Model Checking PA-Processes

Richard Mayr

Institut für Informatik, Technische Universität München,
Arcisstr. 21, D-80290 München, Germany;
e-mail: mayrri@informatik.tu-muenchen.de
Web: http://wwwbrauer.informatik.tu-muenchen.de/mitarbeiter/mayrri
fax: +49 (89) 289-28207/28483

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 subsume both Basic Parallel Processes (BPP) [Chr93] and context-free processes (BPA). They are a simple model for infinite state concurrent systems.
We show that the model checking problem for the branching time temporal logic *EF* is decidable for PA-processes.

Keywords: PA-processes, model checking, process algebras, tableau systems

1 Introduction

The Process Algebra PA is a simple model of infinite state concurrent systems. It has operators for nondeterministic choice, parallel composition, sequential composition and recursion. PA-processes and Petri nets are incomparable, meaning that neither model is more expressive than the other one. Unlike BPPs, 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).

Here we study the model checking problem for PA-processes. This is the problem of deciding if a given PA-process satisfies a property coded as a formula in a certain temporal logic.

For BPPs the situation is already fairly clear. It has been shown in [EK95] that the model checking problem for BPPs is undecidable for the branching time temporal logic EG, whose formulae are built out of the boolean operators, *EX* (for some successor) and EG (for some path always in the future). On the other hand the model checking problem is decidable for the logic *EF*, that uses the boolean operators, and the temporal operators *EX* and *EF* (for some path eventually in the future). Therefore, the logic *EF* (also called UB^- in [Esp] and [May96b]), seems to be the largest branching time logic with a decidable model checking problem. The model checking problem for BPPs and *EF* (UB^-) is PSPACE-complete [May96a, May96b].

Here we show that the model checking problem with the logic EF is decidable even for PA-processes. In section 2 we define PA-processes. In section 3 we describe the tableau system that solves the model checking problem, while in section 4 we prove its soundness and completeness. In Section 5 we describe a small example. Section 6 describes a possible extension of the logic EF by adding constraints on sequences of actions. The paper closes with a section on open problems and related work.

2 PA-Processes

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

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

Convention: We always work with equivalence classes of terms modulo commutativity of parallel composition. Also we define that $\epsilon.E = E$ and $E\|\epsilon = E$.

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 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 transition relation $\xrightarrow{a}$ is extended to sequences of actions $\xrightarrow{\sigma}$ in the standard way. If the sequence σ is of no account, then we just write $\xrightarrow{*}$.

BPPs are the subset of PA-processes without sequential composition, while context-free processes are the subset of PA-processes without parallel composition.

Unlike for PA-processes there is a one-to-one correspondence between BPPs and a class of labelled Petri nets, the *communication-free nets* [Esp]. In these nets every transition has exactly one input place with an arc labelled by 1.

3 The Tableau System

Model checking algorithms can be divided into two classes: iterative algorithms and tableau-based algorithms. The iterative algorithms compute all the states of the system which have the desired property, and usually yield higher efficiency in the worst case. The tableau-based algorithms are designed to check whether a particular expression has a temporal property. This is called local model checking which avoids the investigation of for the verification irrelevant parts of the process being verified. Therefore this method is applicable for the verification of systems with infinite state spaces. In local model checking the proof system is developed in a goal directed fashion (top down). A property holds iff there is a proof tree with a successful leaf which witnesses this truth. The algorithm for the following problem is tableau-based and decides the truth of an EF-formula for a PA-process by examining only finitely many states.

3.1 The Temporal Logic *EF*

The branching time temporal logic EF of [Esp, Esp96] is used to describe properties of PA-processes. We fix a countably infinite set of atomic actions Act. The syntax of the calculus is as follows:

$$\Phi \stackrel{\text{def}}{=} a \;\mid\; \neg\Phi \;\mid\; \Phi_1 \wedge \Phi_2 \;\mid\; \Diamond\Phi$$

where $a \in Act$ ranges over atomic actions. For convenience disjunction and another modal operator $\Box$ can be added by defining $\Box := \neg\Diamond\neg$.

Let $\mathcal{F}$ be the set of all EF-formulae. Let Ω be the set of all processes in the process algebra. The denotation $||\Phi||$ of a formula Φ is the set of processes inductively defined by the following rules:

$$\begin{array}{ll}
||a|| & := \{t \mid \exists t \xrightarrow{a} t'\} \\
||\neg\Phi|| & := \Omega - ||\Phi|| \\
||\Phi_1 \wedge \Phi_2|| & := ||\Phi_1|| \cap ||\Phi_2|| \\
||\Diamond\Phi|| & := \{t \mid \exists t \xrightarrow{\sigma} t'.\ t' \in ||\Phi||\}
\end{array}$$

The property $t \in ||\Phi||$ is also denoted by $t \models \Phi$. An instance of the model checking problem is a PA process algebra, a term t in the algebra and an EF-formula Φ. The question is if $t \models \Phi$.

In order to simplify the presentation we have left out the one-step nexttime-operator EX for now. In Section 6 we'll show that it can be added to the logic without causing any problems. In this framework this operator is often denoted by $\langle a \rangle$, with $a \in Act$ and defined by

$$||\langle a \rangle \Phi|| := \{t \mid \exists t \xrightarrow{a} t' \in ||\Phi||\}$$

The decidability results carry over to the logic that includes the nexttime-operator (see Section 6).

While the model checking problem with *EF* is undecidable for general Petri nets [Esp], it is decidable and PSPACE-complete for BPPs [May96a, May96b]. Here we show that model checking with the logic *EF* is decidable for PA-processes.

