Alternating Context-Free Languages and Linear Time μ-Calculus with Sequential Composition

Martin Lange [1]

Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstr. 67
D-80538 München, Germany

Abstract

This paper shows that linear time μ-calculus with sequential composition defines exactly those properties that are expressible with alternating context-free grammars for ω-words. This helps to understand the expressive power of modal μ-calculus with a chop operator and provides a logical characterisation of the class of alternating context-free languages.

1 Introduction

Beginning with the work of Büchi who showed that Monadic Second-Order Logic (MSO) defines exactly the class of regular languages, several connections between logics and classes of formal languages have been found. One important result concerns modal μ-calculus $\mathcal{L}_\mu$ which was shown to be equi-expressive to the bisimulation invariant fragment of MSO, [8].

Fixed Point Logic with Chop, FLC, is an extension of Kozen's modal μ-calculus $\mathcal{L}_\mu$, [9], which was introduced in [12]. It was shown to genuinely increase $\mathcal{L}_\mu$'s expressive power since it is able to express certain properties that are not even context-free. Some of these properties are interesting for verification purposes for example the fact that on every path of a transition system the number of *receive* actions never exceeds the number of *send* actions so far, [10].

However, FLC's exact expressive power is unknown. Here, we approach this question by defining its linear time equivalent LFLC, Linear Time Fixed

[1] Email: mlange@informatik.uni-muenchen.de

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

Point Logic with Chop, which can also be seen as linear time μ-calculus enriched with sequential composition. We show that LFLC defines exactly those languages which are generated by alternating context-free ω-grammars with parity conditions (ω-ACFGs).

The nonterminals of a classical context-free grammar (CFG) can be seen as existential. They generate a word iff there is a rule for them that generates a word. Alternating grammars feature universal nonterminals as well which generate a word iff all rules for this nonterminal generate the word. They have been studied in [11] for example where it is stated incorrectly that alternating CFGs generate exactly those languages accepted by alternating pushdown automata. This was corrected in [7] where APDAs are shown to be equi-expressive to *linear-erasing* alternating context-free grammars. There, the size of sentential forms in a derivation is linearly bounded by the size of the generated word.

Another form of alternation for context-free grammars was introduced in [13]. There, quantification is over those words that derive a certain word. I.e. in the derivation relation $w \Rightarrow^* v$ one quantifies over w in order to define whether v can be generated. We take the approach of [11] and quantify over those words v that are derived in one step from w where w contains a nonterminal. In other words, [13] considers alternation *over* grammars which gives them context-sensitivity since certain nonterminals are only replaced if they occur in a word of a certain form. We consider alternation *in* a grammar which keeps them truly context-free.

ω-grammars generalise ordinary grammars by allowing infinite words to be generated as well. Context-free ω-grammars have been studied in [3] for example.

There are several possible ways to specify what it means for a grammar to generate an infinite word. We are interested in LFLC which has extremal least and greatest fixed points that can be nested in a formula. It is known from work on automata for the modal μ-calculus that a *parity* condition captures nested fixed points of different types best, [4]. There, each state of the automaton is labelled with a parity index and the acceptance of a word depends on the parity of the least index which is visited infinitely often. We use parity indices for nonterminals of a grammar. This makes the translations between grammars and formulas easiest. It remains to see whether this notion is equivalent to Rabin or Streett or even Büchi conditions for ω-ACFGs.

The result that the class of LFLC definable languages coincides with ω-ACFL, the class of alternating context-free ω-languages, provides insight into both formalisms. Since the emptiness problem for alternating context-free languages is undecidable, LFLC is undecidable, too. LFLC's model checking problem is, however, easily seen to be decidable. Decidability of the word problem for ACFGs is not surprising since ACFL are contained in the class of context-sensitive languages (CSL). However, this gives a non-optimal upper PSPACE bound for ACFL's word problem.

We sketch a model checker for LFLC that runs in PTIME for finite words. Since there are context-sensitive languages whose fixed grammar word problem is PSPACE-complete already we get a separation result between ACFL and CSL unless PTIME = PSPACE.

It is known that ACFL strictly subsumes CFL. This and the former give a characterisation of which properties can be formalised in LFLC. These results carry over to FLC to some degree since it is easy to see that the set of labellings along paths in a model for an FLC formula is LFLC definable.

Finally, we show how to translate an LFLC formula into an alternating pushdown automaton. This is a generalisation of the translation from context-free grammars to Pushdown Automata over finite words, [2]. The translation from Pushdown Automata to context-free grammars is not easily generalisable to the framework of infinite words and alternation. [7] hints that the trick of letting the grammar guess a state that the automaton will be in fails since alternating automata can be in several states simultaneously. Furthermore, in the presence of alternation a word does not necessarily have a left-most derivation. Their result for linear-erasing ACFGs is not easily generalisable to ω-ACFGs since it depends heavily on the presence of end markers for generated words and, hence, on their finiteness.

2 Alternating Languages, Grammars and Pushdown Automata

For a thorough introduction to the theory of grammars and formal languages see [6] and [14]. Here we will just recall the definitions that will be used later on.

An *alphabet* Σ is a finite set of symbols $\{a, b, \ldots\}$. A *word* $w \in \Sigma^*$ is a finite sequence of symbols from Σ, an ω-*word* $w \in \Sigma^\omega$ an infinite one. The empty sequence is denoted by ϵ. $|w|_a$ denotes the number of occurrences of a in w. A *language* L is a subset of Σ^*, an ω-language a subset of Σ^ω. $\Sigma^\infty := \Sigma^* \cup \Sigma^\omega$.

Let $w, v \in \Sigma^\infty$. The sequential composition of w and v is wv if $w \in \Sigma^*$ and w if $w \in \Sigma^\omega$. The set of subwords of w is $Sub(w) = \{v \mid \exists v_1 \in \Sigma^*, \exists v_2 \in \Sigma^\infty, \text{ s.t. } w = v_1 v v_2\}$.

A *morphism* between two alphabets is a mapping $m : \Sigma_1 \to \Sigma_2^*$. Morphism extend to words and languages. If $w = a_0 a_1 \ldots \in \Sigma^\infty$ then $h(w) := h(a_0)h(a_1)\ldots$, and for a language $L \subseteq \Sigma^\infty$, $h(L) := \{h(w) \mid w \in L\}$. A *substitution* $s : \Sigma_1 \to 2^{\Sigma_2}$ maps each letter of an alphabet to a language. It is extended to words and languages in the same way as morphisms are.

A *grammar* is a quadruple (N, Σ, P, S) where N is a finite set of symbols $\{X, Y, \ldots\}$ called *nonterminals*, $S \in N$ the starting symbol, and P a finite set of production rules of the form $P \subset (N \cup \Sigma)^* \times (N \cup \Sigma)^*$. With a type of grammar we associate a derivation relation $\Rightarrow \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$ that explains which words can be generated by the nonterminals of a grammar.

