quad \frac{t_1.t_2 \succ t'_1.t'_2.\ldots.t'_k, \Gamma}{t_1 \succ \epsilon, t_2 \succ t'_1.\ldots.t'_k, \Gamma \quad \ldots \quad t_1 \succ t'_1.\ldots.t'_i, t_2 = t'_{i+1}.\ldots.t'_k, \Gamma \quad \ldots}$$

for $i = 1, \ldots, k-1$ and no t'_j is a sequential composition

$$\text{PP} \quad \frac{t_1\|t_2 \succ t'_1\|t'_2\|\ldots\|t'_k, \Gamma}{\ldots \; t_1 \succ \|M, t_2 \succ \|(\{t'_1, \ldots, t'_k\} - M), \Gamma \; \ldots} \quad \text{for every } M \subseteq \{t'_1, \ldots, t'_k\}$$

and no t'_i is a parallel composition

$$\text{Step} \quad \frac{X \succ t, \Gamma}{s_1 \succ t, \Gamma \quad \ldots \quad s_k \succ t, \Gamma} \quad \text{for } X \neq t \text{ and } X \rightarrow s_i$$

$$\text{E1} \quad \frac{t \succ t, \Gamma}{\Gamma}$$

$$\text{E2} \quad \frac{t = t, \Gamma}{\Gamma}$$

Note that the applicable rule and the result of the rule application is uniquely determined by the sequent.

Definition 22. A node n consisting of a sequence of formulae Γ is a terminal node if one of the following conditions is satisfied.

1. Γ is empty
2. $\Gamma = \epsilon \succ t, \Gamma'$ and $t \neq \epsilon$.
3. $\Gamma = (t = t'), \Gamma'$ and $t \neq t'$.
4. On the path from the root-node to the node n, there is a node n' with a sequence Γ' s.t. $\Gamma' \subseteq \Gamma$. (Every formula in Γ' occurs in Γ.)

Terminals of type 1 are successful, while terminals of types 2–4 are unsuccessful.

Lemma 23. *The tableau is finite.*

Proof There are only finitely many subterms of the root-term and only finitely many subterms of the terms t on the right hand side of rules $X \xrightarrow{a} t$. So there are only finitely many different sequents in the tableau, because of the "cleanup"-condition. So no branch in the tableau can have infinite length, due to termination condition 4. As the tableau is finitely branching the result follows. □

Now we show the soundness and completeness of the tableau-rules.

Lemma 24. *The antecedent of an instance of a tableau rule is true if and only if at least one of the consequents is true.*

Proof

SP Assume $t_1.t_2 \xrightarrow{\sigma} t_3||t_4$. If t_1 is not reduced to ϵ in σ then $t_1.t_2 \xrightarrow{\sigma} t_1'.t_2$ for some t_1'. This cannot be equivalent to $t_3||t_4$, because the outermost operator is different. Therefore $\sigma = \sigma_1\sigma_2$ s.t. $t_1 \xrightarrow{\sigma_1} \epsilon$ and $t_2 \xrightarrow{\sigma_2} t_3||t_4$. The other direction is obvious.

PS If $t_1||t_2 \xrightarrow{\sigma} t_3.t_4$ then t_1 or t_2 must be reduced to ϵ in σ, because otherwise the outermost operator of the result would be parallel composition instead of sequential composition. So σ must consist of two subsequences σ_1 and σ_2 s.t. $t_1 \xrightarrow{\sigma_1} \epsilon$ and $t_2 \xrightarrow{\sigma_2} t_3.t_4$ or vice versa. The other direction is obvious.

SS If $t_1.t_2 \succ t_1'.t_2'.\ldots.t_k'$ is true, then there are two cases:

1. If t_1 is reduced to ϵ in the rest of the sequence, then $t_2 \succ t_1'.t_2'.\ldots.t_k'$ must be true.
2. If t_1 is not reduced to ϵ in the rest of the sequence, then t_1 must develop into a prefix $t_1'.\ldots.t_i'$ of $t_1'.\ldots.t_k'$. In this case t_2 cannot change in the sequence and must therefore be equal to $t_{i+1}'.\ldots.t_k'$. As $t_2 \neq \epsilon$ it follows that $i < k$.

As no t_j' is a sequential composition, it isn't possible that t_1 develops into a part of a t_j' and t_2 into the rest. The other direction is trivial.

PP If $t_1||t_2 \succ t_1'||t_2'||\ldots||t_k'$ then t_1 must develop into the parallel composition of a (possibly empty) subset of $\{t_1', \ldots, t_k'\}$ and t_2 into the parallel composition of the rest. As no t_j' is a parallel composition, it isn't possible that t_1 develops into a part of a t_j' and t_2 into the rest.