Definition 1. $\mathcal{F}_d \subset \mathcal{F}$ is defined as the set of all *EF*-formulae with a nesting-depth of modal operators $\diamond$ of at most d. (It follows that formulae in $\mathcal{F}_0$ contain no modal operators.)

In order to simplify the notation we use some abbreviations:
Let $A = \{a_1, \ldots, a_n\} \subseteq Act$ be a set of atomic actions, then

$$t \models A \;:\Leftrightarrow\; t \models a_1 \wedge \ldots \wedge a_n$$

and

$$t \models -A \;:\Leftrightarrow\; t \models \neg a_1 \wedge \ldots \wedge \neg a_n$$

The decidability proof of the model checking problem is done by induction on the nesting depth d of modal operators in the formula. For a term t and a formula $\Phi \in \mathcal{F}_d$ the algorithm builds a finite tableau for $t \models \Phi$ by using properties of the form $t' \models F'$ with $F' \in \mathcal{F}_{d-1}$ as side conditions.

First we reduce the problem to a simpler form.

Definition 2. The set of conjunctive formulae $\mathcal{F}^c \subset \mathcal{F}$ is the smallest set of formulae satisfying the following conditions:

1. $A^+ \wedge -A^-$ is a conjunctive formula for $A^+, A^- \subseteq Act$
2. $A^+ \wedge -A^- \wedge \bigwedge_{i \in I} \diamond \Psi_i \wedge \bigwedge_{j \in J} \neg \diamond \Upsilon_j$ is a conjunctive formula if $A^+, A^- \subseteq Act$ and $\Psi_i \in \mathcal{F}^c$ and $\Upsilon_j \in \mathcal{F}^c$.

Let $\mathcal{F}_d^c := \mathcal{F}_d \cap \mathcal{F}^c$.

A formula Φ is in *normal form* if $\Phi = \bigvee_{i \in I} \diamond \Psi_i$ s.t. the Ψ_i are conjunctive formulae. $\mathcal{F}_d^n \subset \mathcal{F}_d$ are the formulae in normal form in $\mathcal{F}_d$.

Lemma 3. *Any EF-formula $\Phi = \diamond \Psi$ is equivalent to a formula in normal form.*

Proof By induction on the nesting-depth d of modal operators in Ψ.

1. If $d = 0$ then Ψ doesn't contain any modal operators, so it can be transformed into disjunctive normal form $\bigvee_{i \in I} A_i^+ \wedge -A_i^-$. Therefore Φ is equivalent to $\bigvee_{i \in I} \diamond (A_i^+ \wedge -A_i^-)$. This is a formula in normal form.

2. Now $d > 0$. By induction hypothesis we can transform all subformulae $\Diamond\varphi$ of Ψ into normal from, obtaining a formula Ψ'. Then transform Ψ' into disjunctive normal form $\Psi'' = \bigvee_{i\in I}\gamma_i$. Thus Φ is equivalent to $\Phi' = \Diamond(\bigvee_{i\in I}\gamma_i) = \bigvee_{i\in I}\Diamond\gamma_i$. This is in normal form, because all γ_i are conjunctive formulae.

$\square$

Lemma 4. *Every model checking problem for EF is decidable iff it is decidable for all formulae* $\Diamond\Phi$ *with* $\Phi \in \mathcal{F}^c$*.*

Proof If it is decidable for formulae of the form $\Diamond\Psi$ with $\Psi \in \mathcal{F}^c$, then it is decidable for formulae in normal form and thus by Lemma 3 for all formulae of the form $\Diamond\Phi$. Simple boolean operations yield the decidability of the whole model checking problem. The other direction is trivial. $\square$

In the sequel all *EF*-formulae will be conjunctive formulae. Let $\Phi \in \mathcal{F}^c_d$. Then $\Diamond\Phi$ has the form $\Diamond(A^+ \wedge -A^- \wedge \bigwedge_{i\in I}\Diamond\Psi_i \wedge \bigwedge_{j\in J}\neg\Diamond\Upsilon_j)$ where $A^+ \subseteq Act$, $A^- \subseteq Act$ and $\Psi_i \in \mathcal{F}^c_{d-1}$ and $\Upsilon_j \in \mathcal{F}^c_{d-1}$.

Remark. In the definition of PA algebras we assumed that every occurring variable is defined. It follows that in the other representation there is at least one rule $X \xrightarrow{a} G$ in Δ for every X. Therefore a PA-process cannot perform any action if and only if it is empty. This means that $t \models \Diamond(-Act) \iff \exists t \xrightarrow{\sigma} \epsilon$.

3.2 Decomposition

For the construction of a finite tableau that solves the model checking problem it is necessary to split the problem into several smaller subproblems. We do this by showing that properties of a PA-process can be expressed by properties of its subprocesses.

Lemma 5. *Let* t_1, t_2 *be PA-terms and* Φ *in* $\mathcal{F}^c_d$*. Then a set* I *and terms* $\Phi^1_i, \Phi^2_i \in \mathcal{F}^c_d$ *can be effectively constructed s.t.*

$$t_1 \| t_2 \models \Diamond\Phi \;\Leftrightarrow\; \bigvee_{i\in I} t_1 \models \Diamond\Phi^1_i \wedge t_2 \models \Diamond\Phi^2_i$$

Proof $\Diamond\Phi = \Diamond(A^+ \wedge -A^- \wedge \bigwedge_{i\in I}\Diamond\Psi_i \wedge \bigwedge_{j\in J}\neg\Diamond\Upsilon_j)$ with $A^+, A^- \subseteq Act$, $\Psi_i, \Upsilon_j \in \mathcal{F}^c_{d-1}$. The proof is done by induction on d.

