Model Checking the Full Modal Mu-Calculus for Infinite Sequential Processes

Olaf Burkart[*]
LFCS, University of Edinburgh
JCMB, King's Buildings
Edinburgh EH9 3JZ, UK
<*olaf@dcs.ed.ac.uk*>

Bernhard Steffen
FMI, Universität Passau
Innstraße 33
94032 Passau, Germany
<*steffen@fmi.uni-passau.de*>

Infinite-state systems, context-free processes, pushdown processes, regular graphs, modal mu-calculus, model-checking.

Abstract

In this paper we develop a new elementary algorithm for model-checking infinite sequential processes, including *context-free processes*, *pushdown processes*, and *regular graphs*, that decides the full modal mu-calculus. Whereas the actual model checking algorithm results from considering conditional semantics together with backtracking caused by alternation, the corresponding correctness proof requires a stronger framework, which uses *dynamic environments* modelled by finite-state automata.

1 Introduction

Over the past decade model-checking has emerged as a powerful tool for the automatic analysis of concurrent systems. Whereas model-checking for finite-state systems is nowadays well-established, the theory for infinite systems is a current research topic (cf. [Esp96]). Since even weak branching time logics are undecidable for infinite-state systems incorporating parallel operators [Esp94, EK95], much work has focused on the verification of *sequential processes.* The strongest results obtained so far show the decidability of monadic second order logic (MSOL) for the infinite binary tree [Rab69], pushdown transition graphs [MS85], regular graphs [Cou90], and rational restricted recognizable graphs [Cau96]. However, all decision procedures are non-elementary and thus not applicable to practical problems. Moreover, MSOL is usually too expressive, since it

[*]This work was supported during my stay at IRISA by the European Community under HCM grant ERBCHBGCT 920017, and during my stay at the LFCS by the DAAD under grant D/95/14834 of the NATO science committee.

allows to distinguish even bisimilar models. For these reasons, the modal mu-calculus, a powerful temporal logic combining modal operators with least and greatest fixpoints, is seen as an attractive alternative for specifying behavioural properties. Choosing the modal mu-calculus, we do not even loose expressiveness wrt. bisimulation semantics, since recently, this logic was identified as the bisimulation closed fragment of MSOL [JW96].

The model-checking problem for sequential processes and the modal mu-calculus was first considered in [BS92]. The authors developed an iterative model-checking algorithm that decides the *alternation-free* part of the modal μ-calculus for *context-free* processes based on a conditional formulation of the semantics of μ-formulas. Moreover, in [HS94] it is shown how this can be done using tableaux-based techniques, allowing local model checking. Finally, the approach was also extended to the strictly larger classes of *pushdown processes* [BS95] and *regular graphs* [BQ96]. Since alternation of fixpoints gives, however, rise to a strict hierarchy [Bra96] the problem of model-checking the full modal mu-calculus has still been open. Only recently, Walukiewicz presented a first elementary model-checking algorithm for pushdown processes based on games [Wal96].

In this paper we develop an alternative algorithm which, essentially, arises as a combination of extending the standard iterative model-checking techniques with *conditional reasoning*, in order to capture sequential model structures in an *alternation-free* setting [BS92, BS95, BQ96], and the observation that alternating fixpoints require some kind of *backtracking*, as it is known from regular model checking (cf. e.g. [CKS92]).

Whereas the actual model checker results directly from this combination, the corresponding correctness proof requires a stronger framework, which uses *dynamic environments*. In contrast to the "standard" assertions, which suffice algorithmically, dynamic environments also explicitly model valuations of variables that occur *free* in the actual fixpoint computation. This explicit treatment is necessary in order to establish the link between the result of the fixpoint iteration and the semantics of the full modal mu-calculus.

Fortunately, all this additional complexity is only required for the proof and need not be considered for an implementation. In fact, the actual algorithm can be implemented on top of the algorithms of [BS92, BS95, BQ96] covering the alternation-free case. Taking $|\mathcal{C}|$ as the number of transitions, and $|Q|$ as the branching degree in the finite sequential process representation, as well as $|\Phi|$ as the size of the formula, and ad as the alternation depth of the formula under consideration, the overall complexity[1] is

$$O(\,|\Phi| * (|Q| * |\mathcal{C}|)^{\mathrm{ad}(\Phi)+1} * 2^{|\Phi| * (\mathrm{ad}(\Phi)+|Q|)}\,).$$

The plan of the paper is now as follows. The next section describes the class of processes we will consider, and presents the full modal mu-calculus. Subsequently, we develop our model-checking algorithm which is proved to be correct in Section 4. The final section contains our conclusions and directions for future research.

[1]In this paper we neglect the optimization of [LBC$^+$94] which exploits monotonicity arguments and would reduce $\mathrm{ad}(\Phi)$ to $\mathrm{ad}(\Phi)/2$.

2 Processes and Specifications

Infinite sequential processes comprise context-free processes, pushdown processes, and regular graphs. In this paper we will mainly concentrate on the model-checking problem for context-free processes, as the extension to pushdown processes, respectively regular graphs, can be obtained following the lines of [BS95], respectively [BQ96].

2.1 Context-Free Processes

As usual, we consider *labelled transition graphs* as models for the behaviour of concurrent systems, since they allow to represent the underlying interleaving semantics of many process calculi.

Definition 2.1 *A* labelled transition graph *is a triple* $\mathcal{T} = (\mathcal{S}, Act, \rightarrow)$ *where* $\mathcal{S}$ *is the set of* states*,* Act *is the set of* transition labels *(or* actions*), and* $\rightarrow \subseteq \mathcal{S} \times Act \times \mathcal{S}$ *is the* transition relation*.*

In particular, we are interested in classes of infinite transition graphs which can be finitely represented by labelled rewrite systems.

Definition 2.2 *A* labelled rewrite system *is a triple* $\mathcal{R} = (V, Act, R)$ *where* V *is an* alphabet*,* Act *is a set of* labels*, and* $R \subseteq V^* \times Act \times V^*$ *is a finite set of* rewrite rules*. If the rewrite rules are of the form* $R \subseteq V \times Act \times V^*$ *the rewrite system is called* alphabetic*.*

In the remainder of the paper, a rewrite rule $(u, a, v) \in R$ is also written as $u \xrightarrow{a} v$. Moreover, we will denote a rewrite system simply by R if V and Act are clear from the context. In general, rewrite systems are used to define a rewrite relation on words of V^* where a rewrite rule may be applied at any position. The technical development of this paper concentrates on rewritings of the following restricted form.

Definition 2.3 *Let* $\mathcal{R} = (V, Act, R)$ *be a rewrite system. Then the* prefix rewriting relation *of* R *is defined by*

$$\underset{R}{\longmapsto} =_{\mathrm{df}} \{ (uw, a, vw) \mid (u \xrightarrow{a} v) \in R, w \in V^* \},$$

and the labelled transition graph $\mathcal{T}_R =_{\mathrm{df}} (V^*, Act, \longmapsto_R)$ *is called the* prefix transition graph *of* R*. By abuse of notation, we will henceforth write* $uw \xrightarrow{a} vw$ *instead of* $uw \longmapsto^a_R vw$*.*

An alphabetic rewrite system which is interpreted wrt. prefix rewriting is called a *context-free system*, and a *context-free process* is then the rooted prefix transition graph of a context-free system. Note that the states of a context-free process are words over V, and we will henceforth use lower greek letters $\alpha, \beta, \ldots$ to denote them. One standard example for a context-free process is the prefix transition graph of $\mathcal{C}_{ex} = \{ A \xrightarrow{a} AB,\ A \xrightarrow{b} \epsilon,\ B \xrightarrow{b} \epsilon \}$ rooted at A. According to the prefix rewriting defined above the variable A generates the labelled transition graph of Figure 1.

$$\begin{tikzcd}
A \arrow[r, "a"] \arrow[d, "b"'] & AB \arrow[r, "a"] \arrow[d, "b"'] & AB^2 \arrow[r, "a"] \arrow[d, "b"'] & AB^3 \arrow[d, "b"'] & \cdots \\
\varepsilon & B \arrow[l, "b"'] & B^2 \arrow[l, "b"'] & B^3 \arrow[l, "b"'] & \cdots
\end{tikzcd}$$

Figure 1: The prefix transition graph for the alphabetic rewrite system $\mathcal{C}_{ex}$.

2.2 The Modal μ-Calculus

Nowadays it is widely accepted that system properties can conveniently be expressed by temporal logic formulas. Particularly, the modal μ-calculus as introduced by Kozen [Koz83] is a powerful branching time logic. It combines standard modal logic with least and greatest fixpoint operators which allows to express very complex temporal properties within this formalism. Due to its expressiveness and its conciseness the μ-calculus can be regarded as the "assembly language" of temporal logics. Formulas of the μ-calculus, given in positive form, are defined by the following grammar

$$\Phi ::= \texttt{tt} \mid \texttt{ff} \mid X \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid [a]\Phi \mid \langle a \rangle \Phi \mid \mu X.\Phi \mid \nu X.\Phi$$

where X ranges over a (countable) set of variables *Var*, and a over a set of actions *Act*. We will use $L\mu$ to denote the set of all μ-calculus formulas.

