A Decidable Non-Regular Modal Fixpoint Logic

Florian Bruse
School of Electrical Engineering and Computer Science, Universität Kassel, Germany

Martin Lange
School of Electrical Engineering and Computer Science, Universität Kassel, Germany

Abstract

Fixpoint Logic with Chop (FLC) extends the modal μ-calculus with an operator for sequential composition between predicate transformers. This makes it an expressive modal fixpoint logic which is capable of formalising many non-regular program properties. Its satisfiability problem is highly undecidable. Here we define Visibly Pushdown Fixpoint Logic with Chop, a fragment in which fixpoint formulas are required to be of a certain form resembling visibly pushdown grammars. We give a sound and complete game-theoretic characterisation of FLC's satisfiability problem and show that the games corresponding to formulas from this fragment are stair-parity games and therefore effectively solvable, resulting in 2EXPTIME-completeness of this fragment. The lower bound is inherited from PDL over Recursive Programs, which is structurally similar but considerably weaker in expressive power.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Program specifications

Keywords and phrases formal specification, temporal logic, expressive power

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2021.23

1 Introduction

Modal fixpoint logics form the backbone of many program specification languages. The most prominent example is the modal μ-calculus $\mathcal{L}_\mu$ [15] which extends modal logic with least and greatest fixpoint quantifiers in order to express limit behaviour. Modal logic as a basis ensures bisimulation invariance which is a desirable property for program logics.

The two main decision problems associated with program logics are model checking and satisfiability checking. Whilst model checking for logics like $\mathcal{L}_\mu$ is easily seen to be decidable using fixpoint iteration for instance, satisfiability checking is computationally, combinatorically and conceptually more difficult. A relatively easy proof of the decidability of $\mathcal{L}_\mu$'s satisfiability problem is via a translation into Monadic Second-Order Logic (MSO) on trees using bisimulation-invariance [16]. MSO is decidable over infinite trees, but this only gives a non-elementary upper bound. Over time, this has been improved [26] until the problem could finally be placed in the complexity class EXPTIME using a translation into alternating tree automata [6].

The semantics of such automata can be phrased in terms of two-player games, and this can also be done directly to formulas. Hence, there are also game-theoretic characterisations of $\mathcal{L}_\mu$'s satisfiability problem as finite Rabin or parity games, which gives alternative decidability results [24, 9].

The EXPTIME upper bound is optimal; a matching lower bound is inherited from Propositional Dynamic Logic (PDL) – a modal logic that can be translated linearly into $\mathcal{L}_\mu$ and whose satisfiability problem is EXPTIME-complete [7]. The translation shows that PDL is in fact also a modal fixpoint logic with the fixpoints being implicitly present in the definition of regular languages used in modalities as in $\langle a^*b^* \rangle p$ for instance.


© Florian Bruse and Martin Lange;
licensed under Creative Commons License CC-BY 4.0

Regularity is also the answer to the question after $\mathcal{L}_\mu$'s expressiveness. The translation into MSO on trees shows that it can only express properties which are regular in the sense that they can be recognised by a finite automaton. $\mathcal{L}_\mu$ is in fact the largest such program logic as it is equi-expressive to the bisimulation-invariant fragment of MSO [12].

There is obviously a link between the decidability of satisfiability and the restriction of the expressiveness to regularity only. However, regularity on the expressiveness side does not constitute the border between decidability and undecidability, as there are program logics with the capability of expressing non-regular properties whose satisfiability problem remains decidable. The natural extension PDL[CFL] of PDL to *all* context-free (rather than just regular) programs is undecidable [10] but early on it has been observed that its extension by *some* context-free languages like $L_1 := \{a^n b^n \mid n \geq 1\}$ leads to a decidable program logic [14, 11].

Some such extension, for instance with the similar CFL $L_2 := \{a^n b a^n \mid n \geq 1\}$ lead to undecidability, though. This may seem odd at first sight but it makes perfect sense in the light of a later finding, namely that the extension of PDL with *all visibly pushdown* context-free languages PDL[VPL], is decidable [20]. Note that L_1 is a visibly pushdown language (VPL) but L_2 is not.

PDL[VPL] and $\mathcal{L}_\mu$ are incomparable in expressive power. The latter can express all regular and no non-regular properties, the former can express some non-regular *path* properties and by no means all regular properties. A typical PDL[VPL] property not expressible in $\mathcal{L}_\mu$ is $\langle L_1 \rangle p$ stating "*there is a path with a label from* $a^n b^n$ *ending in a p-state.*" A typical regular property not expressible in PDL[VPL] (or even its extension with the Δ-operator [25]) is $\mu X.\mathsf{win} \vee \Diamond\Box X$ defining the set of winning positions for the beginning player in a turn-based two-player game.

In this paper we show that the boundary of undecidability can be pushed up even further. We introduce a modal fixpoint logic which embeds both PDL[VPL] and $\mathcal{L}_\mu$ – the currently known peaks in the hierarchy of decidable program logics – and is thus strictly more expressive than either of them. The logic is obtained as a fragment of Fixpoint Logic with Chop (FLC) [23], an extension of $\mathcal{L}_\mu$ in which subformulas denote predicate transformers rather than predicates. Syntactically, this extension is comparable to the step taken from right-linear to context-free grammars, and the presence of conjunctions then makes satisfiability undecidable in general. However, we define a fragment by restricting the syntax such that, structurally, formulas resemble visibly pushdown grammars. This helps to retain decidability. We call this fragment Visibly Pushdown Fixpoint Logic with Chop (vpFLC). Conceptually, it is close to PDL[VPL] but the restriction to pure (visibly pushdown) path properties is lifted, and the typical visibly-pushdown principles can be combined more or less freely with modal operators to form properties like "*there is an* $n \geq 1$ *s.t.* $\langle a \rangle^n [c] \langle b \rangle^n p$ *holds.*" Henceforth, we will call this the $\langle a \rangle^n [c] \langle b \rangle^n$-property. From what seems to almost be a pure side-effect of this construction, vpFLC can express all regular properties, too. Hence, it genuinely pushes the limits of decidability amongst modal fixpoint logics as it extends both $\mathcal{L}_\mu$ and PDL[VPL] which are, as said above, incomparable in expressive power. Hence, vpFLC is at least as expressive as their union. Recently, it has even been shown to be strictly more expressive than this union, since there even are non-regular (and therefore non-$\mathcal{L}_\mu$-expressible) properties like $\langle a \rangle^n [b]^n p$ which which can expressed in vpFLC but not in PDL[CFL] and therefore also not in PDL[VPL] [1].

In Sect. 2 we recall necessary preliminaries from program logics, formal languages and games. In Sect. 3 we formally define vpFLC and argue that it extends both PDL[VPL] and $\mathcal{L}_\mu$ such that we obtain a 2EXPTIME lower bound for satisfiability inherited from the

former. In Sect. 4 we define satisfiability games which are sound and complete for the full logic FLC, and show that for the fragment vpFLC they are decidable, leading to a matching 2EXPTIME upper bound. In Sect. 5 we conclude with remarks on further work in this area.

2 Preliminaries

2.1 Fixpoint Logic with Chop

Labelled transition systems. Let $\mathcal{P} = \{p, q, \ldots\}$ be a set of *atomic propositions* and $\Sigma = \{a, b, \ldots\}$ be a finite set of *action names*. A labeled transition system (LTS) over $\mathcal{P}$ and Σ is a $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$ s.t. $\mathcal{S}$ is a set of *states* with a designated initial state $s_0 \in S$, and $\to \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$ is the *transition relation*. We simply write $s \xrightarrow{a} t$ instead of $(s, a, t) \in \to$. Finally, $\lambda : \mathcal{S} \to 2^{\mathcal{P}}$ assigns a *label* to each state in the form of the atomic propositions which are supposed to be true in that respective state.

Syntax. Let $\mathcal{P}$ and Σ be as above. Let $\mathcal{V}$ be a countably infinite set of variable names. Formulas of FLC over $\mathcal{P}$, Σ and $\mathcal{V}$ are given by the following grammar.

$$\varphi \quad ::= \quad q \mid \overline{q} \mid X \mid \tau \mid \langle a \rangle \mid [a] \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid \varphi ; \varphi$$

where $q \in \mathcal{P}$, $a \in \Sigma$ and $X \in \mathcal{V}$.

We will write κ for either μ or ν. To save parentheses we introduce the convention that the operator ; binds stronger than $\wedge$ which binds stronger than $\vee$. We also use the abbreviations $\mathtt{ff} := q \wedge \overline{q}$ and $\mathtt{tt} := q \vee \overline{q}$ for some $q \in \mathcal{P}$. Note that FLC does not contain a negation operator to ensure well-definedness of fixpoints. Nevertheless, FLC is closed under complements, cf. [23] for details. Hence, we may also use Boolean operators like $\to, \leftrightarrow$ or even $\neg$ directly in examples when it helps to make formulas more understandable.

The set $Sub(\varphi)$ of subformulas of φ is defined as usual, with $Sub(\kappa X.\psi) = \{\kappa X.\psi\} \cup Sub(\psi)$ etc. For technical convenience with the satisfiability games introduced in Sect. 4, we assume that $\mathtt{tt}$ is always part of $Sub(\varphi)$ for any φ, cf. also the definition of the semantics below. Let $Sub^{\vee}(\varphi) = \{\psi_0 \vee \psi_1 \mid \psi_0 \vee \psi_1 \in Sub(\varphi)\}$ be the set of *disjunctive* subformulas of φ.

Formulas of the form q or $\overline{q}$ are called *literals*; those of the form $\langle a \rangle$ or $[a]$ are called *modalities*.

