Separating the Expressive Power of Propositional Dynamic and Modal Fixpoint Logics

Eric Alsmann Florian Bruse Martin Lange
School of Electrical Engineering and Computer Science
University of Kassel, Germany
eric.alsmann@student.uni-kassel.de florian.bruse@uni-kassel.de martin.lange@uni-kassel.de

We investigate the expressive power of the two main kinds of program logics for complex, non-regular program properties found in the literature: those extending propositional dynamic logic (PDL), and those extending the modal μ-calculus. This is inspired by the recent discovery of a decidable program logic called *Visibly Pushdown Fixpoint Logic with Chop* which extends both the modal μ-calculus and PDL over visibly pushdown languages, which, so far, constituted the ends of two pillars of decidable program logics.

Here we show that this logic is not only more expressive than either of its two fragments, but in fact even more expressive than their union. Hence, the decidability border amongst program logics has been properly pushed up. We complete the picture by providing results separating all the PDL-based and modal fixpoint logics with regular, visibly pushdown and arbitrary context-free constructions.

1 Introduction

Program Logics. Modal logics play a major role in formal program specification and verification, least because modal logics are tightly linked to the notion of bisimulation-invariance which is deemed to be *the* notion of behavioural equivalence for state-based programs, resp. systems dynamically evolving over time.

Basic modal logic is inadequate for formal program specification since it lacks the ability to even express the simplest forms of program correctness like safety ("*no bad state can be reached*"), termination ("*every run finally ends*"), etc. This is of course due to the fact that a basic modal formula of modal depth k can only "see" the next k levels of successors in a transition system.

This shortcoming has led to the design and study of several extensions of basic modal logic for formal specification and verification. Ultimately, the key to retaining bisimulation-invariance but increasing the expressive power to include typical (un-)desired program properties is the addition of fixpoint constructs. These come in one of two forms: either as explicit fixpoint quantifiers – the most prominent example of such an extension being the modal μ-calculus $\mathscr{L}_\mu$ [12] – or implicitly in the form of temporal operators – the most prominent examples here being temporal logics like LTL [21] and CTL [9].

Another form of implicit fixpoint operator(s) in an extension of basic modal logic is found in Propositional Dynamic Logic (PDL). This comprises, in fact, a family of formalisms, parametrised by generalisations of the accessibility relation in a labelled transition system (LTS), represented as a class of formal languages. Not surprisingly, the most prominent example of this family is PDL[REG], typically just called PDL, which can be seen as basic modal logic over an Kleene algebra of accessibility relations in an LTS [10].

The expressiveness of the logics mentioned so far is well-understood. Most notably, they are all incomparable in expressiveness, and all of them can be embedded into $\mathscr{L}_\mu$ which is therefore strictly

V. Castiglioni and O. Dardha (Eds.): Combined Workshop on Expressiveness in Concurrency and Structural Operational Semantics (EXPRESS/SOS 2021)
EPTCS 339, 2021, pp. 10–26, doi:10.4204/EPTCS.339.4

more expressive than any of them. Example properties witnessing the strictness are also well-known, and their inexpressibility in one of these logics is typically not difficult to prove formally:

- PDL can only combine eventuality properties with existential path quantification; hence, it cannot express the CTL-property $\mathsf{AF}q$ stating "*q holds on all paths at some point*" (which is also expressible in LTL);
- LTL can only quantify over all paths on the top-level, hence it cannot state $\mathsf{EX}q \wedge \mathsf{EX}\neg q$ stating "*there is a successor satisfying q and one that does not satisfy q*" (which is also expressible in PDL as $\langle - \rangle q \wedge \langle - \rangle \neg q$;
- CTL cannot state "*q holds only finitely often on all paths*" which is possible in LTL, and it cannot say "*q holds after every even number of steps*" which is expressed by the PDL formula $[(\Sigma\Sigma)^*]q$ over LTS with edge labels from Σ.

Non-Regular Program Logics. The fact that $\mathscr{L}_\mu$ embeds them all means that their expressiveness is limited by *regularity* in the sense that each property definable in these logics can also be specified by a finite tree automaton or a (bisimulation-invariant) formula of Monadic Second-Order Logic. While regular expressiveness is sufficient for many program specification and verification tasks in the form of safety, liveness, fairness properties, there are situations in formal verification where expressiveness beyond regularity is required. This has led to the design of specification logics beyond $\mathscr{L}_\mu$ or PDL.

- A non-regular PDL-like specification logic is easily obtained by extending the Kleene algebra of accessibility relations, resp. the class of regular languages in modal operators, to larger language classes like the context-free ones, resulting in PDL[CFL] [11]. It can state properties like "*there is a path of the form $a^n b^n$ for some $n \geq 1$*".
- Fixpoint Logic with Chop (FLC) [19] extends $\mathscr{L}_\mu$ with an operator for sequential composition. This enables it to express context-free properties like the only mentioned above for PDL[CFL] but also other non-regular ones like "*all paths end after the same number of steps*." In fact, FLC embeds PDL[CFL] [15].
- Assume-guarantee properties like "*every execution of program P by n steps, can be matched by an execution of program Q by $n+1$ steps*" are not-regular and have led to invention of *Higher-Order Fixpoint Logic* (HFL) [25], an extension of $\mathscr{L}_\mu$ by a typed λ-calculus. This captures FLC on a very low type level and stretches far beyond that.

A common feature of such extensions – at least when not done carefully – is the loss of decidability of the satisfiability problem: PDL[CFL] is highly undecidable [11], and this transfers to FLC and HFL. On the other hand, their model checking problems over finite LTS remains decidable, making them suitable for automatic program verification. For PDL[CFL] it is even polynomial [13], whereas for FLC is it EXPTIME-complete [14], and for general HFL it is non-elementary [4].

Decidable Non-Regular Program Logics. A discovery in formal language theory has opened up some interesting possibilities in the realm of non-regular program logics: the class of *visibly pushdown languages* (VPL) over some visibly pushdown alphabet partitioning alphabet symbols into those that cause push-, pop- and internal state changes in a corresponding pushdown automaton, forms a subset of CFL that enjoys almost the same closure and decidability properties as the class REG [2]. This is even robust in the sense that the corresponding PDL over this class, PDL[VPL], is a genuinely non-regular program logic whose satisfiability problem is actually decidable, namely 2EXPTIME-complete [16].

This comprised the state-of-the-art until recently: the largest (w.r.t. expressiveness) explicit-fixpoint logic that was still known to be decidable was $\mathscr{L}_\mu$, as known extensions thereof were undecidable. Regarding implicit-fixpoint logics in the form of propositional dynamic ones, it was possible to push the decidability frontier genuinely into the realm of non-regular logics by the definition of PDL[VPL]. A similar construction was possible for a temporal logic called *Recursive* CTL (RecCTL) [5] which can be seen as a CTL-like variant of FLC: it was possible to define a (non-regular) fragment which could be shown to be decidable by a simple satisfiability-preserving translation into PDL[VPL] [5].

A common feature of these decidable logics is the modular use of visibly-pushdown effects. This is best seen in PDL[VPL] where VPL can either be used inside an existential or a universal modality, but no mixing of modalities across visibly pushdown languages is possible. Hence, PDL[VPL] can express properties like $\langle a^n b^n \rangle p$, but cannot state "*there is an a-path of length n for some n, such that all following b-paths of length n for the same n end in a state satisfying p*" because this requires a change of modality *within* the VPL $a^n b^n$. We will write $\langle a^n \rangle [b^n]$ to denote this property succinctly, even though this is no well-formed formula of any (PDL-like) logic.

Since decidability of VPL-based logics can be obtained by a reduction to visibly pushdown games [17], there is little reason to believe that the strict use of VPLwithin a single modality is actually required for decidability. In fact, very recently a fragment of FLC, called vpFLC, has been constructed which allows free use of modality changes within VPLs, and whose satisfiability problem is still decidable [6].

Contribution. The aim of this paper is to investigate the expressive power of program logics, specifically those around the decidability border, and to show that the existing inclusions between them are strict. This is already known for a few of them, either because of long-standing results like the strict inclusions between regular logics and their non-regular extensions, or because of rather trivial observations. Note for instance that it has been known for a long time that PDL[CFL] and $\mathscr{L}_\mu$ are incomparable w.r.t. expressiveness. Hence, FLC, which subsumes them both, must subsume both of them strictly. This is fair enough, but also slightly non-satisfactory, when one considers examples witnessing the strictness that result from this kind of reasoning.