2.2.1 Standard Semantics

Usually, formulas are interpreted with respect to a labelled transition graph, and the meaning of a formula is then the set of states where the formula is true. The more specific, equivalent interpretation given in the following only applies to transition graphs of context-free systems, although it may also be generalized to arbitrary labelled transition graphs.

Given $\mathcal{T}_R = (V^*, Act, \rightarrow)$, and a valuation $\mathcal{V} : Var \rightarrow 2^{V^*}$, the inductive definition below stipulates when a context-free process $\alpha \in V^*$ has the property Φ, written as $\alpha \models_{\mathcal{V}} \Phi$. If α fails to satisfy Φ, we will write $\alpha \not\models_{\mathcal{V}} \Phi$.

$$\begin{array}{lll}
\alpha \models_{\mathcal{V}} \texttt{tt} & & \\
\alpha \not\models_{\mathcal{V}} \texttt{ff} & & \\
\alpha \models_{\mathcal{V}} X & \text{iff} & \alpha \in \mathcal{V}(X) \\
\alpha \models_{\mathcal{V}} \Phi_1 \vee \Phi_2 & \text{iff} & \alpha \models_{\mathcal{V}} \Phi_1 \text{ or } \alpha \models_{\mathcal{V}} \Phi_2 \\
\alpha \models_{\mathcal{V}} \Phi_1 \wedge \Phi_2 & \text{iff} & \alpha \models_{\mathcal{V}} \Phi_1 \text{ and } \alpha \models_{\mathcal{V}} \Phi_2 \\
\alpha \models_{\mathcal{V}} \langle a \rangle \Phi & \text{iff} & \exists\, \alpha'.\ \alpha \xrightarrow{a} \alpha' \text{ and } \alpha' \models_{\mathcal{V}} \Phi \\
\alpha \models_{\mathcal{V}} [a]\Phi & \text{iff} & \forall\, \alpha'.\ \alpha \xrightarrow{a} \alpha' \text{ implies } \alpha' \models_{\mathcal{V}} \Phi \\
\alpha \models_{\mathcal{V}} \mu X.\Phi & \text{iff} & \forall\, S \subseteq V^*.\ (\forall\, \beta \in V^*.\ \beta \models_{\mathcal{V}[X \mapsto S]} \Phi \text{ implies } \beta \in S) \text{ implies } \alpha \in S \\
\alpha \models_{\mathcal{V}} \nu X.\Phi & \text{iff} & \exists\, S \subseteq V^*.\ (\forall\, \beta \in V^*.\ \beta \in S \text{ implies } \beta \models_{\mathcal{V}[X \mapsto S]} \Phi) \text{ and } \alpha \in S
\end{array}$$

where $\mathcal{V}[\,X \mapsto S\,]$ is the valuation resulting from $\mathcal{V}$ by updating the binding of X to S.

The clauses for the fixpoints are a reformulation of the Tarski-Knaster theorem which states that the least fixpoint is the intersection of all pre-fixpoints and the greatest fixpoint is the union of all post-fixpoints. As a consequence, states satisfy a fixpoint formula iff they satisfy the *unfolding* of the formula, i.e. $\alpha \models_{\mathcal{V}} \sigma X.\Phi$ iff $\alpha \models_{\mathcal{V}} \Phi[\sigma X.\Phi/X]$ where $\sigma \in \{\,\mu, \nu\,\}$ and $\Phi[\Psi/X]$ denotes the simultaneous replacement of all free occurrences of X in Φ by Ψ.

The satisfaction relation defined above is independent of the valuation if the considered formula has no free variables in which case we will drop the index $\mathcal{V}$. We extend our satisfaction relation, moreover, to sets of formulas by writing $\alpha \models \Gamma$ if $\alpha \models \Phi$, for all $\Phi \in \Gamma$. Finally, we observe that the usual denotation of formulas as the set of states where the formula holds is obtained in our presentation by $[\![\Phi]\!]_{\mathcal{V}} = \{\,\alpha \mid \alpha \models_{\mathcal{V}} \Phi\,\}$.

Next we define some standard notions which will allow us to deal with occurrences of subformulas in a given formula.

Definition 2.4 (Binding) *A formula Φ is called* well named *if every fixpoint operator in Φ binds a distinct variable, and free variables are distinct from bound variables. With each well named formula Φ we then associate its* binding function $\mathcal{D}_\Phi$ *which assigns to every bound variable X of Φ the unique subformula $\sigma X.\Psi(X)$ of Φ, called the* binding definition *of X in Φ.*

From now on we assume that every formula is well named.

Definition 2.5 (Dependency order, Expansion) *Given a formula Φ, we define the* dependency order *over the bound variables of Φ, denoted by $\leq_\Phi$, as the least partial order such that if X occurs free in $\mathcal{D}_\Phi(Y)$ then $X \leq_\Phi Y$. Moreover, for every subformula Ψ of Φ, we define the* expansion *of Ψ with respect to $\mathcal{D}_\Phi$ as:*

$$\langle\!\langle \Psi \rangle\!\rangle_{\mathcal{D}_\Phi} =_{\mathrm{df}} \Psi\ [\,\mathcal{D}_\Phi(X_n)/X_n\,] \ldots [\,\mathcal{D}_\Phi(X_1)/X_1\,]$$

where the sequence $(X_1, \ldots, X_n)$ is a linear ordering of all bound variables of Φ compatible with the dependency order, i.e. if $X_i \leq_\Phi X_j$ then $i \leq j$.

Definition 2.6 (Subformula relation, Closure) *The* subformula relation *on $L\mu$, denoted by $\preceq$, is the least partial order on $L\mu$ such that $\Psi_i \preceq \Psi_1 \vee \Psi_2$, $\Psi_i \preceq \Psi_1 \wedge \Psi_2$, $\Psi \preceq \langle a \rangle \Psi$, $\Psi \preceq [a]\Psi$, $\Psi \preceq \mu X.\Psi$, and $\Psi \preceq \nu X.\Psi$, for $i = 1, 2$ and $a \in Act$. Given a formula Φ, we define the* closure *of Φ as $CL(\Phi) = \{\,\Psi \mid \Psi \preceq \Phi\,\}$. Furthermore, if $CL(\Phi) = \{\,\Psi_1, \ldots, \Psi_n\,\}$ we will henceforth assume that the subformulas Ψ_i are linearly ordered compatible with $\preceq$, i.e. if $\Psi_i \preceq \Psi_j$ then $i \geq j$.*

Finally, we define a measure for the complexity of a formula depending on the number of intertwined alternating fixpoint operators.

Definition 2.7 (Alternation Depth)
A formula Φ is said to be in the classes Σ_0 and Π_0 iff it contains no fixpoint operators. To form the class Σ_{n+1}, take $\Sigma_n \cup \Pi_n$, and close under (i) boolean and modal combinators, (ii) $\mu X.\Phi$, for $\Phi \in \Sigma_{n+1}$, and (iii) substitution of $\Phi' \in \Sigma_{n+1}$ for a free variable of $\Phi \in \Sigma_{n+1}$ provided that no free variable of Φ' is captured by Φ; and dually for Π_{n+1}. The (Niwinski) alternation depth *of a formula Φ, denoted by $ad(\Phi)$, is then the least n such that $\Phi \in \Sigma_{n+1} \cap \Pi_{n+1}$.*

Example 2.8 *Consider the closed formula $\Phi_{ex} = \mu X.\nu Y.[b]X \wedge [a]Y$ which expresses intuitively that "on every (infinite) $\{ a, b \}$-path only finitely many b-transitions can occur". This formula has alternation depth 2, and its closure is the set of formulas $\{ \Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Psi_6, \Psi_7 \}$ where*

$$\begin{array}{lcllcllcl}
\Psi_1 & = & \mu X.\nu Y.[b]X \wedge [a]Y & \Psi_4 & = & [b]X & \Psi_6 & = & X \\
\Psi_2 & = & \nu Y.[b]X \wedge [a]Y & \Psi_5 & = & [a]Y & \Psi_7 & = & Y \\
\Psi_3 & = & [b]X \wedge [a]Y & & & & & &
\end{array}$$

When interpreted with respect to the transition graph of Figure 1 the semantics of Φ_{ex} is the set of states $AB^* \cup B^*$. Note, however, that the semantics of the inner fixpoint formula $\nu Y.[b]X \wedge [a]Y$ is not continuous wrt. the valuation of X. This can be seen by applying

$$f(S) =_{\mathrm{df}} [\![\nu Y.[b]X \wedge [a]Y]\!]^{\mathcal{T}}_{[X \mapsto S]}$$

to the finite approximants $B^{0\ldots j} =_{\mathrm{df}} \{ \epsilon, B, B^2, \ldots, B^j \}$. Since $f(B^{0\ldots j}) = B^{0\ldots j+1}$, for any j, we obtain $\bigcup_{j \geq 0} f(B^{0\ldots j}) = B^*$ as the limit of iteration over the naturals, while one further iteration yields $f(B^*) = AB^* \cup B^*$. Hence the function f is not continuous.

2.2.2 Assertion-Based Semantics

As pointed out in [BS92], context-free processes can be verified by considering Hoare-logic style pre-condition/post-condition pairs of sets of formulas for each of the nonterminals occurring in the context-free system. A triple $\{ \Gamma \}\ \alpha\ \{ \Delta \}$ is then interpreted as α satisfies all formulas of Γ if we assert that after termination of α exactly the set of formulas Δ holds. This intuition is formally captured by the following definition of *assertion-based semantics* which generalises standard semantics by taking into account the set of formulas which hold after termination of a process.

