UNIKASSEL
VERSITÄT

Bachelor's Thesis

Inexpressibility Results for Propositional Dynamic Logic over Context-Free Programs

Eric Alsmann

Research Group

Theoretical Computer Science / Formal Methods

Prof. Dr. Martin Lange

supervised by

Dr. Florian Bruse

July 2021

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

Ich versichere außerdem, dass die digital abgegebene Version der Arbeit mit den gedruckten Versionen übereinstimmt.

(Eric Alsmann)

Contents

Chapter 1

Introduction

Modal logics form the base of many program verification languages. Most of these logics, for example propositional dynamic logic over regular languages (PDL[REG]) [4] and the modal μ-calculus ($\mathcal{L}_\mu$) [6], have their expressibility limits in regular properties. Pushing the limits to context-free languages generally leads to undecidability of their satisfiability problem like in fixed point logic with chop (FLC) [8] or propositional dynamic logic over context-free languages (PDL[CFL]) [5]. There has been work done in order to come as near as possible to the line of decidability in non-regular properties for example with PDL[VPL], which uses visibly push-down languages, a fragment of context-free languages where every alphabet symbol has only one designated function when it comes to altering the stack. Although PDL[VPL] can express non-regular properties, its satisfiability problem is decidable [7]. In [3], the authors introduce a decidable non-regular fragment of FLC, vpFLC, which embeds PDL[VPL] and $\mathcal{L}_\mu$. It restricts FLC in a way similar to visibly push-down languages. It has been non-constructively proven that PDL[VPL] must be strictly less expressive than vpFLC.

The aim of this thesis is to prove this strict inclusion with the help of the concrete property $\langle a \rangle^n [b]^n \mathbf{ff}$. Intuitively this property expresses that there exists an a-path of length n such that the path ends in a state from which every b-path of the same length n ends in a state where no proposition holds. The idea behind this property is the hypothesis that PDL[CFL] and PDL[VPL] cannot express non-regular properties which are spread across multiple modal operators. This means that the non-regularity of properties expressed in either of these languages is locally restricted to one modal operator. However, in [3] it was shown that this property is expressible in vpFLC. We will show that this property is not expressible in PDL[CFL] and therefore, also not in PDL[VPL]. By doing this we will also separate the expressive power of PDL[REG], PDL[VPL] and PDL[CFL].

Chapter 2

Preliminaries

2.1 Words and Languages

An *alphabet* Σ is a finite non-empty set of symbols. A *word* over the alphabet Σ is a finite sequence of symbols $w_1 \cdots w_n$ with $w_i \in \Sigma$. Hence, we define $|w| = n$ as the length of w and ϵ as the empty word. Σ^* is the set of all possible words over Σ.

A set $L \subseteq \Sigma^*$ is called a *language* over Σ. A set of languages $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ is called a *language class*.

2.2 Regular Languages

A finite automaton $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ is a tuple where Q is a non-empty finite set of states, Σ is the alphabet, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, $I \subseteq Q$ is the non-empty set of start states and $F \subseteq Q$ is a set of end states. We say $w \in L(\mathcal{A})$ for some $w = w_1 \dots w_n \in \Sigma^*$ if and only if there exist $q_1 \dots, q_{n-1} \in Q$, $q_0 \in I$ and $q_e \in F$ such that $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \dots q_{n-1} \xrightarrow{w_n} q_e$. For $w = \epsilon$, $w \in L(\mathcal{A})$ if and only if $q_0 \in F$ for some $q_0 \in I$. Hence we define δ^* as a extension of δ to words:

$$(p, w, q) \in \delta^* \Leftrightarrow p \xrightarrow{w_1} p_1 \xrightarrow{w_1} \dots \xrightarrow{w_n} q$$

for $w \in \Sigma^*$ and $p_1, \dots, p_{n-1} \in Q$.

We call a language L over an alphabet Σ regular if and only if there exists a finite automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$. The class of all regular languages is called REG.

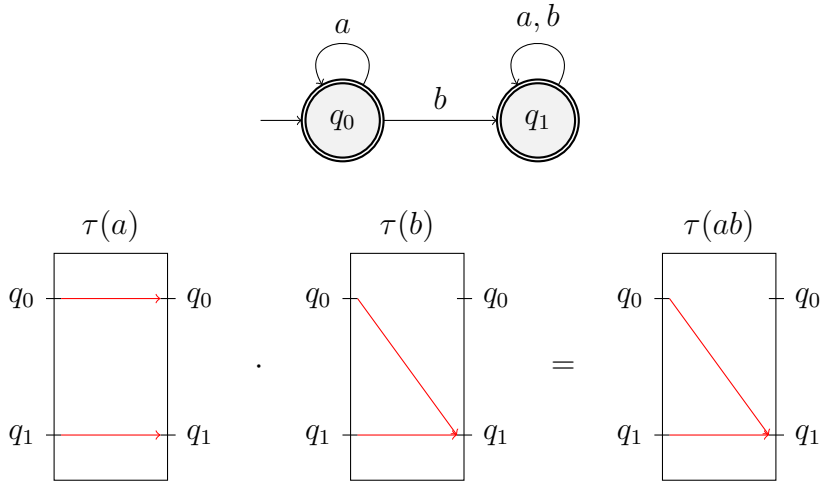


Figure 2.1: Transition profile example.

2.2.1 Transition Profiles

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. A *transition profile* is a relation $\tau_{\mathcal{A}} \subseteq Q \times Q$. Simply speaking a transition profile shows all possible states which can be reached by reading a word w in $\mathcal{A}$. The transition profile for a word $u \in \Sigma^*$ is defined as $\tau_{\mathcal{A}}(u) = \{(s,t) \mid (s,u,t) \in \delta^*\}$. There are $2^{|Q|^2}$ different transition profiles. Let $T_{\mathcal{A}}$ be the set of all transition profiles. We define a operator $\cdot : T_{\mathcal{A}}^2 \to T_{\mathcal{A}}$ on transition profiles with $\tau_1 \cdot \tau_2 = \{(s,t) \mid \exists r \in Q : (s,r) \in \tau_1 \wedge (r,t) \in \tau_2\}$. Intuitively, the operator combines two transition profiles such that two states are connected, if we get from the first state to the second state by reading the word from the first transition profile first and after that the word from the second transition profile in $\mathcal{A}$. Another property of the operator $\cdot$ is that it is multiplicative. That means for some word $w = uv \in \Sigma^*$, $\tau_{\mathcal{A}}(w) = \tau_{\mathcal{A}}(uv) = \tau_{\mathcal{A}}(u) \cdot \tau_{\mathcal{A}}(v)$. We call a transition profile $\tau \in T_{\mathcal{A}}$ reachable if there is a word $u \in L(\mathcal{A})$ with $\tau_{\mathcal{A}}(u) = \tau$.

Example 1. Consider the finite automaton $\mathcal{A}$ from Figure 2.1. We will now look at the transition profiles of $\mathcal{A}$ when reading the word ab. Therefore, we need to combine the transition profiles for a and b. We now see by looking at $\tau(ab)$, that ab is accepted by $\mathcal{A}$, because there is a connection from q_0 to q_1 and q_0 is a start state and q_1 is an end state.