Definition 2.1 A (*parity*) *ω-grammar* is a quintuple (N, Σ, P, S, p) where (N, Σ, P, S) is an ordinary grammar with $p : N \to \mathbb{N}$ a function that assigns a natural number to each nonterminal. p determines which nonterminals can be used infinitely often in the derivation of a word, namely those X for which there is a nonterminal Y which occurs infinitely often and which has a smaller or equal even parity index. A formal definition of the derivation relation for a parity ω-grammar can be found below. Note that an ordinary grammar can be seen as an ω-grammar with $p(X) = 1$ for all $X \in N$. W.l.o.g. we assume $X \neq Y$ implies $p(X) \neq p(Y)$.

A *context-free grammar* (CFG) has production rules of the form $P \subset N \times (N \cup \Sigma)^*$. A grammar is called *context-sensitive* (CSG) if for every $(w, v) \in P$: either $|w| \leq |v|$ or $w = S$ and $v = \epsilon$.

A $(N, \Sigma, P, S, p, \lambda)$ is an *alternating context-free ω-grammar* (ω-ACFG) if (N, Σ, P, S, p) is a context-free ω-grammar in the above sense and $\lambda : N \to \{\exists, \forall\}$ marks each nonterminal as either existential or universal. Again, an ordinary ω-grammar can be simulated with $\lambda(X) = \exists$ for all $X \in N$. We will write

$$X \to v_1 \mid \ldots \mid v_k \qquad \text{if } \lambda(X) = \exists \text{ and } (X, v_i) \in P \text{ for all } i = 1, \ldots, k$$

$$X \to v_1 \& \ldots \& v_k \qquad \text{if } \lambda(X) = \forall \text{ and } (X, v_i) \in P \text{ for all } i = 1, \ldots, k$$

Definition 2.2 The *derivation* for an ω-ACFG $(N, \Sigma, P, S, p, \lambda)$ is a (possibly infinite tree) $\mathcal{T}$ with node set $\mathcal{N}$. Node $n \in \mathcal{N}$ are labelled with sentential forms $v \in (\Sigma \cup N)^*$. For $n \in \mathcal{N}$ let $l(n)$ be its label. The following must hold.

- if n is the root of $\mathcal{T}$ then $l(n) = S$.
- if n is a leaf of $\mathcal{T}$ then $l(n) = a$ for some $a \in \Sigma \cup \{\epsilon\}$.
- for each node n of $\mathcal{T}$ with $l(n) = X$ for some $X \in N$:
  - if $\lambda(X) = \exists$ then there is exactly one son n' of n and $l(n') = v$ with $(X, v) \in P$.
  - if $\lambda(X) = \forall$ and $X \to v_1 \& \ldots \& v_k$ are all rules for X then n has sons $n_1, \ldots, n_k$ and $l(n_i) = v_i$ for $i = 1, \ldots, k$.
- for each other node n of $\mathcal{T}$: if $l(n) = v = x_1 \ldots x_k$ where $x_i \in \Sigma \cup N$ then n has sons $n_1, \ldots, n_k$ and $l(n_i) = x_i$ for $i = 1, \ldots, k$.
- For an infinite path $\pi = n_0 n_1 \ldots$ let

$$\mathit{inf}(\pi) := \{X \mid \text{there are infinitely many } i \in \mathbb{N} \text{ s.t. } l(n_i) = X\}$$

For each path π in $\mathcal{T}$ we require $\mathit{min}\{p(X) \mid X \in \mathit{inf}(\pi)\}$ to be even.

Definition 2.3 A *matching* θ between a word $w \in \Sigma^\infty$ and a derivation tree $\mathcal{T}$ with node set $\mathcal{N}$ is a mapping of type $\theta : \mathcal{N} \to \mathit{Sub}(w)$, defined as follows.

- if n is the root of $\mathcal{T}$ then $\theta(n) = w$.
- for each node $n \in \mathcal{N}$ with $l(n) = X$ for some $X \in N$ and $\theta(n) = w'$:

$\theta(n') = w'$ for each son v of X.

- for each other node $n \in \mathcal{N}$ with $l(n) = v \in (\Sigma \cup N)^*$: if $\theta(n) = w'$ and n has sons $n_1, \ldots, n_k$ and $w' = w_1 \ldots w_k$ then $\theta(n_i) = w_i$ for each $i = 1, \ldots, k$.

A matching θ of a word $w \in \Sigma^*$ and a finite tree $\mathcal{T}$ is *successful* if for every leaf $n \in \mathcal{N}$: $\theta(n) = l(n)$.

To denote that θ is a successful matching between w and $\mathcal{T}$ we write $\theta(\mathcal{T}) = w$.

To call a matching between an infinite word and an infinite tree successful we need another definition.

Definition 2.4 Let $\mathcal{T}$ be an infinite tree with node set $\mathcal{N}$ and $n \in \mathcal{N}$. For each $n' \in \mathcal{N}$ with sons $n_1, \ldots, n_k$ we define

$$\mathit{left}_n(n') = \{n'\} \cup \begin{cases} \emptyset & \text{if } n = n' \\ \bigcup_{i=1}^{m} \mathit{left}_n(n_i) & \text{if } n \in \mathit{left}_n(n_m) \end{cases}$$

Then, $\mathit{left}_{\mathcal{T},n} := \mathit{left}_n(n_0)$ is the set of nodes that occur *left* of n in $\mathcal{T}$ where n_0 is the root of $\mathcal{T}$.

Let $\mathcal{T}|_n$ be the subtree of $\mathcal{T}$ with node set $\mathit{left}_{\mathcal{T},n}$ and $l(n) := \epsilon$.

A matching θ between $w \in \Sigma^\omega$ and an infinite tree $\mathcal{T}$ with node set $\mathcal{N}$ is successful if for every node $n \in \mathcal{N}$ with $\mathit{left}_{\mathcal{T},n} < \infty$ there is a finite prefix v of w and $\theta(\mathcal{T}|_n) = v$. Again, we write $\theta(\mathcal{T}) = w$ to express that θ is a successful matching between the infinite word w and the infinite tree $\mathcal{T}$.

Without this, infinite derivations without leaves would be successfully labelled by any infinite word. However, the corresponding grammar should generate the empty language.

Definition 2.5 Let G be an ω-ACFG. G generates the language $L(G)$ if there is a derivation tree $\mathcal{T}$ for G with node set $\mathcal{N}$ and

$$L(G) := \{w \in \Sigma^\infty \mid \exists \theta : \mathcal{N} \to \mathit{Sub}(w), \text{ s.t. } \theta(\mathcal{T}) = w\}$$

Given a fixed grammar G one can regard the languages $L(X)$ generated by a nonterminal X by replacing the start symbol S with X in the definition of G.