Given $\mathcal{T}_{\mathcal{C}} = (V^*, Act, \longmapsto_{\mathcal{C}})$, and a valuation $\mathcal{V} : Var \to 2^{V^*}$, the inductive definition below stipulates when a context-free process $\alpha \in V^*$ has the property Φ under the hypothesis that after termination of α the formulas Δ hold, written as $\alpha \models_{\mathcal{V}} (\Phi, \Delta)$. If α fails to satisfy Φ under the hypothesis Δ, we will write $\alpha \not\models_{\mathcal{V}} (\Phi, \Delta)$. First we have

$\epsilon \models_{\mathcal{V}} (\Phi, \Delta) \quad$ iff $\quad \Phi \in \Delta$

and then, for $\alpha \neq \epsilon$,

$$
\begin{array}{lll}
\alpha \models_{\mathcal{V}} (\mathtt{tt}, \Delta) & & \\
\alpha \not\models_{\mathcal{V}} (\mathtt{ff}, \Delta) & & \\
\alpha \models_{\mathcal{V}} (X, \Delta) & \text{iff} & \alpha \in \mathcal{V}(X) \\
\alpha \models_{\mathcal{V}} (\Phi_1 \vee \Phi_2, \Delta) & \text{iff} & \alpha \models_{\mathcal{V}} (\Phi_1, \Delta) \text{ or } \alpha \models_{\mathcal{V}} (\Phi_2, \Delta) \\
\alpha \models_{\mathcal{V}} (\Phi_1 \wedge \Phi_2, \Delta) & \text{iff} & \alpha \models_{\mathcal{V}} (\Phi_1, \Delta) \text{ and } \alpha \models_{\mathcal{V}} (\Phi_2, \Delta) \\
\alpha \models_{\mathcal{V}} (\langle a \rangle \Phi, \Delta) & \text{iff} & \exists\, \alpha'.\ \alpha \xrightarrow{a} \alpha' \text{ and } \alpha' \models_{\mathcal{V}} (\Phi, \Delta) \\
\alpha \models_{\mathcal{V}} ([a] \Phi, \Delta) & \text{iff} & \forall\, \alpha'.\ \alpha \xrightarrow{a} \alpha' \text{ implies } \alpha' \models_{\mathcal{V}} (\Phi, \Delta) \\
\alpha \models_{\mathcal{V}} (\mu X.\Phi, \Delta) & \text{iff} & \forall\, S \subseteq V^*.\ (\forall\, \beta \in V^*.\ \beta \models_{\mathcal{V}[X \mapsto S]} (\Phi, \Delta) \text{ implies } \beta \in S) \\
& & \qquad\qquad\qquad \text{implies } \alpha \in S \\
\alpha \models_{\mathcal{V}} (\nu X.\Phi, \Delta) & \text{iff} & \exists\, S \subseteq V^*.\ (\forall\, \beta \in V^*.\ \beta \in S \text{ implies } \beta \models_{\mathcal{V}[X \mapsto S]} (\Phi, \Delta)) \\
& & \qquad\qquad\qquad \text{and } \alpha \in S
\end{array}
$$

As in the case of the standard semantics, we will use $\alpha \models_{\mathcal{V}} (\Gamma, \Delta)$ to denote $\alpha \models_{\mathcal{V}} (\Phi, \Delta)$, for all $\Phi \in \Gamma$.

The usefulness of the assertion-based semantics is underpined by the following proposition [BS92] which states that, firstly, the assertion-based semantics extend the standard semantics, and secondly, that they allow to reason compositionally about context-free processes.

Proposition 2.9 *The assertion-based semantics is*

1. *an* extension *of standard semantics, i.e. given a closed formula Φ, we have,*

$$\alpha \models \Phi \quad \textit{iff} \quad \alpha \models (\Phi, \Delta_\epsilon)$$

for $\Delta_\epsilon = \{\, \Psi \in CL(\Phi) \mid \epsilon \models \langle\!|\Psi|\!\rangle_{\mathcal{D}_\Phi} \,\}$.

2. compositional *wrt. context-free processes, i.e. for all $\Delta, \Gamma \subseteq L\mu$,*

$$\alpha\beta \models (\Gamma, \Delta) \quad \textit{iff} \quad \exists\, \Sigma \subseteq L\mu.\ \alpha \models (\Gamma, \Sigma) \textit{ and } \beta \models (\Sigma, \Delta)$$

The effectiveness of our algorithm, which is presented in the next section, relies, in particular, on Proposition 2.9.1, as it shows that Φ can be verified by taking into account merely the semantics of all subformulas of Φ.

3 The Model-Checking Algorithm

In this section we develop our model-checking algorithm which allows us to verify closed μ-formulas with arbitrary alternation depth for context-free processes in exponential time. In fact, the algorithm coincides with a backtracking extension of the model-checker of [BS92] which deals only with the alternation-free fragment of the modal μ-calculus. The correctness proof, presented in Section 4, requires, however, a stronger new framework involving *dynamic environments* which capture the valuations of free variables on the context-free transition graph.

$$
\begin{array}{lcl|lcl}
[\![A]\!]^{\mathtt{tt}}_{\mathcal{V}} & = & \lambda(\Delta).1 & [\![A]\!]^{\Psi_1\vee\Psi_2}_{\mathcal{V}} & = & [\![A]\!]^{\Psi_1}_{\mathcal{V}} \sqcup [\![A]\!]^{\Psi_2}_{\mathcal{V}} \\
[\![A]\!]^{\mathtt{ff}}_{\mathcal{V}} & = & \lambda(\Delta).0 & [\![A]\!]^{\Psi_1\wedge\Psi_2}_{\mathcal{V}} & = & [\![A]\!]^{\Psi_1}_{\mathcal{V}} \sqcap [\![A]\!]^{\Psi_2}_{\mathcal{V}} \\
[\![A]\!]^{X}_{\mathcal{V}} & = & \lambda(\Delta).\mathcal{V}(X,A) & [\![A]\!]^{\langle a\rangle\Psi}_{\mathcal{V}} & = & \sqcup_{A \xrightarrow{a} \alpha} [\![\alpha]\!]^{\Psi}_{\mathcal{V}} \\
 & & & [\![A]\!]^{[a]\Psi}_{\mathcal{V}} & = & \sqcap_{A \xrightarrow{a} \alpha} [\![\alpha]\!]^{\Psi}_{\mathcal{V}}
\end{array}
$$

$$
\begin{array}{lcl}
[\![A]\!]^{\mu X.\Psi}_{\mathcal{V}} & = & \mathtt{sel}_A(\sqcap\{\, (h^X_{A_1},\dots,h^X_{A_n}) \mid \forall_{1\le i\le n}\ [\![A_i]\!]^{\Psi}_{\mathcal{V}[(X,A_j)\mapsto h^X_{A_j},1\le j\le n]} \sqsubseteq h^X_{A_i} \,\}) \\
[\![A]\!]^{\nu X.\Psi}_{\mathcal{V}} & = & \mathtt{sel}_A(\sqcup\{\, (h^X_{A_1},\dots,h^X_{A_n}) \mid \forall_{1\le i\le n}\ h^X_{A_i} \sqsubseteq [\![A_i]\!]^{\Psi}_{\mathcal{V}[(X,A_j)\mapsto h^X_{A_j},1\le j\le n]} \,\})
\end{array}
$$

$$
\begin{array}{lcl}
[\![\epsilon]\!]^{\Psi}_{\mathcal{V}}(\Delta) & = & \mathtt{mem}_{\Psi}(\Delta) \\
[\![A\alpha]\!]^{\Psi}_{\mathcal{V}}(\Delta) & = & [\![A]\!]^{\Psi}_{\mathcal{V}}(\{\, \Upsilon \in CL(\Phi) \mid [\![\alpha]\!]^{\Upsilon}_{\mathcal{V}}(\Delta) = 1 \,\})
\end{array}
$$

Figure 2: The property transformer scheme.

3.1 The Property Transformer Scheme

Given a context-free system $\mathcal{C}$ and a closed formula Φ, each nonterminal $A \in V = \{ A_1, \dots, A_n \}$ defines a mapping $[\![A]\!] : 2^{CL(\Phi)} \to 2^{CL(\Phi)}$ from post- to pre-conditions. As we are, however, in particular interested in the question whether a given subformula $\Psi \in CL(\Phi)$ belongs to the pre-condition set or not, we refine this notion by defining the following functions, called *characteristic property transformers* (CPT).

$$
[\![A]\!]^{\Psi}(\Delta) =_{\mathrm{df}} \begin{cases} 1 & \text{if } A \models (\Psi, \Delta) \\ 0 & \text{otherwise} \end{cases}
$$

Writing $\mathbb{B}$ for the usual lattice of boolean values, characteristic property transformers are elements of the boolean lattice consisting of all functions from $2^{CL(\Phi)}$ to $\mathbb{B}$, where the ordering, and the meet and join operations respectively, are defined argument-wise.