2.3 Pushdown Languages

A pushdown automaton (PDA) is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with Q being the non-empty set of states, Σ the alphabet, Γ the stack alphabet, $q_0 \in Q$ the start state and $F \subseteq Q$ the set of end states. Furthermore, $\perp \in \Gamma$ is the symbol for the empty

stack. The transition relation δ ist defined as follows:

$$\delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*$$

A configuration of $\mathcal{A}$ is a tuple in $(Q \times \Sigma^* \times \Gamma^*)$. For a configuration (q, w, γ), q is the current state, w is the remaining word to read and γ is the current stack. The set of all possible configurations is called $\mathcal{C}$. We now define the relation $\vdash_{\mathcal{A}} \subseteq \mathcal{C} \times \mathcal{C}$ as follows:

$$\vdash_{\mathcal{A}} := \{((q, aw, g\gamma), (q', w, \gamma'\gamma)) \mid (q, a, g, q', \gamma') \in \delta\}$$

$\vdash_{\mathcal{A}}^*$ is the reflexive and transitive closure of $\vdash_{\mathcal{A}}$. We now say a word $w \in \Sigma^*$ is accepted by $\mathcal{A}$ if and only if $(q_0, w, \bot) \vdash_{\mathcal{A}}^* (f, \epsilon, \gamma)$ for some $f \in F$ and $\gamma \in \Gamma^*$. The set of all words accepted by $\mathcal{A}$ is called $L(\mathcal{A})$.

We call a language $L \subseteq \Sigma^*$ context-free if and only if it exists a PDA $\mathcal{A}$ such that $L = L(\mathcal{A})$. We call the set of all context-free languages CFL.

2.3.1 Visibly Pushdown Languages

Visibly pushdown automata [2] are a restriction to general pushdown automata introduced earlier. Generally speaking, they restrict the behavior of the stack depending on the alphabet symbol read. Either a symbol is used in transitions which put a symbol on the stack, pop a symbol of the stack or don't alter the stack at all. These tasks have to be determined before constructing the automaton. Therefore the alphabet is split in three disjunct sets, one for every task. We call this type of alphabet a *visibly pushdown alphabet*. Σ_{call} denotes the set of call symbols, Σ_{ret} the set of return symbols and Σ_{int} the set of internal symbols.

A *visibly pushdown automaton* VPA is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q is a non-empty set of states, $q_0 \in Q$ is a designated start state and $F \subseteq Q$ is a set of end states. Furthermore, $\Sigma = \Sigma_{ret} \cup \Sigma_{call} \cup \Sigma_{int}$ is the visibly pushdown alphabet. Γ is a finite set of symbols called the stack alphabet and $\bot \in \Gamma$ is the special empty stack symbol. Respectively the transition function $\delta = \delta_{call} \cup \delta_{ret} \cup \delta_{int}$ is also partitioned into three sets:

- $\delta_{call} \subseteq Q \times \Sigma_{call} \times Q \times \Gamma$
- $\delta_{ret} \subseteq Q \times \Sigma_{ret} \times \Gamma \times Q$
- $\delta_{int} \subseteq Q \times \Sigma_{int} \times Q$.

We define the acceptance behavior of a VPA analogue to a general PDA:

$$\begin{aligned}\vdash_{\mathcal{A}} := &\{((p, aw, g\gamma), (q, w, \gamma)) \mid (p, a, g, q) \in \delta_{\text{ret}}\} \\ &\cup \{((p, aw, \gamma), (q, w, g\gamma)) \mid (p, a, q, g) \in \delta_{\text{call}}\} \\ &\cup \{((p, aw, \gamma), (q, w, \gamma)) \mid (p, a, q) \in \delta_{\text{int}}\}.\end{aligned}$$

We call a set of words L over an alphabet Σ a visibly pushdown language if and only if it exists a visibly pushdown automaton $\mathcal{A}$ with $L(\mathcal{A}) = L$. We call the set of all visibly pushdown languages VPL.

Visibly pushdown automata are a restriction to general pushdown automata, because a symbol can only be added to the stack if a call symbol was read and a symbol can only be removed from the stack when a return symbol was read. That means one alphabet symbol can only contribute as a return symbol or a call symol, but never as both types at the same time. Hence the language $L = \{a^n b a^n \mid n \in \mathbb{N}\}$ is not a visibly pushdown language, but a general pushdown language. Every VPA can easily be translated into a PDA, but there are languages, which are accepted by a PDA, but not by an VPA (for example L). This implies the following strict inclusion: VPL $\subset$ CFL.

2.4 Propositional Dynamic Logic

2.4.1 Labelled Transition Systems

Let Σ be an alphabet and AP be a set of atomic propositions. A *labelled transition system* (LTS) is a tuple $(S, \rightarrow, q_0, L)$, where S is a non-empty set of states, $\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation and $q_0 \in S$ is a designated start state. Furthermore, $L : S \rightarrow \mathcal{P}(\text{AP})$ is a function, which assigns every state a label in the form of atomic propositions valid at this state. We write $s \xrightarrow{a} t$ for $(s, a, t) \in \rightarrow$. For simplicity we extend the operator $\rightarrow$ to words:

$$(s, w, t) \in \rightarrow \text{ for } w \in \Sigma^* \Leftrightarrow s \xrightarrow{w_1} q_1 \xrightarrow{w_2} q_2 \cdots q_{n-1} \xrightarrow{w_n} t$$

for $|w| = n$ and $q_1, \ldots, q_{n-1} \in S$. We call this the reflexive and transitive closure of the relation $\rightarrow$. In Figure 2.2 we can see an example for such a LTS.

2.4.2 Syntax

Let Σ and AP be defined as above. Let $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ be a language class. We call PDL[$\mathcal{C}$] the *propositional dynamic logic* over $\mathcal{C}$. Formulas over PDL[$\mathcal{C}$] are defined

as:

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

where $p \in \text{AP}$ und $L \in \mathcal{C}$. We may use boolean operators $\rightarrow, \leftrightarrow$ as well as $\mathbf{tt}$ and $\mathbf{ff}$ in the usual way.

The operators $\langle L\rangle$ and $[L]$ are called modal operators. Hence, we define the *modal depth* $\text{md}(\varphi)$ of a formula φ as:

- $\text{md}(\varphi) = 0$ for formulas without any modal operators
- $\text{md}(\neg\varphi) = \text{md}(\varphi)$
- $\text{md}(\varphi \circ \psi) = \max\{\text{md}(\varphi), \text{md}(\psi)\}$ for $\circ \in \{\wedge, \vee\}$
- $\text{md}(\langle L\rangle\varphi) = \text{md}([L]\,\varphi) = \text{md}(\varphi) + 1$

2.4.3 Semantics

Let $\varphi \in \text{PDL}[\mathcal{C}]$ be a formula and $\mathcal{T} = (S, \rightarrow, q_0, L)$ be a LTS. We define the satisfiability relation ($\models$) for some state $s \in S$ of $\mathcal{T}$ as follows:

- $s \models p$ iff $p \in L(s)$
- $s \models \neg\varphi$ iff not $s \models \varphi$
- $s \models \varphi \vee \psi$ iff $s \models \varphi$ or $s \models \psi$
- $s \models \varphi \wedge \psi$ iff $s \models \varphi$ and $s \models \psi$
- $s \models \langle L\rangle\varphi$ iff $\exists w \in L : \exists s' \in S : s \xrightarrow{w} s'$ and $s' \models \varphi$
- $s \models [L]\,\varphi$ iff $\forall w \in L : \forall s' \in S : s \xrightarrow{w} s' \Rightarrow s' \models \varphi$

We say $\mathcal{T} \models \varphi$ if and only if $q_0 \models \varphi$. Furthermore, $[L]\,\varphi$ can als be expressed as $\neg\langle L\rangle\neg\varphi$

Simply speaking $\langle L\rangle\varphi$ can be written as, there exists a path from state s in $\mathcal{T}$ such that the path is labelled with a word $w \in L$ and the end state of the path satisfies the formula φ. Hence $[L]\,\varphi$ can be understood as: there is no path which is labelled with $w \in L$ such that the end state of the path does not satisfy φ or all paths labelled with a word $w \in L$ end in a state which satisfies φ.