Formulas are assumed to be well-named in the sense that no variable is bound by a μ or a ν more than once in a given formula. Our main interest is with formulas that do not have free variables, in which case there is a function $fp_{\varphi} : \mathcal{V} \cap Sub(\varphi) \to Sub(\varphi)$ that maps each variable X to its unique defining fixpoint formula $\kappa X.\psi$ in φ.

Given two variables $X, Y \in Sub(\varphi)$ for some φ, we write $X <_{\varphi} Y$ if Y occurs free in $fp_{\varphi}(X)$. A variable X is called *outermost* among a set of variables $V \subseteq \mathcal{V} \cap Sub(\varphi)$ if it is maximal in V w.r.t. $<_{\varphi}$.

Semantics. Given an LTS $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$, an *environment* $\rho : \mathcal{V} \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ assigns to each variable a function from sets of states to sets of states in $\mathcal{T}$. We write $\rho[X \mapsto f]$ for the function that maps X to f and agrees with ρ on all other arguments.

The semantics $[\![\cdot]\!]^{\mathcal{T}}_{\rho} : 2^{\mathcal{S}} \to 2^{\mathcal{S}}$ of an FLC formula, relative to an LTS $\mathcal{T}$ and an environment, is such a function. It is monotone with respect to the inclusion ordering on $2^{\mathcal{S}}$. Such functions together with the partial order given by

$$f \sqsubseteq g \quad \text{iff} \quad \forall T \subseteq \mathcal{S} : f(T) \subseteq g(T)$$

form a complete lattice with joins $\sqcup$ and meets $\sqcap$ – defined as the pointwise intersection, resp. union. By the Knaster-Tarski Theorem [27] the least and greatest fixpoints of functionals $F : (2^{\mathcal{S}} \to 2^{\mathcal{S}}) \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ exist. They are used to interpret fixpoint formulas of FLC. The semantics is then inductively defined as follows.

$$\begin{aligned}
[\![q]\!]^{\mathcal{T}}_\rho &= _ \mapsto \{s \in \mathcal{S} \mid q \in \lambda(s)\} & [\![\langle a\rangle]\!]^{\mathcal{T}}_\rho &= T \mapsto \{s \in \mathcal{S} \mid \exists t \in T \text{ s.t. } s \xrightarrow{a} t\}\\
[\![\overline{q}]\!]^{\mathcal{T}}_\rho &= _ \mapsto \{s \in \mathcal{S} \mid q \notin \lambda(s)\} & [\![[a]]\!]^{\mathcal{T}}_\rho &= T \mapsto \{s \in \mathcal{S} \mid \forall t : s \xrightarrow{a} t \Rightarrow t \in T\}\\
[\![Z]\!]^{\mathcal{T}}_\rho &= \rho(Z) & [\![\mu X.\varphi]\!]^{\mathcal{T}}_\rho &= \bigsqcap\{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ mon., } [\![\varphi]\!]^{\mathcal{T}}_{\rho[X\mapsto f]} \sqsubseteq f\}\\
[\![\tau]\!]^{\mathcal{T}}_\rho &= T \mapsto T & [\![\nu X.\varphi]\!]^{\mathcal{T}}_\rho &= \bigsqcup\{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ mon., } f \sqsubseteq [\![\varphi]\!]^{\mathcal{T}}_{\rho[X\mapsto f]}\}\\
[\![\varphi;\psi]\!]^{\mathcal{T}}_\rho &= [\![\varphi]\!]^{\mathcal{T}}_\rho \circ [\![\psi]\!]^{\mathcal{T}}_\rho & [\![\varphi \vee \psi]\!]^{\mathcal{T}}_\rho &= [\![\varphi]\!]^{\mathcal{T}}_\rho \sqcup [\![\psi]\!]^{\mathcal{T}}_\rho\\
& & [\![\varphi \wedge \psi]\!]^{\mathcal{T}}_\rho &= [\![\varphi]\!]^{\mathcal{T}}_\rho \sqcap [\![\psi]\!]^{\mathcal{T}}_\rho
\end{aligned}$$

For any FLC formula φ, any LTS $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$ and any environment ρ let $\|\varphi\|^{\mathcal{T}}_\rho := [\![\varphi]\!]^{\mathcal{T}}_\rho(\mathcal{S})$. We call this the set of positions in $\mathcal{T}$ *defined* by φ and ρ. We also write $\mathcal{T}, s \models_\rho \varphi$ if $s \in \|\varphi\|^{\mathcal{T}}_\rho$, resp. $\mathcal{T} \models_\rho \varphi$ if $\mathcal{T}, s_0 \models_\rho \varphi$. If φ is closed we may omit ρ in both kinds of notation. We say that $\mathcal{T}$ is a *model* of a closed formula φ if $\mathcal{T} \models \varphi$. A formula is *satisfiable* if it has a model.

Two formulas φ and ψ are *equivalent*, written $\varphi \equiv \psi$, iff their semantics are the same, i.e. for every environment ρ and every LTS $\mathcal{T}$: $[\![\varphi]\!]^{\mathcal{T}}_\rho = [\![\psi]\!]^{\mathcal{T}}_\rho$. Two formulas φ and ψ are *weakly equivalent*, written $\varphi \approx \psi$, iff they define the same set of states in an LTS, i.e. for every ρ and every $\mathcal{T}$ we have $\|\varphi\|^{\mathcal{T}}_\rho = \|\psi\|^{\mathcal{T}}_\rho$. Hence, we have $\varphi \approx \varphi;\mathtt{tt}$ for any φ.

▶ **Example 1.** Take the $\langle a\rangle^n[c]\langle b\rangle^n$-property mentioned in the introduction. It is definable in FLC via $\varphi_{acb} := \langle a\rangle;(\mu X.[c] \vee \langle a\rangle;X;\langle b\rangle);\langle b\rangle;p$. For better readability, i.e. to stay closer to the syntax of more familiar fixpoint logics like $\mathcal{L}_\mu$ we drop the sequential composition operator when its left argument is a modality. This is semantically sound as the standard translation from $\mathcal{L}_\mu$ to FLC replaces $\langle a\rangle\varphi$ with $\langle a\rangle;\varphi$ etc. Then we can write this formula as $\langle a\rangle(\mu X.[c] \vee \langle a\rangle X;\langle b\rangle);\langle b\rangle p$.

To see that this truly formalises the desired property, we need two principles. They are simple consequences of the functional semantics and established truths in the theory of FLC.

1. Sequential composition is associative and left-commutes with Boolean operators, e.g. $(\varphi \vee \psi);\chi \equiv \varphi;\chi \vee \psi;\chi$.
2. The fixpoint unfolding principle holds, e.g. $\mu X.\varphi \equiv \varphi[\mu X.\varphi/X]$.

Then we have

$$\begin{aligned}
\langle a\rangle(\underbrace{\mu X.[c] \vee \langle a\rangle X;\langle b\rangle}_{\varphi_X});\langle b\rangle p &\equiv \langle a\rangle([c] \vee \langle a\rangle\varphi_X;\langle b\rangle);\langle b\rangle p\\
&\equiv \langle a\rangle[c]\langle b\rangle p \vee \langle a\rangle\langle a\rangle\varphi_X;\langle b\rangle\langle b\rangle p\\
&\equiv \langle a\rangle[c]\langle b\rangle p \vee \langle a\rangle\langle a\rangle([c] \vee \langle a\rangle\varphi_X;\langle b\rangle);\langle b\rangle\langle b\rangle p\\
&\equiv \langle a\rangle[c]\langle b\rangle p \vee \langle a\rangle^2[c]\langle b\rangle^2 p \vee \langle a\rangle^3([c] \vee \langle a\rangle\varphi_X;\langle b\rangle);\langle b\rangle^3 p\\
&\equiv \ldots \equiv \bigvee_{n\geq 1} \underbrace{\langle a\rangle \ldots \langle a\rangle}_{n}[c]\underbrace{\langle b\rangle \ldots \langle b\rangle}_{n} p
\end{aligned}$$

▶ **Example 2.** Consider $\varphi_{\mathsf{ubn}} := \nu X.\tau \wedge X;\langle a\rangle$. Remember that $\varphi_{\mathsf{ubn}} \approx \varphi_{\mathsf{ubn}};\mathtt{tt}$ by definition of the semantics. Using a similar unfolding approach as above, we can see that

$$\varphi_{\mathsf{ubn}};\mathtt{tt} \equiv \bigwedge_{n\geq 0} \langle a\rangle^n \mathtt{tt}$$

stating "*there are a-paths of unbounded lengths*." This is not a regular property either: take for instance $\varphi_{\mathsf{ubn}}; \mathtt{tt} \wedge \mu Y.[a]Y$, i.e. its conjunction with the property stating that all a-paths are of finite length. This formula is satisfiable but does not have finite models, hence, $\varphi_{\mathsf{ubn}}; \mathtt{tt}$ cannot be expressed in $\mathcal{L}_\mu$ which has the finite model property.

Later, we will need the notion of guardedness in FLC formulas which plays the same role as it does in $\mathcal{L}_\mu$, requiring that fixpoint variables occur "behind" modal operators. For FLC, this is easy to define formally.

▶ **Definition 3.** *Let $\varphi \in$ FLC. An occurrence of a fixpoint variable X is* guarded*, if it occurs in the right argument ψ_2 of a sequential composition $\psi_1; \psi_2$ within its defining fixpoint formula $fp_\varphi(X)$ such that ψ_1 does not contain τ. The formula φ itself is guarded if all occurrences of fixpoint formulas in it are guarded.*

For instance, φ_{acb} from Ex. 1 is guarded but φ_{ubn} from Ex. 2 is not. This lifts the notion of guardedness – useful for decision procedures – from $\mathcal{L}_\mu$ to FLC in the most obvious way.