- PDL[CFL] is designed to express some non-regular properties, whereas $\mathscr{L}_\mu$ can only express regular ones. Hence, any genuinely non-regular property in FLC witnesses the strictness of the inclusion of $\mathscr{L}_\mu$ in FLC.
- On the other hand, PDL-based logics cannot express properties that involve unbounded modality alternation like $(\langle a \rangle [b])^n := \mu X.p \vee \langle a \rangle [b] X$ (defining the winning region for one of the players in a turn-based two-player game). Hence, any such formula also witnesses the strict inclusion of PDL[CFL] in FLC. This is not very satisfactory, as this does not shed light onto the true non-regular difference of these genuinely non-regular program logics.

In this paper we complete the study into the (non-expressiveness) of properties expressible in program logics around the decidability border. The hierarchy formed by them is shown in Fig. 1, naming specific properties witnessing the strictness of the corresponding inclusion as well as pointing to their origin. The picture is made complete by results in this paper, providing new separation results as well as tighter separation results in the form of witnessing properties which genuinely do not rely on fairly trivial inexpressibility results between the regular and non-regular world. To denote these properties we use a fairly intuitive notation like $\langle a^n b^n \rangle$ etc. In the end, we can see that all the inclusions in Fig. 1 are strict. In particular,

- ... propositional dynamic logics and modal fixpoint logics can be separated using using properties of unbounded modality alternation (e.g. $(\langle a \rangle [b])^n$) or, further up in the hierarchy, those in which

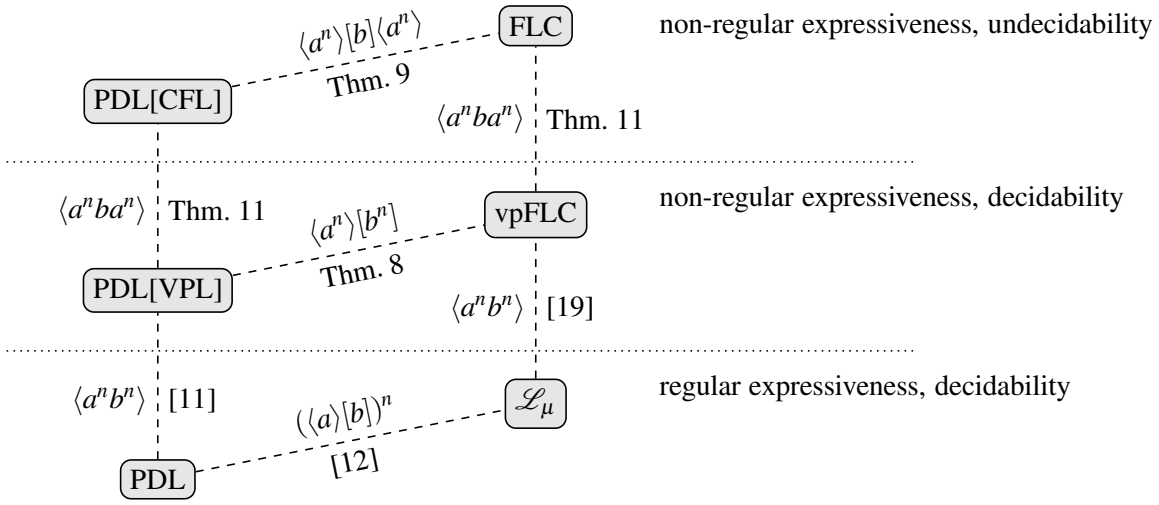

Figure 1: The (strict) hierarchy of propositional dynamic and modal fixpoint logics betwen PDL and FLC.

modal alternation and limited counting is intertwined (e.g. $\langle a^n\rangle[b^n]$ or $\langle a^n\rangle[b]\langle a^n\rangle$). As a result, vpFLC is not only strictly more expressive than PDL[VPL] and $\mathscr{L}_\mu$ (which was known before), but even strictly more expressive than their union.

- ... the visibly pushdown based logics (middle band) can be separated from the general non-regular ones (top band) using properties based on visibly pushdown languages. This may not sound surprising but is not a triviality as program properties using non-visibly-pushdown languages could, in theory, be composable by complex formulas using visibly pushdown languages only. Here we show that this is indeed not the case.

Organisation. In Sect. 2 we recall necessary preliminaries from formal languages, propositional dynamic and modal fixpoint logics. In Sect. 3 we provide the missing results that separate the propositional dynamic logics from the modal fixpoint logics, i.e. the left column from the right column in Fig. 1. This concerns the non-regular parts, as the separation of $\mathscr{L}_\mu$ from PDL is well-known.

In Sect. 4 we then show how to leverage the undecidability of the satisfiability problem for such logics into an expressiveness gap, thus separating the top row from the middle row in Fig. 1. Again, the separation of the middle row from the bottom row was known already, with proofs relying for instance on the finite model properties of the regular logics PDL and $\mathscr{L}_\mu$. We conclude with remarks on further work in Sect.

2 Preliminaries

2.1 Languages

An *alphabet* is a finite, nonempty set of *letters*. A *word* over some alphabet Σ is a finite sequence $w = w_1 \cdots w_n$ of letters from Σ. The empty word is denoted by ε. The length $|w|$ of $w = w_1 \cdots w_n$ is n.

The set of all Σ-words is denoted by Σ^*, the set of all non-empty words is denoted by Σ^+. A Σ-*language* is a subset of Σ^*. We denote concatenation of words and languages by juxtaposition as usual.

A *visibly pushdown alphabet* is an alphabet Σ that is partitioned into three sets $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$ of *call*, *return* and *internal* symbols.

Finite Automata. A (nondeterministic) *finite automaton* (NFA) is a tuple $(Q, \Sigma, \delta, q_I, F)$ where Q is a finite, nonempty set of states, Σ is an alphabet, $q_I \in Q$ is the *initial state*, $F \subseteq Q$ is the set of accepting states and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

A *run* of some NFA $\mathscr{A}$ on a word $w = a_1 \cdots a_n \in \Sigma^*$ is a sequence $q_0, \ldots, q_n \in Q^*$ such that, for all $0 \leq i < n$, we have that $(q_i, a_i, q_{i+1}) \in \delta$. We also say that $\mathscr{A}$ has a run from q_0 to q_n over w. Such a run is called accepting if $q_0 = q_I$ and $q_n \in F$. A word w is accepted by $\mathscr{A}$ if there is an accepting run of $\mathscr{A}$ on w. We write $L(\mathscr{A})$ for the set of words accepted by $\mathscr{A}$. A finite automaton $(Q, \Sigma, \delta, q_i, F)$ is called *deterministic*, if for all $q \in Q, a \in \Sigma$, there is exactly one $q' \in Q$ such that $(q, a, q') \in \delta$. It is well-known that, for each NFA, there is a DFA that accepts exactly the same language, and has at most exponentially more states. A language is called *regular* if there is a finite automaton that accepts it. We write REG for the class of regular languages (over a given alphabet).

Pushdown Automata. A *pushdown automaton* is a tuple $(Q, \Sigma, \Gamma, \bot, \delta, q_I, F)$ where Q, Σ, q_I, F are as in the case of finite automata, Γ is the *stack alphabet* such that $\Gamma \cap \Sigma = \emptyset$, $\bot \notin \Gamma \cup \Sigma$ is the stack bottom symbol, $\delta \subseteq (Q \times \Sigma \times \Gamma \times \Gamma^* \times Q) \cup (Q \times \Sigma \times \{\bot\} \times \Gamma^* \times Q)$ is the transition relation.

A *run* of some pushdown automaton (PDA) $\mathscr{A}$ on a word $w = a_1 \cdots a_n \in \Sigma^*$ is a finite sequence $(q_0, \gamma_0), \ldots, (q_n, \gamma_n) \in (Q \times \Gamma^*)^*$ of states and stack contents such that, for all $0 \leq i < n$ either $\gamma_i = \varepsilon$ and $(q_i, a_i, \bot, \gamma_{i+1}, q_{i+1}) \in \delta$ or $\gamma_i = \gamma_i' \gamma'$ with $|\gamma'| = 1$ and $\gamma_{i+1} = \gamma_i' \gamma''$ and $(q_i, a_i, \gamma', \gamma'', q_{i+1}) \in \delta$. A run is *accepting* if $q_0 = q_I, \gamma_0 = \varepsilon$ and $q_n \in F$. A word w is accepted by $\mathscr{A}$ if there is an accepting run of $\mathscr{A}$ on w. We write $L(\mathscr{A})$ for the language of words accepted by $\mathscr{A}$. A language is called *context-free* if it is accepted by some PDA. We write CFL for the class of context-free languages. Note that every NFA can be extended to a PDA by ignoring the stack, hence every regular language is context-free. Also, context-free languages over a unary (size 1) alphabet are known to be regular [20].