Example 2.6 The ω-grammar $G = (\{S, B\}, \{a, b\}, P, S, p, \lambda)$ with

$$P = \{\ S \to aBS,\ \ B \to b \mid aBB\ \}$$

and $p(S) = 0$, $p(B) = 2$, and $\lambda(A) = \lambda(B) = \exists$ generates the language

$$L(G) = \{w \in \Sigma^\omega \mid \forall v \in \Sigma^* : \text{if } w = v \ldots \text{ then } |v|_b \leq |v|_a\ \}$$

The labelled derivation tree of the word $w = (aabb)^\omega \in L(G)$ is sketched in Figure 1. We write the the matching and the labelling of a node n as $\theta(n) : l(n)$.

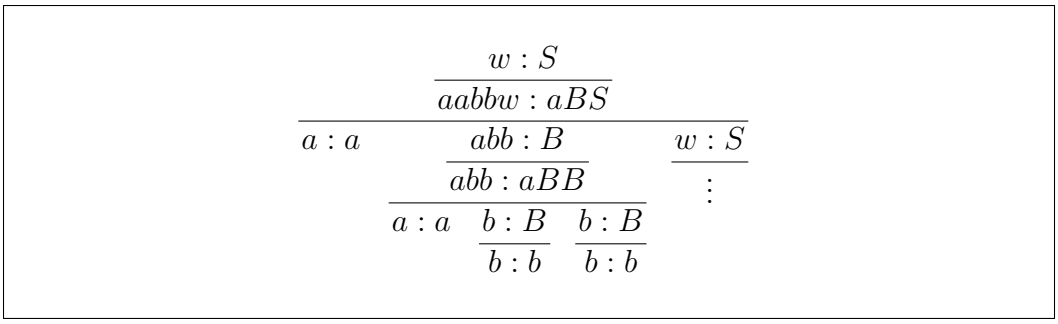

Fig. 1. The labelled derivation tree for Example 2.6.

CFL, ω-CFL, and ω-ACFL are the classes of all languages generated by context-free grammars, context-free ω-grammars and alternating context-free ω-grammars.

An ω-ACFG is in *Chomsky Normal Form* (CNF) if all productions are of the form $P \subset N \times (\Sigma \cup \{\epsilon\} \cup N^2)$. It is easy to see that the construction which transforms a context-free grammar into CNF is applicable to ω-ACFGs as well.

Lemma 2.7 *For every ω-ACFG G there is an ω-ACFG G' in CNF with $L(G') = L(G)$.*

Proof. We appeal to the same construction as it is used for context-free grammars, [6]. A production of the form $X \to x_1 \dots x_k$, where $x_i \in N \cup \Sigma$ is translated into productions $X \to X_1 X_2'$, $X_{k-1}' \to X_{k-1} X_k$, $X_{i-1}' \to X_i X_i'$ for $i = 3, \dots, k-1$, and $X_i \to x_i$ for $i = 1, \dots, k$. The parity indices for the new nonterminals X_i and X_i', $i = 1, \dots, k$ are the same as the one for X. Then, visiting an X_i infinitely often is possible iff it is possible to visit X infinitely often. To obtain a grammar in which parity indices are unique one can increase them at the end, s.t. $p(X_i) = p(X) + 2 * n$ for some $n \in \mathbb{N}$. Parity indices of other nonterminals must be increased by at least $2 * n + 2$.

Finally, for each new nonterminal there is exactly one production. Therefore they can be characterised as either existential or universal. □

Definition 2.8 An *alternating pushdown automaton with a parity condition* (ω-APDA) is a tuple $\mathfrak{A} = (Q, \Sigma, \Gamma, q_0, S_0, \delta, p, \lambda)$ where

- Q is a finite set of *states*,
- Σ is the finite *input alphabet*,
- Γ is a finite alphabet of *stack symbols*,
- $q_0 \in Q$ is the *initial state*,
- $S_0 \in \Gamma$ is the stack's *start symbol*,
- $p : Q \to \mathbb{N}$ is a *parity* function as above,
- $\lambda : Q \to \{\exists, \forall\}$ marks states as existential or universal as above, and
- $\delta : (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \to 2^{(Q \times \Gamma^*)}$ is the *transition relation*.

A *configuration* is a triple $(q, w, \gamma) \in Q \times \Sigma^\infty \times \Gamma^*$. The *computation* of $\mathfrak{A}$ on a $w \in \Sigma^\infty$ is a tree of configurations with the root (q_0, w, S_0). $(q', w, \gamma'\gamma)$ is a successor of $(q, aw, S\gamma)$ if $(q', \gamma') \in \delta(q, a, S)$, and (q', w, γ) is a successor of $(q, w, S\gamma)$ if $(q', \gamma') \in \delta(q, \epsilon, S)$.

If $\lambda(q) = \exists$ then every configuration (q, w, γ) has exactly one son in the computation which is a successor of (q, w, γ). If $\lambda(q) = \forall$ then all successors of (q, w, γ) are included as sons in the computation.

$\mathfrak{A}$ accepts w if there is a computation tree s.t.

- every leaf is of the form (q, ϵ, ϵ) for some q, and
- the smallest parity index of a state that occurs infinitely often on an infinite path is even.

$L(\mathfrak{A})$ is the set of all words accepted by $\mathfrak{A}$. ω-APDA will also be called the class of all languages accepted by alternating pushdown automata.

3 Linear Time μ-Calculus with Sequential Composition

Let *Var* a set of propositional variables. Formulas of LFLC are interpreted over finite or infinite words $w \in \Sigma^\infty$. The syntax is close to those of other μ-calculi like modal μ-calculus or linear time μ-calculus μ-LIN, [15].

$$\varphi \quad ::= \quad \mathtt{tt} \mid \mathtt{ff} \mid a \mid Z \mid \epsilon \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu Z.\varphi \mid \nu Z.\varphi \mid \varphi;\varphi$$

where Z ranges over *Var*, and a over Σ.[2] A formula is *closed* if it has no free variables, i.e. every Z occurs in the scope of a quantifier $\sigma \in \{\mu, \nu\}$. It is *well-named* if variable names are unique, i.e. no formula can contain a subformula $\sigma Z.\psi$ at two different positions. Thus, given a formula φ there is a function $fp_\varphi : Var \to \text{LFLC}$ that maps every variable to its unique fixed point formula. If φ is clear from the context we simply write fp.

A variable Y depends on Z in φ if Z occurs free in $fp_\varphi(Y)$. A variable is called *outermost* in φ if it does not depend on any other variable. $\varphi[\psi/Z]$ denotes φ with every occurrence of Z in it replaced by ψ. We implicitly assume that the resulting formula is well-named.

The set of subformulas $Sub(\varphi)$ of an LFLC formula φ is defined as follows.

$$\begin{aligned}
Sub(\varphi) &= \{\varphi\} \quad \text{if } \varphi \in \{\mathtt{tt}, \mathtt{ff}, \epsilon\} \cup \Sigma \cup Var \\
Sub(\varphi_0 \vee \varphi_1) &= \{\varphi_0 \vee \varphi_1\} \cup Sub(\varphi_0) \cup Sub(\varphi_1) \\
Sub(\varphi_0 \wedge \varphi_1) &= \{\varphi_0 \vee \varphi_1\} \cup Sub(\varphi_0) \cup Sub(\varphi_1) \\
Sub(\varphi_0;\varphi_1) &= \{\varphi_0;\varphi_1\} \cup Sub(\varphi_0) \cup Sub(\varphi_1) \\
Sub(\mu Z.\varphi) &= \{\mu Z.\varphi\} \cup Sub(\varphi) \\
Sub(\nu Z.\varphi) &= \{\nu Z.\varphi\} \cup Sub(\varphi)
\end{aligned}$$

[2] FLC has a construct `term` in [12] which is called τ in [10]. Here, we use the symbol ϵ to keep in line with formal language theory.