2.2 Visibly Pushdown Systems and Languages

A *visibly pushdown alphabet* is a partition $\Sigma = \Sigma_{\mathsf{int}} \uplus \Sigma_{\mathsf{call}} \uplus \Sigma_{\mathsf{ret}}$ of an action set into three categories of designated *internal-*, *call-*, resp. *return* symbols.

The concepts introduced in the following are always defined relative to a fixed visibly pushdown alphabet which will not be mentioned over and over again. Moreover, when connecting multiple ones, for instance when explaining equivalence between two visibly pushdown automata, the underlying alphabet is always assumed to be partitioned in the same way for all participating entities.

Visibly pushdown systems. A *visibly pushdown frame* (VPF) over a visibly pushdown alphabet Σ, partitioned accordingly, is a (Q, Γ, q_0, δ) where Q is a finite set of states, $q_0 \in Q$ is a designated starting state, Γ is a finite stack alphabet including a designated bottom-of-stack symbol $\bot$, and $\delta = \delta_{\mathsf{call}} \cup \delta_{\mathsf{int}} \cup \delta_{\mathsf{ret}}$ with

$$\delta_{\mathsf{call}} \subseteq Q \times \Gamma \times \Sigma_{\mathsf{call}} \times Q \times \Gamma \ , \quad \delta_{\mathsf{int}} \subseteq Q \times \Gamma \times \Sigma_{\mathsf{int}} \times Q \times \Gamma \ , \quad \delta_{\mathsf{ret}} \subseteq Q \times (\Gamma \setminus \{\bot\}) \times \Sigma_{\mathsf{ret}} \times Q$$

is its transition table. A *visibly pushdown system* (VPS) over a set $\mathcal{P}$ of atomic propositions is a $\mathcal{A} = (Q, \Gamma, q_0, \delta, \lambda)$ where (Q, Γ, q_0, δ) are a visibly-pushdown frame and $\lambda : Q \to 2^{\mathcal{P}}$ assigns to each state a set of atomic propositions.

Such a VPS gives rise to a generally infinite-state LTS $\mathcal{T}_{\mathcal{A}} = (Q \times \Gamma^* \bot, \to, (q_0, \bot), \lambda')$ where $\lambda'(q, \gamma) := \lambda(q)$, and transitions are given as follows. Let $q, p \in Q$, $B, C \in \Gamma$, $\gamma \in \Gamma^*$ and $a \in \Sigma$.

$$\begin{aligned}
(q, B\gamma) &\xrightarrow{a} (p, C\gamma) && \text{if } (q, B, a, p, C) \in \delta_{\mathsf{int}} \\
(q, B\gamma) &\xrightarrow{a} (p, CB\gamma) && \text{if } (q, B, a, p, C) \in \delta_{\mathsf{call}} \\
(q, B\gamma) &\xrightarrow{a} (p, \gamma) && \text{if } (q, B, a, p) \in \delta_{\mathsf{ret}}
\end{aligned}$$

Remember that $\Sigma_{\mathsf{call}} \cap \Sigma_{\mathsf{int}} = \emptyset$. Hence, if $(q, B, a, p, C) \in \delta_{\mathsf{call}}$ and $(q, B, a', p, C) \in \delta_{\mathsf{int}}$ then $a \neq a'$. The underlying directed graph $(Q \times \Gamma^* \bot, \to)$ is also called the *configuration graph* of the VPS $\mathcal{A}$.

A *path* in (the LTS associated with the) VPS $\mathcal{A}$ is, as usual, an infinite sequence alternating between states and actions $\pi = (q_0, \gamma_0), a_0, (q_1, \gamma_1), a_1, \ldots$ s.t. $(q_i, \gamma_i) \xrightarrow{a_i} (q_{i+1}, \gamma_{i+1})$ for all $i \geq 0$. We write $Paths(\mathcal{T}_{\mathcal{A}})$ for the set of all paths through $\mathcal{T}_{\mathcal{A}}$ starting in the initial state $(q_0, \bot)$.

For such a path π, the set of states (of the underlying VPS) occurring infinitely often is $\inf(\pi) = \{q \in Q \mid \text{there are infinitely many } i \text{ s.t. } q = q_i\}$. The *word* of such a path π is $word(\pi) = a_0 a_1 \ldots \in \Sigma^\omega$.

Let $(q_0, B_0\gamma_0), (q_1, B_1\gamma_1), \ldots$ be the projection of such a path π to the states, ignoring actions and giving names to the top stack symbols in each position. Such a position $(q_i, B_i\gamma_i)$ in π is a *stair-position* if for every $j \geq i$: $|\gamma_j| \geq |\gamma_i|$. I.e. in stair positions the top-most stack symbol may become replaced but not removed. Moreover the content of the stack below the top-most symbol persists throughout the entire remainder of the path since changing it would require the top-most stack symbol to be removed. For a path π, let $stairs(\pi)$ be the projection of π onto stair-positions, i.e. $stairs(\pi) = (q_0, \bot), (q_{i_1}, B_{i_1}\gamma_{i_1}), \ldots$ for some $0 < i_1 < i_2 < \ldots$. Note that each infinite path has infinitely many stair positions since the bottom-of-stack symbol $\bot$ never gets removed.

Visibly pushdown automata. A *visibly pushdown Büchi automaton* (VPA) is a $\mathcal{A} = (Q, \Gamma, q_0, \delta, F)$ s.t. (Q, Γ, q_0, δ) is a VPF and $F \subseteq Q$. Note that this can be seen as a VPS over a singleton atomic proposition fin marking those states that belong to F. Hence, a VPA also gives rise to an LTS $\mathcal{T}_\mathcal{A}$ just like a VPS does. The two are distinguished only because of their different pragmatic use: a VPS is a finite representation of an infinite-state LTS, serving as a model for branching-time properties expressible in logics like FLC; a VPA is a finite representation of an ω-language, obtained as follows.

The *language* of the VPA $\mathcal{A} = (Q, \Gamma, q_0, \delta, F)$ is

$$L(\mathcal{A}) = \{word(\pi) \mid \pi \in Paths(\mathcal{T}_\mathcal{A}), F \cap \inf(\pi) \neq \emptyset\}$$

consisting of all paths in the associated LTS which traverse states from F infinitely often.

A *stair-parity automaton* (SPA) is a $\mathcal{A} = (Q, \Gamma, q_0, \delta, \Omega)$ where (Q, Γ, q_0, δ) is a VPF and $\Omega : Q \to \mathbb{N}$. The *language* of an SPA is

$$L(\mathcal{A}) = \{a_0 a_1 \ldots \in \Sigma^\omega \mid \text{there is } \pi = (q_0, \gamma_0), a_0, (q_1, \gamma_1), a_1, \ldots \in Paths(\mathcal{T}_\mathcal{A}) \text{ s.t. } stairs(\pi) = (q_{i_0}, \gamma_{i_0}), (q_{i_1}, \gamma_{i_1}), \ldots \text{ and } \limsup_{j \to \infty} \Omega(q_{i_j}) \text{ is even}\} .$$

Thus, a word $w = a_0 a_1 \ldots$ is accepted by a SPA if there is a path π through the LTS associated with the SPA s.t. the word along the transitions through π is w and the maximal priority which occurs infinitely often in stair positions along π is an even one.

An SPA $\mathcal{A} = (Q, \Gamma, q_0, \delta, \Omega)$ is *deterministic* (DSPA) if for every $q, p, p' \in Q$, $B, C, C' \in \Gamma$ and $a \in \Sigma$ we have

- if $(q, B, a, p, C), (q, B, a, p', C') \in \delta_{\mathsf{call}} \cup \delta_{\mathsf{int}}$ then $p = p'$ and $C = C'$, and
- if $(q, B, a, p), (q, B, a, p') \in \delta_{\mathsf{ret}}$ then $p = p'$.

The purpose of the introduction of the complicated stair-parity acceptance condition is the fact that (nondeterministic) VPA are not determinisable, not even with Muller acceptance conditions [2]. However, when the range of interpretation of the acceptance condition is restricted as it is done in SPA, then determinisation becomes possible. Moreover, DSPA are easy to complement.

We measure the size of a VPA or SPA over the VPF (Q, Γ, q_0, δ) as $|Q| + |\Gamma| + |\delta|$.

▶ **Proposition 4** ([21]). *For every VPA $\mathcal{A}$ of size n there is a DSPA $\mathcal{D}$ of size $2^{\mathcal{O}(n^2)}$ s.t. $L(\mathcal{D}) = \overline{L(\mathcal{A})}$.*

The main purpose of automata determinisation in the context of game solving is their ability to reduce games with abstract winning conditions to those with very concrete ones, see Cor. 6 below.

2.3 Games

Graph games. Two-player games played on directed graphs play a fundamental role in system's verification based on formal logic. We will use the game-theoretic approach to characterise FLC's satisfiability problem in Sect. 4 and to show decidability of the fragment defined in Sect. 3. This uses particular games known as stair-parity games defined below.

An (edge-labeled) *game* over a set Σ of labels is a $\mathcal{G} = (V, V_\exists, V_\forall, E, v_0, W)$ s.t. (V, E) is a directed graph. The nodes are also called *configurations* or *positions* of the game (arena). The edge relation $E \subseteq V \times \Sigma \times V$ is usually assumed to be total in the sense that for each $v \in V$ and $a \in \Sigma$ there is a $u \in V$ with $(v, a, u) \in E$. This is not a strict requirement, though, as there are easy ways to transform games without total edge relations into those with. $V_\exists$ and $V_\forall$ partition the set of positions into those owned by player $\exists$, resp. player $\forall$. $W \subseteq (V \times \Sigma)^\omega$ is the *winning condition* for player $\exists$.

A *play* is an alternating sequence of positions and actions $\pi = v_0, a_0, v_1, a_1, \ldots$ starting in the initial position v_0 and satisfying $(v_i, a_i, v_{i+1}) \in E$ for all $i \geq 0$. Player $\exists$ wins π if $\pi \in W$, otherwise it is won by player $\forall$.