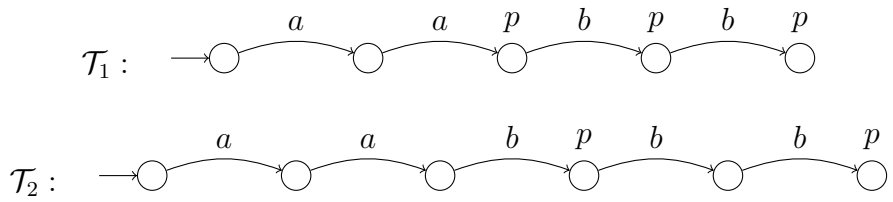


Figure 2.2: The example structure for PDL[REG] and PDL[CFL].

Example 2. Over PDL[REG] the formula $\varphi = \langle a^* \rangle [b^*] p$ expresses that there exists an a^*-path such that at all states which are reachable with a b^*-path, p holds. Hence for the structures in Figure 2.2, $\mathcal{T}_1 \models \varphi$, but $\mathcal{T}_2 \not\models \varphi$

Example 3. Over PDL[CFL] the formula $\varphi = \langle L \rangle p$ for $L = \{a^n b^n \mid n \in \mathbb{N}\}$ expresses that there exists a path labelled with a word from L such that on the end state of the path p holds. Hence in Figure 2.2, $\mathcal{T}_1 \models \varphi$, but $\mathcal{T}_2 \not\models \varphi$.

Chapter 3

A Better Pumping Lemma

In this chapter we will show that, for each finite set of regular languages $L_1, \ldots, L_m$ we can find numbers $k, l \in \mathbb{N}$ for which all those languages behave in the same way on the word a^k and a^{k+l}. The idea for this argument comes from the traditional pumping lemma for regular languages:

Lemma 4 (Rabin, Scott [10])**.** *For every regular language $L \in REG$ exists a $n \in \mathbb{N}$ such that for all $w \in L$ with $|w| \geq n$ exist a partition $w = xyz$ with the following properties:*

$$|xy| \leq n, \quad |y| \geq 1, \quad xy^i z \in L \ \forall i \in \mathbb{N}$$

For each word of a regular language which is long enough, the pumping lemma for regular languages ensures the existence of a partial word which can be multiplied any number of times and still be part of the language. We now try to extend this argument to sets of regular languages. Due to the fact that this is in general not possible, we will proof a less strongly argument over unary words. We want to find numbers $k, l \in \mathbb{N}$ such that for all $n \geq k$ holds: if one of the regular languages accepts the word a^n, it also accepts a^{n+l}.

Definition 5 (Synchronous Automaton)**.** *Let $L_1, \ldots, L_m$ be regular languages. For each language L_i exists a finite automaton $\mathcal{A}_i = (Q_i, \Sigma, \delta_i, I_i, F_i)$ with $L(\mathcal{A}_i) = L_i$. Without loss of generality we assume that all state sets are pairwise disjoint. We now define the synchronous automaton of these languages as:*

$$\mathcal{M} = (Q_1 \cup \cdots \cup Q_m, \Sigma, \delta_1 \cup \cdots \cup \delta_m, I_1 \cup \cdots \cup I_m, F_1 \cup \cdots \cup F_m)$$

We define $|\mathcal{M}| = |Q_1 \cup \ldots Q_m|$.

Although $\mathcal{M}$ looks like the union automaton, we are not interested in the acceptance behavior of the automaton. We are particularly interested in the different transition profiles the automaton reaches. The automaton can be interpreted as the synchronous execution of all separate automata at the same time. There are $2^{|\mathcal{M}|^2}$

different transition profiles of $\mathcal{M}$. If we execute $\mathcal{M}$ on a^*, it follows with a pigeon-hole principle argument, that at some point $\mathcal{M}$ has to go through a transition profile twice. These arguments are summarized in the following lemma:

Lemma 6. *For every finite set of regular languages $L_1, \dots L_m$, there exist numbers $k, l \in \mathbb{N}$ with $k + l \leq 2^{|\mathcal{M}|^2} + 1$ such that $\tau_{\mathcal{M}}(a^{k+n}) = \tau_{\mathcal{M}}(a^{k+l+n})$ for all $n \in \mathbb{N}$. $\mathcal{M}$ is the synchronous automaton of these regular languages.*

Proof. Let $L_1, \dots, L_m \in \text{REG}$ be regular languages and $\mathcal{M}$ the synchronous automaton constructed as in Definition 5. Furthermore let $\tau_{\mathcal{M}}(a)$ be the transition profile of $\mathcal{M}$ when reading a. For $p = 2^{|\mathcal{M}|^2} + 1$ we observe the sequence of transition profiles by synchronous execution of all finite automata in $\mathcal{M}$:

$$\tau_{\mathcal{M}}(a) \to \tau_{\mathcal{M}}(aa) \to \cdots \to \tau_{\mathcal{M}}(a^p)$$

It follows that $\tau_{\mathcal{M}}(a^i) = \tau_{\mathcal{M}}(a^j)$ for $i, j \leq p$ with a simple pigeon-hole principle argument. There are only $2^{|\mathcal{M}|^2}$ many possible transition profiles. Hence, because $p = 2^{|\mathcal{M}|^2} + 1$, at least one transition profile must occur twice. We now set $k = i$ and $l = j - i$. It is obvious now that the following argument holds for all $n \in \mathbb{N}$:

$$\tau_{\mathcal{M}}(a^{k+n}) = \tau_{\mathcal{M}}(a^k) \cdot \tau_{\mathcal{M}}(a^n) = \tau_{\mathcal{M}}(a^{k+l}) \cdot \tau_{\mathcal{M}}(a^n) = \tau_{\mathcal{M}}(a^{k+l+n})$$

□

We will use this argument in later sections in the following form:

Corollary 7. *Let $L_1, \dots L_m$ be a finite set of regular languages and $k, l \in \mathbb{N}$ defined as in Lemma 6 then:*

$$a^{k+n}w \in L_i \Leftrightarrow a^{k+l+n}w \in L_i$$

for $w \in \Sigma^*$, $n \in \mathbb{N}$ and $i \in \{1, \dots, m\}$.

This follows directly from Lemma 6. Because when we are considering the transition profiles we get:

$$\tau_{\mathcal{M}}(a^{k+n}w) = \tau_{\mathcal{M}}(a^{k+n}) \cdot \tau_{\mathcal{M}}(w) = \tau_{\mathcal{M}}(a^{k+l+n}) \cdot \tau_{\mathcal{M}}(a^w) = \tau_{\mathcal{M}}(a^{k+l+n}w)$$

Example 8. Consider the two finite automata $\mathcal{A}$ and $\mathcal{B}$ as in Figure 3.1 over the unary alphabet $\Sigma = \{a\}$. We now look at the transition profiles of the synchronous automaton $\mathcal{M}$ of $\mathcal{A}$ and $\mathcal{B}$, which can also be seen in Figure 3.1.