More importantly, they can be obtained as a fixpoint solution of an appropriate function scheme, called the *property transformer scheme* (PTS). This scheme is defined by the rules given in Figure 2, and consists of two parts. The first part copes with the structure of the context-free system, as well as with the semantics of the formula, and defines an equation for each pair $(A, \Psi) \in V \times CL(\Phi)$. The second part deals with the empty process according to the first clause of the assertion-based semantics, as well as with composed processes according to Proposition 2.9.2. Whereas the rules for the basic cases mimic directly the semantics of the subformula, the fixpoint related equations are slightly more complicated and require a simultaneous computation of all their corresponding transformers[2]. $\mathtt{sel}_A$ then simply selects the A component of the resulting tuple. The other auxiliary function, $\mathtt{mem}_{\Psi}$, tests the membership of Ψ in the given set of formulas. It returns 1 if $\Psi \in \Delta$ and 0 otherwise.

The overall structure of the model-checking algorithm consists now of the following three steps.

[2]Note that valuations map now pairs of variables and nonterminals to characteristic property transformers.

1. Given a context-free system $\mathcal{C}$ and a closed μ-formula Φ construct the property transformer scheme according to the rules given in Figure 2.
2. Solve the (finite) fixpoint problem for the property transformer scheme.
3. Check whether $[\![A_1]\!]^{\Phi}(\Delta_\epsilon) = 1$ where A_1 is the root of the context-free system, and $\Delta_\epsilon = \{\, \Psi \in CL(\Phi) \mid \epsilon \models \langle\!|\Psi|\!\rangle_{\mathcal{D}_\Phi} \,\}$.

In Section 4 we prove that the second step of the algorithm computes transformers which reflect the assertion-based semantics, while Proposition 2.9.1 now ensures that the third step solves the model-checking problem, as we have

$$[\![A_1]\!]^{\Phi}(\Delta_\epsilon) = 1 \quad \text{iff} \quad A_1 \models (\Phi, \Delta_\epsilon) \quad \text{iff} \quad A_1 \models \Phi$$

Moreover, the ordinary semantics of Φ can be obtained from the set of CPT's by means of $[\![\Phi]\!] = \{\, \alpha \in V^* \mid [\![\alpha]\!]^{\Phi}(\Delta_\epsilon) = 1 \,\}$. In particular, the semantics of Φ is always a regular set of states.

3.2 Complexity

As expected, the required backtracking for alternating μ-formulas yield a worst-case time complexity for the algorithm, which is exponentially worse (in the alternation depth) than the estimation given for the alternation-free case [BS92, BS95].

Theorem 3.1 (Complexity)
Let $\mathcal{C}$ be a context-free system, and Φ be a closed μ-formula. Then the worst-case time complexity of solving the property transformer scheme is

$$O(\,|\Phi| * (|\mathcal{C}| * 2^{|\Phi|})^{ad(\Phi)+1}\,)$$

Proof: For the complexity analysis we assume wlog. that the context-free system is given in 3-GNF, i.e. that each right-hand side of the context-free system has length at most 2.

First observe that the right-hand side of a single PTS-equation can be computed argument-wise for each of the $2^{|\Phi|}$ arguments $\Delta \in CL(\Phi)$. The most expensive right-hand side computations result from PTS-equations related to formulas containing a modality. In this case any evaluation of $[\![\alpha]\!]^{\Psi}_{\mathcal{V}}$ takes time $O(\,|\Phi|\,)$, and for a fixed Ψ, we have at most $|\mathcal{C}|$ of these composed expressions. Thus fixing a subformula Ψ_k, the evaluation of all its PTS right-hand sides for all arguments has time complexity $O(\,|\mathcal{C}| * |\Phi| * 2^{|\Phi|}\,)$.

Moreover, there are now at most $|\mathcal{C}|$ CPT's each having a maximal chain length of approximants of $2^{|\Phi|}$. Thus the overall worst-case time complexity for computing the fixpoint at level $k \in [1, n]$ can be estimated by

$$TC_k = O(\,|\mathcal{C}| * 2^{|\Phi|} * (TC_{k+1} + |\mathcal{C}| * |\Phi| * 2^{|\Phi|})\,)$$

where $TC_{n+1} = 0$. Using induction, and thereafter standard techniques to refine the fixpoint computation by dealing with subformulas occurring in between alternation simultaneously we hence obtain the overall time complexity

$$TC_1 = O(\,|\Phi| * (|\mathcal{C}| * 2^{|\Phi|})^{\mathrm{ad}(\Phi)+1}\,)$$

□

3.3 An Example

In this section we illustrate our model-checking algorithm by verifying the μ-formula Φ_{ex} of Example 2.8 for the context-free process $\mathcal{C}_{ex}$ given in Figure 1. Figure 3 shows the property transformer scheme PTS_{ex} constructed from $\mathcal{C}_{ex}$ and Φ_{ex}, as well as the corresponding fixpoint computation. To shorten the example we have evaluated $\{ \Psi_2, \ldots, \Psi_7 \}$ simultaneously, as they occur in the same alternation level, and we have used simply Ψ_6 instead of $\mathtt{mem}_{\Psi_6}$. At the bottom of the figure we, finally, apply the resulting characteristic property transformer $[\![A]\!]^{\Psi_1}$ to the set of subformulas satisfied by ϵ. Since the application yields true, we have thus verified that A satisfies indeed the property represented by Ψ_1.

4 The Correctness Proof

The observation that the fixpoint computation for μ-formulas with only one sort of fixpoint can be done simultaneously was used in [BS92, BS95] to verify closed alternation-free formulas by merely exploring the assertion-based semantics. For formulas with alternation, however, we need a stronger framework which takes also into account the bindings of free variables where the parity may alternate. These bindings will be dealt with by *dynamic environments* which are decribed in terms of deterministic finite-state automata introduced in Section 4.1. Subsequently, in Section 4.2 we show that dynamic environments may represent the semantic solution of the model-checking problem, and that there exists an iterative algorithm which admits the computation of the semantic solution.

4.1 Dynamic Environments

In the presence of formulas containing free variables the simple composition property of Proposition 2.9.1 no longer captures correctly the behaviour of context-free processes wrt. the specification at hand. This defect is eliminated by the slight modification given below.

$$\{ \Gamma, \mathcal{V}' \} \; \alpha\beta \; \{ \Delta, \mathcal{V} \} \quad \text{iff} \quad \exists \; \Sigma, \mathcal{V}'' \; \{ \Gamma, \mathcal{V}' \} \; \alpha \; \{ \Sigma, \mathcal{V}'' \} \text{ and } \{ \Sigma, \mathcal{V}'' \} \; \beta \; \{ \Delta, \mathcal{V} \}$$

Intuitively, the modified composition rule expresses that in addition to assertions also *environments* must be adapted when considered at intermediate states. In general, the valuation $\mathcal{V}''$ is obtained from $\mathcal{V}$ by right cancellation of β, i.e. for all $X \in \mathit{dom}(\mathcal{V})$, $\mathcal{V}''(X) = (\mathcal{V}(X) \cap V^*\beta)\beta^{-1}$. As an example, $\alpha\beta \in \mathcal{V}(X)$ would imply $\alpha \in \mathcal{V}''(X)$.

In the remainder of this section we fix now a context-free system $\mathcal{C}$, and a formula Φ with closure $\{ \Psi_1, \ldots, \Psi_n \}$. Our aim is to develop a formalism, the *dynamic environments*, which faithfully models the adaptations of valuations needed for composition. Dynamic environments will be partitioned into levels $k \in [1, n]$ where a dynamic environment of level k defines the valuations for $\{ \Psi_1, \ldots, \Psi_k \}$. This change from valuations for variables to valuations for subformulas is reflected in the semanics by adding the rule "if $\Psi \in \mathit{dom}(\mathcal{V})$ then $(\alpha \models_{\mathcal{V}} (\Psi, \Delta)$ if $\alpha \in \mathcal{V}(\Psi))$". The original model-checking problem is then reduced to a corresponding fixpoint problem on the finite domain of dynamic

$$\mathcal{C}_{ex} = \left\{ \begin{array}{lcl} A & \xrightarrow{a} & AB \\ A & \xrightarrow{b} & \epsilon \\ B & \xrightarrow{b} & \epsilon \end{array} \right\} \quad \Phi_{ex} = \left\{ \begin{array}{lcl} \Psi_1 & \overset{\mu}{=} & \Psi_2 \\ \Psi_2 & \overset{\nu}{=} & \Psi_3 \\ \Psi_3 & = & \Psi_4 \wedge \Psi_5 \\ \Psi_4 & = & [b]\Psi_6 \\ \Psi_5 & = & [a]\Psi_7 \\ \Psi_6 & = & \Psi_1 \\ \Psi_7 & = & \Psi_2 \end{array} \right\} \quad PTS_{ex} = \left\{ \begin{array}{lcllcl} \llbracket A \rrbracket^{\Psi_1} & \overset{\mu}{=} & \llbracket A \rrbracket^{\Psi_2} & \llbracket B \rrbracket^{\Psi_1} & \overset{\mu}{=} & \llbracket B \rrbracket^{\Psi_2} \\ \llbracket A \rrbracket^{\Psi_2} & \overset{\nu}{=} & \llbracket A \rrbracket^{\Psi_3} & \llbracket B \rrbracket^{\Psi_2} & \overset{\nu}{=} & \llbracket B \rrbracket^{\Psi_3} \\ \llbracket A \rrbracket^{\Psi_3} & = & \llbracket A \rrbracket^{\Psi_4} \sqcap \llbracket A \rrbracket^{\Psi_5} & \llbracket B \rrbracket^{\Psi_3} & = & \llbracket B \rrbracket^{\Psi_4} \sqcap \llbracket B \rrbracket^{\Psi_5} \\ \llbracket A \rrbracket^{\Psi_4} & = & \llbracket \epsilon \rrbracket^{\Psi_6} & \llbracket B \rrbracket^{\Psi_4} & = & \llbracket \epsilon \rrbracket^{\Psi_6} \\ \llbracket A \rrbracket^{\Psi_5} & = & \llbracket AB \rrbracket^{\Psi_7} & \llbracket B \rrbracket^{\Psi_5} & = & 1 \\ \llbracket A \rrbracket^{\Psi_6} & = & \llbracket A \rrbracket^{\Psi_1} & \llbracket B \rrbracket^{\Psi_6} & = & \llbracket B \rrbracket^{\Psi_1} \\ \llbracket A \rrbracket^{\Psi_7} & = & \llbracket A \rrbracket^{\Psi_2} & \llbracket B \rrbracket^{\Psi_7} & = & \llbracket B \rrbracket^{\Psi_2} \end{array} \right\}$$