The semantics of a formula is inductively defined using an environment $\rho : Var \to 2^{\Sigma^\infty}$. With $\rho[Z \mapsto V]$ we denote the function that maps Z to V and agrees with ρ on all other arguments.

$$\begin{array}{rclcrcl}
[\![\mathtt{tt}]\!]_\rho & = & \Sigma^\infty & & [\![\mathtt{ff}]\!]_\rho & = & \emptyset \\
[\![a]\!]_\rho & = & \{a\} & & [\![Z]\!]_\rho & = & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & = & [\![\varphi]\!]_\rho \cup [\![\psi]\!]_\rho & & [\![\epsilon]\!]_\rho & = & \{\epsilon\} \\
[\![\varphi \wedge \psi]\!]_\rho & = & [\![\varphi]\!]_\rho \cap [\![\psi]\!]_\rho & & & & \\
[\![\varphi ; \psi]\!]_\rho & = & \{v \mid \exists v_1 \in [\![\varphi]\!]_\rho, \exists v_2 \in [\![\psi]\!]_\rho \text{ s.t. } v = v_1 v_2\} & & & & \\
[\![\mu Z.\varphi]\!]_\rho & = & \bigcap \{V \subseteq \Sigma^\infty \mid [\![\varphi]\!]_{\rho[Z \mapsto V]} \subseteq V\} & & & & \\
[\![\nu Z.\varphi]\!]_\rho & = & \bigcup \{V \subseteq \Sigma^\infty \mid V \subseteq [\![\varphi]\!]_{\rho[Z \mapsto V]}\} & & & &
\end{array}$$

A formula $\varphi(Z)$ with a free variable Z defines a function $[\![\varphi(Z)]\!]_\rho : 2^{\Sigma^\infty} \to 2^{\Sigma^\infty}$ if $\rho(Z) = \mathtt{undef}$. It is monotone in the sense that $V_1 \subseteq V_2$ implies $[\![\varphi(Z)]\!]_{[Z \mapsto V_1]} \subseteq [\![\varphi(Z)]\!]_{[Z \mapsto V_2]}$. Hence, according to [16] fixed points of these functions exist.

We write $w \models_\rho \varphi$ iff $w \in [\![\varphi]\!]_\rho$. If φ is closed we can drop ρ and write $w \models \varphi$.

Two formulas φ and ψ are *equivalent* in LFLC, written $\varphi \equiv \psi$, iff $\forall \rho$: $[\![\varphi]\!]_\rho = [\![\psi]\!]_\rho$. The language of a closed formula for a given alphabet Σ is the set of all its models, $L(\varphi) := [\![\varphi]\!]_\rho$. Overloading notation we let LFLC also denote the class of all languages definable by an LFLC formula, i.e. $\{L(\varphi) \mid \varphi \in \text{ LFLC }\}$. It will be easy to derive from the context whether the set of formulas or the class of languages is meant.

For correctness proofs we need to introduce *approximants* of fixed point formulas. Let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ with λ being a limit ordinal. For a $\nu Z.\varphi$ we define $Z^0 := \mathtt{tt}$, $Z^{\alpha+1} := \varphi[Z^\alpha/Z]$ and $Z^\lambda := \bigwedge_{\alpha<\lambda} Z^\alpha$. Dually, for a $\mu Y.\varphi$: $Y^0 := \mathtt{ff}$, $Y^{\alpha+1} := \varphi[Y^\alpha/Y]$ and $Y^\lambda := \bigvee_{\alpha<\lambda} Y^\alpha$. The next lemma is a standard result about approximants.

Lemma 3.1 *$w \models \mu Y.\varphi$ iff $\exists \alpha \in \mathbb{O}\mathrm{rd}$ s.t. $w \models Y^\alpha$. $w \not\models \nu Z.\varphi$ iff $\exists \alpha \in \mathbb{O}\mathrm{rd}$ s.t. $w \not\models Z^\alpha$. In both cases α is not a limit ordinal.*

Example 3.2 Let $\Sigma = \{a, b\}$. LFLC can define $L = \{w \in \Sigma^\omega \mid \forall v \in \Sigma^* :$ if $w = v \ldots$ then $|v|_b \leq |v|_a$ $\}$, compare with Example 2.6. L consists of all infinite words for which the number of bs never exceeds the number of a's in every prefix. The formula for L is

$$\varphi := \nu Y.a; (\nu Z.b \vee a; Z; Z); Y$$

φ is best understood by unfolding fixed points, i.e. following a word w through φ's syntax tree and moving to the corresponding fixed point formula whenever a variable is reached. φ says that the word must begin with an a. This is

$$\frac{v \vdash \varphi_0 \vee \varphi_1}{v \vdash \varphi_i}\ i \in \{0,1\} \qquad \frac{v \vdash \varphi_0 \wedge \varphi_1}{v \vdash \varphi_0 \qquad v \vdash \varphi_1} \qquad \frac{v \vdash \sigma Z.\varphi}{v \vdash Z}$$

$$\frac{v \vdash Z}{v \vdash \varphi} \quad \text{if } fp(Z) = \sigma Z.\varphi \qquad \frac{v \vdash \varphi_0;\varphi_1}{v_0 \vdash \varphi_0 \qquad v_1 \vdash \varphi_1}\ v = v_0 v_1$$

Fig. 2. The model checking tableau rules.

necessary since $\Sigma = \{a, b\}$ and a word beginning with a b would violate the property described above.

The following suffix can be a single b or another a in which case it is followed by two words of the same kind. By unfolding each time an a is visited a Z gets replaced with two Zs, i.e. the number of Zs still present in the unfolded formula counts the number of bs that are still possible without exceeding the number of as so far. Consequently, every time a b is seen a Z has simply been deleted from the unfolded formula.

Finally, if all Zs are "used", i.e. the prefix so far contained as many bs as as then this can only be suffixed by another word of the language. Hence the Y at the end which leads to the beginning of the formula again.

4 Model checking LFLC

An LFLC model checking tableau for a word w and a closed formula φ is a tree of configurations of $\Sigma^\infty \times Sub(\varphi)$. A configuration is written $v \vdash \psi$ and its intended meaning is simply $v \models \psi$. The rules are presented in Figure 2. Each tableau for w and φ has the root $w \vdash \varphi$. A branch of a tableau is successful if it ends on a leaf $a \vdash a$ with $a \in \Sigma \cup \{\epsilon\}$ or $v \vdash \mathtt{tt}$, or if it is infinite and the outermost variable that occurs infinitely often on this branch is of type ν. A tableau is successful if all its branches are successful. If there is a successful tableau for w and φ then we simply write $w \vdash \varphi$.