The first repeating transition profile of $\mathcal{M}$ is $\tau_{\mathcal{M}}(aaa) = \tau_{\mathcal{M}}(a)$. This means in this case we choose $k = 1$ and $l = 2$. Hence for every $n \in \mathbb{N}$ $\tau_{\mathcal{M}}(a^{1+n}) = \tau_{\mathcal{M}}(a^{3+n})$. It follows that for example $a^2 \in L(\mathcal{B})$ and $a^{2+2m} \in L(\mathcal{B})$ for every $m \in \mathbb{N}$ too.

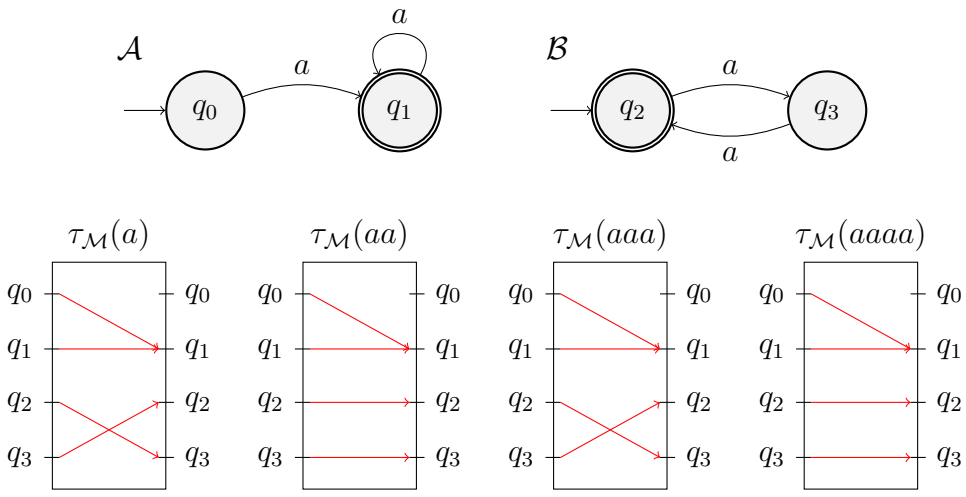


Figure 3.1: The Example for Lemma 6.

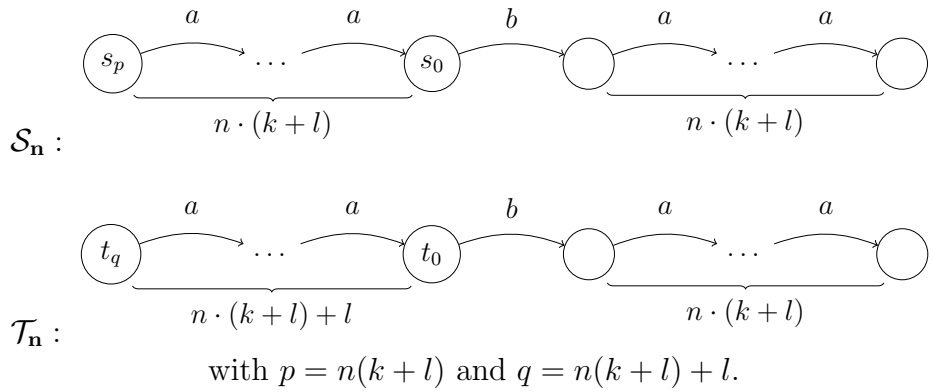
Chapter 4

Separating PDL[REG] and PDL[CFL]

In this chapter we will separate PDL[REG] and PDL[CFL] as well as PDL[REG] and PDL[VPL]. We start by separating PDL[REG] and PDL[VPL]. Let $L = \{a^n ba^n \mid n \in \mathbb{N}\}$ and $\varphi = \langle L \rangle \mathbf{ff}$. L is obviously context-free. For readability we simply write $\langle a^n ba^n \rangle \mathbf{ff}$. We will show that this formula which is obviously in PDL[CFL] cannot be expressed in PDL[REG]. Note that $\langle a^n ba^n \rangle \mathbf{ff}$ is not a formula over PDL[REG], but will be used as an expression for this property.

Lemma 9. *Let P be the property $\langle a^n ba^n \rangle \mathbf{ff}$ for some $n \in \mathbb{N}$. There exists no formula φ over PDL[REG], such that for all labelled transition systems $\mathcal{T}$ holds: $\mathcal{T} \models P \Leftrightarrow \mathcal{T} \models \varphi$*

Proof. Assume there is a formula $\varphi \in$ PDL[REG] which satisfies the claim. Let $L_1, \dots L_m$ be the languages used in the modal operators of the formula. Furthermore, let $n = \mathrm{md}(\varphi)$ be the modal depth of φ and let $k, l \in \mathbb{N}$ be defined according to Lemma 6.



with $p = n(k+l)$ and $q = n(k+l)+l$.

Figure 4.1: The transition systems for separating PDL[REG] and PDL[CFL].

Let $\mathcal{S}_n$ and $\mathcal{T}_n$ be defined as in Figure 4.1. Obviously $\mathcal{S}_n \models P$ and $\mathcal{T}_n \not\models P$. We will now show through induction over d that for all $\varphi \in \mathrm{PDL}[L_1, \ldots, L_m]$ with $\mathrm{md}(\varphi) = d \leq n$ and $i \geq d(k+l)$: $s_i \models \varphi \Leftrightarrow t_{i+l} \models \varphi$ with $s_i \in \mathcal{S}_n$ and $t_i \in \mathcal{T}_n$.

Note that for all j, s_j and t_j satisfy the same formulas, because the structures are isomorphic beginning from these states.

$\mathbf{d = 0}$: The claim holds trivially because all states do not differ in their propositions.

$\mathbf{d > 0}$: $\varphi = \langle L_j \rangle \varphi'$. Then either $s_i \models \varphi$ and it exist $s' \in \mathcal{S}_n$, $w \in L_j$ with $s_i \xrightarrow{w} s'$ and $s' \models \varphi'$, or $t_{i+l} \models \varphi$ and it exist $t' \in \mathcal{T}_n$, $w \in L_j$ with $t_{i+l} \xrightarrow{w} t'$ and $t' \models \varphi'$.

Case 1: $s_i \models \varphi$ and $|w| \geq k + l$. w must be of the form $a^{k+l}w'$. Then with Corollary 7, $a^{k+2l}w' \in L_j$ too. That means we can take the following path in $\mathcal{T}_n$:

$$t_{i+l} \xrightarrow{a^l} t_i \xrightarrow{w'} t'.$$

As mentioned above s_i and t_i satisfy the same formulas and therefore, s' and t' too. Hence, $t' \models \varphi'$.

Case 2: $t_{i+l} \models \varphi$ and $|w| \geq k + l$. w must be of the form $a^{k+l}w'$ and

$$t_{i+l} \xrightarrow{a^l} t_i \xrightarrow{a^k w'} t'.$$

Then with Corollary 7, $a^k w' \in L_j$ too. Because s_i and t_i satisfy the same formulas and $s_i \xrightarrow{a^k w'} s'$, it follows that $s' \models \varphi'$.