A *strategy* for player $p \in \{\exists, \forall\}$ is a $\sigma_p : (V \times \Sigma)^*(V_p \times \Sigma) \to V$ s.t. for all $\rho = v_0, a_0, \ldots, v_n, a_n$ with $v_n \in V_p$ we have $(v_n, a_n, \sigma_p(\rho)) \in E$. I.e. it prescribes, given the *history* of a moment in a play, the next move to player p in terms of a successors of the current position and a given action. A play $\pi = v_0, a_0, v_1, a_1, \ldots$ *adheres* to σ_p if for every $n \in \mathbb{N}$ with $v_n \in V_p$ we have that $v_{n+1} = \sigma_p(v_0, a_0, \ldots, v_n, a_n)$, i.e. in this play, player p has always followed the advice given by σ_p whenever it was her turn to move.

The strategy σ_p is a *winning strategy* for player p, if every play that adheres to σ_p is winning for player p. The problem of *solving* is: given a game $\mathcal{G}$, decide whether or not player $\exists$ has a winning strategy for $\mathcal{G}$.

Stair-Parity Games. A *stair-parity game* (SPG) is a $\mathcal{G} = (Q, Q_\exists, Q_\forall, \Gamma, q_0, \delta, \Omega)$ such that $\mathcal{F} = (Q, \Gamma, q_0, \delta)$ is a VPF with its state space partitioned into $Q = Q_\exists \uplus Q_\forall$, and $\Omega : Q \to \mathbb{N}$ is a priority function as above. It gives rise to the abstract game $(V, V_\exists, V_\forall, v_0, E, W)$ where the arena (V, E) is the configuration graph of $\mathcal{F}$. The initial position is $v_0 = (q_0, \bot)$. The partition of positions belonging to either of the players is derived from the partition $Q_\exists \uplus Q_\forall$ into states belonging to them, s.t. $V_p = Q_p \times \Gamma^* \bot$ for $p \in \{\exists, \forall\}$.

The winning condition derived from Ω is defined similar to the acceptance condition for stair-parity automata, hence the name. A play $\pi = (q_0, \gamma_0), a_0, (q_1, \gamma_1), a_1, \ldots$ belongs to W iff $\limsup_{j \to \infty} \Omega(q_{i_j})$ is even where $i_0, i_1, i_2, \ldots$ is the set of stair positions in π.

The *size* of an SPG is simply defined as $|\delta|$. Note that an SPG can easily be represented using space that is linear in $|\delta|$. Another important parameter for the complexity of analysing games is its *index* which is defined as $|\{\Omega(q) \mid q \in Q\}|$.

▶ **Proposition 5** ([21]). *The problem of solving an SPG is decidable in* EXPTIME.

The proof is by a reduction to the problem of solving a parity game [22, 5] which is exponential in the size of the game but polynomial in its index. There are several algorithms showing that parity games can be solved in time polynomial in their size but exponential in their index, cf. [13], resulting in the EXPTIME upper bound for SPGs. Recent advances showing that parity games can be solved in quasi-polynomial [4] time do not have any impact that is significant for our purposes here.

▶ **Corollary 6.** *Suppose $\mathcal{G}$ is a game of size n, played on an arena that is a VPF with winning condition W s.t. $\overline{W}$ is recognised by a VPA of size $\mathcal{O}(n^c)$. Then $\mathcal{G}$ is solvable in* 2EXPTIME.

Proof. This uses the well-known product construction between a game and a deterministic automaton (cf. [9]) for the winning condition, obtained here through Prop. 4. If the latter is a DSPA, the resulting product becomes a stair-parity game of exponential size, and the doubly exponential bound follows from Prop. 5. ◀

3 Visibly Pushdown Fixpoint Logic with Chop

3.1 Syntax

The definition of the syntax of vpFLC is inspired by the structure of visibly pushdown grammars (VPG) [3] and PDL[VPL] (see Appendix A for a brief introduction). Again, we assume the set of actions to be partitioned into $\Sigma = \Sigma_{\mathsf{int}} \uplus \Sigma_{\mathsf{call}} \uplus \Sigma_{\mathsf{ret}}$.

▶ **Definition 7.** *The syntax of the fragment* vpFLC *of* FLC *is given by the following grammar.*

$$\begin{aligned}\varphi \quad ::= \quad & q \mid \overline{q} \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu X.\varphi \mid \nu X.\varphi \mid \\ & \langle a_{\mathsf{int}} \rangle \mid \langle a_{\mathsf{int}} \rangle ; \varphi \mid \langle a_{\mathsf{call}} \rangle ; \langle a_{\mathsf{ret}} \rangle \mid \langle a_{\mathsf{call}} \rangle ; \varphi ; \langle a_{\mathsf{ret}} \rangle \mid \langle a_{\mathsf{call}} \rangle ; \langle a_{\mathsf{ret}} \rangle ; \varphi \mid \langle a_{\mathsf{call}} \rangle ; \varphi ; \langle a_{\mathsf{ret}} \rangle ; \varphi\end{aligned}$$

where $q \in \mathcal{P}$, $X \in \mathcal{V}$, $a_x \in \Sigma_x$ for $m \in \{\mathsf{int}, \mathsf{call}, \mathsf{ret}\}$, and $\langle a \rangle$ can be either $\langle a \rangle$ or $[a]$. Furthermore, we postulate that the sequential composition operator is right-associative; parentheses are not shown explicitly here for the sake of better readability.

Hence, vpFLC restricts the use of sequential composition such that the left argument is a modality over an internal or call-action. In the latter case, the right argument contains a modality over a return-action, possibly surrounded by formulas. The termination operator τ – the neutral element for sequential composition – is forbidden.

▶ **Example 8.** The $\langle a \rangle^n [c] \langle b \rangle^n$-property is definable in vpFLC provided that $a \in \Sigma_{\mathsf{call}}$, $b \in \Sigma_{\mathsf{ret}}$ and $c \in \Sigma_{\mathsf{int}}$. Reconsider the FLC formula φ_{acb} defined in Ex. 1. It is easily seen to be a vpFLC formula; note how the call-modality $\langle a \rangle$ is paired with the return-modality $\langle b \rangle$ both around the fixpoint variable X as well as the entire fixpoint formula for X.

The formula $\mu X.\langle b \rangle \vee \langle a \rangle ; X ; \langle a \rangle$ stating "*there is a path labeled $a^n b a^n$ for some $n \geq 0$*" is not a vpFLC formula because a cannot be a call- and return-modality at the same time, see also the comment on L_2 not being a VPL in the introduction.

3.2 Expressive Power

The logic vpFLC extends both $\mathcal{L}_\mu$ and PDL[VPL] in expressive power.

▶ **Theorem 9.** *For every formula $\varphi \in \mathcal{L}_\mu \cup$ PDL[VPL] there is a $\psi \in$ vpFLC s.t. $\psi \equiv \varphi$ and $|\psi| = \mathcal{O}(|\varphi|)$.*

Proof. (Sketch) The case of $\varphi \in \mathcal{L}_\mu$ is easy as $\mathcal{L}_\mu$ embeds into FLC straightforwardly via $\langle a \rangle \chi \equiv \langle a \rangle ; \chi$ etc., cf. [23]. By considering all action symbols to be internal, $\Sigma = \Sigma_{\mathsf{int}}$, the resulting formulas fall into vpFLC. In fact, the standard translation produces formulas that fall into the syntax when restricted to literals, Boolean connectives, fixpoints and formulas of the form $\langle a \rangle ; \varphi$ and $[a] ; \varphi$ alone.

The other clauses in the syntax are needed to capture PDL[VPL]. We sketch the translation here for PDL[VPL] without the test operator. A corresponding extension is just a matter of technicality then. Take a PDL[VPL] formula of the form $\langle L \rangle \varphi$ where L is a VPL.

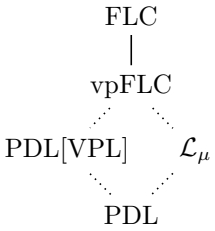

Figure 1 The expressiveness of logics closely related to vpFLC. Dotted lines are strict inclusions.

W.l.o.g. we can assume $\varepsilon \notin L$ because of $\langle\{\varepsilon\} \cup L\rangle\varphi \equiv \varphi \vee \langle L \setminus \{\varepsilon\}\rangle\varphi$. According to [3], L is representable by a context-free grammar with productions of the form $A \to \varepsilon$, $A \to a_{\mathsf{int}}B$, $A \to a_{\mathsf{call}}Ba_{\mathsf{ret}}C$. Using standard ε-elimination, this can be transformed into a grammar with productions in either of the following six forms.

$$A \to a_{\mathsf{int}},\quad A \to a_{\mathsf{int}}B,\quad A \to a_{\mathsf{call}}a_{\mathsf{ret}},\quad A \to a_{\mathsf{call}}Ba_{\mathsf{ret}},\quad A \to a_{\mathsf{call}}a_{\mathsf{ret}}C,\quad A \to a_{\mathsf{call}}Ba_{\mathsf{ret}}C$$

Applying the translation of full PDL[CFL] into unrestricted FLC from [19] results in formulas which fall into vpFLC – note how its syntax resembles the form of these six grammar productions. ◀

For the embedding of PDL[VPL] it would indeed be easier to have τ in the syntax of vpFLC as this would eliminate the need to argue about the elimination of ε, and we would only need two instead of six patterns of productions. However, the inclusion of τ in vpFLC would invalidate the relatively simple argument in the proof of Lemma 19 below which is needed in the decidability proof.

Since PDL[VPL] and $\mathcal{L}_\mu$ are known to be incomparable in expressive power [20] we even have strict inclusion of both of them in vpFLC.