$$t_1 \| t_2 \models \Diamond(A^+ \wedge -A^- \wedge \bigwedge_{i\in I}\Diamond\Psi_i \wedge \bigwedge_{j\in J}\neg\Diamond\Upsilon_j)$$

By definition of *EF* this is equivalent to

$$\begin{aligned}\exists A^+_1 \cup A^+_2 = A^+.\ \exists t_1 \xrightarrow{*} t'_1, t_2 \xrightarrow{*} t'_2.\ t'_1 \models (A^+_1 \wedge -A^-) \wedge t'_2 \models (A^+_2 \wedge -A^-) \wedge \\ \textstyle\bigwedge_{i\in I} t'_1\|t'_2 \models \Diamond\Psi_i \wedge \bigwedge_{j\in J} t'_1\|t'_2 \models \neg\Diamond\Upsilon_j\end{aligned}$$

By induction hypothesis there are K_i, L_j and $\varphi^1_{i,k}, \varphi^2_{i,k}, \delta^1_{j,l}, \delta^2_{i,l} \in \mathcal{F}^c_{d-1}$ s.t. the expression is equivalent to

$$\exists A_1^+ \cup A_2^+ = A^+ . \exists t_1 \xrightarrow{*} t_1', t_2 \xrightarrow{*} t_2'.\ t_1' \models (A_1^+ \wedge -A^-) \wedge t_2' \models (A_2^+ \wedge -A^-) \wedge \\ \bigwedge_{i\in I} (\bigvee_{k \in K_i} t_1' \models \Diamond\varphi^1_{i,k} \wedge t_2' \models \Diamond\varphi^2_{i,k}) \wedge \bigwedge_{j\in J} \neg(\bigvee_{l\in L_j} t_1' \models \Diamond\delta^1_{j,l} \wedge t_2' \models \Diamond\delta^2_{j,l})$$

By De Morgan this is equivalent to

$$\exists A_1^+ \cup A_2^+ = A^+ . \exists t_1 \xrightarrow{*} t_1', t_2 \xrightarrow{*} t_2'.\ t_1' \models (A_1^+ \wedge -A^-) \wedge t_2' \models (A_2^+ \wedge -A^-) \wedge \\ \bigwedge_{i\in I} (\bigvee_{k \in K_i} t_1' \models \Diamond\varphi^1_{i,k} \wedge t_2' \models \Diamond\varphi^2_{i,k}) \wedge \bigwedge_{j\in J} \bigwedge_{l\in L_j} (t_1' \models \neg\Diamond\delta^1_{j,l} \vee t_2' \models \neg\Diamond\delta^2_{j,l})$$

By transformation to disjunctive normal form we get

$$\exists A_1^+ \cup A_2^+ = A^+ . \exists t_1 \xrightarrow{*} t_1', t_2 \xrightarrow{*} t_2'.\ t_1' \models (A_1^+ \wedge -A^-) \wedge t_2' \models (A_2^+ \wedge -A^-) \wedge \\ \bigvee_{F:I\mapsto K_i,\, G\times H \subset J\times L_j} \Big[\bigwedge_{i\in I} t_1' \models \Diamond\varphi^1_{i,F(i)} \wedge t_2' \models \Diamond\varphi^2_{i,F(i)} \wedge \\ \bigwedge_{(j,l)\in G\times H} t_1' \models \neg\Diamond\delta^1_{j,l} \wedge \bigwedge_{(j,l)\in J\times L_j - G\times H} t_2' \models \neg\Diamond\delta^2_{j,l} \Big]$$

Here F is a total function $F : I \mapsto \bigcup_{i\in I} K_i$, s.t. $\forall i \in I.\ F(i) \in K_i$. G and H must satisfy the restriction that if $(j,l) \in G \times H$, then $l \in L_j$. Putting it together again yields

$$\bigvee_{A_1^+ \cup A_2^+ = A^+,\, F:I\mapsto K_i,\, G\times H\subset J\times L_j} \\ t_1 \models \Diamond(A_1^+ \wedge -A^- \wedge \bigwedge_{i\in I} \Diamond\varphi^1_{i,F(i)} \bigwedge_{(j,l)\in G\times H} \neg\Diamond\delta^1_{j,l}) \wedge \\ t_2 \models \Diamond(A_2^+ \wedge -A^- \wedge \bigwedge_{i\in I} \Diamond\varphi^2_{i,F(i)} \bigwedge_{(j,l)\in J\times L_j - G\times H} \neg\Diamond\delta^2_{j,l})$$

This is in normal form. □

Lemma 6. *Let t_1, t_2 be PA-terms and Φ in $\mathcal{F}^c_d$. Then sets N, P, Q and terms $\alpha, \beta_n \in \mathcal{F}^c_d$ and $\gamma_p, \delta_q \in \mathcal{F}^c_{d-1}$ can be effectively constructed s.t. $t_1.t_2 \models \Diamond\Phi$ iff*

$$t_1 \models \Diamond(-Act) \wedge t_2 \models \Diamond\Phi \vee \\ t_1 \models \neg\Diamond(-Act) \wedge t_1 \models \Diamond\Phi \vee \\ t_1 \models \Diamond\alpha \vee \bigvee_{n\in N} \Big[t_1 \models \Diamond\beta_n \wedge \bigwedge_{p\in P(n)} t_2 \models \Diamond\gamma_p \wedge \bigwedge_{q\in Q(n)} t_2 \models \neg\Diamond\delta_q \Big]$$

Proof by induction on d.
If $d = 0$ then $\Diamond\Phi = \Diamond(A^+ \wedge -A^-)$. The first two cases of the above disjunction

are clear. The only remaining case is $t_1 \rightarrow t_1' \neq \epsilon.\ t_1' \models (A^+ \wedge -A^-)$. (Here $\exists t_1' \xrightarrow{*} \epsilon$.) Choose $\alpha = \mathit{false}$, $N = Act$, $\beta_a = A^+ \cup \{a\} \wedge -A^-$, $P(a) = Q(a) = \emptyset$ for every $a \in Act$.