Case 3: $|w| < k + l$. w must be of the form $a^{|w|}$. We can now apply our induction hypothesis and assume that $s_{i-|w|}$ and $t_{i+l-|w|}$ cannot be distinguished by φ' with $\mathrm{md}(\varphi') = d - 1$.

Note that if no modal operator has a witness with length greater $k + l$, φ can only distinguish a LTS in the first $d(k + l)$ nodes. Therefore, $\mathcal{T}_n \models \varphi$ holds trivially.

The cases of φ being a boolean combination of subformulas, follow directly from the induction hypothesis.

□

For separating PDL[REG] and PDL[VPL] we consider another property. Let $L = \{a^n b^n \mid n \in \mathbb{N}\}$ and $\varphi = \langle L \rangle \mathbf{ff}$. L is a well known example for a visibly push-down language. We will show that there is no formula in PDL[REG] such that this property could be expressed.

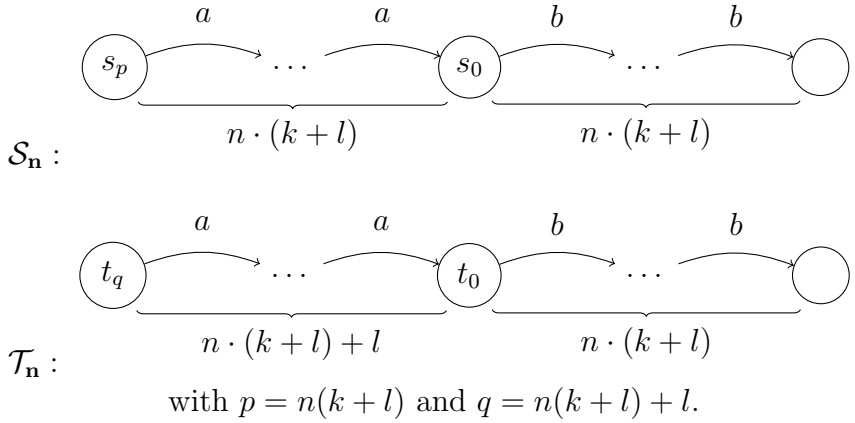


Figure 4.2: The LTS for Lemma 10.

Lemma 10. *Let P be the property $\langle a^n b^n \rangle \mathbf{ff}$ for some $n \in \mathbb{N}$. There exists no formula φ over PDL[REG] such that for all linear transition systems $\mathcal{T}$ holds: $\mathcal{T} \models P \Leftrightarrow \mathcal{T} \models \varphi$*

Proof. This proof works analogous to Lemma 9. Assume there is a formula $\varphi \in$ PDL[REG] which satisfies the claim. Let $L_1, \ldots, L_m$ be the languages used in the modal operators of the formula. Furthermore, let $n = \mathrm{md}(\varphi)$ be the modal depth of φ and let $k, l \in \mathbb{N}$ be defined according to Lemma 6.

Let $\mathcal{S}_n$ and $\mathcal{T}_n$ be defined as in Figure 4.2. Obviously $\mathcal{S}_n \models P$ and $\mathcal{T}_n \not\models P$. It can be shown through induction over d that for all $\varphi \in \mathrm{PDL}[L_1, \ldots, L_m]$ with $\mathrm{md}(\varphi) = d \leq n$ and $i \geq d(k+l)$: $s_i \models \varphi \Leftrightarrow t_{i+l} \models \varphi$ with $s_i \in \mathcal{S}_n$ and $t_i \in \mathcal{T}_n$.

This proof by induction works in the same way as the induction in Lemma 9. Hence we will omit the full induction proof in this case and refer to Lemma 9. □

Theorem 11. *PDL[REG] is strictly less expressive than PDL[CFL].*

Proof. The inclusion of PDL[REG] in PDL[CFL] is trivial, because it is well known that REG ⊂ CFL. Hence, every PDL[REG] formula can also be seen as a PDL[CFL] formula. The strict inclusion follows from Lemma 9. □

Theorem 12. *PDL[REG] is strictly less expressive than PDL[VPL].*

Proof. In this case the inclusion of PDL[REG] in PDL[VPL] is trivial too, because every regular language can be represented as a finite automaton and every finite automaton can be seen as a VPA, which only uses internal symbols. Furthermore, we have shown in Lemma 10 that there is a property which can be expressed in PDL[VPL], but not in PDL[REG]. It follows that PDL[REG] is strictly less expressive than PDL[VPL]. □

Chapter 5

Separating vpFLC and PDL[CFL]

As mentioned in the introduction, this chapter will be about separating vpFLC and PDL[CFL]. Let P be the property $\langle a\rangle^n[b]^n\mathbf{ff}$. Intuitively P expresses the following property. For some $n \in \mathbb{N}$, there exists a path labelled with a^n, such that from the end state of that path all paths labelled with b^n end in a state where no proposition holds. It was shown in [3] that P is expressible in vpFLC. The proof that P is not expressible in PDL[CFL] will use several arguments introduced in the previous sections.

We start by proving the indistinguishability of two LTS, which will later appear as part of a bigger structure.

Lemma 13. *For each $\varphi \in$ PDL[CFL] with modal depth $n \in \mathbb{N}$ exist $k, l \in \mathbb{N}$ such that φ cannot distinguish the two LTS from Figure 5.1:*

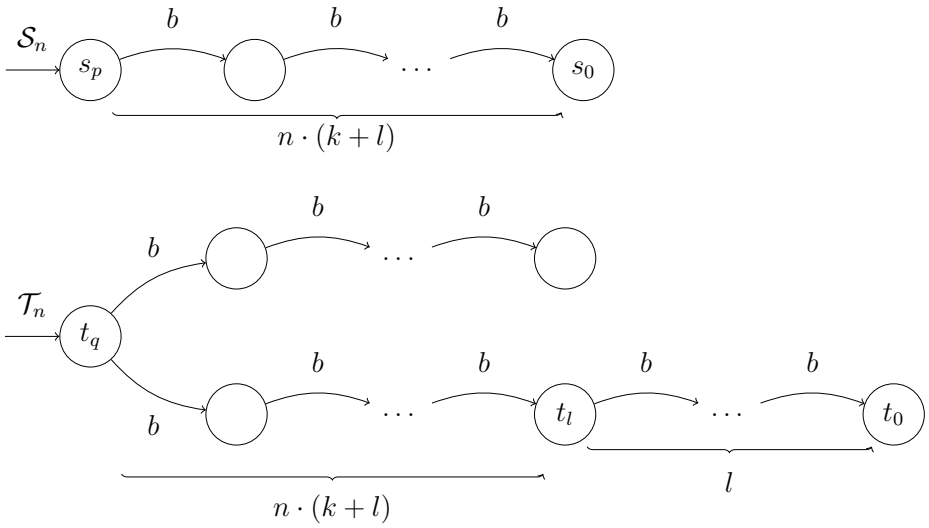
Proof. Let $L_1, \ldots, L_m$ be the pushdown languages used in φ. Without loss of generality we assume that for all languages L_i must hold that $L_i \subseteq L(b^*)$. Due to the regularity of pushdown languages over unary alphabets [9], we can find numbers $k, l \in \mathbb{N}$ for any finite set of pushdown languages used in φ defined as in Lemma 6.