A *visibly pushdown automaton* (VPA) [18, 2] is a pushdown automaton such that its alphabet Σ is a visibly pushdown alphabet and, moreover, for $(q, a, \gamma, \gamma', q') \in \delta$, we have that either

- $a \in \Sigma_c$ and $|\gamma'| = 2$,
- $a \in \Sigma_r$, $\gamma \neq \bot$, and $|\gamma'| = 0$,
- $a \in \Sigma_i$ and $|\gamma'| = 1$,

and, finally, a run is only accepting if the final configuration has an empty stack. A language over some visibly pushdown alphabet is called *visibly pushdown* if it is accepted by some VPA. We write VPL for the class of visibly pushdown languages (over a given visibly pushdown alphabet), Clearly every visibly pushdown language is context-free. Moreover, every regular language is visibly-pushdown over the alphabet that regards all letters as internal.

Note that this definition introduces visibly pushdown languages without so-called pending calls and returns, i.e. such that only words are accepted that are balanced and well-nested w.r.t. symbols from Σ_c and Σ_r. This is done to make the definition of vpFLC (see Sec. 2.3 below) more accessible.

Derivatives. Let L be a Σ-language and let $a \in \Sigma$. The a-derivative of L is the set $\Delta_a(L) := \{w \in \Sigma^* \mid aw \in L\}$ [22]. It is easy to see that derivatives of regular languages are regular again by manipulating the

associated deterministic finite automaton, i.e. making the a-successor of the initial state the new initial state, cf. [8].

We now argue that the a-derivative of a context-free language is context-free again.

Lemma 1. *Let $L \in$ CFL over some alphabet Σ and $a \in \Sigma$. Then $\Delta_a(L) \in$ CFL.*

Proof. Suppose $\mathscr{A} = (Q, \Sigma, \Gamma, \bot, \delta, q_I, F)$ is a PDA with $L(\mathscr{A}) = L$. Let $T = \{(q_i, a, \bot, \gamma, q) \in \delta \mid \gamma \in \Gamma^*, q \in Q\}$ be the set of transitions available to $\mathscr{A}$ upon reaching an a as the first letter of a word. For each $t = (q_i, a, \bot, \gamma, q) \in T$, if $\gamma = \varepsilon$, define the automaton $\mathscr{A}_t = (Q \cup \{q'\}, \Sigma, \Gamma, \bot, \delta_t, q', F)$ with

$$\delta_t = \delta \cup \{(q', b, \bot, \gamma', q'') \mid (q', b, \bot, \gamma', q'') \in \delta\}$$

and if $\gamma = \gamma'\gamma''$ with $|\gamma''| = 1$, define the automaton $\mathscr{A}_t = (Q \cup \{q'\}, \Sigma, \Gamma, \bot, \delta_t, q', F)$ with

$$\delta_t = \delta \cup \{(q', b, \bot, \gamma'\gamma''', q'') \mid (q', b, \gamma'', \gamma''', q'') \in \delta\}.$$

Intuitively, $\mathscr{A}_t$ simulates the case of $\mathscr{A}$ reading an a and taking t as its first transition, and then proceeds like $\mathscr{A}$ would on the rest of the word. Hence, $\mathscr{A}_t$ accepts only words in $\Delta_a(L(\mathscr{A}))$.

Conversely, each word in $\Delta_a(L(\mathscr{A}))$ is accepted by at least one of the $\mathscr{A}_t$. Using the well-known closure of context-free languages under union, we obtain the desired statement. □

2.2 Propositional Dynamic Logic

Labelled Transition Systems. Let $\mathscr{P}$ be a set of atomic propositions and let Σ be an alphabet, in the context of propositional dynamic logics often referred to as the set of *atomic programs*.

A *labelled transition system* (LTS) is a tuple $\mathscr{T} = (S, \to, \ell)$ where S is a, potentially infinite, set of *states*, $\to \subseteq S \times \Sigma \times S$ is the transition relation, and $\ell \colon S \to 2^{\mathscr{P}}$ labels each state with the set of propositions that hold at it. Given an LTS, we write $s \xrightarrow{a} t$ to denote that $(s, a, t) \in \to$. This extends to Σ-words via $s \xrightarrow{w_1 w_2} t$ iff there is s' with $s \xrightarrow{w_1} s'$ and $s' \xrightarrow{w_2} t$, and $s \xrightarrow{\varepsilon} t$ iff $s = t$. We write $s \xrightarrow{L} t$ if there is $w \in L$ such that $s \xrightarrow{w} t$. Sometimes we consider labelled transition systems with a designated initial state. In a drawing, this will be marked by an ingoing edge with no source.

A (finite) path π in an LTS is a sequence $s_1, \ldots, s_n$ of states such that, for all $1 \leq i < n$, there is $a_i \in \Sigma$ such that $s_i \xrightarrow{a_i} s_{i+1}$. In this case, the path is said to be labelled by $w = a_1 \cdots a_{n-1}$. Note that $s \xrightarrow{w} t$ iff there is a w-labelled path from s to t.

Syntax. Let $\mathscr{P}$ and Σ be as before. Let C be a, not necessarily finite, set of Σ-languages. The syntax of Propositional Dynamic Logic over C (PDL[C]) is defined by the following grammar:

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle L \rangle \varphi \mid [L]\varphi$$

where $p \in \mathscr{P}$ and $L \in C$. The auxiliary formulas $\mathtt{tt}$ and $\mathtt{ff}$ are defined as usual via $p \vee \neg p$, resp. $p \wedge \neg p$ for arbitrary $p \in \mathscr{P}$. We are particularly interested in those logics in which $C =$ REG, VPL or CFL, resulting in the logics

- *Propositional Dynamic Logic of Regular Programs* (PDL[REG]) [10], sometimes only called PDL,
- *Propositional Dynamic Logic of Context-Free Programs*[1] (PDL[CFL]) [11], and

[1] Originally it was called *Propositional Dynamic Logic of Non-Regular Programs* which is of course slightly misleading as it hardly comprises *all* non-regular programs.

- *Propositional Dynamic Logic of Recursive Programs* (PDL[VPL]) [16].

The *modal depth* $md(\varphi)$ of a formula φ measures the nesting depth of modal operators and is defined inductively via

$$\begin{aligned} md(p) &= 0 \\ md(\langle L\rangle\varphi) = md([L]\varphi) = md(\neg\varphi) &= 1 + md(\varphi) \\ md(\varphi_1 \vee \varphi_2) = md(\varphi_1 \wedge \varphi_2) &= \max(md(\varphi_1), md(\varphi_2)) \end{aligned}$$

Semantics. Given an LTS $\mathscr{T} = (S, \to, \ell)$, any PDL[$C$]-formula over matching $\mathscr{P}$ and Σ defines a subset $[\![\varphi]\!]^{\mathscr{T}}$ of S, inductively defined via

$$\begin{aligned} [\![p]\!]^{\mathscr{T}} &= \{s \in S \mid p \in \ell(s)\} & [\![\neg\varphi]\!]^{\mathscr{T}} &= S \setminus [\![\varphi]\!]^{\mathscr{T}} \\ [\![\varphi_1 \vee \varphi_2]\!]^{\mathscr{T}} &= [\![\varphi_1]\!]^{\mathscr{T}} \cup [\![\varphi_2]\!]^{\mathscr{T}} & [\![\varphi_1 \wedge \varphi_2]\!]^{\mathscr{T}} &= [\![\varphi_1]\!]^{\mathscr{T}} \cup [\![\varphi_2]\!]^{\mathscr{T}} \\ [\![\langle L\rangle\varphi]\!]^{\mathscr{T}} &= \{s \mid \text{ ex. } t \in [\![\varphi]\!]^{\mathscr{T}}, \text{ s.t. } s \xrightarrow{L} t\} & [\![[L]\varphi]\!]^{\mathscr{T}} &= \{s \mid \text{ f.a. } t \text{ s.t. } s \xrightarrow{L} t, \text{ then } t \in [\![\varphi]\!]^{\mathscr{T}}\}. \end{aligned}$$

We write $\mathscr{T}, s \models \varphi$ to denote that $s \in [\![\varphi]\!]^{\mathscr{T}}$. We say that $\mathscr{T}$ is a model of φ to denote that the initial state of $\mathscr{T}$ is in $[\![\varphi]\!]^{\mathscr{T}}$.