Fixpoint computation for PTS_{ex}:

	$\llbracket A \rrbracket^{\Psi_1}$	$\llbracket A \rrbracket^{\Psi_2}$	$\llbracket A \rrbracket^{\Psi_3}$	$\llbracket A \rrbracket^{\Psi_4}$	$\llbracket A \rrbracket^{\Psi_5}$	$\llbracket A \rrbracket^{\Psi_6}$	$\llbracket A \rrbracket^{\Psi_7}$	$\llbracket B \rrbracket^{\Psi_1}$	$\llbracket B \rrbracket^{\Psi_2}$	$\llbracket B \rrbracket^{\Psi_3}$	$\llbracket B \rrbracket^{\Psi_4}$	$\llbracket B \rrbracket^{\Psi_5}$	$\llbracket B \rrbracket^{\Psi_6}$	$\llbracket B \rrbracket^{\Psi_7}$
1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
2		Ψ_6	Ψ_6	Ψ_6	1	0	1		Ψ_6	Ψ_6	Ψ_6	1	0	1
3		0	0	Ψ_6	0	0	Ψ_6		Ψ_6	Ψ_6	Ψ_6	1	0	Ψ_6
4		0	0	Ψ_6	0	0	0		Ψ_6	Ψ_6	Ψ_6	1	0	Ψ_6
5	0	1	1	1	1	1	1	Ψ_6	1	1	1	1	1	1
6		Ψ_6	Ψ_6	Ψ_6	1	0	1		Ψ_6	Ψ_6	Ψ_6	1	Ψ_6	1
7		Ψ_6	Ψ_6	Ψ_6	Ψ_6	0	Ψ_6		Ψ_6	Ψ_6	Ψ_6	1	Ψ_6	Ψ_6
8	Ψ_6	1	1	1	1	1	1	Ψ_6	1	1	1	1	1	1
9		Ψ_6	Ψ_6	Ψ_6	1	Ψ_6	1		Ψ_6	Ψ_6	Ψ_6	1	Ψ_6	1
10		Ψ_6	Ψ_6	Ψ_6	Ψ_6	Ψ_6	Ψ_6		Ψ_6	Ψ_6	Ψ_6	1	Ψ_6	Ψ_6

Standard model-checking for ϵ delivers $\epsilon \models \Delta_\epsilon =_{\mathrm{df}} \{\, \Psi_1, \Psi_2, \Psi_3, \Psi_4, \Psi_5, \Psi_6, \Psi_7 \,\}$. Overall we thus obtain:

$$A \models \Psi_1 \quad \text{iff} \quad A \models (\Psi_1, \Delta_\epsilon) \quad \text{iff} \quad \llbracket A \rrbracket^{\Psi_1}(\Delta_\epsilon) = 1 \quad \text{iff} \quad \mathtt{mem}_{\Psi_6}(\Delta_\epsilon) = 1 \quad \text{iff} \quad \Psi_6 \in \Delta_\epsilon$$

Figure 3: The verification of Φ_{ex} for $\mathcal{C}_{ex}$.

environments, such that the semantics of the original formula is captured by the final environment of level n.

Definition 4.1 (Dynamic Environment)
A dynamic environment $\bar{\mathcal{A}}_k$ *of level* $k \in [1, n]$ *is a sequence of deterministic finite-state automata* $\mathcal{A}_i = (Q_{\mathcal{A}_i}, V, \delta_{\mathcal{A}_i}, F_{\mathcal{A}_i})$, $i = 1, \ldots, k$, *where*

- $Q_{\mathcal{A}_i} = (2^{CL(\Phi)})^i$ *are the state sets of the automata,*
- V *is the input alphabet,*
- $\delta_{\mathcal{A}_i} : Q_{\mathcal{A}_i} \times V \to Q_{\mathcal{A}_i}$ *are the transition functions obeying the constraints*

  $$\delta_{\mathcal{A}_i}(\bar{\Delta}_i, A) = \bar{\Gamma}_i \quad \textit{implies} \quad \delta_{\mathcal{A}_{i-1}}(\bar{\Delta}_{i-1}, A) = \bar{\Gamma}_{i-1}$$

  where $\bar{\Delta}_i$ *denotes* $(\Delta_1, \ldots, \Delta_i)$, *and*
- $F_{\mathcal{A}_i} = \{\, \bar{\Delta}_i \in Q_{\mathcal{A}_i} \mid \Psi_i \in \Delta_i \,\}$ *is the set of accepting states.*

Denoting the transitive closure of $\delta_{\mathcal{A}_i}$, *as usual, also by* $\delta_{\mathcal{A}_i}$ *the language accepted by* $\mathcal{A}_i$ *starting in the state* $\bar{\Delta}_i$ *is* $\mathcal{L}_{\mathcal{A}_i}(\bar{\Delta}_i) = \{\, \alpha \in V^* \mid \delta_{\mathcal{A}_i}(\bar{\Delta}_i, \tilde{\alpha}) \in F_{\mathcal{A}_i} \,\}$ *where* $\tilde{\alpha}$ *is the reverse of* α[3].

A dynamic environment $\bar{\mathcal{A}}_k$ *together with a state* $\bar{\Delta}_k$ *is then interpreted as an environment which defines valuations for* $\Psi_1, \ldots, \Psi_k$ *by means of*

$$\begin{aligned}
&\bar{\mathcal{A}}_1\langle\bar{\Delta}_1\rangle =_{\mathrm{df}} [\, \Psi_1 \mapsto \mathcal{L}_{\mathcal{A}_1}(\bar{\Delta}_1) \,] \\
&\bar{\mathcal{A}}_k\langle\bar{\Delta}_k\rangle =_{\mathrm{df}} \bar{\mathcal{A}}_{k-1}\langle\bar{\Delta}_{k-1}\rangle\ [\, \Psi_k \mapsto \mathcal{L}_{\mathcal{A}_k}(\bar{\Delta}_k) \,] \quad \textit{for}\ \ 2 \le k \le n
\end{aligned}$$

Dynamic environments are a convenient formalism to describe the semantics of μ-formulas on context-free processes since they model compositionality simply by transitions in the finite automata.

Lemma 4.2 *Let* $\{\, \Gamma, \mathcal{V}' \,\}\ A\ \{\, \Delta, \bar{\mathcal{A}}_k\langle\bar{\Delta}_k\rangle \,\}$. *Then*

1. *For all* $i \le k$, $\Psi_i \in \Gamma$ *iff* $A \in \bar{\mathcal{A}}_k\langle\bar{\Delta}_k\rangle(\Psi_i)$, *and*
2. $\mathcal{V}' = \bar{\mathcal{A}}_k\langle\delta_{\mathcal{A}_k}(\bar{\Delta}_k, A)\rangle$.

The first property expresses that a dynamic environment of level k captures the semantics of all subformulas up to level k, while the second property states that the environment to be considered in the pre-condition of A coincides with the interpretation of the A-successor of $\bar{\Delta}_k$ in $\mathcal{A}_k$.