Now $d > 0$. We can assume that $\Diamond\Phi = \Diamond(A^+ \wedge -A^- \wedge \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j)$ with $A^+, A^- \subseteq Act$, $\Psi_i, \Upsilon_j \in \mathcal{F}^c_{d-1}$. The first two cases of the above disjunction are obvious. In the third case we have:

$$\exists t_1 \xrightarrow{*} t_1' \neq \epsilon.\ t_1' \models A^+ \wedge -A^- \wedge t_1'.t_2 \models \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j$$

This is equivalent to

$$\begin{aligned}
t_1 &\models \Diamond(A^+ \wedge -A^- \wedge \neg\Diamond(-Act) \wedge \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j) \vee \\
t_2 &\models \bigwedge_{j\in J} \neg\Diamond\Upsilon_j \wedge \bigvee_{a\in Act} \exists t_1 \xrightarrow{*} t_1'.\Big[t_1' \models (A^+ \cup \{a\} \wedge -A^-) \wedge \\
&\qquad t_1' \models \bigwedge_{j\in J} \bigwedge_{b\in Act} \neg\Diamond(\Upsilon_j \wedge b) \wedge t_1'.t_2 \models \bigwedge_{i\in I} \Diamond\Psi_i\Big]
\end{aligned}$$

As $\Psi_i \in \mathcal{F}^c_{d-1}$ there are (by induction hypothesis) $\alpha_i, \beta_n \in \mathcal{F}^c_{d-1}$ and $\gamma_p, \delta_q \in \mathcal{F}^c_{d-2}$ s.t. this is equivalent to

$$\begin{aligned}
&t_1 \models \Diamond(A^+ \wedge -A^- \wedge \neg\Diamond(-Act) \wedge \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j) \vee \\
&\qquad t_2 \models \bigwedge_{j\in J} \neg\Diamond\Upsilon_j \wedge \\
&\bigvee_{a\in Act} \exists t_1 \xrightarrow{*} t_1'.\Big[t_1' \models (A^+ \cup \{a\} \wedge -A^-) \wedge t_1' \models \bigwedge_{j\in J} \bigwedge_{b\in Act} \neg\Diamond(\Upsilon_j \wedge b) \wedge \\
&\qquad \bigwedge_{i\in I} \Big(t_2 \models \Diamond\Psi_i \vee t_1' \models \Diamond\alpha_i \vee \\
&\qquad \bigvee_{n\in N_i} (t_1' \models \Diamond\beta_n \wedge \bigwedge_{p\in P(n)} t_2 \models \Diamond\gamma_p \wedge \bigwedge_{q\in Q(n)} t_2 \models \neg\Diamond\delta_q)\Big)\Big]
\end{aligned}$$

This requires some explanation. The case that t_1' cannot be reduced to ϵ is already considered in the first line of this formula. So we can assume that $t_1' \models \Diamond(-Act)$. Therefore in the application of the induction hypothesis we only need to add the formula $t_2 \models \Diamond\Psi_i$.

By transformation to disjunctive normal form we get

$$\begin{aligned}
&t_1 \models \Diamond(A^+ \wedge -A^- \wedge \neg\Diamond(-Act) \wedge \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j) \vee \\
&\qquad t_2 \models \bigwedge_{j\in J} \neg\Diamond\Upsilon_j \wedge \\
&\bigvee_{a\in Act} \exists t_1 \xrightarrow{*} t_1'.\Big[t_1' \models (A^+ \cup \{a\} \wedge -A^-) \wedge t_1' \models \bigwedge_{j\in J} \bigwedge_{b\in Act} \neg\Diamond(\Upsilon_j \wedge b) \wedge
\end{aligned}$$

$$\bigvee_{I',I''\subseteq I, F:(I-(I'\cup I''))\mapsto N_i} \bigwedge_{i\in I'} t_2 \models \Diamond\Psi_i \bigwedge_{i\in I''} t_1' \models \Diamond\alpha_i$$
$$\bigwedge_{i\in I-(I'\cup I'')} t_1' \models \Diamond\beta_{F(i)} \bigwedge_{i\in I-(I'\cup I'')} \bigwedge_{k\in P(F(i))} t_2 \models \Diamond\gamma_k$$
$$\bigwedge_{i\in I-(I'\cup I'')} \bigwedge_{k\in Q(F(i))} t_2 \models \neg\Diamond\delta_k\Big]$$

Here F is a total function from $I-(I'\cup I'')$ to $\bigcup_{i\in I} N_i$ s.t. $\forall i.\ F(i)\in N_i$.