We will now show by induction over the modal depth d that for all formulas $\varphi \in \text{PDL}[L_1, \ldots, L_m]$ with $\text{md}(\varphi) = d \leq n$ and $i \geq d(k + l)$: $s_i \models \varphi \Leftrightarrow t_{i+l} \models \varphi$ with $s_i \in \mathcal{S}_n$ and $t_{i+l} \in \mathcal{T}_n$.

Note that for all f, s_f and t_f satisfy the same formulas, because the structures are isomorphic beginning from these states.

$\mathbf{d = 0}$: The claim holds trivially because none of the states differ in their propositions.

$\mathbf{d > 1}$: $\varphi = \langle L_j\rangle\varphi'$ with $\text{md}(\varphi') = n-1$. Then either $s_i \models \varphi$ and it exist s_f, $w \in L_j$ with $s_i \xrightarrow{w} s_f$ and $s_f \models \varphi'$, or $t_{i+l} \models \varphi$ and it exist t_f, $w \in L_j$ with $t_{i+l} \xrightarrow{w} t_f$ and $t_f \models \varphi'$.



with $p = n(k+l)$ and $q = n(k+l)+l$.

Figure 5.1: The LTS used in Lemma 13.

Case 1: $s_i \models \varphi$ and $i = p$, i.e., s_i is the first node in $\mathcal{S}_n$. In this case we take to isomorphic path in the upper arm of $\mathcal{T}_n$. The end state of this path is isomorphic to s_f.

Case 2: $t_{i+l} \models \varphi$ and $i + l = q$. If t_f is in the upper arm of $\mathcal{T}_n$, we choose the equivalent path in $\mathcal{S}_n$. Due to isomorphism of the remaining structures $s_f \models \varphi'$. If s_f is in the lower arm, we apply Case 4.

Case 3: $s_i \models \varphi$ and $|w| \geq k+l$. w must be of the form $b^{|w|}$ and with Corollary 7, $b^{|w|+l} \in L_j$ too. We can take the following path in $\mathcal{T}_n$:

$$t_{i+l} \xrightarrow{a^l} t_i \xrightarrow{w} t_f$$

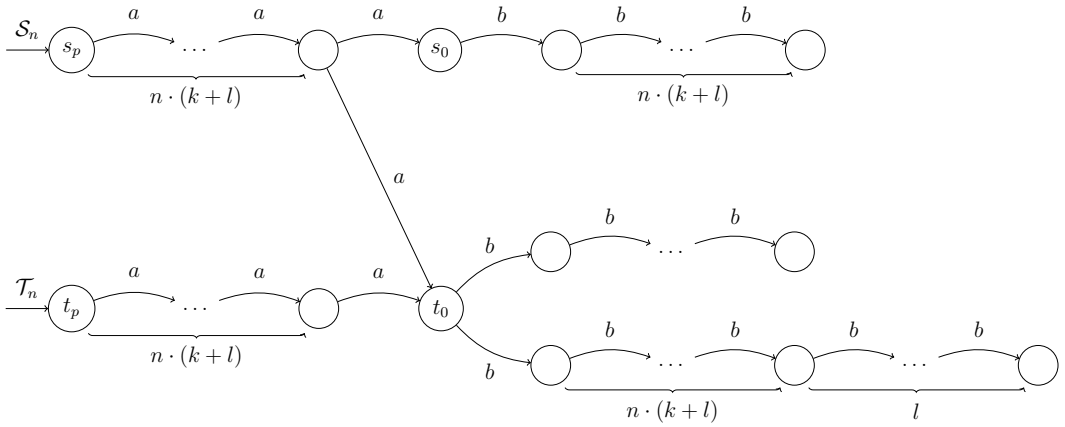
As mentioned above s_f and t_f satisfy the same formulas and therefore, $t_f \models \varphi'$.

Case 4: $t_{i+l} \models \varphi$ and $|w| \geq k + l$. w must be of the form $b^{|w|}$ and with Corollary 7, $b^{|w|-l} \in L_j$ too. Therefore, we can take the following path in $\mathcal{S}_n$:

$$s_i \xrightarrow{b^{|w|-l}} s_f$$

As mentioned above s_f and t_f satisfy the same formulas and therefore, $t_f \models \varphi'$.

Case 5: $|w| < k + l$. w must be of the form $b^{|w|}$. We can now apply our induction hypothesis and assume that $s_{i-|w|}$ and $t_{i+l-|w|}$ cannot be distinguished by φ' with $\mathrm{md}(\varphi') = d - 1$.



with $p = n(k + l)$.

Figure 5.2: The LTS for Lemma 14.

Note that if no modal operator has a witness with length greater $k + l$, φ can only distinguish a LTS in the first $d(k + l)$ nodes. Therefore, $\mathcal{T}_n \models \varphi$ holds trivially.

The cases of φ being a boolean combination of subformulas, follow directly from the induction hypothesis.

□

With the help of this lemma, we can now prove the actual claim.

Lemma 14. *Let P be the property $\langle a \rangle^n [b]^n \mathbf{ff}$ for some $n \in \mathbb{N}$. There exists no formula φ over PDL[VPL], such that for all labelled transition systems $\mathcal{T}$ holds: $\mathcal{T} \models P \Leftrightarrow \mathcal{T} \models \varphi$*

Proof. Assume there was a formula $\varphi \in$ PDL[VPL] which satisfies the claim. Let $L_1, \ldots L_m$ be the languages used in the modal operators of the formula. Furthermore, let $n = \mathrm{md}(\varphi)$ be the modal depth of φ.

We construct the LTS $\mathcal{S}_n$ and $\mathcal{T}_n$ as seen in Figure 5.2 by doing an intersection of all languages $L_1, \ldots, L_m$ with $L(b^*)$. By doing this we get the languages $\widetilde{L}_1, \ldots, \widetilde{L}_m$, which are all unary. It is known that unary context-free languages are regular [9]. We now define k, l as in Lemma 6 by considering the languages $\widetilde{L}_1, \ldots, \widetilde{L}_m$.

We will now show by induction over the modal depth d that for all formulas $\varphi \in$ PDL[CFL] with $\mathrm{md}(\varphi) = d$ and $i \geq d(k + l)$ holds: $s_i \models \varphi \Leftrightarrow t_i \models \varphi$ with $s_i \in \mathcal{S}_n$ and $t_i \in \mathcal{T}_n$ (Figure 5.2).

d = 0: The claim holds trivially, because none of the states differ in their propositions.

d > 0: $\varphi = \langle L_j \rangle \varphi'$. Then either $s_i \models \varphi$ and it exist s', $w \in L_j$ with $s_i \xrightarrow{w} s'$ and $s' \models \varphi'$, or $t_i \models \varphi$ and it exist t', $w \in L_j$ with $t_i \xrightarrow{w} t'$ and $t' \models \varphi'$.

Case 1: $s_i \models \varphi$ and $s' = s_0$. In this case $t_i \xrightarrow{w} t_0$ and it follows with Lemma 13 that s_0 and t_0 cannot by distinguished by φ'.

Case 2: $t_i \models \varphi$ and $t' = t_0$. Analogue to Case 1.

Case 3: $s_i \models \varphi$ and $s' = s_f$ for $f > 0$. In this case we apply our induction hypothesis and assume that s_f and t_f cannot be distinguished by φ'.