[3]Here we have to use $\tilde{\alpha}$ as the automaton has to model the above mentioned right cancellation

The granularity of the transition functions of dynamic environments is not sufficient to obtain a match between the semantic and the iterative intuition behind the model checking problem. We therefore split these transition functions into *characteristic transition functions* (CTF) $\delta^{i,j}$ as follows.

$$\delta^{i,j}(\bar{\Delta}_i, A) = \begin{cases} 1 & \text{if } \Psi_j \in \delta^i_{\mathcal{A}_i}(\bar{\Delta}_i, A) \\ 0 & \text{otherwise} \end{cases}$$

where $\delta^j_{\mathcal{A}_i}(\bar{\Delta}_i, A) =_{\mathrm{df}} \Gamma_j$ if $\delta_{\mathcal{A}_i}(\bar{\Delta}_i, A) = \bar{\Gamma}_i$. CTF's can naturally be extended to words over V by means of

$$\begin{array}{lcl} \delta^{i,j}(\bar{\Delta}_i, \epsilon) & = & \mathtt{mem}_{\Psi_j}(\Delta_i) \\ \delta^{i,j}(\bar{\Delta}_i, \alpha A) & = & \delta^{i,j}(\delta_{\mathcal{A}_i}(\bar{\Delta}_i, A), \alpha) \end{array}$$

The split into characteristic transition functions allows us to view a dynamic environment $\bar{\mathcal{A}}_k$ as a matrix of CTF's as depicted below.

$$\begin{array}{ccccccc} \delta^{1,1} & \delta^{1,2} & \dots & \delta^{1,k} & \dots & \delta^{1,n} \\ \vdots & & \ddots & & & \vdots \\ \delta^{k,1} & \delta^{k,2} & \dots & \delta^{k,k} & \dots & \delta^{k,n} \end{array}$$

This matrix can be systematically extended to a matrix for $\bar{\mathcal{A}}_{k+1}$ with new row $(\delta^{k+1,1}, \dots, \delta^{k+1,n})$ by means of a fixpoint computation such that the final result will capture the semantics of the formula Φ on the given process. During this computation we need to update a characteristic transition function by a function of the same column. The arity of the function is then adapted by either suppressing or by replicating arguments. Formally, we define

$$\begin{array}{ll} (\delta^{i,j} \uparrow k)\ (\bar{\Delta}_k, A) =_{\mathrm{df}} \delta^{i,j}(\bar{\Delta}_i, A) & \text{where } k \geq i, \text{ and} \\ (\delta^{i,j} \downarrow k)\ (\bar{\Delta}_k, A) =_{\mathrm{df}} \delta^{i,j}(\bar{\Delta}_i, A) & \text{where } k \leq i \text{ and } \Delta_j = \Delta_k, \text{ for } j = k+1, \dots, i. \end{array}$$

Intuitively, the first line describes that a CTF of level i which is used at level $k \geq i$ ignores the local formula sets $\Delta_{i+1}, \dots, \Delta_k$, while the second line expresses that a CTF of level i which is used at level $k \leq i$ takes as arguments $\Delta_1, \dots, \Delta_k$, and replicates Δ_k $i-k$ times. To simplify the subsequent presentation of the algorithm we will, however, use $\delta^{i,j}$ also for the adapted CTF's.

As will be elaborated on in the next subsection, the matrices defined above are adequate for proving our main result, Theorem 4.7, i.e. the equivalence of the semantic and the iterative algorithm presented in Section 4.2, because it is possible to "synchronize" their corresponding computations on the diagonal.

4.2 Semantic and Iterative Solutions

Given the semantics of the formulas $\Psi_1, \dots, \Psi_{k-1}$ in terms of a dynamic environment $\bar{\mathcal{A}}_{k-1}$ we will now consider the semantics of the remaining formulas $\Psi_k, \dots, \Psi_n$.

Definition 4.3 (Semantic Solutions)
We call $\bar{\mathcal{A}}_k$, for $k \in [1, n]$, the semantic solution *of $\bar{\mathcal{A}}_{k-1}$, written as $\mathcal{S}(\bar{\mathcal{A}}_{k-1})$, if the transition function of $\mathcal{A}_k$ satisfies*

$$\delta_{\mathcal{A}_k}(\bar{\Delta}_k, A) = \bar{\Gamma}_k \quad \textit{iff} \quad (\Gamma_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Gamma}_{k-1}\rangle) \; A \; (\Delta_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Delta}_{k-1}\rangle).$$

Moreover, we call $(\bar{\mathcal{A}}_k, \ldots, \bar{\mathcal{A}}_n)$ the semantic solutions *of $\bar{\mathcal{A}}_{k-1}$, denoted by $\bar{\mathcal{S}}(\bar{\mathcal{A}}_{k-1})$, if $\bar{\mathcal{A}}_i = \mathcal{S}(\bar{\mathcal{A}}_{i-1})$, for $i \in [k, n]$.*

It turns out that the semantic solution respects the standard substitution lemma.

Lemma 4.4 *Let $(\Gamma_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Gamma}_{k-1}\rangle) \; A \; (\Delta_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Delta}_{k-1}\rangle)$ and let $\bar{\mathcal{A}}_k$ be the* semantic *solution of $\bar{\mathcal{A}}_{k-1}$. Then*

$$(\Gamma_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Gamma}_{k-1}\rangle[\,\Psi_k \mapsto \mathcal{L}_{\mathcal{A}_k}(\bar{\Gamma}_k)\,]) \; A \; (\Delta_k, \bar{\mathcal{A}}_{k-1}\langle\bar{\Delta}_{k-1}\rangle[\,\Psi_k \mapsto \mathcal{L}_{\mathcal{A}_k}(\bar{\Delta}_k)\,])$$

Corollary 4.5 (Diagonal Consistency)
If $\bar{\mathcal{A}}_k, \ldots, \bar{\mathcal{A}}_n$ are the semantic solutions of $\bar{\mathcal{A}}_{k-1}$ then $\delta^{i,j} = \delta^{j,j}$, for $i \in [k, n], j \in [1, n]$.

Due to this corollary we may simply identify the semantic solutions $\bar{\mathcal{A}}_k, \ldots, \bar{\mathcal{A}}_n$ with the characteristic transition functions $\delta^{k,k}, \ldots, \delta^{n,n}$. Conversely, given $\bar{\mathcal{A}}_{k-1}$ and (arbitrary) $\delta^{k,k}, \ldots, \delta^{n,n}$ we define

- $\bar{\mathcal{A}}_{k-1} \oplus (\delta^{k,k}, \ldots, \delta^{n,n})$ as $\bar{\mathcal{A}}_{k-1}$ augmented by $\delta^{k,j} =_{\mathrm{df}} \delta^{j,j}$, for $j \in [1, n]$, and
- $\bar{\mathcal{A}}_{k-1} \otimes (\delta^{k,k}, \ldots, \delta^{n,n})$ as $\bar{\mathcal{A}}_{k-1}$ augmented by $\delta^{i,j} =_{\mathrm{df}} \delta^{j,j}$, for $i \in [k, n], j \in [1, n]$.

Let us finally sketch the resulting (conceptual) algorithm which iteratively computes the semantic solutions for $\bar{\mathcal{A}}_{k-1}$. Given $\bar{\mathcal{A}}_{k-1}$, we would like to compute $\delta^{i,j}$ for $i \in [k, n], j \in [1, n]$. By Corollary 4.5 we already know that $\delta^{i,j} = \delta^{j,j}$ for $i \in [k, n], j \in [1, k-1]$. The remaining characteristic transition functions are then computed level-wise by a two-level fixpoint computation. During the inner-level computation we have fixed some approximant $\delta^{k,k}$ and vary the values of $\delta^{k,k+1}, \ldots, \delta^{k,n}$. The idea is that $(\delta^{k,1}, \ldots, \delta^{k,n})$ together with $\bar{\mathcal{A}}_{k-1}$ defines a dynamic environment $\bar{\mathcal{A}}_k$ for which we can compute the semantic solutions $\theta^{k+1,k+1}, \ldots, \theta^{n,n}$ by induction. We may therefore update $\delta^{k,k+1}, \ldots, \delta^{k,n}$ by $\theta^{k+1,k+1}, \ldots, \theta^{n,n}$, and repeat this iteration until we reach consistency. In the outer-level fixpoint computation we may now update the fixed $\delta^{k,k}$ by evaluating the characteristic transition function for the "unfolding" of Ψ_k in the current setting, and start the inner fixpoint computation again. Our main theorem then states that if we have reached consistency also at the outer-level then the iterative and the semantic solutions for $\bar{\mathcal{A}}_{k-1}$ coincide.

Formally, given a dynamic environment $\bar{\mathcal{A}}_{k-1}$, we define a monotone function $GI_k(\delta^{k,k})$ called the *global iteration function* at level k with respect to $\delta^{k,k}$ by

$$GI_k(\delta^{k,k})(\delta^{k+1,k+1}, \ldots, \delta^{n,n}) =_{\mathrm{df}} (\theta^{k+1,k+1}, \ldots, \theta^{n,n}).$$

where $(\theta^{k+1,k+1}, \ldots, \theta^{n,n})$ are the semantic solutions of $\bar{\mathcal{A}}_k = \bar{\mathcal{A}}_{k-1} \oplus (\delta^{k,k}, \ldots, \delta^{n,n})$. Accordingly, $\mu GI_k(\delta^{k,k})$ denotes now the least fixpoint of $GI_k(\delta^{k,k})$, while $\nu GI_k(\delta^{k,k})$ denotes the greatest. Notice, moreover, that both fixpoints are monotone wrt. $\delta^{k,k}$.