2.3 Fixpoint Logic with Chop

We assume some familiarity with the modal μ-calculus $\mathscr{L}_\mu$. The yardstick in the world of modal fixpoint logics to measure the expressive power of propositional dynamic logics against has turned out to be Fixpoint Logic with Chop [19], the extension of $\mathscr{L}_\mu$ by a sequential composition operator. This is since some kind of sequential composition operator is needed in many language classes beyond the regular languages, and FLC is, in some sense, a minimal extension of $\mathscr{L}_\mu$ by such an operator. As FLC is by far less known than $\mathscr{L}_\mu$, we include its definition here. It is also needed to explain its fragment vpFLC, comprising the largest currently known modal fixpoint logic with a decidable satisfiability problem [6].

Syntax. Let $\mathscr{P}$ and Σ be as above. Let $\mathscr{V}$ be a countably infinite set of variable names. Formulas of FLC over $\mathscr{P}$, Σ and $\mathscr{V}$ 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 $q \in \mathscr{P}$, $a \in \Sigma$ and $X \in \mathscr{V}$. Note that FLC does not have a negation operator.

Semantics. Let $\mathscr{T} = (S, \to, \ell)$ be an LTS. An *environment* $\eta : \mathscr{V} \to (2^S \to 2^S)$ assigns to each variable a function from sets of states to sets of states in $\mathscr{T}$. We write $\eta[X \mapsto f]$ for the function that maps X to f and agrees with η on all other arguments.

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

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

form a complete lattice with joins $\sqcup$ and meets $\sqcap$ – defined as the pointwise union, resp. intersection. By the Knaster-Tarski Theorem [23] the least and greatest fixpoints of monotone functionals $F : (2^S \to$

$2^S) \to (2^S \to 2^S)$ exist. They are used to interpret fixpoint formulas of FLC. The semantics is then inductively defined as follows.

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

Here, $\circ$ denotes function composition.

For any FLC formula φ, any LTS $\mathscr{T} = (S, \to, \ell)$ with initial state s_0 and any environment η let $\|\varphi\|^{\eta}_{\mathscr{T}} := [\![\varphi]\!]^{\eta}_{\mathscr{T}}(S)$. We call this the set of positions in t *defined* by φ and η. We also write $\mathscr{T}, s \models_{\eta} \varphi$ if $s \in \|\varphi\|^{\eta}_{\mathscr{T}}$, resp. $\mathscr{T} \models_{\eta} \varphi$ if $\mathscr{T}, s_0 \models_{\eta} \varphi$. If φ is closed we may omit η in both kinds of notation. We say that $\mathscr{T}$ is a *model* of a closed formula φ if $\mathscr{T} \models \varphi$. 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 η and every LTS $\mathscr{T}$: $[\![\varphi]\!]^{\eta}_{\mathscr{T}} = [\![\psi]\!]^{\eta}_{\mathscr{T}}$. 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 η and every $\mathscr{T}$ we have $\|\varphi\|^{\eta}_{\mathscr{T}} = \|\psi\|^{\eta}_{\mathscr{T}}$. Hence, we have $\varphi \approx \varphi; \mathtt{tt}$ for any φ.

Visibly Pushdown FLC.

Definition 2. Let $\Sigma = \Sigma_c \cup \Sigma_r \cup \Sigma_i$ be a visibly pushdown alphabet. The syntax of the fragment vpFLC of FLC is given by the following grammar.

$$\begin{aligned}
\varphi \quad ::= \quad & q \mid \overline{q} \mid X \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mu X.\psi \mid \nu X.\psi \mid \\
& [\![a_i]\!] \mid [\![a_i]\!];\varphi \mid [\![a_c]\!];[\![a_r]\!] \mid [\![a_c]\!];\varphi;[\![a_r]\!] \mid [\![a_c]\!];[\![a_r]\!];\varphi \mid [\![a_c]\!];\varphi;[\![a_r]\!];\varphi
\end{aligned}$$

where $q \in \mathscr{P}$, $X \in \mathscr{V}$, $a_m \in \mathscr{A}_x$ for $m \in \{i, c, r\}$, and $[\![a]\!]$ 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.

The definition alongside the visibly pushdown alphabet ensures that sequential composition, in particular when involving multiple composition operators, appears only "guarded" by modal operators. Hence, when exploring several formulas at the same time, e.g. in a tableau or a satisfiability game (cf. [6]), unfolding modal operators in lockstep will ensure that these formulas always have similar amounts of nested chop operators, and, hence, are of similar size. This makes such a satisfiability game a stair-parity game and, hence decidable.

It is open whether the definition of vpFLC can be relaxed to allow pending calls and returns.

2.4 Separating Properties

The properties witnessing the separation results discussed in this paper have – to some degree – been mentioned in the introduction already, and are also shown in Fig. 1. Here we defined them formally as formulas of the corresponding logics.

$\langle a^n b^n \rangle$. This property simply states "*there is a path labelled with n a's, followed by n b's, for some $n \geq 1$, ending in a state where p holds*". It should be clear that this can be expressed in PDL[CFL] since $L_{a^n b^n} := \{a^n b^n \mid n \geq 1\}$ is a CFL. Hence, $\langle a^n b^n \rangle := \langle L_{a^n b^n} \rangle p$ is a formula doing so.

Note that $L_{a^n b^n}$ is in fact a VPL over the visibly pushdown alphabet with a being a call and b a return symbol. Hence $\langle a^n b^n \rangle \in$ PDL[VPL].

An FLC formula formalising this property is $(\mu Z.\langle a \rangle; \langle b \rangle \vee \langle a \rangle; Z; \langle b \rangle); p$. It is best understood by unfolding the fixpoint formula using

$$\begin{aligned}
\underbrace{(\mu Z.\langle a \rangle; \langle b \rangle \vee \langle a \rangle; Z; \langle b \rangle)}_{\psi_Z}; p &\equiv (\langle a \rangle; \langle b \rangle \vee \langle a \rangle; \psi_Z; \langle b \rangle); p \equiv \langle a \rangle; \langle b \rangle; p \vee \langle a \rangle; \psi_Z; \langle b \rangle; p \\
&\equiv \langle a \rangle; \langle b \rangle; p \vee \langle a \rangle; (\langle a \rangle; \langle b \rangle \vee \langle a \rangle; \psi_Z; \langle b \rangle); \langle b \rangle; p \\
&\equiv \langle a \rangle; \langle b \rangle; p \vee \langle a \rangle; \langle a \rangle; \langle b \rangle; \langle b \rangle; p \vee \langle a \rangle; \langle a \rangle; \psi_Z; \langle b \rangle; \langle b \rangle; p \\
&\vdots \\
&\equiv \bigvee_{n \geq 1} \underbrace{\langle a \rangle; \ldots, \langle a \rangle}_{n \text{ times}}; \underbrace{\langle b \rangle; \ldots; \langle b \rangle}_{n \text{ times}}; p.
\end{aligned}$$

Clearly, this property cannot be expressed in PDL[REG] [11].

$\langle a^n b a^n \rangle$. This is very similar to the previous property with the CFL $L_{a^n b a^n} = \{a^n b a^n \mid n \geq 1\}$ instead, hence $\langle a^n b a^n \rangle$ it is a PDL[CFL] property. By the standard translation into FLC, it is also expressible there, for instance as $\varphi_{a^n b a^n} := (\mu Z.\langle a \rangle; \langle b \rangle; \langle a \rangle \vee \langle a \rangle; Z; \langle a \rangle); p$. Using fixpoint unfolding as before, one can see that it is equivalent to $\bigvee_{n \geq 1} \langle a \rangle^n; \langle b \rangle; \langle a \rangle^n; p$.

Note that $L_{a^n b a^n}$ is not a VPL since a would need to be both a call and a return symbol in order to be recognisable using a pushdown automaton. Likewise, $\varphi_{a^n b a^n}$ is not a vpFLC formula as $\langle a \rangle$ occurs both in front of and behind a recursion variable. Again, this would require a to be both a call and a return symbol. We formally show this in Thm. 11.

$(\langle a \rangle [b])^n$. This property is supposed to state something occurring in the definition of winning regions in alternating two-player reachability games of unbounded iteration, namely that there is some number $n \geq 1$ of moves such that player 1 can make a move such that no matter how player 2 responds, player 1 can make another move, etc. until after $2n$ moves a states satisfying p is reached. This is expressed by the $\mathscr{L}_\mu$ formula $\mu X.\langle a \rangle [b](p \vee X)$ or, using the standard translation into FLC [19], as $(\mu X.\langle a \rangle; [b](p \vee X)); \mathtt{tt}$ which is also a vpFLC formula. However, this property cannot be expressed in PDL[REG] [12].