The other rules *Step*, E_1 and E_2 are obvious. □

Lemma 25. *A PA-process t_2 is reachable from a PA-process t_1 iff there is a successful tableau with root $t_1 \succ t_2$.*

Proof

1. If the tableau is successful, then there is a successful (empty) leaf that is certainly true. By Lemma 24 the root sequent is also true.
2. If t_2 is reachable from t_1 then there exists a sequence σ of minimal length n s.t. $t_1 \xrightarrow{\sigma} t_2$. We prove by induction on lexicographically ordered pairs $(n, size(t_1) + size(t_2))$ that there is a successful path in the tableau. The base case is trivial. Otherwise apply the tableau rules. By Lemma 24 at least one of the consequents must be true, and by induction hypothesis there must be a successful tableau for it. This construction cannot terminate with termination condition 4, because this would contradict the minimality of the length of σ.

□

The tableau system does not yield an NP-algorithm, because it can contain branches of exponential length. However, this can be avoided by a slight modification. To show containment in NP we modify the tableau system by avoiding some redundant parts.

Theorem 26. *The reachability problem for PA-processes is NP-complete.*

Proof By using the same algorithm as in the proof of Theorem 8 we can decide $\mathcal{E}(X)$ for every variable X in polynomial time. Using this knowledge we can decide $t \succ \epsilon$ using $O(length(t))$ time. Now in the tableau system formulae of the form $t \succ \epsilon$ are decided immediately when they appear, using only polynomial time. Now if $t_1 \succ t_2$ is true, then there is a successful branch of polynomial length. As the formulae on this branch only have polynomial length the branch can be described in polynomial space. We can nondeterministically guess a branch of polynomial size and check in polynomial time if it is a valid successful branch. To do this we use the tableau rules and the polynomial algorithm for deciding $t \succ \epsilon$ on the way.

As the reachability problem is NP-hard even for the weaker model of BPP [Esp95] the result follows. □

6 Conclusion

All three investigated problems for PA-processes have the same complexities as for BPPs. While reachability is NP-complete, partial deadlock reachability and partial livelock reachability can be decided in polynomial time.

Tableau systems often do not yield very efficient algorithms, because the tableaux contain many redundant parts. The tableau in section 3 is an example for this. On the other hand the efficiency of tableau methods can be increased significantly by using small non-tableau algorithms to decide side-conditions during the construction of the tableau. This technique has been used in the modified tableau method for the reachability problem in the proof of Theorem 26.

Tableau techniques are a common tool for the analysis of systems described by Petri nets or process algebras. They have been used for model checking problems for various temporal logics [BEM96, May96c, May96a]. The three problems studied in this paper can be seen as special cases of a general model checking problem for PA-processes. This problem has recently been shown to be decidable with the help of tableau techniques [May96a]. However, the algorithm in [May96a] has an extremely high complexity (non-elementary) and thus it is useful to consider these subproblems here.

Although the tableau-methods presented here are sound and complete, they are not intended to be used as a decision procedure, but rather as a proof method that can be implemented in the framework of a theorem prover with human interaction.

References

[BEM96] J. Bradfield, J. Esparza, and A. Mader. An effective tableau system for the linear time μ-calculus. In F. Meyer auf der Heide and B. Monien, editors, *Proceedings of ICALP'96*, number 1099 in LNCS. Springer Verlag, 1996.

[Chr93] S. Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, Edinburgh University, 1993.

[Esp95] Javier Esparza. Petri nets, commutative context-free grammars and basic parallel processes. In Horst Reichel, editor, *Fundamentals of Computation Theory*, number 965 in LNCS. Springer Verlag, 1995.

[Esp96] J. Esparza. More infinite results. In B. Steffen and T. Margaria, editors, *Proceedings of INFINITY'96*, number MIP-9614 in Technical report series of the University of Passau. University of Passau, 1996.

[May84] E. Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal of Computing*, 13:441–460, 1984.

[May96a] Richard Mayr. Model checking pa-processes. Technical Report TUM-I9640, TU-München, December 1996.

[May96b] Richard Mayr. Semantic reachability for simple process algebras. In B. Steffen and T. Margaria, editors, *Proceedings of INFINITY'96*, number MIP-9614 in Technical report series of the University of Passau. University of Passau, 1996.

[May96c] Richard Mayr. A tableau system for model checking petri nets with a fragment of the linear time μ-calculus. Technical Report TUM-I9634, TU-München, October 1996.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.