Case 4: $t_i \models \varphi$ and $t' = t_f$ for $f > 0$. Analogue to Case 2.

Case 5: $t_i \models \varphi$. Because of the connection between s_1 and t_0, we can also go from s_i with w to s' and $s' = t'$.

Case 6: $s_i \models \varphi$ and s' is one of the states on the right hand side of the lower structure. This case is analogue to Case 4, because we can simply take the same path in $\mathcal{S}_n$ and then $s' = t'$.

Case 7: $s_i \models \varphi$ and s' is one of the state on the right hand side of the upper structure. We notice that the upper arm of the lower structure is isomorphic to the arm in the upper structure. That means for every state in the upper arm, we can find a state in the lower arm which is reachable with the same path, such that both states satisfy the same formulas. In this case we choose the equivalent state to s' in the lower isomorphic arm t'. Obviously $t_i \xrightarrow{w} t'$.

The cases of φ being a boolean combination of subformulas, follow directly from the induction hypothesis.

This results in the fact that there is no formula over PDL[CFL] which can distinguish the LTS $\mathcal{S}_n$ and $\mathcal{T}_n$ for any model depth n.

□

Theorem 15. *PDL[VPL] is strictly less expressive than vpFLC.*

Proof. The inclusion of PDL[VPL] in vpFLC was shown in [3]. Due to Lemma 14 the property $P = \langle a \rangle^n [b]^n \mathbf{ff}$ is not expressible in PDL[CFL] and therefore not in PDL[VPL] due to the obvious inclusion PDL[VPL] $\subseteq$ PDL[CFL]. However, it was shown in [3] that P is expressible in vpFLC. Hence PDL[VPL] is strictly less expressive than vpFLC.

□

Theorem 16. *PDL[CFL] and vpFLC are incomparable in terms of expressive power.*

Proof. We showed in Lemma 14 that there is a property which is expressible in vpFLC, but not in PDL[CFL]. The other direction is presented in [1].

□

Chapter 6

Separating PDL[VPL] and PDL[CFL]

This last chapter aims to separate PDL[VPL] and PDL[CFL] by proving that $\langle a^n b a^n \rangle \mathbf{ff}$ is not expressible in PDL[VPL]. For this proof we will take advantage of the following property over visibly pushdown languages: Every visibly pushdown language, which is unary except from constant many other symbols is regular. We notice that the strategy is the same as for the previous proofs. We will try to reduce problems involving context-free languages to smaller problems over regular cases. This property is proven in the following lemma:

Lemma 17. *Every visibly pushdown language $L \in VPL$ with $L \subseteq L(a^*ba^*)$ is regular.*

Proof. Let $L \in$ VPL be a visibly pushdown language with $L \subseteq L(a^*ba^*)$. By definition there exists a visibly pushdown automaton $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_o, F)$ with $L(\mathcal{A}) = L$. The structure of $\mathcal{A}$ depends on the choice of the visibly pushdown alphabet $\Sigma = \Sigma_{\text{int}} \cup \Sigma_{\text{ret}} \cup \Sigma_{\text{call}}$.

Case 1: $a, b \in \Sigma_{\text{int}}$. In this case, by definition of a VPA, $\delta : Q \times \Sigma_{\text{int}} \to Q$. Hence δ has exactly the same form of the transition function of a finite automaton. Therefore $\mathcal{A}$ can trivially be understood as a finite automaton, ignoring the stack alphabet.

Case 2: $a \in \Sigma \setminus \Sigma_{\text{int}}, b \in \Sigma_{\text{int}}$. In this case an a can alter the stack either by pushing a symbol to the stack or by popping a symbol from the stack. In both cases the stack can either never be read from or never be written to, because b is internal. Hence, this case also collapses to a finite automaton, by simply ignoring all stack actions.

Case 3: $b \in \Sigma \setminus \Sigma_{\text{int}}, a \in \Sigma_{\text{int}}$. Analogue to Case 2.

Case 4: $a \in \Sigma_{\text{call}}, b \in \Sigma_{\text{ret}}$. In this case we will build a finite automaton $\mathcal{A}'$ which accepts the same language as $\mathcal{A}$. The idea behind this construction is that we take advantage of the stack restrictions of $\mathcal{A}$. Because a is a call symbol and b is a return symbol, the stack can only be read from once. Therefore the finite automaton will remember the current top stack symbol. We define $\mathcal{A}' = ((Q \times \Gamma), \Sigma, \delta', I \times \{\bot\}, F \times \Gamma)$ with $\delta' \subseteq (Q \times \Gamma) \times \Sigma \times (Q \times \Gamma)$ and:

$$\delta' = \{((p,g), a, (q,g')) \mid (p,a,q,g') \in \delta_{call}\}$$
$$\cup \{((p,g), b, (q,\bot)) \mid (p,b,g,q) \in \delta_{ret}\}$$

We will now show that $L(\mathcal{A}') = L(\mathcal{A})$:

$\subseteq$: Let $w \in L(\mathcal{A}')$ be a word accepted by $\mathcal{A}'$. Therefore, there exists a sequence of states such that:

$$(q_0, \bot) \xrightarrow{w_1} (q_1, g_1) \to \ldots \xrightarrow{w_n} (q_n, g_n)$$

for $|w| = n, q_0 \in I$, $q_n \in F$ and $g_i \in \Gamma$. By definition of δ' for every transition $(p,g) \xrightarrow{a} (q,g')$, there exists a equivalent transition in δ with $(p,a,q,g') \in \delta_{\text{call}}$. For $(p,g) \xrightarrow{b} (q,\bot)$ we use $(p,a,g,q) \in \delta_{\text{ret}}$ and do not alter the stack of $\mathcal{A}$. Hence the sequence of configurations:

$$(q_0, w_1 ... w_n, \bot) \vdash (q_1, w_2 ... w_n, g_1 \bot) \vdash \cdots \vdash (q_n, \epsilon, g_n \gamma)$$

is accepted in $\mathcal{A}$.

$\supseteq$: Let $w \in \mathcal{A}$ with $|w| = n$. We know by definition that $w \in L(a^*ba^*)$. Let $i \in \{1, \ldots, n\}$ be the index with $w_i = b$. Note that, because b only occurs once and is the only return symbol, only one state transition depends on the current top stack symbol. This state transition happens after reading w_i. Because w is accepted by $\mathcal{A}$, we find a sequence of configurations:

$$(q_0, w, \bot) \vdash \cdots \vdash (q_{i-1}, bw', g_{i-1} ... g_0 \bot) \vdash \cdots \vdash (q_n, \epsilon, g_n ... g_i g_{i-2} ... \bot)$$

with $q_0 \in I$ and $q_n \in F$. By definition of δ', $w \in L(\mathcal{A}')$ with:

$$(q_0, \bot) \xrightarrow{w_1} (q_1, g_1) \ldots (q_i, g_i) \xrightarrow{w_i} (q_{i+1}, \bot) \ldots (q_n, g_n)$$