▶ **Corollary 10.** *Both* PDL[VPL] *and* $\mathcal{L}_\mu$ *are strictly less expressive than* vpFLC.

In fact, vpFLC is even more expressive than the union of PDL[VPL] and $\mathcal{L}_\mu$, cf. [1]. Fig. 1 shows the hierarchy of expressiveness amongst the modal fixpoint logics mentioned here. The undecidability of FLC– as opposed to vpFLC's decidability shown in the following section – gives a strong indication that FLC is strictly more expressive than vpFLC: there can, at least, be no *effective* translation from FLC to vpFLC.

Another consequence of the embeddings, which happen to be polynomial, is the inheritance of lower complexity bounds of the satisfiability problems from the embedded logics. For $\mathcal{L}_\mu$ it is "only" EXPTIME-hard, this is already the case for the smaller PDL [7]. However, the satisfiability problem for PDL[VPL] is even 2EXPTIME-hard [20].

▶ **Corollary 11.** *The satisfiability problem for* vpFLC *is* 2EXPTIME*-hard.*

4 Satisfiability Games

4.1 Satisfiability Games for FLC

Let $\chi \in$ FLC be closed. The *satisfiability game* $\mathcal{G}^{\mathsf{sat}}(\chi)$ is played between the *verifier* (**V**) and the *refuter* (**R**). A position in this game is a set of stacks, and a stack is a sequential composition $\psi_1; \ldots; \psi_m$ of subformulas of χ. A position $\mathcal{C}$ is written $\gamma_1, \ldots, \gamma_n$, resp. $\varphi_1;\gamma_1\ ,\ \ldots\ ,\ \varphi_n;\gamma_n$ if we want to refer to the tops, resp. *heads* of the stacks particularly.

$$(\vee)\ \frac{(\varphi_0 \vee \varphi_1);\gamma\ ,\ \Phi}{\varphi_i;\gamma\ ,\ \Phi}\ \mathbf{V} : i \in \{0,1\} \qquad (\wedge)\ \frac{(\varphi_0 \wedge \varphi_1);\gamma\ ,\ \Phi}{\varphi_0;\gamma\ ,\ \varphi_1;\gamma\ ,\ \Phi} \qquad (;)\ \frac{(\varphi_0;\varphi_1);\gamma\ ,\ \Phi}{\varphi_0;\varphi_1;\gamma\ ,\ \Phi}$$

$$(\tau)\ \frac{\tau;\gamma\ ,\ \Phi}{\gamma\ ,\ \Phi} \qquad (\mathsf{fp})\ \frac{\kappa X.\varphi;\gamma\ ,\ \Phi}{X;\gamma\ ,\ \Phi} \qquad (\mathsf{var})\ \frac{X;\gamma\ ,\ \Phi}{\varphi;\gamma\ ,\ \Phi}\ \text{if } fp_\chi(X) = \kappa X.\varphi$$

$$(\mathsf{mod})\ \frac{\ell_1;\gamma_1,\ldots,\ell_n;\gamma_n\ ,\ \langle a_1\rangle;\gamma'_1,\ldots,\langle a_m\rangle;\gamma'_m\ ,\ [b_1];\gamma''_1,\ldots,[b_k];\gamma''_k}{\gamma'_i, \{\gamma''_j \mid b_j = a_i\}}\ \mathbf{R} : i \in \{1,\ldots,m\}$$

Figure 2 The rules of the FLC satisfiability games.

We may abbreviate several elements in a set of elements, writing such a position for instance as $\varphi;\gamma\ ,\ \Phi$, for instance when the particular shapes of elements other than $\varphi;\gamma$ are of no particular concern.

The *initial* position of the game $\mathcal{G}^{\mathsf{sat}}(\chi)$ is $\mathcal{C}_0 = \chi;\mathtt{tt}$, i.e. it only contains a single stack whose head is χ and rest is the single formula $\mathtt{tt}$. The game then proceeds according to the rules shown in Fig. 2. They all, apart from (mod), operate on the head of one stack, transforming this within a game position. If the head formula is a disjunction, then player $\mathbf{V}$ performs a choice with rule $(\vee)$ to replace it by one of its disjuncts. The others are deterministic and do not require a player to make a choice. Rule (mod), though, requires player $\mathbf{R}$ to make one. It is only applicable when all heads in the current position are either literals or modal formulas. The conclusion of this rule is called an a-child of its premise if $a_i = a$ for the i chosen by $\mathbf{R}$.

The next step to be taken is to explain under which condition a play is won by one of the players. This requires a technical definition, though.

▶ **Definition 12.** *Suppose $\pi = \Phi_0, \Phi_1, \ldots$ is a* play *of $\mathcal{G}^{\mathsf{sat}}(\chi)$. A* thread *in π is a sequence $\pi = \gamma_0\ , \gamma_1, \ldots$ of stacks s.t. for all $i \geq 0$: $\gamma_i \in \Phi_i$ and the rule played in Φ_i either*

- *operates on a formula different to the top of the stack γ_i and $\gamma_{i+1} = \gamma_i$, or*
- *operates on the top of γ_i, and γ_{i+1} results from γ_i through this rule application.*

Let $\varphi_0, \varphi_1, \ldots$ be the sequence of topmost formulas in the stacks of π, i.e. $\gamma_i = \varphi_i;\gamma'_i$ for some γ'_i. Let $i_0, i_1, \ldots$ be the set of stair positions in π' according to the definition in Sect. 2.2.

We say that π is a μ-thread, if the outermost variable occurring infinitely often in the sequence $\varphi_{i_0}, \varphi_{i_1}, \ldots$ is of type μ. Otherwise, it is a ν-thread.

Threads trace the evolution of single formulas with their appended stacks through a play, and μ-threads witness the existence of a non-well-founded least fixpoint construct in the configurations of a play. These should be avoided in the search for a model. In other words, player $\mathbf{V}$– who aims to prove satisfiability of the input formula – should avoid μ-threads in her proof (in form of a winning strategy).

▶ **Definition 13.** *The play $\pi = \Phi_0, \Phi_1, \ldots$ of $\mathcal{G}^{\mathsf{sat}}(\chi)$ is won by player $\mathbf{R}$, if*

- *there is an n s.t. $\Phi_n = q;\gamma\ , \overline{q};\gamma'\ , \Phi$ for some $q \in \mathcal{P}$ and some γ, γ', Φ; or*
- *π contains a μ-thread.*

Otherwise, player $\mathbf{V}$ wins π.

Next we show that the FLC satisfiability games are sound and complete. We make use of model checking games for FLC [17]; for convenience they are presented in Appendix B.

▶ **Theorem 14.** *If χ is satisfiable then player* $\mathbf{V}$ *has a winning strategy for* $\mathcal{G}^{\mathsf{sat}}(\chi)$.

Proof. Suppose there is an LTS $\mathcal{T}$ with a state s_0 s.t. $\mathcal{T}, s_0 \models \chi$. By definition we also have $\mathcal{T}, s \models \chi; \mathtt{tt}$. A strategy $\sigma_{\mathbf{V}}$ for $\mathbf{V}$ can be described as follows. She annotates each configuration $\gamma_1, \dots, \gamma_m$ with a state s from $\mathcal{T}$, written $s \vdash \gamma_1, \dots, \gamma_m$. We then call such an annotated configuration *safe* if $\mathbf{V}$ has a winning strategy in $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}}(\chi)$ starting from $s \vdash \gamma_i$ for all i. By completeness of the model checking games, she has a winning strategy σ' for $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}}(\chi)$ starting in $s_0 \vdash \chi; \mathtt{tt}$ and, hence, this annotated initial configuration is safe.

The key observation here is that $\mathbf{V}$ can preserve safety by consulting σ': whenever she is required to choose a disjunct in an annotated configuration $s \vdash (\varphi_0 \vee \varphi_1); \gamma, \Phi$, there is a corresponding choice of a disjunct under σ' in a configuration $s \vdash (\varphi_0 \vee \varphi_1); \gamma$. Moreover, the deterministic rules and $\mathbf{R}$'s choices preserve safety in general.

Now suppose $\pi = s_0 \vdash \Phi_0,\ s_1 \vdash \Phi_1, \dots$ is an (annotated) play adhering to this strategy $\sigma_{\mathbf{V}}$. It remains to be seen that it is won by $\mathbf{V}$. The second key observation is that any thread in this play, together with the state annotation corresponds to a play in $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}}(\chi)$ that adheres to σ'. By assumption, it is won by $\mathbf{V}$. If it is an infinite play then the outermost variable occurring infinitely often in stair positions is of type ν, i.e. it is a ν-thread. In other words, π cannot contain a μ-thread and is therefore won by $\mathbf{V}$. π cannot be won in finite time by player $\mathbf{R}$ either because there cannot be a safe configuration of the form $s \vdash q; \gamma,\ \overline{q}; \gamma,\ \Phi$, as by soundness of the model checking games we would have $s \models q$ and $s \not\models q$ at the same time. Hence, π is won by $\mathbf{V}$, showing that $\sigma_{\mathbf{V}}$ is indeed a winning strategy. ◀

Soundness does not hold in general: take for instance $(\nu X.X) \wedge (\mu Y.Y)$ which is unsatisfiable. Not every play will exhibit a μ-thread, though, as one may get caught up in the unwinding of $\nu X.X$. Guardedness (cf. Def. 3) excludes this as it requires rule (mod) to be played between each two unfoldings of any fixpoint formula.

▶ **Theorem 15.** *Let χ be guarded. If player* $\mathbf{V}$ *has a winning strategy for* $\mathcal{G}^{\mathsf{sat}}(\chi)$ *then χ is satisfiable.*

Proof. Consider the tree of all plays adhering to $\mathbf{V}$'s winning strategy σ. Guardedness ensures that each play uses rule (mod)infinitely often. Moreover, note that this tree only branches through applications of rule (mod) since this is the only one that gives player $\mathbf{R}$ the ability to perform choices.