Theorem 4.1 *$w \models \varphi$ iff $w \vdash \varphi$.*

Proof. Suppose $w \vdash \varphi$ but $w \not\models \varphi$. In this case the root of the tableau is called *false*. The rules are backwards sound, i.e. falsity can be pushed through the tableau along one branch. This cannot be a finite branch since successful branches end in true leaves only.

Suppose therefore it is an infinite one. This must contain a false node $v \vdash \nu Z.\psi$. By Lemma 3.1, Z can be interpreted as the least approximant Z^α s.t. $v \not\models Z^\alpha$. Pushing falsity further down this branch creates a contradiction since, by well-foundedness of the ordinals, it must eventually reach a node $v' \vdash Z^0$ which cannot be false since $Z^0 \equiv \mathtt{tt}$.

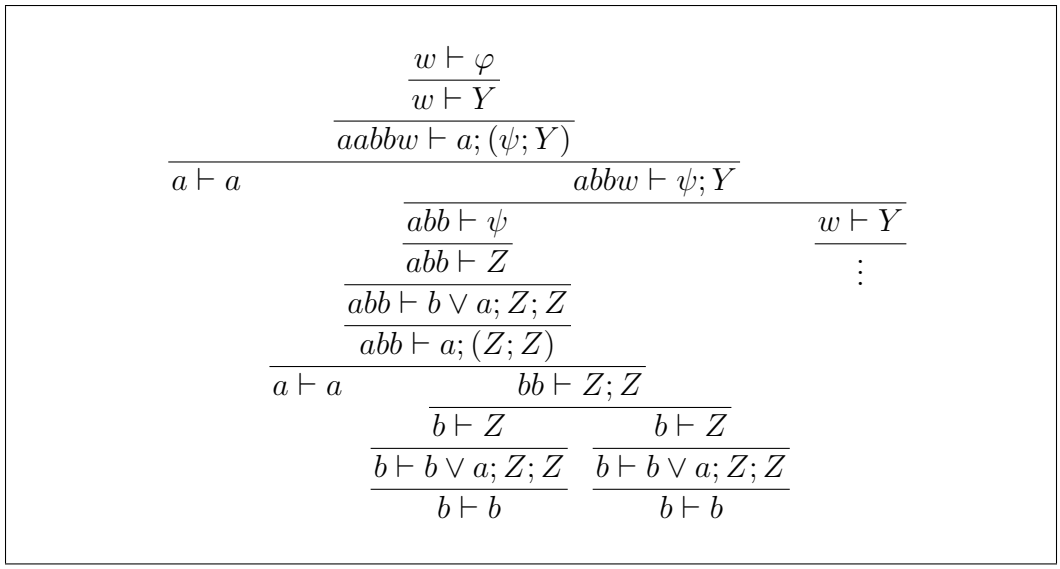

Fig. 3. The tableau for Example 4.2.

For the converse implication suppose $w \models \varphi$. We build a successful tableau for $w \vdash \varphi$. The root is called *true* since the intended semantical relation holds. The tableau rules can always be applied preserving truth. For example, if $v \models \psi_0 \vee \psi_1$ then $v \models \psi_i$ for some $i \in \{0, 1\}$ and, thus, the tableau construction can continue with the true node $v \vdash \psi_i$. In case of a conjunction or a sequential composition the parameters can be chosen s.t. both successor are true. Unfolding of fixed points and variables obviously preserves truth.

This way, every finite branch will be successful since they end in true leaves only. Suppose there is an infinite branch on which the outermost variable Y that occurs infinitely often is of type μ. Then the first occurrence of Y in a $v \vdash Y$ is interpreted with the least α s.t. $v \models Y^\alpha$. But then following the branch will eventually result in a true configuration which is interpreted as $v' \vdash Y^0$ which is a contradiction since $Y^0 \equiv \mathtt{ff}$ and the configuration cannot be true. Therefore, every outermost variable occurring infinitely often must be of type ν. Consequently, the tableau must be successful, i.e. $w \vdash \varphi$. □

Example 4.2 Take the formula $\varphi := \nu Y.a; (\nu Z.b \vee a; Z; Z); Y$ from Example 3.2. A successful tableau for φ and the ω-word $w = (aabb)^\omega$ is sketched in Figure 3. Note that $w = aabbw$. We use the abbreviation $\psi := \nu Z.b \vee a; Z; Z$.

5 LFLC and ω-ACFL

Theorem 5.1 *ω-ACFL $\subseteq$ LFLC.*

Proof. Let $L \in \omega$-ACFL. Then there exists an ω-ACFG $G = (N, \Sigma, P, S, p, \lambda)$ in CNF s.t. $L = L(G)$. We will associate with each nonterminal X an eponymous propositional variable X. Note that they are linearly ordered by their parity indices $p(X)$.

We process the nonterminals in decreasing order of parity index starting with the greatest. Let $X \to w_1 \mid \ldots \mid w_m$ be all productions for X with $\lambda(X) = \exists$. This is then translated to $\varphi_X = \sigma X.\psi_1 \vee \psi_2 \vee \ldots \vee \psi_m$, where $\sigma = \mu$ if $p(X)$ is odd, $\sigma = \nu$ if $p(X)$ is even, and for each $i = 1, \ldots, m$

- $\psi_i = w_i$ if $w_i = a$ for any $a \in \Sigma$ or $w_i = \epsilon$.
- $\psi_i = \psi_1; \psi_2$ if $w_i = Y_1 Y_2$, where for $j \in \{1, 2\}$:
  - $\psi_j = \varphi_{Y_j}$ if Y_j has been processed already,
  - $\psi_j = Y_j$ otherwise.

In the same way $X \to w_1 \,\&\, \ldots \,\&\, w_k$ is translated into $\varphi_X = \sigma X.\psi_1 \wedge \ldots \wedge \psi_k$. This way, an acyclic system of equations for formulas ψ_X is obtained which can be made into a single formula φ_S by substituting right sides of defining equations for their corresponding formulas in other equations. Note that the resulting formulas can be simplified using $\sigma X.\varphi \equiv \varphi$ if $X \notin Sub(\varphi)$.

Define $\varphi_G := \varphi_S$. It might contain free variables in case S did not have the least parity index. Suppose X is free in φ_S. Then replace the latter with $\varphi_S[\varphi_X/X]$. Note that φ_X cannot contain S as a free variable since $p(X) < p(S)$. Variables can be renamed to preserve well-naming of formulas. Continue this process until the new φ_G does not contain any free variables anymore. Termination is guaranteed since the parity indices of new free variables are always decreased. Note that this process does not change the language accepted by φ_G because of $\sigma Z.\varphi \equiv \varphi[\sigma Z.\varphi/Z]$.