Putting it together again yields

$$t_1 \models \Diamond(A^+ \wedge -A^- \wedge \neg\Diamond(-Act) \wedge \bigwedge_{i\in I} \Diamond\Psi_i \wedge \bigwedge_{j\in J} \neg\Diamond\Upsilon_j) \vee$$
$$\bigvee_{a\in Act, I',I''\subseteq I, F:(I-(I'\cup I''))\mapsto N_i} \Big[\bigwedge_{j\in J} t_2 \models \neg\Diamond\Upsilon_j \wedge \bigwedge_{i\in I'} t_2 \models \Diamond\Psi_i$$
$$\bigwedge_{i\in I-(I'\cup I'')} \bigwedge_{k\in P(F(i))} t_2 \models \Diamond\gamma_k$$
$$\bigwedge_{i\in I-(I'\cup I'')} \bigwedge_{k\in Q(F(i))} t_2 \models \neg\Diamond\delta_k \ \wedge\ t_1 \models \Diamond(A^+\cup\{a\} \wedge -A^- \wedge$$
$$\bigwedge_{j\in J} \bigwedge_{b\in Act} \neg\Diamond(\Upsilon_j \wedge b) \wedge$$
$$\bigwedge_{i\in I''} \Diamond\alpha_i \wedge \bigwedge_{i\in I-(I'\cup I'')} \Diamond\beta_{F(i)})\Big]$$

This has the desired form. □

3.3 The Tableau-rules

Now we can define the rules for the construction of a tableau that decides $t \models \Diamond\Phi$ for $\Phi \in \mathcal{F}_d^c$. In this construction we assume that we can already decide all problems of the form $t' \models \Diamond\Psi$ or $t' \models \neg\Diamond\Psi$ for any $\Psi \in \mathcal{F}_{d-1}^c$. In the base case of $d=0$ this condition is trivially satisfied, as $\mathcal{F}_{-1}^c = \emptyset$. Also we assume that we can decide problems of the form $t \models \Diamond(-Act)$. (This is equivalent to $\exists t \xrightarrow{\sigma} \epsilon$).

Lemma 7. *Let t be a PA-term. It is decidable if $t \models \Diamond(-Act)$.*

Proof The algorithm proceeds by successively marking variables as being reducible to ϵ. First mark all variables X s.t. $\exists X \xrightarrow{a} \epsilon$. Then mark all variables Y s.t. $\exists X \xrightarrow{a} G$ where all variables occurring in G are already marked. Repeat this until no new variables can be marked. Then $t \models \Diamond(-Act)$ iff all variables occurring in t are marked. □

The nodes in the tableau are marked with sets of expressions of the form $t \vdash \Phi$, where t is a PA-term and Φ an EF-formula. Such sets are denoted by Γ. These sets of expressions at the nodes are interpreted conjunctively, while the branches in the tableau are interpreted disjunctively. The tableau is successful iff there is a successful branch.

$$\text{PAR}\quad \frac{\{t_1 \| t_2 \vdash \Diamond\Phi\} \cup \Gamma}{(\text{see Lemma 5}) \cup \Gamma}$$

$$\text{SEQ}\quad \frac{\{t_1 . t_2 \vdash \Diamond\Phi\} \cup \Gamma}{(\text{see Lemma 6}) \cup \Gamma}$$

$$\text{Step}\quad \frac{\{X \vdash \Diamond\Phi\} \cup \Gamma}{\{X \vdash \Phi\} \cup \Gamma \quad \{t_1 \vdash \Diamond\Phi\} \cup \Gamma \quad \ldots \quad \{t_n \vdash \Diamond\Phi\} \cup \Gamma} \quad \text{for } X \xrightarrow{a} t_i$$

$$\wedge\quad \frac{\{t \vdash \Phi \wedge \Psi\} \cup \Gamma}{\{t \vdash \Phi, t \vdash \Psi\} \cup \Gamma}$$

$$\vee\quad \frac{\{t \vdash \Phi \vee \Psi\} \cup \Gamma}{\{t \vdash \Phi\} \cup \Gamma \quad \{t \vdash \Psi\} \cup \Gamma}$$

$$\text{Induct1}\quad \frac{\{t \vdash \Diamond\Psi\} \cup \Gamma}{\Gamma} \quad \text{if } \Psi \in \mathcal{F}^c_{d-1} \text{ and } t \models \Diamond\Psi$$

$$\text{Induct2}\quad \frac{\{t \vdash \neg\Diamond\Psi\} \cup \Gamma}{\Gamma} \quad \text{if } \Psi \in \mathcal{F}^c_{d-1} \text{ and not } t \models \Diamond\Psi$$

$$\text{Term1}\quad \frac{\{t \vdash \Diamond(-Act)\} \cup \Gamma}{\Gamma} \quad \text{if } \exists t \xrightarrow{\sigma} \epsilon$$

$$\text{Term2}\quad \frac{\{t \vdash \neg\Diamond(-Act)\} \cup \Gamma}{\Gamma} \quad \text{if } \nexists t \xrightarrow{\sigma} \epsilon$$

$$\text{Act1}\quad \frac{\{t \vdash A^+\} \cup \Gamma}{\Gamma} \quad \text{if } \forall_{a \in A^+} \exists t \xrightarrow{a} t'$$

$$\text{Act2}\quad \frac{\{t \vdash -A^-\} \cup \Gamma}{\Gamma} \quad \text{if } \forall_{a \in A^-} \nexists t \xrightarrow{a} t'$$

To avoid any unnecessary growth of the proof tree we define that the rules $\wedge$, $\vee$, Induct1, Induct2, Term1, Term2, Act1 and Act2 take precedence over all the other rules (PAR, SEQ and Step).

The following property follows immediately from the definition of the tableau-rules and Lemma 5 and Lemma 6.