$\langle a^n \rangle [b^n]$. This is similar to the property $\langle a^n b^n \rangle$ but here the second part "of the path" is universally quantified, i.e. it asks for the existing of an a-path of some length $n \geq 1$ such that all b-paths of length n following it end in a state satisfying p. This is easily expressed in FLC by changing the corresponding $\langle b \rangle$ to a $[b]$ in the formula for $\langle a^n b^n \rangle$, resulting in $(\mu Z.\langle a \rangle; [b] \vee \langle a \rangle; Z; [b]); p$. Since the original formula was already vpFLC, so is this one, as vpFLC treats existential and universal modalities equally. However, we show in Thm. 8 that this property cannot be expressed in PDL[CFL].

$\langle a^n \rangle [b] \langle a^n \rangle$. This asks for the existence of an a path of length n, such that all b-successors of the target state have an emerging a-path of length n again to some state satisfying p. Again, an FLC formula for this property is easily obtained from one for $\langle a^n b a^n \rangle$ by changing a corresponding modality,

resulting in $(\mu Z.\langle a\rangle;[b];\langle a\rangle \vee \langle a\rangle;Z;\langle a\rangle);p$. This is also no vpFLC formula for the same reason that a cannot have two roles in a visibly pushdown alphabet. We use this property to separate PDL[CFL] and FLC in Thm. 9 by showing that it cannot be expressed in the former.

3 Separating Propositional Dynamic and Modal Fixpoint Logics

In this section we prove that particular properties separate the expressive power of the modal fixpoint logics in the middle and upper band of Fig. 1 from the propositional dynamic logics in these bands, namely the properties $\langle a^n\rangle[b]^n$ and $\langle a^n\rangle[b]\langle a^n\rangle$ as defined in the previous section. The proofs use a special case of the well-known Pumping Lemma for regular languages [22]. Note that this is used in the context of propositional dynamic logics over context-free languages in the setting where the alphabet is only unary so that context-free languages boil down to regular ones anyway.

3.1 The Pumping Lemma for Unary Languages

The Pumping Lemma for regular languages states that, for any regular language L, there is n such that any word $w \in L$ with $|w| \geq n$ can be partitioned into $w = uvx$ with $|uv| \leq n$ and $|v| \geq 1$ such that $uv^i x \in L$ for all $i \in \mathbb{N}$. This follows from the fact that there must be a finite automaton for L, a DFA $\mathscr{A}$ in fact, that accepts it. Any run that over a word that is longer that the number of states of $\mathscr{A}$, some state must be visited more than once, and the section of the word between these occurrences can be "pumped" (up or down). However, the partition depends on the word in question. Moreover, given several regular languages, the partitions can differ even for words that are in the intersection of all the languages. The situation becomes more predictable in the setting of unary alphabets, i.e. those of the form $\{a\}$, in which case pumping constants can be found that work for all languages simultaneously. Before we show this, we need the following definition:

Definition 3. Let $L_1,\ldots,L_n$ be regular languages over the same alphabet Σ, and for $1 \leq i \leq n$, let $\mathscr{A}_i = (Q_i,\Sigma,\delta_i,q_I^i,F_i)$ be finite automata for L_i, respectively. A *simultaneous transition profile* for $\mathscr{A}_1,\ldots,\mathscr{A}_n$ is a relation $\tau \subseteq \bigcup_{1\leq i\leq n}(Q_i \times Q_i)$. Each word $w \in \Sigma^*$ defines such a simultaneous transition profile via $\tau_a = \bigcup_{1\leq i\leq n}\{(q,q') \mid (q,a,q') \in \delta_i\}$, and $\tau_{av} = \tau_a\tau_v = \bigcup_{1\leq i\leq n}\{(q,q') \mid$ ex. q'' s.t. $(q,q'') \in \tau_a, (q'',q') \in \tau_v\}$.

Note that if q,q' are states in Q_i, then $(q,q') \in \tau_w$ iff there is a run of $\mathscr{A}_i$ from q to q' over w.

Lemma 4. *Let $L_1,\ldots,L_n$ be regular Σ-languages, and let $a \in \Sigma$. Then there are m and $k > 0$ such that, for any $l \geq m+k$ and for all $1 \leq i \leq n, j \in \mathbb{N}$ we have that $a^l \in L_i$ iff $a^{l+j\cdot k} \in L_i$.*

Proof. Let $\mathscr{A}_1,\ldots,\mathscr{A}_n$ be finite automata for $L_1,\ldots,L_n$ as in Def. 3. By cardinality reasons, no more than $2^{|Q_1|^2+\cdots+|Q_n|^2}$ many transition profiles exist. Hence, if one enumerates the transition profiles for a,a^2,a^3 etc., upon reaching a word of the length $2^{|Q_1|^2+\cdots+|Q_n|^2}+1$, one transition profile τ must have occurred twice. Note that, if a transition profile occurs twice, the entire sequence of profiles between the two occurrences will occur again, i.e. the sequence of transition profiles is ultimately periodic. Let m be such that $\tau = \tau_m$ is the first occurrence of this profile, and k such that τ_{m+k} is the second occurrence of this profile. Then m and k are as in the lemma: let $l \geq m+k$ and let $w^l \in L_i$. Then $\mathscr{A}_i$ has an accepting run over w^l, i.e. a run from q_I^i to $q \in F_i$. Let q' be the state in the run after reading a^m. By definition of q', there is a run of $\mathscr{A}_i$ over a^{l-m} from q' to q. Since $\tau_m = \tau_{m+k}$, there is also a run of $\mathscr{A}_i$ from q_I^i to q' over a^{m+k}. By combining these two runs, we obtain a run from q_I^i to q' over a^{m+k}, and then a run from q' to q over a^{l-m}, which is an accepting run over $a^{m+k+l-m} = a^{l+k}$. The rest of the claim is by repeated application of the previous argument. □

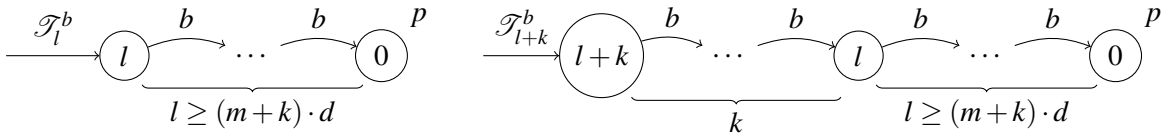

Figure 2: Transition Systems $\mathscr{T}_l^b$ and $\mathscr{T}_{l+k}^b$ for $l \geq (m+k)\cdot d$.

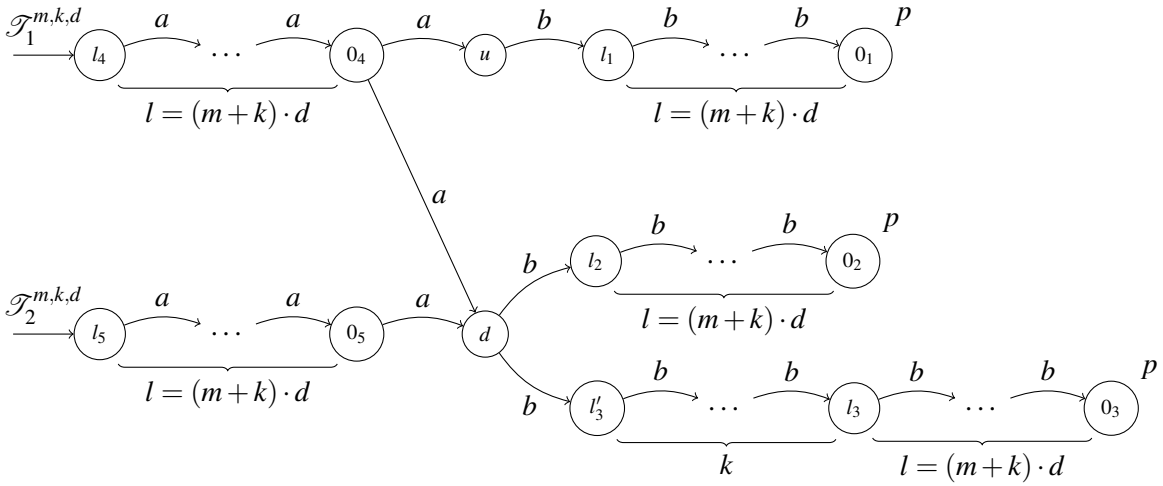

Figure 3: Transition Systems $\mathscr{T}_1^{m,k,d}$ and $\mathscr{T}_2^{m,k,d}$.