We extract a tree model $\mathcal{T}_\sigma$ from this tree of plays by collapsing each segment $C_i, \dots, C_j$ into a state $\langle C_i, \dots, C_j \rangle$ s.t. before C_i and in C_j rule (mod) has been applied. Transitions are given as $\langle C_i, \dots, C_j \rangle \xrightarrow{a} \langle C_{j+1}, \dots, C_k \rangle$ if C_{j+1} is an a-child of C_j in rule (mod). The labelling is given by $q \in \lambda(\langle C_i, \dots, C_j \rangle)$ if $q; \gamma \in C_j$ for some γ. Write $\vec{C}_0$ for the initial state of $\mathcal{T}_\sigma$ according to this segmentation.

It remains to be seen that $\mathcal{T}_\sigma, \vec{C}_0 \models \chi$. Suppose it was not the case. By soundness of $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}_\sigma}(\chi)$, $\mathbf{R}$ would have a winning strategy $\sigma_{\mathbf{R}}$ for this game starting in $\vec{C}_0 \vdash \chi; \mathtt{tt}$. As in the proof of Thm. 14, the key observation is that any infinite play adhering to $\sigma_{\mathbf{R}}$ is a μ-thread in a play adhering to σ, contradicting the assumption that σ is a winning strategy for $\mathbf{V}$ in $\mathcal{G}^{\mathsf{sat}}(\chi)$. Moreover, suppose $\mathbf{R}$ won a finite play with $\sigma_{\mathbf{R}}$ by ending in a configuration $\vec{C} \vdash q; \gamma$, i.e. $\vec{C} \not\models q$ but $q \in \lambda(\vec{C})$ by construction. This also contradicts the assumption that σ would be a winning strategy. The other cases of winning finite plays are handled equally. Hence, we must indeed have $\mathcal{T}_\sigma, \vec{C}_0 \models \chi$ finishing the proof. ◀

▶ **Example 16.** Consider $\varphi = (\mu Y.[c] \vee \nu X.\langle a \rangle; Y; \langle b \rangle; X) \wedge \nu Z.[a]; (Z \wedge \langle c \rangle; \mathtt{tt}); [b]$. The initial part of one play of the game $\mathcal{G}^{\mathsf{sat}}(\varphi)$ is shown in Fig. 3. For sake of brevity, rules that operate on separate stacks in a configuration have been compressed into a single step from one configuration to a next one, and the first rule application is not shown.

$$(\mu Y.[c] \lor \nu X.\langle a\rangle; Y; \langle b\rangle; X); \mathtt{tt} \;,\; (\nu Z.[a]; (Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$\text{(fp), (fp)}$$
$$Y; \mathtt{tt} \;,\; Z; \mathtt{tt}$$
$$\text{(var), (var)}$$
$$([c] \lor \nu X.\langle a\rangle; Y; \langle b\rangle; X); \mathtt{tt} \;,\; ([a]; (Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$(\lor), (;)$$
$$(\nu X.\langle a\rangle; Y; \langle b\rangle; X); \mathtt{tt} \;,\; [a]; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$\text{(fp)}$$
$$X; \mathtt{tt} \;,\; [a]; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$\text{(var)}$$
$$(\langle a\rangle; Y; \langle b\rangle; X); \mathtt{tt} \;,\; [a]; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$(;)$$
$$\langle a\rangle; (Y; \langle b\rangle; X); \mathtt{tt} \;,\; [a]; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$\text{(mod)}$$
$$(Y; \langle b\rangle; X); \mathtt{tt} \;,\; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); \mathtt{tt}$$
$$(;), (;)$$
$$Y; (\langle b\rangle; X); \mathtt{tt} \;,\; (Z \land \langle c\rangle; \mathtt{tt}); [b]; \mathtt{tt}$$
$$\text{(var)}, (\land)$$
$$([c] \lor \nu X.\langle a\rangle; Y; \langle b\rangle; X); (\langle b\rangle; X); \mathtt{tt} \;,\; Z; [b]; \mathtt{tt} \;,\; \langle c\rangle; \mathtt{tt}; [b]; \mathtt{tt}$$
$$(\lor), \text{(var)}$$
$$[c]; (\langle b\rangle; X); \mathtt{tt} \;,\; ([a]; (Z \land \langle c\rangle; \mathtt{tt}); [b]); [b]; \mathtt{tt} \;,\; \langle c\rangle; \mathtt{tt}; [b]; \mathtt{tt}$$
$$(;)$$
$$[c]; (\langle b\rangle; X); \mathtt{tt} \;,\; [a]; ((Z \land \langle c\rangle; \mathtt{tt}); [b]); [b]; \mathtt{tt} \;,\; \langle c\rangle; \mathtt{tt}; [b]; \mathtt{tt}$$
$$\text{(mod)}$$
$$(\langle b\rangle; X); \mathtt{tt} \;,\; \mathtt{tt}; [b]; \mathtt{tt}$$
$$(;)$$
$$\langle b\rangle; X; \mathtt{tt} \;,\; \mathtt{tt}; [b]; \mathtt{tt}$$
$$\text{(mod)}$$
$$X; \mathtt{tt}$$
$$\vdots$$

Figure 3 Initial part of a play on the formula from Ex. 16 with some rule applications compressed.

It would be cumbersome to try to express the property formalised by φ in English words; in fact, φ is constructed specifically to exemplify the mechanics of the satisfiability games. The grey background highlights the evolution of one of the stacks, particularly its elements up to stack height 2. One can see that the stack reaches height 3 inbetween, and it is there that Y occurs in head position for the second time. This is not a stair position, though. The play could be continued s.t. both X and Y occur infinitely often in head positions but it is only the inner X which does so in stair positions. Hence, this forms a ν-thread.

4.2 Satisfiability Games for Guarded vpFLC are Stair-Parity Games

Thms. 14 and 15 give a reduction from FLC's satisfiability problem to the problem of solving particular games on infinite-state spaces. Undecidability of the former transfers to these games which are therefore not algorithmically solvable for arbitrary FLC formulas. They are, however, for guarded vpFLC formulas. We prove this by revealing them as special stair-parity games. As a first step in this direction, we reformulate them as turn-based games. Fix some $\chi \in$ vpFLC over some visibly pushdown alphabet Σ.

▶ **Definition 17.** *Configurations of the* turn-based satisfiability game $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ *are of the same form as those in the original* FLC *satisfiability game* $\mathcal{G}^{\mathsf{sat}}(\chi)$. *The initial configuration is also* $\chi; \mathtt{tt}$. *The rules, however, deviate.*

Let $f : Sub^{\lor}(\chi) \to \{0,1\}$. *A configuration* C' *is the* f*-normalisation of a configuration* C, *if* C' *is obtained by repeatedly applying rules* $(\land)$, $(;)$, (fp) *and* (var) *to* C *until all heads of all stacks are either literals or modalities. Likewise, when some head of a stack is of a form* $\psi_0 \lor \psi_1$ *then it gets replaced by* ψ_i *using rule* $(\lor)$ *where* $i = f(\psi_0 \lor \psi_1)$.

In a configuration C, *the turn-based game* $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ *proceeds as follows.*

1. **V** *selects a function* $f : Sub^{\lor}(\chi) \to \{0,1\}$,
2. **R** *applies rule* (mod) *to the* f*-normalisation of* C.

The following is not difficult to see; $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ can be seen as an efficient way of playing $\mathcal{G}^{\mathsf{sat}}(\chi)$ with several rules being played at once.

▶ **Lemma 18.** *Player* **V** *wins* $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ *iff she wins* $\mathcal{G}^{\mathsf{sat}}(\chi)$.

Proof. Note that all that the definition of $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ does is to condense consecutive choices **V** does in selecting disjuncts and the applications of deterministic rules into a single move by player **V**. All that is needed then is to see that the syntax of vpFLC guarantees the f-normalisation of any configuration to exist (uniquely), for any f. In a sense, vpFLC formulas are guarded, i.e. between two unfoldings of a fixpoint variable, rule (mod) needs to be played. In $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$, this just happens in two consecutive steps requiring a **V** choice followed by a **R** choice. Hence, winning strategies can easily be transferred between these two games. ◀

The next goal is to show that turn-based satisfiability games are in fact stair-parity games. There are two obstacles to overcome. Note that positions in stair-parity games contain a single stack, whereas satisfiability games on arbitrary FLC formulas can contain an unbounded number of stacks.

1. There needs to be a representation of the set of stacks as a single one. This is possible because on vpFLC formulas, the sizes of all stacks do not deviate by much. Secondly, we can use a standard encoding via transition profiles in order to represent the unboundedly many stacks by one of fixed width.
2. The winning condition for **V** needs to be phrased as a stair-parity condition.

To tackle the first problem, we call a configuration $\gamma_1, \dots, \gamma_m$ *k-aligned* for $k \geq 0$, if the stack heights differ by at most k: $||\gamma_i| - |\gamma_j|| \leq k$ for all i, j. Note that the initial configuration of $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ is always 0-aligned as it contains only one stack. The first key observation here is the following. It uses the fact that sequential composition is forced by the syntax to be right-associative, and that a child under the modal rule is built from stacks all starting with modalities for the same action symbol.

▶ **Lemma 19.** *The following holds.*

a) *The f-normalisation C' of a 0-aligned configuration C is 1-aligned, and*

b) *the a-child C'' of any 1-aligned configuration C' in rule* (mod) *is 0-aligned, for any $a \in \Sigma$.*

Proof. Part (b) is easier to see: note that all stacks in C'' are suffixes of a stack in C' which started with a modality containing a. The syntax of vpFLC can only create non-0-aligned configurations by making use of two different actions, though. Note how, for instance, a call-modality $\langle a \rangle$ can only be used in a formula of the form $\alpha_{\mathsf{call}}; \varphi; \alpha_{\mathsf{ret}}$ according to the syntax, and this is the case for any other stack head which is either $\langle a \rangle$ or $[a]$ for the same $a \in \Sigma$.