In order to define in a second step the *local iteration function* LI_k at level k we first consider the CTF $[\![\Psi_k]\!]_{\bar{\mathcal{A}}_n} : Q_{\mathcal{A}_k} \times V \to \{0,1\}$ representing the "unfolding" of a formula Ψ_k wrt. a dynamic environment $\bar{\mathcal{A}}_n$. It is given by

$$[\![\Psi_k]\!]_{\bar{\mathcal{A}}_n} =_{\mathrm{df}} \begin{cases} \lambda(\bar{\Delta}_k, A).1 & \text{if } \Psi_k = \mathtt{tt} \\ \lambda(\bar{\Delta}_k, A).0 & \text{if } \Psi_k = \mathtt{ff} \\ \bar{\mathcal{A}}_n[i,i] & \text{if } \Psi_k = X \text{ and } \Psi_i = \sigma X.\Psi_j \\ \bar{\mathcal{A}}_n[i,i] \sqcup \bar{\mathcal{A}}_n[j,j] & \text{if } \Psi_k = \Psi_i \vee \Psi_j \\ \bar{\mathcal{A}}_n[i,i] \sqcap \bar{\mathcal{A}}_n[j,j] & \text{if } \Psi_k = \Psi_i \wedge \Psi_j \\ \lambda(\bar{\Delta}_k, A). \bigsqcup_{A \xrightarrow{a} \alpha} \bar{\mathcal{A}}_n[i,i](\bar{\Delta}_k, \alpha) & \text{if } \Psi_k = \langle a \rangle \Psi_i \\ \lambda(\bar{\Delta}_k, A). \bigsqcap_{A \xrightarrow{a} \alpha} \bar{\mathcal{A}}_n[i,i](\bar{\Delta}_k, \alpha) & \text{if } \Psi_k = [a] \Psi_i \\ \bar{\mathcal{A}}_n[i,i] & \text{if } \Psi_k = \sigma X.\Psi_i \end{cases}$$

where $\bar{\mathcal{A}}_n[i,j]$ denotes the CTF of $\bar{\mathcal{A}}_n$ ocurring in row i, and in column j.

Now let σ be the parity[4] of Ψ_k, $\sigma GI_k(\delta^{k,k}) = (\delta^{k+1,k+1}, \ldots, \delta^{n,n})$, and $(\bar{\mathcal{A}}_{k+1}, \ldots, \bar{\mathcal{A}}_n)$ be the semantic solutions of $\bar{\mathcal{A}}_k = \bar{\mathcal{A}}_{k-1} \oplus (\delta^{k,k}, \ldots, \delta^{n,n})$. Then we define

$$LI_k(\delta^{k,k}) =_{\mathrm{df}} [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n}$$

As LI_k is again monotone, we may build its least fixpoint μLI_k, as well as its greatest fixpoint νLI_k. Finally, we use GI_k and LI_k to define the *iterative* solutions of $\bar{\mathcal{A}}_{k-1}$, written as $\bar{\mathcal{I}}(\bar{\mathcal{A}}_{k-1})$, by

$$\bar{\mathcal{I}}(\bar{\mathcal{A}}_{k-1}) =_{\mathrm{df}} (\vartheta^{k,k}, \ldots, \vartheta^{n,n})$$

where σ is the parity of Ψ_k, $\vartheta^{k,k} = \sigma LI_k$, and $(\vartheta^{k+1,k+1}, \ldots, \vartheta^{n,n}) = \sigma GI_k(\vartheta^{k,k})$.

Based on the following lemma, which states that the semantic solutions are a fixpoint of GI_k and LI_k, the key to our algorithm is now an iterative characterization of the semantic solutions.

Lemma 4.6 *Let $\bar{\mathcal{S}}(\bar{\mathcal{A}}_{k-1}) = (\bar{\mathcal{A}}_k, \ldots, \bar{\mathcal{A}}_n)$ with CTF's $(\theta^{k,k}, \ldots, \theta^{n,n})$. Then we have*

1. *$GI_k(\theta^{k,k})(\theta^{k+1,k+1}, \ldots, \theta^{n,n}) = (\theta^{k+1,k+1}, \ldots, \theta^{n,n})$, and*
2. *$LI_k(\theta^{k,k}) = \theta^{k,k}$.*

Theorem 4.7 *For any given dynamic environment $\bar{\mathcal{A}}_k$ the semantic and the iterative solutions coincides, i.e. $\bar{\mathcal{S}}(\bar{\mathcal{A}}_k) = \bar{\mathcal{I}}(\bar{\mathcal{A}}_k)$.*

[4]For non-fixpoint formulas we can take σ to be either μ or ν as the semantics of the subformulas of Ψ_k do not depend on the semantics of Ψ_k itself.

Proof: The theorem is shown by induction on k. For the induction base where $k = n$ there is nothing to show, as $\bar{\mathcal{S}}(\bar{\mathcal{A}}_n) = () = \bar{\mathcal{I}}(\bar{\mathcal{A}}_n)$. Now consider a dynamic environment $\bar{\mathcal{A}}_{k-1}$ with $k-1 < n$, and assume the theorem holds for all $i \in [k, n]$. The induction step for $\Psi_k = \texttt{tt}, \texttt{ff}, X, \Psi_i \vee \Psi_j, \Psi_i \wedge \Psi_j, \langle a \rangle \Psi_i$, and $[a]\Psi_i$ is proved using the definition of LI_k and the fact that the semantics of occurring proper subformulas do not depend on the semantics of Ψ_k itself. We therefore prove the induction step only for the case where Ψ_k is a minimal fixpoint operator $\mu X.\Psi_l$, since the case for maximal fixpoints is completely dual.

Let $\theta = (\theta^{k,k}, \ldots, \theta^{n,n})$ be the semantic solutions of $\bar{\mathcal{A}}_{k-1}$, and $\bar{\mathcal{S}}_k = \bar{\mathcal{A}}_{k-1} \oplus \theta$. Moreover, let $\vartheta = (\vartheta^{k,k}, \ldots, \vartheta^{n,n})$ be the iterative solutions of $\bar{\mathcal{A}}_{k-1}$, i.e. $\vartheta^{k,k} = \mu LI_k$ and $(\vartheta^{k+1,k+1}, \ldots, \vartheta^{n,n}) = \mu GI_k(\vartheta^{k,k})$, as well as $\bar{\mathcal{I}}_k = \bar{\mathcal{A}}_{k-1} \oplus \vartheta$.

$\boxed{\textbf{(I)}\ (\vartheta^{k,k}, \ldots, \vartheta^{n,n}) \sqsubseteq (\theta^{k,k}, \ldots, \theta^{n,n})}$

To show the inequality we construct inductively a chain of dynamic environments $\bar{\mathcal{A}}_n^{(i)}$, for $i \geq 0$, as follows. Let

$$\bar{\mathcal{A}}_n^{(i)} = \bar{\mathcal{A}}_{k-1} \otimes (\vartheta^{(i)k,k}, \ldots, \vartheta^{(i)n,n})$$

where

$$\begin{array}{rclrcl} \vartheta^{(0)k,k} & = & \theta^{k,k}, & (\vartheta^{(0)k+1,k+1}, \ldots, \vartheta^{(0)n,n}) & = & \mu GI_k(\vartheta^{(0)k,k}) \\ \vartheta^{(i+1)k,k} & = & [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n^{(i)}}, & (\vartheta^{(i+1)k+1,k+1}, \ldots, \vartheta^{(i+1)n,n}) & = & \mu GI_k(\vartheta^{(i+1)k,k}) \end{array}$$

Claim : $\bar{\mathcal{A}}_n^{(i)}$, for $i \geq 0$, is a descending chain of dynamic environments.

Proof: First note that $\vartheta^{(i)k,k} \sqsupseteq \vartheta^{(i+1)k,k}$ implies $\bar{\mathcal{A}}_n^{(i)} \sqsupseteq \bar{\mathcal{A}}_n^{(i+1)}$, since by definition

$$(\vartheta^{(i)k+1,k+1}, \ldots, \vartheta^{(i)n,n}) = \mu GI_k(\vartheta^{(i)k,k}) \sqsupseteq \mu GI_k(\vartheta^{(i+1)k,k}) = (\vartheta^{(i+1)k+1,k+1}, \ldots, \vartheta^{(i+1)n,n}).$$

The claim is then proved by showing

(4.7.1) $\bar{\mathcal{A}}_n^{(0)} \sqsupseteq \bar{\mathcal{A}}_n^{(1)}$, and

(4.7.2) $\bar{\mathcal{A}}_n^{(i-1)} \sqsupseteq \bar{\mathcal{A}}_n^{(i)}$ implies $\bar{\mathcal{A}}_n^{(i)} \sqsupseteq \bar{\mathcal{A}}_n^{(i+1)}$.

(4.7.1) Since by Lemma 4.6.1 $GI_k(\theta^{k,k})(\theta^{k+1,k+1}, \ldots, \theta^{n,n}) = (\theta^{k+1,k+1}, \ldots, \theta^{n,n})$ we see that

$$(\theta^{k+1,k+1}, \ldots, \theta^{n,n}) \sqsupseteq \mu GI_k(\theta^{k,k}) = (\vartheta^{(0)k+1,k+1}, \ldots, \vartheta^{(0)n,n}),$$

and hence

$$\vartheta^{(0)k,k} = \theta^{k,k} = [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n^{\theta}} \sqsupseteq [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n^{(0)}} = \vartheta^{(1)k,k}.$$

Thus we obtain $\bar{\mathcal{A}}_n^{(0)} \sqsupseteq \bar{\mathcal{A}}_n^{(1)}$.