To show that $L(\varphi_G) = L(G)$ we consider the derivation tree $\mathcal{T}$ of G and a word $w \in L(G)$. There is obviously a successful matching between w and $\mathcal{T}$. This can be made into a tableau for w and φ_G. Let n be a node in the derivation tree with matching $\theta(n)$ and labelling $l(n) = x_1 \ldots x_k \in (N \cup \Sigma)^*$. This corresponds to a node in the tableau of the form

$$\theta(n) \vdash \varphi_{x_1}; \ldots; \varphi_{x_k}$$

where $\varphi_{x_i} = a$ if $x_i = a$ for some $a \in \Sigma$ and $i \in \{1, \ldots, k\}$. The overall structure of the tableau is the same as the one of the derivation tree. The point where a 1-1 correspondence between nodes of the derivation and nodes of the tableau breaks down is the first occurrence of each nonterminal as a variable in the tableau. Suppose $X \in N$ is matched with $\theta(n)$ at some node n with $l(n) = X$ in $\mathcal{T}$. The translation above introduces a fixed point quantifier for X. Thus, the tableau will have an additional node $\theta(n) \vdash \sigma X. \ldots$ whose only son is $\theta(n) \vdash X$.

Remember that for each path π in $\mathcal{T}$ the minimal parity index $p(X)$ that occurs infinitely often for an $X \in inf(\pi)$ is even. By the translation above the corresponding variable X in the tableau is of type ν. It remains to show that, if there is an odd parity index $p(Y)$ for a nonterminal $Y \in inf(\pi)$ then the variable Y depends on X. Clearly, $p(Y) > p(X)$. Hence, Y has been processed before X. If there is a $(Y, vXw) \in P$ for some $v, w \in (N \cup \Sigma)^*$ then X will occur as a free variable in φ_Y. On the other hand, when X gets

processed, φ_Y exists already and if there is a rule $(X, vYw) \in P$ for some v, w then φ_Y will be plugged into φ_X. Thus, Y will depend on X in φ_G. If there are no such rules then the variables are incomparable and it is impossible for both to occur infinitely often on a tableau branch. Equally, they cannot occur both infinitely often in the derivation tree.

We conclude that there is a successful tableau for $w \vdash \varphi_G$. By Theorem 4.1, $w \in L(\varphi_G)$.

Suppose on the other hand that $w \in L(\varphi_G)$. By Theorem 4.1 $w \vdash \varphi_G$. Again, the tableau can be seen as a representation of a successful matching between w and a derivation tree $\mathcal{T}$ for G. Take an infinite path in the tableau. The outermost variable will be of type ν. By the construction of φ_G it corresponds to a nonterminal X with $p(X)$ being even. Suppose there is another variable Y with an odd $p(Y)$ that occurs infinitely on a tableau branch. Then X is outermost among X and Y. By the construction of φ_G, Y must have been processed before X since $\mu Y. \ldots$ occurs inside φ_X. But this is only if $p(Y) > p(X)$ and thus the corresponding branch in the derivation tree is successful as well. Hence, $w \in L(G)$. □

Example 5.2 Translating G of Example 2.6 yields the formula

$$\varphi_G = \nu S.a; (\nu B.b \vee a; B; B); S$$

which the same as φ of Example 3.2 up to variable renaming. Note that in the proof of Theorem 5.1 the grammar G is assumed to be in CNF. Therefore, the derivation tree is binary as well as every tableau is. This is not the case for the derivation tree in Figure 1 and the corresponding tableau of Figure 3. But G of Example 2.6 is not in CNF.

Theorem 5.3 *LFLC* $\subseteq$ *ω-ACFL.*

Proof. Let $\varphi_0 \in$ LFLC. We construct an ω-ACFG $G = (N, \Sigma, P, S, p, \lambda)$ s.t. $L(G) = L(\varphi_0)$. For each subformula of φ_0 we use a nonterminal, $N := \{X_\psi \mid \psi \in \mathit{Sub}(\varphi_0)\}$. φ_0 itself serves as the starting symbol, $S := X_{\varphi_0}$. Productions are given by the following rules. For every $\psi \in \mathit{Sub}(\varphi_0)$:

$$\begin{array}{lcl}
\psi = a & \leadsto & X_a \to a \\
\psi = \epsilon & \leadsto & X_\epsilon \to \epsilon \\
\psi = \mathtt{tt} & \leadsto & X_{\mathtt{tt}} \to \epsilon \mid a_1 X_{\mathtt{tt}} \mid \ldots \mid a_k X_{\mathtt{tt}} \quad \text{if } \Sigma = \{a_1, \ldots, a_k\} \\
\psi = \mathtt{ff} & \leadsto & X_{\mathtt{ff}} \to a \,\&\, b \quad \text{for some } a \neq b \\
\psi = Z & \leadsto & X_Z \to X_\varphi \quad \text{if } \mathit{fp}(Z) = \sigma Z.\varphi \\
\psi = \psi_0 \vee \psi_1 & \leadsto & X_{\psi_0 \vee \psi_1} \to X_{\psi_0} \mid X_{\psi_1} \\
\psi = \psi_0 \wedge \psi_1 & \leadsto & X_{\psi_0 \wedge \psi_1} \to X_{\psi_0} \,\&\, X_{\psi_1}
\end{array}$$

$$\psi = \sigma Z.\varphi \quad \leadsto \quad X_{\sigma Z.\varphi} \to X_Z$$
$$\psi = \psi_0;\psi_1 \quad \leadsto \quad X_{\psi_0;\psi_1} \to X_{\psi_0} X_{\psi_1}$$

The categorisation of nonterminals is given by $\lambda(X_\psi) = \forall$ iff $\psi = \psi_0 \wedge \psi_1$ for some ψ_i or $\psi = \mathtt{ff}$. What remains is the definition of the parity indices p. This is simply done by enumerating all subformulas in a depth-first search of the syntax tree of φ_0. Formally, each subformula ψ will receive a number $n(\psi)$ s.t. for all $\varphi, \psi \in \mathit{Sub}(\varphi_0)$:

- $\varphi \in \mathit{Sub}(\psi)$ implies $n(\varphi) > n(\psi)$
- $\varphi = \nu Z.\psi$ implies $n(\varphi)$ is even.
- $\varphi = \mu Z.\psi$ implies $n(\varphi)$ is odd.

Then, the parity index of a nonterminal is easily given by $p(X_\psi) := n(\psi)$.

To show that $L(G) = L(\varphi_0)$ we use the observation from the proof of Theorem 5.1 that a successful tableau for w and φ_0 is nothing else than a successfully labelled derivation of G with w. Again, the important point is the correspondence between infinite branches of the tableau and the derivation.