For part (a) observe that sequential compositions in the syntax of vpFLC (apart from those directly linked to call- and return-modalities) are right-associative. Hence, rule (;) can increase the size of a stack by one but not any more as further sequential compositions can only occur in the right argument but not the left. ◀

This reduces the complexity of a potentially unbounded number of stacks of different height to a potentially unbounded number of stacks of essentially the same height. Next we observe that it can be reduced further to a bounded number of stacks of (almost) equal height. This follows from the fact that the game rules operate on the stack heads only of which there are at most n different ones. Additional stacks can only be created using rule (∧) which duplicates the rest of the stack that it operates on.

▶ **Lemma 20.** *Let $C = \gamma_1, \dots, \gamma_m$ be a configuration of $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ and $n := |\chi|$. There are $1 \leq i_1, \dots, i_n \leq m$ s.t. for all $j \in \{1, \dots, m\}$ there is a $j' \in \{i_1, \dots, i_n\}$ s.t. γ_j and $\gamma_{j'}$ differ at most in their heads.*

Consequently, the m stacks in a configuration can only have been grown out of n different ones in the previous configuration. This suggests a compact representation of a configuration $\Phi = \gamma_1, \dots, \gamma_m$ with $\gamma_i = \psi_{i,h}; \dots; \psi_{i,0}$ as a single stack over the stack alphabet $\Gamma_\chi = 2^{[n]\times[n]}$. Assume some enumeration $\varphi_0, \dots, \varphi_{n-1}$ of $Sub(\chi)$. Then Φ can be represented as $g_h; \dots; g_1$ where, for each $i = h, \dots, 1$: $(j,k) \in g_i$ if there is j' s.t. $\psi_{j',i} = \varphi_j$ and $\psi_{j',i-1} = \varphi_k$. So in order to reconstruct the stack γ_i, one starts with the index of its head and follows the connections through $g_h; \dots; g_1$ from left to right. This representation technique is also used in the determinisation of VPA [2] and in decision procedures [8].

To overcome the second obstacle of matching $\mathbf{V}$'s winning condition as a stair parity condition, we employ automata-theory. We define a new such alphabet

$$\Sigma^\chi := \{\mathsf{ch}_f \mid f \in Sub^\vee(\chi) \to \{0,1\}\} \cup \{\mathsf{mod}_{\langle a \rangle} \mid \langle a \rangle \in Sub(\chi)\}$$

s.t. $\mathsf{ch}_f \in \Sigma^\chi_{\mathsf{int}}$ for all f, and $\mathsf{mod}_{\langle a \rangle} \in \Sigma^\chi_x$ if $a \in \Sigma_x$. Note that any play of $\mathcal{G}^{\mathsf{sat}}_{\mathsf{tb}}(\chi)$ can easily be represented by a Σ'^ω-word of the form $\mathsf{ch}_{f_0}, \mathsf{mod}_{\langle a_0 \rangle}, \mathsf{ch}_{f_1}, \mathsf{mod}_{\langle a_1 \rangle}, \dots$. The Σ^χ-symbols uniquely determine the players' alternating choices and, vice-versa, given such a symbolic representation of a play and the initial configuration, all others can be reconstructed uniquely.

▶ **Lemma 21.** *There is a VPA $\mathcal{A}^{\mathsf{thr}}_\chi$ over Σ_χ of size $\mathcal{O}(|\chi|)$ which accepts exactly those (symbolic representations of) plays which contain a μ-thread.*

Proof. $\mathcal{A}^{\mathsf{thr}}_\chi$ stores the current head of a stack in its state, starting with χ. Its stack alphabet is $Sub(\chi) \times \{0,1\}$. Upon reading symbols from Σ^χ, it guesses which thread to follow in the play encoded in the input word and maintains its stack accordingly. The additional bit in its stack symbols is used to guess which positions are stair positions. It also guesses the outermost μ-variable X which will presumably be seen infinitely often in stair positions. Final states are those in which this variable is seen as the head of the stack it follows, when the automaton's internal stack shows that the current position is a stair position. $\mathcal{A}^{\mathsf{thr}}_\chi$ fails immediately, when it should pop a symbol from its internal stack which was marked to be pushed in a stair position, and when a ν-variable Y is seen in a stair position s.t. $X \prec_\chi Y$. ◀

Putting all this together gives an upper bound for vpFLC's satisfiability problem matching the doubly exponential lower bound from Cor. 11.

▶ **Corollary 22.** *The satisfiability problem for* vpFLC *is decidable in* 2EXPTIME.

Proof. Thms. 14, 15, Lemma 18 constitute an exponential reduction from general satisfiability games for vpFLC formulas to turn-based ones. Lemmas 19 and 20 show that the arena of these turn-based games is a visibly pushdown frame. Lemma 21 states that the winning condition for player $\mathbf{R}$, i.e. the complement of that of player $\mathbf{V}$ can be recognised by a VPA. The statement then follows from Cor. 6. ◀

5 Conclusion and Further Work

We have presented a decidable fragment of the otherwise undecidable modal fixpoint logic FLC. It makes use of visibly-pushdown principles in order to achieve decidability. However, these principles are built into the logic syntactically and mixed with modal operators unlike similar program logics like PDL[VPL] where the visibly-pushdown principles are separated from the modal part of the logic in the form of so-called (recursive) programs. Consequently,

a logic like PDL[VPL] can only make use of such visibly-pushdown principles in expressing properties of some or all paths. Unlike that, in vpFLC these principles can be used to form genuine branching properties. This extends the expressive power notably, making vpFLC a strict superlogic of both PDL[VPL] and the modal μ-calculus and thus pushing the decidability border amongst modal fixpoint logics further up.

The strictness of this extension from PDL[VPL] to vpFLC is a simple consequence of the fact that PDL[VPL] cannot express every $\mathcal{L}_\mu$-definable property. Hence, we have explicit witnesses for the strictness of this inclusion in the form of regular properties like the one mentioned in the introduction about winning two-player games and requiring an unbounded nesting of diamond- and box-formulas in a PDL-like logic. We believe that there are also non-regular properties separating vpFLC from PDL[VPL], for instance the $\langle a\rangle^n[c]\langle b\rangle^n$-property mentioned in the introduction. A detailed study classifying what properties are expressible in vpFLC but not even in PDL[CFL] is left for further research.

There is no formal separation of vpFLC from general FLC in expressive power other than the strong indication given by the decidability border between these two. We strongly believe that vpFLC is in fact less expressive than FLC, and that something like "*there is an n s.t. $\langle a\rangle^n\langle b\rangle\langle a\rangle^n p$ holds*" is not expressible in the former (while it is an easy exercise to formulate it in FLC). Again, in order to prove this formally, techniques need to be developed that better capture the expressive power of such expressive modal fixpoint logics. As a starting point, it seems not to be too difficult to separate vpFLC from FLC over the class of infinite words where the former may only be able to express visibly-pushdown word languages, while the latter can express all properties from the Boolean closure of ω-CFLs [18].

It is possible, albeit technically tedious, to encode multiple stacks in one when they are k-aligned for some $k > 1$. This would allow the syntax of vpFLC to be a bit more relaxed, enabling more formulas to be specified, without breaking the general proof structure. For instance, it may be possible to include τ in the syntax of vpFLC as it can be useful to specify particular properties like "*there is an $n \geq 0$ s.t. $\langle a\rangle^n[b]^n p$ holds*" via $(\mu X.\tau \vee \langle a\rangle; X; [b]); p$. There is no reason why the expressibility of such properties should break decidability, but the free use of τ would require a stronger version of Lemma 19 for the decidability proof to still be valid.

References

1. E. Alsmann, F. Bruse, and M. Lange. Separating the expressive power of propositional dynamic and modal fixpoint logics, 2021. Submitted.
2. R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proc. 36th Annual ACM Symp. on Theory of Computing, STOC'04*, pages 202–211. ACM, 2004. doi:10.1145/1007352.1007390.
3. J. Baran and H. Barringer. A grammatical representation of visibly pushdown languages. In *Proc. 14th Int. Workshop on Logic, Language, Information and Computation, WoLLIC'07*, volume 4576 of *LNCS*, pages 1–11. Springer, 2007. doi:10.1007/978-3-540-73445-1_1.
4. C. S. Calude, S. Jain, B. Khoussainov, W. Li, and F. Stephan. Deciding parity games in quasipolynomial time. In *Proc. 49th Annual ACM SIGACT Symp. on Theory of Computing, STOC'17*, pages 252–263. ACM, 2017. doi:10.1145/3055399.3055409.
5. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1991. IEEE. doi:10.1109/sfcs.1991.185392.
6. E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 2000. doi:10.1137/s0097539793304741.
7. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979. doi:10.1016/0022-0000(79)90046-1.

8 O. Friedmann, F. Klaedtke, and M. Lange. Ramsey-based inclusion checking for visibly pushdown automata. *ACM Trans. Comput. Log.*, 16(4):34, 2015. doi:10.1145/2774221.

9 O. Friedmann and M. Lange. The modal μ-calculus caught off guard. In *20th Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX'11*, volume 6793 of *LNCS*, pages 149–163. Springer, 2011. doi:10.1007/978-3-642-22119-4_13.

10 D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, 1983. doi:10.1016/0022-0000(83)90014-4.

11 D. Harel and D. Raz. Deciding properties of nonregular programs (preliminary version). In *Proc. 31st Annual Symp. on Foundations of Computer Science, FOCS'90*, volume II, pages 652–661, St. Louis, Missouri, 1990. IEEE. doi:10.1109/fscs.1990.89587.