Proposition 8. *For any instance of a tableau-rule the antecedent is true iff one of the consequents is true.*

Definition 9. (Termination conditions) A node n consisting of a set of formulae Γ is a terminal node if one of the following conditions is satisfied:

1. Γ is empty
2. $t \vdash \Diamond\Psi \in \Gamma$ with $\Psi \in \mathcal{F}^c_{d-1}$ and $t \not\models \Diamond\Psi$
3. $t \vdash \neg\Diamond\Psi \in \Gamma$ with $\Psi \in \mathcal{F}^c_{d-1}$ and $t \models \Diamond\Psi$
4. $t \vdash \Diamond(-Act) \in \Gamma$ and $\not\exists t \xrightarrow{\sigma} \epsilon$
5. $t \vdash \neg\Diamond(-Act) \in \Gamma$ and $\exists t \xrightarrow{\sigma} \epsilon$
6. $t \vdash A^+ \in \Gamma$ and $\exists a \in A^+.\ \not\exists t \xrightarrow{a} t'$
7. $t \vdash -A^- \in \Gamma$ and $\exists a \in A^-.\ \exists t \xrightarrow{a} t'$
8. There is a previous node n' in the same branch that is marked with set Γ' s.t. $\Gamma = \Gamma'$

Terminals of type 1 are successful, while terminals of type 2–8 are unsuccessful.

4 Soundness and Completeness

Lemma 10. *If the root node has the form $t \vdash \Diamond\Phi$, then for every node n in the tableau at least one of the following conditions is satisfied:*

- *A tableau rule is applicable*
- *The node is a terminal node.*

Proof The only problematic cases are the formulae of the form $t \vdash \neg\Diamond\Phi$. If such a formula occurs, then it must be due to the rules SEQ or Step. By definition of the rule Step and Lemma 6 we know that $\Phi \in \mathcal{F}^c_{d-1}$. Therefore the node is a terminal node or one of the rules Induct2 or Term2 is applicable. □

Lemma 11. *The tableau is finite.*

Proof There are only finitely many formulae in $\mathcal{F}^c_d$ and only finitely many rules $X \xrightarrow{a} t$ with only finitely many subterms of the terms t. So there are only finitely many different sets of expressions of the form $t \vdash \Phi$ in the tableau. Therefore the branches of the tableau can only have finite length, because of termination condition 8. As the tableau is finitely branching, the result follows. □

Now we prove the soundness and completeness of the tableau.

Lemma 12. *Let $\Phi \in \mathcal{F}^c_d$. If there is a successful tableau with root $t \vdash \Diamond\Phi$, then $t \models \Diamond\Phi$.*

Proof A successful tableau has a successful branch ending with a node marked by the empty set of formulae. As these sets are interpreted conjunctively this node is true. By Proposition 8 all its ancestor-nodes must be true and thus the root-node must be true as well. □

Lemma 13. *Let t be a PA-term, $\Phi \in \mathcal{F}^c_d$ and Γ a set of formulae. If $t \models \Diamond\Phi$ then there is a sequence of rule applications s.t. there is a path from a node marked $\{t \vdash \Diamond\Phi\} \cup \Gamma$ to a node marked Γ.*

Proof by induction on lexicographically ordered pairs (x, y) where x is the length of the shortest sequence σ s.t. $t \xrightarrow{\sigma} t'$ and $t' \models \Phi$, and y is the size of t.

The construction of the tableau is done in rounds. Each round consists of an application of one of the rules SEQ, PAR or Step, followed by several applications of the rules $\wedge$, $\vee$, Induct1, Induct2, Term1 and Term2 to clear away unnecessary formulae (Remember that these rules take precedence over the rules SEQ, PAR and Step). As the node is true, at least one of its successors (at the end of the round) must be true.

SEQ If this rule was used, then the successor has the form $\Gamma \cup \Gamma'$, where all members of Γ' are of the form $t' \vdash \Diamond\Phi'$ where t' is smaller than t. This means that y is now smaller. An analysis of the proof of Lemma 6 shows that the value of x cannot have increased. The result follows from the induction hypothesis.

PAR In this case the successor has the form $\{t_1 \vdash \Diamond\Phi_1, t_2 \vdash \Diamond\Phi_2\} \cup \Gamma$ s.t. t_1 and t_2 are smaller than t. It follows from the proof of Lemma 5 that the value of x has not increased, while the value of y is smaller. Applying the induction hypothesis twice yields the desired result.

Step Here we have two subcases:

1. If the first branch of the Step-rule is true, then applications of the rules $\wedge$, $\vee$, Induct1, Induct2, Term1, Term2, Act1 and Act2 directly lead to a node marked by Γ.
2. Otherwise choose the true successor that corresponds to the shortest sequence σ (see above). Here the value of y may have increased, but the value of x has decreased by 1, and thus we can apply the induction hypothesis.

This construction cannot be stopped by termination condition 8, because this would contradict the minimality of the length of σ. $\square$

Corollary 14. *If $t \models \Diamond\Phi$ for $\Phi \in \mathcal{F}_d^c$, then there is a successful tableau with root $t \vdash \Diamond\Phi$.*

Proof Applying Lemma 13 for the special case of an empty set Γ yields that a node can be reached that is marked by the empty set. The branch from the root-node to this node is successful and thus there is a successful tableau. $\square$

Lemma 15. *Let t be a PA-term and $\Phi \in \mathcal{F}_d^c$. $t \models \Diamond\Phi$ iff there is a successful tableau for $t \vdash \Diamond\Phi$.*

Proof Directly from Lemma 12 and Corollary 14. $\square$

Theorem 16. *The model checking problem for PA-processes and the logic EF is decidable.*

Proof By Lemma 4 it suffices to prove decidability for formulae of the form $\Diamond\Phi$ with Φ in $\mathcal{F}_d^c$ for any d. We prove this by induction on d. By Lemma 15 and Lemma 11 it suffices to construct a finite tableau. During the construction we need to decide problems of the form $t' \models \Diamond\Psi$ for $\Psi \in \mathcal{F}_{d-1}^c$ and problems of the form $t \models \Diamond(-Act)$. The first one is possible by induction hypothesis, and the second one by Lemma 7. $\square$

5 Example

In this section we describe a tiny example of the model checking problem. Let a process algebra be described by the following set of rules Δ:

$$X \xrightarrow{a} X \| X$$

$$X \xrightarrow{c} \epsilon$$

$$Y \xrightarrow{b} X.Y$$

The operator $\Box$ (meaning always) can be expressed in the logic EF by defining $\Box := \neg\Diamond\neg$. By using the algorithm derived from the tableau system described in subsection 3.3 we can show a property of the process $(X\|Y)$.

$$(X\|Y) \models \Diamond(\Box\Diamond(a) \ \wedge \ \Box\Diamond(b) \ \wedge \ \neg\Diamond(a \wedge b))$$

This means that this process can reach a state s.t. from then on it can always get back into states where it can do action a or action b, but never both, as a and b are then mutually exclusive.