Suppose the tableau has an infinite branch. Then the outermost variable Z that occurs infinitely often is of type ν, i.e. in the derivation it corresponds to a nonterminal X_Z whose parity index $p(X_Z)$ is even. Suppose on the path where X_Z occurs infinitely often, another X occurs infinitely often as well. Then $X = X_\psi$ for some $\psi \in \mathit{Sub}(\mathit{fp}(Z))$ for otherwise Z would not be outermost on this particular branch. But then the enumeration of subformulas assigned a greater value to ψ, i.e. $n(\psi) > n(Z)$. Thus, the corresponding branch in the derivation is successful as well. The other implication holds because of the same argument, too. □

6 Some Properties of LFLC and ω-ACFL

Theorem 6.1 *μ-LIN $\subsetneq$ LFLC.*

Proof. The syntax of μ-LIN is usually given as

$$\varphi \quad ::= \quad \mathtt{tt} \mid \mathtt{ff} \mid a \mid Z \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$$

where a ranges over Σ. The semantics is almost the same as it is for LFLC. The *next* operator $\bigcirc\varphi$ can be defined in LFLC as $(\bigvee_{a\in\Sigma} a);\varphi$. Note that the semantics of a proposition a in μ-LIN is usually defined as

$$[\![a]\!]_. = \{w \in \Sigma^\infty \mid \exists v \in \Sigma^\infty, w = av\}$$

while the semantics of a proposition a in LFLC is the word a only. But the above can be modelled in LFLC by $a;\mathtt{tt}$.

The translation is similar to the one from $\mathcal{L}_\mu$ into FLC given in [12].

It is well-known that μ-LIN can only describe regular languages. Therefore, LFLC can express strictly more than μ-LIN. □

Theorem 6.2 *LFLC and LFLC^k are undecidable for every $k \in \mathbb{N}$.*

Proof. This follows from the translation of context-free grammars into LFLC formulas given in Theorem 5.1. Let L_1 and L_2 be two context-free languages. $L_1 \cap L_2 \neq \emptyset$ iff $\varphi_{L_1} \wedge \varphi_{L_2}$ is satisfiable. But the intersection problem for context-free languages is known to be undecidable, [6]. The translation even yields a non-alternating formula. Note that only μ variables are needed for finite words. Therefore, LFLC^0 is already undecidable.

Note that in the same way undecidability of FLC is proved in [12] with a translation from context-free processes. □

Theorem 6.3 *ω-CFL $\subsetneq$ LFLC.*

Proof. This is based on an observation from [12] that $L_0 = \{ a^n b^n c^n \mid n \in \mathbb{N} \}$ is LFLC-definable. The corresponding formula is

$$\varphi_0 := \psi_{a,b}; \chi_c \wedge \chi_a; \psi_{b,c}$$

where $\psi_{a,b} := \mu X.\epsilon \vee a; X; b$ and $\chi_c := \mu X.\epsilon \vee c; X$. On the other hand, it is well-known that L_0 is not context-free, [6].

Now, let $\Sigma = \{a, b, c, d\}, L = \{a^n b^n c^n d^\omega \mid n \in \mathbb{N}\}$. Clearly, $L = L_0\{d^\omega\}$ and therefore $L = L(\varphi_0; (\nu Z.d; Z))$. Suppose now that L is ω-context-free. By [3], $L = \bigcup_{i=1}^{k} U_i V_i^\omega$ for some $k \in \mathbb{N}$ and some $U_i, V_i \in$ CFL with $i \in \{1, \ldots, k\}$. All $V_i = \{d^{k_i}\}$ for some $k_i \in \mathbb{N}$ because no a, b or c occurs infinitely often in a word of L. Therefore $U_i = \{a^n b^n c^n d^{m_{i,n}} \mid n \in \mathbb{N}\}$ for every i and some $m_{i,n}$.

Consider the mapping $s : a \mapsto \{a\}, b \mapsto \{b\}, c \mapsto \{c\}, d \mapsto \{\epsilon\}$. Clearly, s is a substitution with $s(x) \in$ CFL for every $x \in \Sigma$. Also, $s(U_i) \notin$ CFL for every U_i. Since CFL is closed under substitutions with context-free languages, [5], the U_i cannot be context-free. Thus, L cannot be ω-context-free. □

Theorem 6.4 *LFLC is not effectively closed under morphisms. There is no algorithm that, given a morphism $h : \Sigma_1 \to \Sigma_2^*$ and a $\varphi \in$ LFLC computes a $\varphi_h \in$ LFLC s.t. $L(\varphi_h) = h(L(\varphi))$.*

Proof. The word problem for type-0 grammars is known to be undecidable. Take $w \in \Sigma^*$ and a type-0 grammar G. Then there exist effectively computable deterministic context-free grammars G_1 and G_2 and a morphism h s.t. $w \in L(G)$ iff $w \in h(L(G_1) \cap L(G_2))$, [5]. According to Theorem 5.1 there is a φ s.t. $L(\varphi) = L(G_1) \cap L(G_2)$. An effectively computable ψ with $L(\psi) = h(L(\varphi))$ could be used to solve the word problem for G. □

Example 6.5 The formula φ_0 from the proof of Theorem 6.3 can also be used to hint the failure of closure under morphisms. Morphisms do not correspond to substitution on formulas, i.e. in general $h(L(\varphi)) \neq L(\varphi[h])$ where

$\varphi[h]$ is used to denote the formula that arises from φ by carrying out all the substitutions of h in φ simultaneously.

Take the morphism h with $h(a) = h(b) = h(c) = c$. Then

$$\begin{aligned} L(\varphi_0[h]) &= L((\psi_{a,b}; \chi_c \wedge \chi_a; \psi_{b,c})[h]) \\ &= L((\psi_{a,b}; \chi_c)[h] \wedge (\chi_a; \psi_{b,c})[h]) \\ &= L(\psi_{c,c}; \chi_c \wedge \chi_c; \psi_{c,c}) \\ &= \{\ c^m \mid m \geq 2n, n \in \mathbb{N}\ \} \cap \{\ c^m \mid m \geq 2n, n \in \mathbb{N}\ \} \\ &= \{\ c^m \mid m \geq 2n, n \in \mathbb{N}\ \} \end{aligned}$$

but $h(L(\varphi_0)) = \{\ c^{3n} \mid n \in \mathbb{N}\ \}$.

Theorem 6.6 *For finite words, model checking LFLC and the word problem for APDA are PTIME-complete.*

Proof. Note that the translations in the proofs of Theorem 5.1 and 5.3 are polynomial if resulting formulas are represented by sharing common subformulas.

An alternating algorithm can decide whether there is a successful tableau for a finite w and φ. The space needed for this is logarithmic in $|w|$ and $|\varphi|$. A configuration consists of a subword and a subformula of the input. A subword v of w can be represented as a pair (i, j) of natural numbers denoting the start and end of v in w. It is also possible to enumerate all subformulas of the input formula s.t. each of them can be given a natural number. Then, the size of a configuration $v \vdash \psi$ is at most $2 \cdot \log |w| + \log |\varphi|$ for an input w, φ. According to [1], the model checking problem can be decided in PTIME.

Note that tableaux for finite words cannot have infinite paths. Therefore, the condition regarding parity indices need not be checked.