12 D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In *CONCUR*, pages 263–277, 1996. doi:10.1007/3-540-61604-7_60.

13 M. Jurdziński. Small progress measures for solving parity games. In H. Reichel and S. Tison, editors, *Proc. 17th Ann. Symp. on Theoretical Aspects of Computer Science, STACS'00*, volume 1770 of *LNCS*, pages 290–301. Springer, 2000. doi:10.1007/3-540-46541-3_24.

14 T. Koren and A. Pnueli. There exist decidable context free propositional dynamic logics. In E. Clarke and D. Kozen, editors, *Proc. of the Workshop on Logics of Programs*, volume 164 of *LNCS*, pages 290–312. Springer, 1983. doi:10.1007/3-540-12896-4_369.

15 D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983. doi:10.1007/BFb0012782.

16 D. Kozen and R. Parikh. A decision procedure for the propositional μ-calculus. In *Proc. Workshop on Logics of Programs*, volume 164 of *LNCS*, pages 313–325. Springer, 1983. doi:10.1007/3-540-12896-4_370.

17 M. Lange. Local model checking games for fixed point logic with chop. In *Proc. 13th Conf. on Concurrency Theory, CONCUR'02*, volume 2421 of *LNCS*, pages 240–254. Springer, 2002. doi:10.1007/3-540-45694-5_17.

18 M. Lange. Temporal logics beyond regularity, 2007. Habilitation thesis, University of Munich, BRICS research report RS-07-13. doi:10.7146/brics.v14i13.22178.

19 M. Lange and R. Somla. Propositional dynamic logic of context-free programs and fixpoint logic with chop. *Information Processing Letters*, 100(2):72–75, 2006. doi:10.1016/j.ipl.2006.04.019.

20 C. Löding, C. Lutz, and O. Serre. Propositional dynamic logic with recursive programs. *J. Log. Algebr. Program*, 73(1-2):51–69, 2007. doi:10.1016/j.jlap.2006.11.003.

21 C. Löding, P. Madhusudan, and O. Serre. Visibly pushdown games. In *Proc. 24th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'04*, volume 3328 of *LNCS*, pages 408–420. Springer, 2004. doi:10.1007/978-3-540-30538-5_34.

22 A.W. Mostowski. Games with forbidden positions. Technical Report 78, Uniwersytet Gdański. Instytut Matematyki, 1991.

23 M. Müller-Olm. A modal fixpoint logic with chop. In *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, volume 1563 of *LNCS*, pages 510–520. Springer, 1999. doi:10.1007/3-540-49116-3_48.

24 D. Niwiński and I. Walukiewicz. Games for the μ-calulus. *TCS*, 163:99–116, 1997. doi:10.1016/0304-3975(95)00136-0.

25 R. S. Streett. Propositional dynamic logic of looping and converse. In *Proc. 13th Symp. on Theory of Computation, STOC'81*, pages 375–383. ACM, 1981. doi:10.1145/800076.802492.

26 R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81(3):249–264, 1989. doi:10.1016/0890-5401(89)90031-x.

27 A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955. doi:10.2140/pjm.1955.5.285.

A PDL over Recursive Programs

We give a brief overview of PDL[VPL]. The underlying action set is a visibly pushdown alphabet $\Sigma = \Sigma_{\mathsf{int}} \uplus \Sigma_{\mathsf{call}} \uplus \Sigma_{\mathsf{ret}}$. Formulas are built as in modal logic, with modalities over VPLs:

$$\varphi \;::=\; q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \langle L \rangle \varphi$$

where $q \in \mathcal{P}$ and L is a (finite representation, for instance in the form of a VPA, of) a VPL. By introducing $\vee$ and $[\cdot]\cdot$ as primitives, formulas can also be given in negation normal form.

Formulas of PDL[VPL] are interpreted over states s of an LTS $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$ in the following way. First we extend the transition relation $\to \subseteq \mathcal{S} \times \Sigma \times \mathcal{S}$ to a path relation $\to \subseteq \mathcal{S} \times \Sigma^* \times \mathcal{S}$ connecting states by words via

- $s \xrightarrow{\varepsilon} s$ for any $s \in \mathcal{S}$, and
- $s \xrightarrow{aw} t$ if there is $u \in \mathcal{S}$ s.t. $s \xrightarrow{a} u$ and $u \xrightarrow{w} t$, for $s, t \in \mathcal{S}$.

Then the semantics can be given as follows.

$$\begin{aligned}
\mathcal{T}, s \models q \quad &\text{iff} \quad q \in \lambda(s) \\
\mathcal{T}, s \models \varphi \wedge \psi \quad &\text{iff} \quad \mathcal{T}, s \models \varphi \text{ and } \mathcal{T}, s \models \psi \\
\mathcal{T}, s \models \neg\varphi \quad &\text{iff} \quad \mathcal{T}, s \not\models \varphi \\
\mathcal{T}, s \models \langle L \rangle \varphi \quad &\text{iff} \quad \text{there is } t \in \mathcal{S} \text{ s.t. } s \xrightarrow{w} t \text{ for some } w \in L \text{ and } \mathcal{T}, t \models \varphi
\end{aligned}$$

PDL[VPL] can then be used to formalise nested visibly-pushdown path properties like $[(L_1)^*]\langle L_1 \rangle \mathtt{tt}$ with $L_1 = \{a^n b^n \mid n \geq 1\}$ as mentioned in the introduction. This formula requires every state that is reachable under a (possibly empty) sequence of words from L_1 to be the source of a path with labels from L_1 again. In particular, it implies (but is not equivalent to) the existence of an infinite path with labels of the form $a^{n_1} b^{n_1} a^{n_2} b^{n_2} \ldots$ for some $n_i \geq i$, $i = 1, \ldots$

B Model Checking Games for FLC

We briefly present the essentials of the model checking games for FLC defined in [17] used in the proofs of Thms. 14 and 15.

Given an LTS $\mathcal{T} = (\mathcal{S}, \to, s_0, \lambda)$ and an FLC-formula χ, the model checking game $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}}(\chi)$ is played by players **V** and **R** on configurations of the form $s \vdash \gamma$ where γ is a *stack* of subformulas of χ just like those used in the satisfiability games. The game starts in the initial configuration $s_0 \vdash \chi; \mathtt{tt}$. The rules are presented in Fig. 4.

Player **V** wins a finite play $s_0 \vdash \varphi_0; \gamma_0,\ s_1 \vdash \varphi_1; \gamma_1,\ \ldots,\ s_n \vdash \varphi_n; \gamma_n$ if

- $\varphi_n = \mathtt{tt}$,
- $\varphi_n = q$ for some $q \in \mathcal{P}$ and $s_n \in \lambda(q)$,
- $\varphi_n = \overline{q}$ for some $q \in \mathcal{P}$ and $s_n \notin \lambda(q)$,
- $\varphi_n = [a]$ for some $a \in \Sigma$ and there is no t s.t. $s_n \xrightarrow{a} t$.

Player **R** wins a finite play $s_0 \vdash \varphi_0; \gamma_0,\ s_1 \vdash \varphi_1; \gamma_1,\ \ldots,\ s_n \vdash \varphi_n; \gamma_n$ if

- $\varphi_n = \mathtt{ff}$,
- $\varphi_n = q$ for some $q \in \mathcal{P}$ and $s_n \notin \lambda(q)$,
- $\varphi_n = \overline{q}$ for some $q \in \mathcal{P}$ and $s_n \in \lambda(q)$,
- $\varphi_n = \langle a \rangle$ for some $a \in \Sigma$ and there is no t s.t. $s_n \xrightarrow{a} t$.

$$\frac{s \vdash (\varphi_0 \vee \varphi_1); \gamma}{s \vdash \varphi_i; \gamma}\ \mathbf{V} : i \in \{0,1\} \qquad \frac{s \vdash (\varphi_0 \wedge \varphi_1); \gamma}{s \vdash \varphi_i; \gamma}\ \mathbf{R} : i \in \{0,1\} \qquad \frac{s \vdash (\varphi_0; \varphi_1); \gamma}{s \vdash \varphi_0; \varphi_1; \gamma}$$

$$\frac{s \vdash \tau; \gamma}{s \vdash \gamma} \qquad \frac{s \vdash \sigma X.\varphi; \gamma}{s \vdash X; \gamma} \qquad \frac{s \vdash X; \gamma}{s \vdash \varphi; \gamma}\ \text{if } fp_\chi(X) = \sigma X.\varphi$$

$$\frac{s \vdash \langle a \rangle; \gamma}{t \vdash \gamma}\ \mathbf{V} : s \xrightarrow{a} t \qquad \frac{s \vdash [a]; \gamma}{t \vdash \gamma}\ \mathbf{R} : s \xrightarrow{a} t$$

Figure 4 The rules of the FLC model checking games.

Additionally, the winner of an infinite play $s_0 \vdash \varphi_0; \gamma_0,\ s_1 \vdash \varphi_1; \gamma_1,\ \ldots$ is determined by the outermost variable X occurring infinitely often amongst the sequence $(\varphi_{i_j})_{j \geq 0}$ where $i_0, i_1, \ldots$ is the sequence of stair positions of the sequence of stacks $\gamma_0, \gamma_1, \ldots$[1] If this variable X is of fixpoint type ν, then $\mathbf{V}$ wins this play. Otherwise, if it is type μ, $\mathbf{R}$ wins the play.

▶ **Proposition 23** ([17]). *Let $\mathcal{T}$ be an LTS and χ be a closed* FLC *formula. Player $\mathbf{V}$ wins $\mathcal{G}^{\mathsf{mc}}_{\mathcal{T}}(\chi)$ iff $\mathcal{T} \models \chi$.*

[1] In [17], this variable is called *stack-increasing*. The terminology of stair positions was coined later in [21].