6 Extensions

In this section we extend the logic EF by constraints on sequences. So far the expression $t \models \Diamond\Phi$ only means that there is a sequence σ s.t. $t \xrightarrow{\sigma} t'$ and $t' \models \Phi$ without saying anything about σ. Now we generalize the operator $\Diamond$ to $\Diamond_C$, where $C : Act^* \mapsto \{true, false\}$ are predicates on finite sequences of actions. Here these functions are called constraints.

The semantics of the modified modal operator $\Diamond_C$ is defined by:

$$\|\Diamond_C\Phi\| = \{t \mid \exists\sigma, t'.\ t \xrightarrow{\sigma} t' \wedge t' \in \|\Phi\| \wedge C(\sigma)\}$$

We'll show that for a special class of constraints C the extended logic is still decidable for PA-processes.

Definition 17. (Decomposable constraints) Let $a \in Act$, $i, k \in \mathbb{N}$ and σ a sequence of actions. $[x]_k$ denotes x modulo k. Decomposable constraints are of the following form

$$C ::= W(\sigma) \geq i \mid W(\sigma) \leq i \mid [W(\sigma)]_k = i \mid C_1 \vee C_2 \mid C_1 \wedge C_2 \mid \mathit{first}(\sigma) = a$$

where $W : Act^* \mapsto \mathbb{N}$ is a function on sequences s.t. $W(\sigma_1\sigma_2) = W(\sigma_1) + W(\sigma_2)$ for all σ_1, σ_2. (This implies that if σ is the empty sequence, then $W(\sigma) = 0$).

These constraints are called "decomposable", because a constraint C on a sequence of actions σ performed by a sequential– or parallel composition of processes t_1 and t_2 can be expressed by constraints on sequences performed by t_1 and t_2.

For example let W be the function that counts the number of a-actions in a sequence. Now if $t_1||t_2 \xrightarrow{\sigma} t_1'||t_2'$ and $[W(\sigma)]_3 = 0$ then there are sequences σ_1, σ_2 s.t. $t_1 \xrightarrow{\sigma_1} t_1'$ and $t_2 \xrightarrow{\sigma_2} t_2'$ and either $W(\sigma_1) = W(\sigma_2) = 0$ or $W(\sigma_1) = 1$ and $W(\sigma_2) = 2$ or $W(\sigma_1) = 2$ and $W(\sigma_2) = 1$.

Definition 18. Let EF_{DC} be the extension of EF by modal operators $\Diamond_C$, where C is a decomposable constraint.

By using decomposability of the constraints the Lemmas 5 and 6 can be extended to the logic EF_{DC}. The tableau method can be adjusted accordingly and thus the logic EF_{DC} is still decidable for PA-processes.

Let λ be the empty sequence of actions. The modified tableau rules are:

$$\text{PAR}\quad \frac{\{t_1||t_2 \vdash \Diamond_C \Phi\} \cup \Gamma}{\{\text{the modified Lemma 5}\} \cup \Gamma}$$

$$\text{SEQ}\quad \frac{\{t_1.t_2 \vdash \Diamond_C \Phi\} \cup \Gamma}{\{\text{the modified Lemma 6}\} \cup \Gamma}$$

$$\text{Split}\quad \frac{\{t \vdash \Diamond_{C_1 \vee C_2} \Phi\} \cup \Gamma}{\{t \vdash \Diamond_{C_1} \Phi\} \cup \Gamma \quad \{t \vdash \Diamond_{C_2} \Phi\} \cup \Gamma}$$

$$\text{Clear}\quad \frac{\{t \vdash \Diamond_{C_1 \wedge C_2} \Phi\}}{t \vdash \Diamond_{C_1} \Phi} \quad \text{if } C_2 \text{ is equal to } \mathit{true}$$

$$\text{Step1}\quad \frac{\{X \vdash \Diamond_C \Phi\} \cup \Gamma}{\{X \vdash \Phi\} \cup \Gamma \quad \{t_1 \vdash \Diamond_{C_1} \Phi\} \cup \Gamma \quad \ldots \quad \{t_n \vdash \Diamond_{C_n} \Phi\} \cup \Gamma}$$

if $C(\lambda)$, $X \xrightarrow{a_i} t_i$ and $C_i = Cons(C, a_i)$

$$\text{Step2}\quad \frac{\{X \vdash \Diamond_C \Phi\} \cup \Gamma}{\{t_1 \vdash \Diamond_{C_1} \Phi\} \cup \Gamma \quad \ldots \quad \{t_n \vdash \Diamond_{C_n} \Phi\} \cup \Gamma}$$

if not $C(\lambda)$, $X \xrightarrow{a_i} t_i$ and $C_i = Cons(C, a_i)$

$$\wedge\quad \frac{\{t \vdash \Phi \wedge \Psi\} \cup \Gamma}{\{t \vdash \Phi, t \vdash \Psi\} \cup \Gamma}$$