For the following lemma, let $\Sigma = \{a,b\}, \mathscr{P} = \{p\}$ and, for $l \in \mathbb{N}$, let $\mathscr{T}_l^b$ be defined as $\mathscr{T}_l^b = (\{0,\ldots,l\}, \to, \ell)$ with $\to = \{(i+1,b,i) \mid 0 \leq i \leq l-1\}$ and $\ell(s) = \{p\}$ iff $s = 0$. See Fig. 2 for a graphical representation.

Lemma 5. *Let $C = \{L_1,\ldots,L_n\}$ where $L_1,\ldots,L_n$ are regular Σ-languages, m and $k > 0$ be their combined pumping indices according to Lemma 4, $d \in \mathbb{N}$, $0 < d' \leq d$ and $l \geq (m+k)\cdot d$. Then, for all $l \geq j \geq (m+k)\cdot d'$, the states j in $\mathscr{T}_l^b$ and $j+k$ in $\mathscr{T}_{l+k}^b$ satisfy the same* PDL[C] *formulas of modal depth at most d'.*

Proof. The proof is by induction over d'. Assume that the result has been proven for all $0 < d'' \leq d'$. Let $j \geq (m+k)\cdot d'$. Let $\varphi = \langle L_i \rangle \psi$ with $md(\psi) \leq d'-1$ and $1 \leq i \leq n$. Assume that state j in $\mathscr{T}_l$ satisfies φ. Then there is $w \in \{b\}^*$ with $|w| \leq j$ such that $w \in L_i$ and the state $j-|w|$ satisfies ψ. There are two cases: If $|w| < m+k$, then $j-|w| \geq (m+k)\cdot(d'-1)$. If $d' = 1$, the result is immediate, since $\ell(s) = \emptyset$ unless $s = 0$ in either LTS. If $d' > 1$, we can use the induction hypothesis to infer that also state $j-|w|+k$ in $\mathscr{T}_{l+k}^b$ satisfies ψ. If $|w| \geq m+k$, then, by Lemma 4, also $b^{|w|+k} \in L_i$, whence state $j+k$ also satisfies $\langle L_i \rangle \psi$ in $\mathscr{T}_{l+k}^b$.

Conversely, let $\varphi = \langle L_i \rangle \psi$ hold at state $j+k$ in $\mathscr{T}_{l+k}^b$. Then there is $w \in \{b\}^*$ with $|w| \leq j+k$ such that $w \in L_i$ and the state $j+k-|w|$ satisfies ψ. Again, there are two cases. If $|w| < m+k$, then, $j+k-|w| \geq (m+k)\cdot(d'-1)+k$. If $d' = 1$ we again refer to the fact that $\ell(s) = \emptyset$ unless $s = 0$. If $d' > 1$ we can use the induction hypothesis to infer that also state $j-|w|$ satisfies ψ in $\mathscr{T}_l^b$. If $|w| \geq m+k$, then, by Lem. 4, also $b^{|w|-k} \in L_i$, whence state j also satisfies $\langle L_i \rangle \psi$ in $\mathscr{T}_l^b$. □

Definition 6. Let $m,k,d > 1$, $l = (m+k)\cdot d$ and $l' = l+k$. The LTS $\mathscr{T}_1^{m,k,d}$ and $\mathscr{T}_2^{m,k,d}$ are defined as in Fig. 3.

Let $\Sigma = \{a,b\}$. Our aim is to show that the property $\langle a^n\rangle[b^n]$ cannot be expressed in PDL[CFL].

Lemma 7. *Let $C = \{L_1,\ldots,L_n\}$ be a collection of context-free languages and, for $1 \le i \le n$, let $L_i' = L_i \cap L(b^*)$ be the (regular) intersection of L_i with $\{b\}^*$. Let m and k be their combined pumping indices as per Lemma 4 applied to $L_1',\ldots,L_n'$. Let $d \ge 1$, $l = (m+k)\cdot d$ and $l' = l+k$.*

Suppose $\mathscr{T}_1$ and $\mathscr{T}_2$ are the LTS $\mathscr{T}_1^{m,k,d}$, respectively $\mathscr{T}_2^{m,k,d}$ as per Def. 6. Then no PDL[C] *formula of modal depth d or less distinguishes $\mathscr{T}_1$ and $\mathscr{T}_2$.*

Proof. Clearly, for $0 \le j \le l$, the states j_1, j_2 and j_3 on the right halves of $\mathscr{T}_1$ and $\mathscr{T}_2$ satisfy the same formula of any modal depth due to the natural isomorphism of the associated structures. Moreover, by Lemma 5, the states j_1, j_2 and $(j+k)_3$ satisfy the same PDL[$L_1',\ldots,L_n'$]-formulas of modal depth $d' > 0$ if $j \ge (m+k)\cdot d'$, and, since the associated sub-LTS contain only b-transitions, also the same PDL[C] formulas of modal depth at most d'. By another application of Lemma 5, we obtain that the states u and d also satisfy the same PDL[C] formulas of modal depth at most d since they satisfy the same formulas of the form $\langle L_i\rangle\psi$.

Towards the claim of the lemma, it remains to show that, for all $0 \le j \le l$, the states l_4 and l_5 satisfy the same formulas of the form $\langle L_i\rangle\psi$, where $1 \le i \le n$ and $md(\psi) \le d$. Let $0 \le j \le l$ and assume that this has been shown for all $j' < j$. Let $1 \le i \le n$ and $md(\psi) \le n$. Consider $\langle L_i\rangle\psi$. In order for it to hold at state j_4, respectively j_5, there must be $w \in L_i$ and some other state reachable via w such that ψ holds at that state. In case that $w \in \{a\}^*$, this state is either u, respectively d, for which the result follows immediately, or the state is j_4', respectively j_5' with $j' \ge j$. In this case, the result follows from the induction hypothesis. Hence, the remaining case is that where $w \in \{a^+b^+\}$ and the witness for ψ is one of the j_1', j_2', j_3'. For the latter two cases, note that any j_2' or j_3' is reachable from j_4 and j_5 via the exact same word, whence the claim immediately follows.

Hence, the interesting case is that where $\langle L_i\rangle\psi$ holds at j_4 due to ψ holding at j_1' with $w \in L_i$ labelling the path from j_4 to j_1'. However, note that w also labels the path from j_5 to j_2', at which ψ also holds due to the sub-structures reachable from j_1' and j_2' being isomorphic. Hence, j_4 and j_5 satisfy exactly the same PDL[C] formulas of the form $\langle L_i\rangle\psi$ modal depth at most d, and, hence exactly the same PDL[C] formulas of modal depth d. In particular, this holds for l_4 and l_5, which is the claim of the lemma. □

Theorem 8. *Property $\langle a^n\rangle[b^n]$ cannot be expressed in* PDL[CFL]. *Hence,* vpFLC $\not\leq$ PDL[CFL].

Proof. Assume to the contrary that there is $\varphi \in$ PDL[CFL] that expresses $\langle a^n\rangle[b^n]$, i.e. φ holds in exactly those LTS that satisfy it. Let d be the modal depth of φ and let $C = L_1,\ldots,L_n$ be a list of the languages used in φ. By Lemma 7, φ cannot distinguish the transition systems $\mathscr{T}_1$ and $\mathscr{T}_2$ in Fig. 3. However, clearly $\mathscr{T}_1$ satisfies $\langle a^n\rangle[b^n]$, while $\mathscr{T}_2$ does not. Hence, $\langle a^n\rangle[b^n]$ cannot be expressed in PDL[CFL].

Conversely, $\langle a^n\rangle[b^n]$ can be expressed in vpFLC as seen in Sec. 2.4. □

Note that, in fact, the result can be strengthened to the Boolean closure of context-free languages, since the intersection of a language in the boolean closure of CFL with a unary alphabet remains regular. Moreover, the theorem also separates PDL[VPL] from vpFLC since the former is a fragment of PDL[CFL].