PTIME-hardness follows from the fact that the word problem for CFGs is PTIME-complete. □

Corollary 6.7 *$\exists L \in$ CSL, s.t. $L \notin \omega$-ACFL, if PTIME $\neq$ PSPACE.*

Proof. The model checking problem for CSL is PSPACE-complete, and there are context-sensitive grammars for which the problem is PSPACE-complete even if the grammar is fixed. □

7 LFLC and ω-APDA

Theorem 7.1 *LFLC $\subseteq \omega$-APDA.*

Proof. Let $\varphi_0 \in$ LFLC. We construct an ω-APDA $\mathfrak{A}$ s.t. $L(\mathfrak{A}) = L(\varphi_0)$. Let $\mathfrak{A} = (Q, \Sigma, \Gamma, q_0, S_0, \delta, p, \lambda)$ where $Q = \Gamma := Sub(\varphi_0) \cup \{\mathtt{tt}\}$, $q_a = S_0 = \mathtt{tt}$, $q_0 := \varphi_0$. The transition relation is defined by case distinction on the state.

$$\begin{array}{llll}
\delta(Y,\epsilon,\psi) & \ni & (\varphi,\psi) \text{ if } fp(Y)=\mu Y.\varphi & \\
\delta(\mathtt{tt},a,\psi) & \ni & (\mathtt{tt},\psi) & \delta(\mathtt{ff},a,\psi) \ni (\mathtt{ff},\psi) \\
\delta(a,a,\psi) & \ni & (\psi,\epsilon) & \delta(\psi_0\vee\psi_1,\epsilon,\psi) \ni (\psi_i,\psi) \quad i\in\{0,1\} \\
\delta(\epsilon,\epsilon,\psi) & \ni & (\psi,\epsilon) & \delta(\psi_0\wedge\psi_1,\epsilon,\psi) \ni (\psi_i,\psi) \quad i\in\{0,1\} \\
\delta(\sigma Y.\varphi,\epsilon,\psi) & \ni & (Y,\psi) & \delta(\psi_0;\psi_1,\epsilon,\psi) \ni (\psi_0,\psi_1;\psi)
\end{array}$$

Note that the content of the stack is only changed in states a, ϵ or $\psi_0;\psi_1$.

Set $\lambda(\psi)=\forall$ iff $\psi=\psi_0\wedge\psi_1$ for some $\psi_0,\psi_1\in Sub(\varphi_0)$. The parity indices are found just as in the proof of Theorem 5.3 by enumerating subformulas. Additionally, we set $p(\mathtt{tt})=0$ and $p(\mathtt{ff})=1$.

Again, to show that $L(\mathfrak{A})=L(\varphi_0)$ we appeal to the fact that an accepting computation of $\mathfrak{A}$ for a word w is nothing else than a tableau for $w\vdash\varphi_0$. There are a few minor technical differences. For example the last configuration on a finite tableau path is $a\vdash a$ which corresponds to a configuration $(a,a,\mathtt{tt})$ in the computation. From then on $\mathfrak{A}$ is going into an infinite but accepting loop on $(\mathtt{tt},\epsilon,\epsilon)$. The definition of the parity indices ensures that successful infinite branches in the tableau correspond exactly to accepting infinite branches in the computation other than those trivial ones. □

8 Further work

It is not clear whether LFLC is closed under complementation because $\varphi;\psi$ quantifies existentially over a position inside a word. Its complement is universal which cannot simply be modelled using sequential composition on arbitrary words with finite formulas. It remains to see whether the complement on Σ^* only is expressible. Note that a negative answer would separate LFLC from the class of context-sensitive languages.

It is known that the alternation hierarchy inside linear time μ-calculus collapses to the first level where alternation of fixed point quantifiers is considered. This is seen using a translation between formulas and Büchi automata forth and back. The translation between LFLC and ω-ACFL does not obviously lead to a collapse. The complexity analysis of the model checking tableaux suggest that fixed point alternation is removable.

References

[1] Chandra, A. K., D. C. Kozen and L. J. Stockmeyer, *Alternation*, Journal of the ACM **28** (1981), pp. 114–133.

[2] Chomsky, N., *Context-free grammars and pushdown storage*, Quarterly Progress Report 65, Research Laboratory of Electronics, Mass (1962).

[3] Cohen, R. S. and A. Y. Gold, *Theory of ω-languages: I: Characterization of ω-context-free languages; II: A study of various models of ω-type generation and recognition*, Journal of Computer and System Sciences **15** (1977), pp. 169–184; 185–208.

[4] Emerson, E. A., C. S. Jutla and A. P. Sistla, *On model checking for the μ-calculus and its fragments*, Theoretical Computer Science **258** (2001), pp. 491–522.

[5] Harrison, M. A., "Introduction to Formal Language Theory," Addison-Wesley, Reading, 1978, 1 edition.

[6] Hopcroft, J. and J. Ullman, "Introduction to Automata Theory, Languages, and Computation," Addison-Wesley, N. Reading, MA, 1980.

[7] Ibarra, O. H., T. Jiang and H. Wang, *A characterization of exponential-time languages by alternating context-free grammars*, TCS **99** (1992), pp. 301–315.

[8] Janin, D. and I. Walukiewicz, *On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic*, in: U. Montanari and V. Sassone, editors, *CONCUR '96: Concurrency Theory, 7th Int. Conf.*, LNCS **1119** (1996), pp. 263–277.

[9] Kozen, D., *Results on the propositional μ-calculus*, TCS **27** (1983), pp. 333–354.

[10] Lange, M. and C. Stirling, *Model checking fixed point logic with chop*, in: M. Nielsen and U. H. Engberg, editors, *Proc. Foundations of Software Science and Computation Structures, FOSSACS'02*, LNCS **2303** (2002), pp. 250–263. URL `http://www.dcs.ed.ac.uk/~martin/papers/fossacs23.pdf`

[11] Moriya, E., *A grammatical characterization of alternating pushdown automata*, TCS **67** (1989), pp. 75–85.

[12] Müller-Olm, M., *A modal fixpoint logic with chop*, in: C. Meinel and S. Tison, editors, *Proc. 16th Annual Symp. on Theoretical Aspects of Computer Science, STACS'99*, LNCS **1563** (1999), pp. 510–520.

[13] Reinhardt, K., *Hierarchies over the context-free languages*, in: *YOUNGCS: 6th International Meeting of Young Computer Scientists (Trends, Techniques, and Problems in Theoretical Computer Science; Machines, Languages, and Complexity; Aspects and Prospects of Theoretical Computer Science)*, 1990.

[14] Rozenberg, G. and A. Salomaa, editors, "Handbook of Formal Languages," Springer Verlag, Berlin, Heidelberg, New York., 1996.

[15] Stirling, C., *Modal and temporal logics*, , **2 (Background: Computational Structures)**, Clarendon Press, Oxford, 1992 pp. 477–563.

[16] Tarski, A., *A lattice-theoretical fixpoint theorem and its application*, Pacific J.Math. **5** (1955), pp. 285–309.