$$\vee\quad \frac{\{t \vdash \Phi \vee \Psi\} \cup \Gamma}{\{t \vdash \Phi\} \cup \Gamma \quad \{t \vdash \Psi\} \cup \Gamma}$$

$$\text{Induct1}\quad \frac{\{t \vdash \Diamond_C \Psi\} \cup \Gamma}{\Gamma} \quad \text{if } \Psi \in \mathcal{F}^c_{d-1} \text{ and } t \models \Diamond_C \Psi$$

$$\text{Induct2}\quad \frac{\{t \vdash \neg\Diamond_C \Psi\} \cup \Gamma}{\Gamma} \quad \text{if } \Psi \in \mathcal{F}^c_{d-1} \text{ and not } t \models \Diamond_C \Psi$$

$$\text{Term1}\ \frac{\{t \vdash \Diamond_C(-Act)\} \cup \Gamma}{\Gamma} \quad \text{if } \exists t \xrightarrow{\sigma} \epsilon \text{ and } C(\sigma)$$

$$\text{Term2}\ \frac{\{t \vdash \neg\Diamond_C(-Act)\} \cup \Gamma}{\Gamma} \quad \text{if } \not\exists t \xrightarrow{\sigma} \epsilon \text{ with } C(\sigma)$$

$$\text{Act1}\ \frac{\{t \vdash A^+\} \cup \Gamma}{\Gamma} \quad \text{if } \forall_{a \in A^+} \exists t \xrightarrow{a} t'$$

$$\text{Act2}\ \frac{\{t \vdash -A^-\} \cup \Gamma}{\Gamma} \quad \text{if } \forall_{a \in A^-}\ \not\exists t \xrightarrow{a} t'$$

In the rules Step1 and Step2 the new constraints C_i are computed from the constraint C and the action a_i by

$$Cons(C_1 \wedge C_2, a) := Cons(C_1, a) \wedge Cons(C_2, a)$$

$$Cons(W(\sigma) \geq i, a) := W(\sigma) \geq i - W(a)$$

$$Cons(W(\sigma) \leq i, a) := W(\sigma) \leq i - W(a)$$

$$Cons([W(\sigma)]_k = j, a) := [W(\sigma)]_k = [j - W(a)]_k$$

$$Cons(first(\sigma) = b, a) := if\ \ a = b\ then\ true\ else\ false$$

The termination conditions are the same as in Definition 9 with the addition of one more unsuccessful one. A node of the form $t \vdash \Diamond_C \Phi$ is an unsuccessful terminal if C is equal to *false*, i.e. $C = C' \wedge false$ or $C = C' \wedge W(\sigma) \leq k$ for some $k < 0$.

Note that only finitely many different constraints can occur in a tableau, because of the definition of the function *Cons*, the rule *Clear* and this new termination condition. Thus the proofs of soundness and completeness of the tableau from section 3 carry over to the extended logic with constraints.

Theorem 19. *The model checking problem for PA-processes and the logic EF_{DC} is decidable.*

With decomposable constraints we can also express the usual one-step next operator by defining

$$\langle a \rangle := \Diamond_C$$

with $C := [first(\sigma) = a \wedge length(\sigma) = 1]$.

7 Conclusion

We have shown decidability of the model checking problem for the branching time temporal logic *EF* and PA-processes. The exact complexity of the problem is left open. While for the special case of BPPs the problem is PSPACE-complete [May96a, May96b] the algorithm described here for PA has superexponential complexity. The algorithm described here is not a generalization of the one for BPP in [May96b]. The PSPACE-algorithm for BPP of [May96b] uses a bounded search, while the algorithm for PA works by decomposition. For a formula of nesting-depth d the complexity of the algorithm derived from the tableau system is d-times exponential in the size of the process description. So the overall complexity of the algorithm is $O(tower(n))$, where $tower(0) := 0$ and $tower(i+1) := 2^{tower(i)}$.

It is interesting to compare the decidability results for branching time logics with the results for linear time logics like LTL and the linear time μ-calculus. While model checking PA-processes with *EF* is decidable, it is undecidable for LTL and the linear time μ-calculus [BH96]. For Petri nets the situation is just the other way round. While model checking Petri nets with *EF* is undecidable [Esp, Esp96], it is decidable for LTL and the linear time μ-calculus [Esp]. This emphasizes the fact that PA-processes and Petri nets are incomparable models of concurrent systems. For the modal μ-calculus the model checking problem is undecidable even for BPPs [Esp, Esp96].

	EF	LTL, linear time μ-calc.	modal μ-calc.
Petri nets	undecidable	decidable, EXPSPACE-hard	undecidable
PA	**decidable**	undecidable	undecidable
BPP	PSPACE-complete	decidable, EXPSPACE-hard	undecidable
finite LTS	polynomial	PSPACE-complete	$\in$ NP $\cap$ co-NP

References

[BH96] A. Bouajjani and P. Habermehl. Constrained properties, semilinear systems, and petri nets. In Ugo Montanari and Vladimiro Sassone, editors, *Proceedings of CONCUR'96*, number 1119 in LNCS. Springer Verlag, 1996.

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

[EK95] J. Esparza and A. Kiehn. On the model checking problem for branching time logics and basic parallel processes. In *CAV'95*, number 939 in LNCS, pages 353–366. Springer Verlag, 1995.

[Esp] J. Esparza. Decidability of model checking for infinite-state concurrent systems. To appear in Acta Informatica.

[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.

[May96a] Richard Mayr. Some results on basic parallel processes. Technical Report TUM-I9616, TU-München, March 1996.

[May96b] Richard Mayr. Weak bisimulation and model checking for basic parallel processes. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS'96)*, number 1180 in LNCS. Springer Verlag, 1996.

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