The previous theorem already supports the intuition that the reason for the ineffability of $\langle a^n\rangle[b^n]$ is not to be found in language-theoretic reasons, but in the alternation of the modal operators, which in some sense "insulates" the front part and the back part of the property, i.e. $\langle a^n\rangle$, respectively $[b^n]$ from

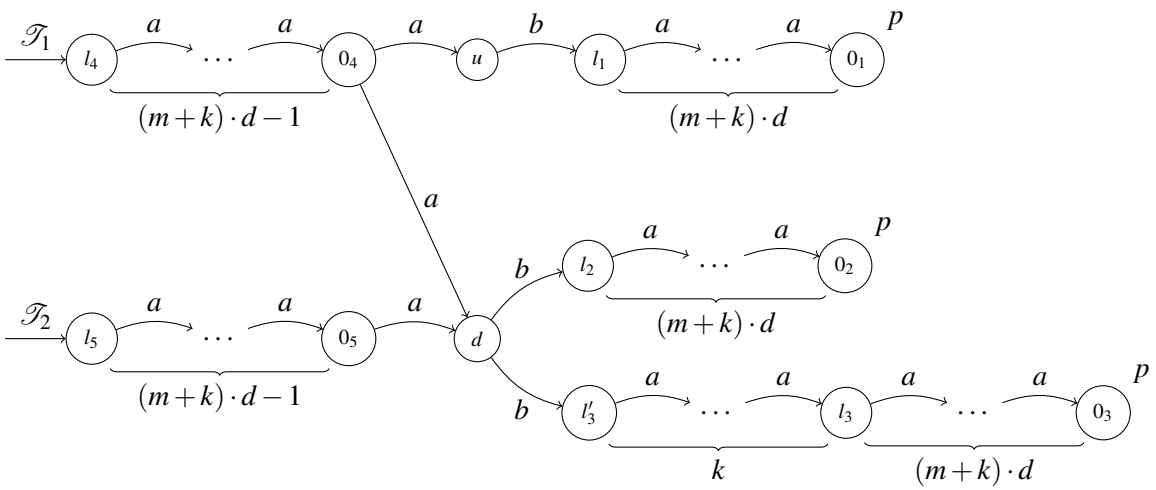

Figure 4: Transition Systems $\mathscr{T}_1^a$ and $\mathscr{T}_2^a$.

each other, preventing the constraint on the joint number of letters n to be "remembered" in the rest of the formula. In order to underline this point we now sketch that the property $\langle a^n\rangle[b]\langle a^n\rangle$ can also not be expressed in PDL[CFL].

Theorem 9. *Property $\langle a^n\rangle[b]\langle a^n\rangle$ cannot be expressed in* PDL[CFL]. *Hence,* FLC $\not\leq$ PDL[CFL].

Proof. (sketch) Consider the structures $\mathscr{T}_1^a$ and $\mathscr{T}_2^a$ in Fig. 4 for m,k,d to be given later. Clearly $\mathscr{T}_1^a$ satisfies $\langle a^n\rangle[b]\langle a^n\rangle$, while $\mathscr{T}_2^a$ does not.

The proof that the two structures cannot be distinguished in PDL[CFL] proceeds via the same pattern as the proof of Thm. 8, i.e. the structures are built depending on the context-free languages $L_1,\ldots,L_n$ used in the hypothetical formula that expressed $\langle a^n\rangle[b]\langle a^n\rangle$, and its modal depth d. In particular, in the proof that the states j_1, j_2 and $(j+k)_3$ satisfy the same formulas of a given modal depth over $L_1,\ldots,L_n$ proceeds in the same pattern by invoking Lemma 5 over the (regular) intersections of $L_1,\ldots,L_n$ with $\{a\}^*$. Also, after establishing that this holds, and that the states u and d satisfy the same PDL$[L_1,\ldots,L_n]$ formulas of modal depth d, the proof for the left part of the structures is the same.

However, it is not as straightforward to establish that u and d satisfy the same PDL$[L_1,\ldots,L_n]$-formulas of modal depth d, since the paths leading out of u and d are not over a unary alphabet. On the other hand, all these paths are labelled by a word starting with exactly one b, followed by a number of a's. Hence, we can equivalently replace any $\langle L_i\rangle\psi$ by $\langle b\rangle\langle\Delta_b(L_i)\rangle\psi$ where $\Delta_b(L_i)$ is the b-derivative of L_i. Note that, by Lemma 1, $\Delta_b(L_i)$ is context-free again and, hence, can now be replaced by its (regular) intersection with $\{a\}^*$, since it is interpreted over paths that contain only a's. Of course, this has to be taken into account when defining m and k via Lemma 4. It is not hard to see that the proof succeeds by defining m and k over the set of languages $L'_1,\ldots,L'_n,L'^b_1,\ldots,L'^b_n$ where L'_i is the intersection of L_i with $\{a\}^*$, and L'^b_i is the intersection of $\Delta_b(L_i)$ with $\{a\}^*$.

On the other hand, we have seen in Sec. 2.4 that $\langle a^n\rangle[b]\langle a^n\rangle$ can be expressed in FLC by the formula $(\mu Z.\langle a\rangle;[b];\langle a\rangle\vee\langle a\rangle;Z;\langle a\rangle);p$. □

4 Separating Decidable and Undecidable Logics

We now show that the inclusions of decidable program logics in Fig. 1 in the undecidable ones are strict. More precisely, we show that PDL[VPL] is strictly less expressive than PDL[CFL], and that vpFLC is strictly less expressive than FLC, which separates the top band in Fig. 1 from the middle band.

It is tempting to conclude an expressivity gap from the decidability gap; however, only a weaker proposition follows: there can be no *computable* equivalence-preserving translation from FLC to vpFLC, resp. from PDL[CFL] to PDL[VPL]. This does not preclude the existence of equivalent formulas in the smaller logic for each of the larger, though. It merely says that they could not be constructed effectively if they exist.

However, with a little bit more observation it is possible to extend this to an expressiveness gap, too.

Lemma 10. *Let p be an atomic proposition. Suppose $\langle a^n b a^n \rangle p$ was expressible in* vpFLC. *Then its satisfiability problem would be undecidable.*

Proof. Harel et al. present a reduction from Post's Correspondence Problem (PCP) to the satisfiability problem for PDL[$a^n b a^n$] [11]. They show how to construct, for every input $\mathscr{I} = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ to PCP, a formula $\varphi_{\mathscr{I}}$ of PDL[$a^n b a^n$] which is satisfiable iff $\mathscr{I}$ has a solution (in the sense of PCP).

Now suppose there was a vpFLC-formula $\varphi_{a^n b a^n}$ equivalent to $\langle a^n b a^n \rangle p$. Since any ordinay PDL[-] formula can easily be expressed in vpFLC, we immediately get a similar reduction from PCP to vpFLC's satisfiability problem: for any instance $\mathscr{I}$ of φ we can construct a vpFLC-formula $\varphi'_{\mathscr{I}}$ in the same way as the PDL[$a^n b a^n$]-formula $\varphi_{\mathscr{I}}$ with the only difference that we use $\varphi_{a^n b a^n}[\psi/p]$ whenever $\varphi_{\mathscr{I}}$ uses $\langle a^n b a^n \rangle \psi$. Clearly, $\varphi'_{\mathscr{I}}$ is equivalent to $\varphi_{\mathscr{I}}$ for any $\mathscr{I}$, and so it is satisfiable iff $\mathscr{I}$ is solvable. □

Note that this argument relies only on the existence of a formula equivalent to $\langle a^n b a^n \rangle p$, not its effective constructibility. In fact, the question after the effective constructibility of $\varphi_{a^n b a^n}$ is meaningless as it is a *fixed* formula and is therefore trivially constructible whenever it exists, as for every fixed formula there is clearly an algorithm which can write it down. Instead, it is the modularity of vpFLC, i.e. the possibility to build formulas by replacing subformulas, which is used in order to handle arbitrary PCP inputs $\mathscr{I}$.

Theorem 11. *We have* FLC $\not\leq$ vpFLC *and* PDL[CFL] $\not\leq$ PDL[VPL].

Proof. Clearly, $\langle a^n b a^n \rangle p$ can be expressed in PDL[CFL]. By [15], it is expressible in FLC. On the other hand, if it was expressible in vpFLC then, by Lemma 10, vpFLC's satisfiability problem would be undecidable contradicting its decidability result from [6].

Likewise $\langle a^n b a^n \rangle p$ cannot be expressible in PDL[VPL] either, as it would then be expressible in vpFLC, too, by the generic embedding of PDL[CFL] into FLC [15] which produces formulas from vpFLC when applied to formulas from PDL[VPL]. Equally, the contradiction can be obtained using the decidability result for PDL[VPL] [16]. □

So the different status of decidability between program logics does not immediately yield a gap in expressiveness, but it can be used to construct one by embedding presumably inexpressible properties in a set of formulas such that its subset of satisfiable ones is decidable in one case and undecidable in the other.

5 Conclusion

Summary. We have completely mapped the structures in the hierarchy of expressiveness amongst program logics for non-regular properties up to context-free ones. The two main strands of logics for such purposes found in the literature are propositional dynamic ones which incorporate formal languages into modal operators, and modal fixpoint logics which can, to some extent, mimic the generation of formal languages through least and greatest fixpoint constructions. We have provided formal proofs of what one may expect, namely that the bounded modality alternation inherent in propositional dynamic logics cannot be overcome by subtle constructions: there are properties which require some – even the minimal – amount of alternation amongst modal operators which cannot be expressed in these propositional dynamic logics. Note that $\langle a^n\rangle[b^n]$ only features one swap from an existential to a universal modality, and $\langle a^n\rangle[b]\langle a^n\rangle$ features the smallest possible amount of one kind of operator: only a single box-modality.