(4.7.2) Now let $\bar{\mathcal{A}}_n^{(i-1)} \sqsupseteq \bar{\mathcal{A}}_n^{(i)}$, for some $i \geq 1$. Since

$$\vartheta^{(i)k,k} = [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n^{(i-1)}} \sqsupseteq [\![\Psi_k]\!]_{\bar{\mathcal{A}}_n^{(i)}} = \vartheta^{(i+1)k,k}$$

by definition, we immediately obtain $\bar{\mathcal{A}}_n^{(i)} \sqsupseteq \bar{\mathcal{A}}_n^{(i+1)}$, as desired. □

Since the domain of all dynamic environments wrt. Φ and $\mathcal{C}$ is finite, we eventually reach a fixpoint $\bar{\mathcal{A}}_n^{(fix)}$ for which we conclude

$$(\vartheta^{k,k}, \ldots, \vartheta^{n,n}) \sqsubseteq (\vartheta^{(fix)k,k}, \ldots, \vartheta^{(fix)n,n}) \sqsubseteq (\vartheta^{(0)k,k}, \ldots, \vartheta^{(0)n,n}) \sqsubseteq (\theta^{k,k}, \ldots, \theta^{n,n}).$$

$\boxed{\textbf{(II)}\ \mathcal{L}_{\bar{\mathcal{I}}_k}(\bar{\Delta}_k) \supseteq \mathcal{L}_{\bar{\mathcal{S}}_k}(\bar{\Delta}_k), \quad \text{for any consistent } \bar{\Delta}_k}$

$\bar{\Delta}_k$ is said to be *consistent* if each Δ_i, for $i \in [1, k]$, is consistent, and Δ_i is called consistent if it satisfies the following conditions: $\texttt{ff} \notin \Delta_i$, $\Psi_1 \vee \Psi_2 \in \Delta_i$ iff $\Psi_1 \in \Delta_i$ or $\Psi_2 \in \Delta_i$, $\Psi_1 \wedge \Psi_2 \in \Delta_i$ iff $\Psi_1 \in \Delta_i$ and $\Psi_2 \in \Delta_i$, and $\sigma X.\Psi \in \Delta_i$ iff $X \in \Delta_i$ iff $\Psi \in \Delta_i$.

As $\Psi_k = \mu X.\Psi_l$, **(II)** follows now from the fixpoint property

$$\alpha \models_{\bar{\mathcal{I}}_k \langle \bar{\Delta}_k \rangle} (\Psi_l, \Delta_k) \quad \text{iff} \quad \alpha \in \mathcal{L}_{\mathcal{I}_k}(\bar{\Delta}_k)$$

which is proved by induction on the length of processes. For the induction base we have

$$\begin{array}{lll} & \epsilon \models_{\bar{\mathcal{I}}_k \langle \bar{\Delta}_k \rangle} (\Psi_l, \Delta_k) & \\ \text{iff} & \Psi_l \in \Delta_k & \\ \text{iff} & \Psi_k \in \Delta_k & \text{by consistency of } \Delta_k \\ \text{iff} & \epsilon \in \mathcal{L}_{\mathcal{I}_k}(\bar{\Delta}_k) & \end{array}$$

The induction step then follows from

$$\begin{array}{lll} & \alpha A \models_{\bar{\mathcal{I}}_k \langle \bar{\Delta}_k \rangle} (\Psi_l, \Delta_k) & \\ \text{iff} & \alpha \models_{\bar{\mathcal{I}}_k \langle \delta_{\mathcal{I}_k}(\bar{\Delta}_k, A) \rangle} (\Psi_l, \delta^k_{\mathcal{I}_k}(\bar{\Delta}_k, A)) & \text{by induction hypothesis on k} \\ \text{iff} & \alpha \in \mathcal{L}_{\mathcal{I}_k}(\delta_{\mathcal{I}_k}(\bar{\Delta}_k, A)) & \text{by induction hypothesis on the length of } \alpha \\ \text{iff} & \alpha A \in \mathcal{L}_{\mathcal{I}_k}(\bar{\Delta}_k) & \end{array}$$

(III) Since **(I)** implies $\mathcal{L}_{\bar{\mathcal{I}}_k}(\bar{\Delta}_k) \subseteq \mathcal{L}_{\bar{\mathcal{I}}_k}(\bar{\Delta}_k)$, we conclude from **(II)** $\mathcal{L}_{\bar{\mathcal{I}}_k}(\bar{\Delta}_k) \subseteq \mathcal{L}_{\bar{\mathcal{I}}_k}(\bar{\Delta}_k)$, for any consistent $\bar{\Delta}_k$. We have hence $\vartheta^{k,k} = \theta^{k,k}$, and by induction hypothesis on k also $\vartheta^{i,i} = \theta^{i,i}$, for $i \in [k+1, n]$.

As Δ_ϵ is consistent, and characteristic transition functions preserve consistency, this concludes the proof. □

The observation that only the characteristic transition functions on the diagonal have to be taken into account when updating $\delta^{k,k}$ wrt. the current dynamic environment $\bar{\mathcal{A}}_n$, allows us to replace the "conceptual" algorithm used in the correctness proof to the "actual" model-checking algorithm presented in the previous section. This optimization is, finally, the key for proving the claimed complexity result.

5 Conclusions and Further Research

In this paper we have presented an iterative, elementary model-checking algorithm for context-free processes which deals with the full modal μ-calculus. This basic algorithm can be extended to the class of pushdown processes following the lines of [BS95], as well as to the class of regular graphs following the lines of [BQ96], respectively. Essentially, both extensions are obtained by taking into account the arity Q of pushdown processes (which corresponds to the number of states in the finite control), respectively regular graphs (which corresponds to the maximal arity of a hyperedge), which yields characteristic property transformers with multiple arguments. For these extensions our algorithm has the worst-time complexity

$$O(\,|\Phi| * (|Q| * |\mathcal{C}|)^{\mathrm{ad}(\Phi)+1} * 2^{|\Phi| * (\mathrm{ad}(\Phi)+|Q|)}\,).$$

Recently, Walukiewicz presented another model-checker for pushdown processes which uses games [Wal96]. His algorithm has a dramatically different complexity estimation. In particular it behaves much worse for increasing degrees of alternation depths.

$$O(\,|\mathcal{C}| * (2^{|Q| * |\Phi| * \mathrm{ad}(\Phi)})^{\mathrm{ad}(\Phi)}\,)$$

Whereas our algorithm directly mimics the behavioural intuition behind sequential processes and, in particular, keeps process and formula structure transparent, which gives a direct handle to extending the underlying process structure, Walukiewics' algorithm intertwines these structures, which, at least, complicates the identification of the modifications necessary for the extensions.

This structural distinction has also an impact on the further extensions we are planning: First, to extend model-checking to the class of rational restricted recognizable graphs as introduced in [Cau96], and second, to develop a local variant. Both extensions will exploit the structural transparency of our approach and, in particular, use the framework of dynamic environments.

References

[BQ96] O. Burkart and Y.-M. Quemener. Model-Checking of Infinite Graphs Defined by Graph Grammars. In *INFINITY '96*, MIP-9614, pages 56–70. Universität Passau, July 1996.

[Bra96] J.C. Bradfield. The Modal mu-Calculus Alternation Hierarchy is Strict. In *CONCUR '96*, LNCS 1119, pages 233–246. Springer, 1996.

[BS92] O. Burkart and B. Steffen. Model Checking for Context-Free Processes. In *CONCUR '92*, LNCS 630, pages 123–137. Springer, 1992.

[BS95] O. Burkart and B. Steffen. Composition, Decomposition and Model-Checking of Pushdown Processes. *Nordic Journal of Computing*, 2:89–125, 1995.

[Cau96] D. Caucal. On Infinite Transition Graphs Having a Decidable Monadic Theory. In *ICALP '96*, LNCS 1099, pages 194–205. Springer, 1996.

[CKS92] R. Cleaveland, M. Klein, and B. Steffen. Faster Model Checking for the Modal Mu-Calculus. In *CAV '92*, LNCS 663, pages 410–422, 1992.

[Cou90] B. Courcelle. Graph Rewriting: An Algebraic and Logic Approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 5, pages 193–242. Elsevier Science Publisher B.V., 1990.

[EK95] J. Esparza and A. Kiehn. On the Model Checking Problem for Branching Time Logics and Basic Parallel Processes. In *CAV '95*, LNCS 939, pages 353–366. Springer, 1995.

[Esp94] J. Esparza. On the Decidability of Model Checking for Several μ-calculi and Petri Nets. In *CAAP '94*, LNCS 787, pages 115–129. Springer, 1994.

[Esp96] J. Esparza. More Infinite Results. In *INFINITY '96, MIP-9614*, pages 4–20. Universität Passau, July 1996.

[HS94] H. Hungar and B. Steffen. Local Model-Checking for Context-Free Processes. *Nordic Journal of Computing*, 1(3):364–385, 1994.

[JW96] D. Janin and I. Walukiewicz. On the Expressive Completeness of the Propositional mu-Calculus wrt. Monadic Second Order Logic. In *CONCUR '96*, LNCS 1119, pages 263–277. Springer, 1996.

[Koz83] D. Kozen. Results on the Propositional μ-Calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[LBC$^+$94] D.E. Long, A. Browne, E.M. Clarke, S. Jha, and W.R. Marrero. An Improved Algorithm for the Evaluation of Fixpoint Expressions. In *CAV '94*, LNCS 818, pages 338–350. Springer, 1994.

[MS85] D.E. Muller and P.E. Schupp. The Theory of Ends, Pushdown Automata, and Second-Order Logic. *Theoretical Computer Science*, 37:51–75, 1985.

[Rab69] R.O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the AMS*, 141:1–35, 1969.

[Wal96] I. Walukiewicz. Pushdown Processes: Games and Model-Checking. In *CAV '96*, LNCS 1102. Springer, 1996.