Further Work. One can, surely, devote an arbitrary amount of time to find further properties that witness the separation of logics presented here. For instance, with the developments leading to Thm. 8 it should not be too difficult to show that $[a^n]\langle b^n\rangle$ cannot be expressed in PDL[CFL] either.

Far more interesting, though, would be to investigate whether such separation techniques could be applied even further up the hierarchy of program logics. Note that PDL is a very generic formalism that is formally defined for *any* language class. Thus, any hierarchy of language classes imposes a hierarchy of PDL-logics, but strictness amongst languages does not immediately transfer to the logics. Instead, more or less sophisticated arguments are needed. As shown here, the argument based on the Pumping Lemma can be used up to the context-free languages, in fact even their Boolean closure. Beyond, for instance for the class CSL of context-sensitive languages, it is unclear whether there are separation results to be discovered in a similar style. So a separation of PDL[CSL] from PDL[CFL] for instance has, as far as we know, not been shown yet.

One may argue that beyond PDL[CFL] and FLC, the question of the strictness of the hierarchy becomes less interesting as these logics are undecidable already. There is, however, still a vast space of program logics with potential applications in formal verification despite undecidability of their satisfiability problems, as decidability of their model checking problems reaches far beyond that. On the modal fixpoint logic strand, even full HFL – which lifts $\mathscr{L}_\mu$ not only to predicate transformers as FLC does, but also to higher-order predicate transformers of arbitrary arity – retains model checking decidability, albeit of complexity that is k-fold exponential in the size of the underlying LTS [4] when k equals the maximal type order of such transformers.

The complexity of model checking propositional dynamic logics is well grounded in formal language theory, as it is polynomially linked to the complexity of the emptiness problem for intersections with regular languages [3], yielding, for instance exponential-time model checking for PDL over indexed languages [1], and doubly exponential-time model checking for PDL over multi-stack visibly pushdown languages [24].

A natural question that arises from the lifting of the decidability gap in satisfiability checking to the expressiveness gap, as done in the previous section, is whether complexity-theoretic gaps can be used for such purposes as well. The answer is of course yes: if the data complexity of two logics is separated by provably different complexity classes then so is their expressivity. This has been used for instance to establish that each HFL^{k+1} is more expressive than HFL^k for $k \geq 1$ [4], making use of the time hierarchy theorem. Likewise, the space hierarchy theorem can be used to separate the so-called tail-recursive fragments of each HFL^k [7]. It remains to be seen, though, if such results can be used to obtain separations from highly expressive PDL-based logics.

References

[1] A. V. Aho (1968): *Indexed Grammars - An Extension of Context-Free Grammars*. *J. ACM* 15(4), pp. 647–671, doi:10.1145/321479.321488.

[2] R. Alur & P. Madhusudan (2004): *Visibly pushdown languages*. In: *Proc. 36th Ann. ACM Symp. on Theory of Computing, STOC'04*, ACM Press, New York, pp. 202–211, doi:10.1145/1007352.1007390.

[3] R. Axelsson & M. Lange (2011): *Formal Language Constrained Reachability and Model Checking Propositional Dynamic Logics*. In: *Proc. 5th Workshop on Reachability Problems, RP'11*, *LNCS* 6945, Springer, pp. 45–57, doi:10.1007/978-3-642-24288-5_6.

[4] R. Axelsson, M. Lange & R. Somla (2007): *The Complexity of Model Checking Higher-Order Fixpoint Logic*. *Logical Methods in Computer Science* 3, pp. 1–33, doi:10.2168/LMCS-3(2:7)2007.

[5] F. Bruse & M. Lange (2020): *Temporal Logic with Recursion*. In: *Proc. 27th Int. Symp. on Temporal Representation and Reasoning, TIME'20*, *LIPIcs* 178, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 6:1–6:14, doi:10.4230/LIPIcs.TIME.2020.6.

[6] F. Bruse & M. Lange (2021): *A Decidable Non-Regular Modal Fixpoint Logic*. In: *Proc. 32nd Int. Conf. on Concurrency Theory, CONCUR'21*, *LIPIcs* 203, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 23:1–23:18, doi:10.4230/LIPIcs.CONCUR.2021.23.

[7] F. Bruse, M. Lange & É. Lozes (2021): *The Complexity of Model Checking Tail-Recursive Higher-Order Fixpoint Logic*. *Fundamenta Informaticae* 178(1–2), pp. 1–30, doi:10.3233/FI-2021-1996.

[8] J. A. Brzozowski (1964): *Derivatives of Regular Expressions*. *J. of the ACM* 11(4), pp. 481–494, doi:10.1145/321239.321249.

[9] E. M. Clarke & E. A. Emerson (1981): *Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic*. In D. Kozen, editor: *Proc. Workshop on Logics of Programs*, *LNCS* 131, Springer, Yorktown Heights, New York, pp. 52–71, doi:10.1007/BFb0025774.

[10] M. J. Fischer & R. E. Ladner (1979): *Propositional Dynamic Logic of Regular Programs*. *Journal of Computer and System Sciences* 18(2), pp. 194–211, doi:10.1016/0022-0000(79)90046-1.

[11] D. Harel, A. Pnueli & J. Stavi (1983): *Propositional Dynamic Logic of Nonregular Programs*. *Journal of Computer and System Sciences* 26(2), pp. 222–243, doi:10.1016/0022-0000(83)90014-4.

[12] D. Kozen (1983): *Results on the Propositional μ-calculus*. *TCS* 27, pp. 333–354, doi:10.1016/0304-3975(82)90125-6.

[13] M. Lange (2005): *Model Checking Propositional Dynamic Logic with All Extras*. *Journal of Applied Logic* 4(1), pp. 39–49, doi:10.1016/j.jal.2005.08.002.

[14] M. Lange (2007): *Three Notes on the Complexity of Model Checking Fixpoint Logic with Chop*. *R.A.I.R.O. – Theoretical Informatics and Applications* 41, pp. 177–190, doi:10.1051/ita:2007011.

[15] M. Lange & R. Somla (2006): *Propositional Dynamic Logic of Context-Free Programs and Fixpoint Logic with Chop*. *Information Processing Letters* 100(2), pp. 72–75, doi:10.1016/j.ipl.2006.04.019.

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

[17] C. Löding, P. Madhusudan & O. Serre (2004): *Visibly Pushdown Games*. In: *Proc. 24th Int. Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'04*, *LNCS* 3328, Springer, pp. 408–420, doi:10.1007/978-3-540-30538-5_34.

[18] K. Mehlhorn (1980): *Pebbling mountain ranges and its application to DCFL-recognition*. In: *Proc. 7th Int. Coll. on Automata, Languages and Programming, ICALP'80*, *LNCS* 85, Springer, pp. 422–435, doi:10.1007/3-540-10003-2_89.

[19] M. Müller-Olm (1999): *A Modal Fixpoint Logic with Chop*. In: *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, *LNCS* 1563, Springer, pp. 510–520, doi:10.1007/3-540-49116-3_48.

[20] R. J. Parikh (1966): *On Context-Free Languages*. *J. of the ACM* 13(4), pp. 570–581, doi:10.1145/321356.321364.

[21] A. Pnueli (1977): *The temporal logic of programs*. In: *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, IEEE, Providence, RI, USA, pp. 46–57, doi:10.1109/SFCS.1977.32.

[22] M. Rabin & D. Scott (1959): *Finite automata and their decision problems*. *IBM Journal of Research and Development* 3, pp. 114–125, doi:10.1147/rd.32.0114.

[23] A. Tarski (1955): *A Lattice-theoretical Fixpoint Theorem and its Application*. *Pacific Journal of Mathematics* 5, pp. 285–309, doi:10.2140/pjm.1955.5.285.

[24] S. La Torre, P. Madhusudan & G. Parlato (2007): *A Robust Class of Context-Sensitive Languages*. In: *Proc. 22nd Conf. on Logic in Computer Science, LICS'07*, IEEE, pp. 161–170, doi:10.1109/LICS.2007.9.

[25] M. Viswanathan & R. Viswanathan (2004): *A Higher Order Modal Fixed Point Logic*. In: *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, *LNCS* 3170, Springer, pp. 512–528, doi:10.1007/978-3-540-28644-8_33.