Case 5: $a \in \Sigma_{\text{ret}}, b \in \Sigma_{\text{call}}$. In this case, the stack can only be written to once, because b only occurs once. We will use a similar construction as in Case 4. We define $\mathcal{A}' = ((Q \times \Gamma), \Sigma, \delta', I \times \{\bot\}, F \times \Gamma)$ with $\delta' \subseteq (Q \times \Gamma) \times \Sigma \times (Q \times \Gamma)$

and:

$$\delta' = \{((p,g),a,(q,\bot)) \mid (p,a,g,q) \in \delta_{ret}\} \\ \cup \{((p,g),b,(q,g')) \mid (p,b,q,g') \in \delta_{call}\}$$

Again we will show that $L(\mathcal{A}') = L(\mathcal{A})$

$\subseteq$: This argument works analogous to the argument in Case 4. For every transition $(p,g) \xrightarrow{a} (q,\bot)$ in $\mathcal{A}'$, there exists an equivalent transition in δ with $(p,a,g,q) \in \delta_{\text{ret}}$ which does not alter the stack of $\mathcal{A}$. For $(p,g) \xrightarrow{b} (q,g')$ we use $(p,b,q,g') \in \delta_{\text{call}}$. Hence the sequence of configurations:

$$(q_0, w_1...w_n, \bot) \vdash (q_1, w_2...w_n, g_1\bot) \vdash \cdots \vdash (q_n, \epsilon, g_n\gamma)$$

is accepted in $\mathcal{A}$.

$\supseteq$: Let $w \in \mathcal{A}$ with $|w| = n$. We know by definition that $w \in L(a^*ba^*)$. Let $i \in \{1,\dots,n\}$ be the index with $w_i = b$. Again note that because b occurs only once and is a call symbol this time, the stack will be empty all the time except from one possible a-transition after having read w_i. Because w is accepted by $\mathcal{A}$ we find a sequence of configurations:

$$(q_0, w, \bot) \vdash \cdots \vdash (q_{i-1}, bw', \bot) \vdash (q_i, w', g_i) \vdash \cdots \vdash (q_n, \epsilon, g_n)$$

with $q_0 \in I$ and $q_n \in F$. Note that $g_n = \bot$ if $w \neq b$. If $w = b$, then $g_n = g_i$. By definition of δ', $w \in L(\mathcal{A}')$ with:

$$(q_0, \bot) \xrightarrow{w_1} (q_1, \bot) \dots (q_{i-1}, \bot) \xrightarrow{w_i} (q_i, g_i) \dots (q_n, g_n)$$

□

With the help of this result, we can now prove the inexpressibility of the mentioned property in PDL[VPL] by reducing the problem to the regular case.

Lemma 18. *Let P be the property $\langle a^nba^n\rangle\mathbf{ff}$ for some $n \in \mathbb{N}$. There exists no formula φ over PDL[VPL], such that for all labelled transition systems $\mathcal{T}$ holds: $\mathcal{T} \models P \Leftrightarrow \mathcal{T} \models \varphi$*

Proof. Assume there is a formula $\varphi \in$ PDL[VPL] which satisfies the claim. Let $L_1,\dots,L_m$ be the visibly pushdown languages used in the formula and $n = \text{md}(\varphi)$ be the modal depth of φ. We will now show that φ cannot distinguish the two LTS from Figure 4.1. Without loss of generality, we assume that all L_i are either a subset from $L(a^*ba^*)$, or $L(b^*)$.

Notice that all used languages must be regular, because context-free languages over unary alphabets are regular and with Lemma 17, all languages which are subset from $L(a^*ba^*)$, too. Therefore, all languages which could be used in φ are regular and with Lemma 9 follows that there cannot be such a formula φ.

□

Theorem 19. *PDL[VPL] is strictly less expressive than PDL[CFL]*

Proof. The inclusion of PDL[VPL] in PDL[CFL] is trivial. This is, because visibly pushdown automata are a special form of general pushdown automata. Hence every formula over PDL[VPL] is also included in PDL[CFL]. The strict inclusions follows from Lemma 18.

□

Chapter 7

Conclusion

An overview on the separation results between modal fixpoint and propositional dynamic logics can be seen in Figure 7.1. We see that FLC is strictly more expressive than all mentioned logics. vpFLC is strictly less expressive than FLC as proved in [1]. We showed the strict inclusion of PDL[VPL] in vpFLC in Theorem 15 and even that vpFLC and PDL[CFL] are incomparable in terms of expressibility. Furthermore, we showed in Theorem 12 that PDL[REG] is strictly included in PDL[VPL].

As stated in the introduction, we proved that propositional dynamic logic over context-free programs cannot express non-regular properties which are spread across multiple modal operators. Intuitively, there is no reason why propositional dynamic logic over context-sensitive languages PDL[CSL] should be able to express these properties. However, the techniques used in this thesis cannot be easily adapted to PDL[CSL]. This is, because we took advantage of the fact that unary languages over CFL are regular. This allowed us to construct transition systems which were long enough that we could use a pumping lemma like property of regular languages to "fool" PDL[CFL] formulas which try to express properties like $\langle a \rangle^n [b]^n \mathbf{ff}$. Over CSL, unary languages are not longer regular. For example $L = \{a^{2^n} \mid n \in \mathbb{N}\}$. Therefore, the mentioned arguments cannot be used anymore. But even though the argument with regularity over unary languages does not hold in CSL, there is no obvious reason why the additional expressiveness of context-sensitive languages should help in expressing such properties.

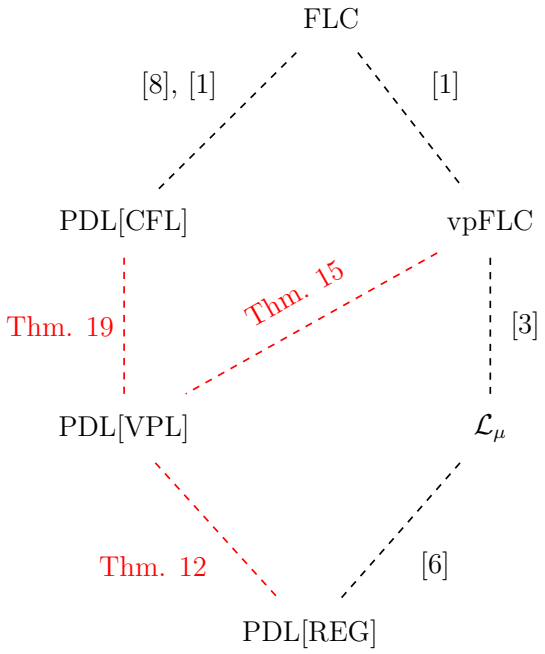


Figure 7.1: Expressiveness of logics mentioned in this thesis. Dashed lines are strict inclusions. Red lines are inclusions proved in this thesis.

Bibliography

[1] E. Alsmann, F. Bruse, and M. Lange. Separating the Expressive Power of Propositional Dynamic and Modal Fixpoint Logics, 2021.

[2] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing - STOC '04*, page 202, Chicago, IL, USA, 2004. ACM Press.

[3] F. Bruse and M. Lange. A Decidable Non-Regular Modal Fixpoint Logic. In *Proc. 32nd Int. Conf. on Concurrency Theory, CONCUR'21*, volume 203 of *LIPIcs*, pages ??–?? Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[4] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, Apr. 1979.

[5] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, Apr. 1983.

[6] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[7] C. Löding, C. Lutz, and O. Serre. Propositional Dynamic Logic with Recursive Programs. *Journal of Logic and Algebraic Programming*, 73:51–69, 2007.

[8] M. Müller-Olm. A Modal Fixpoint Logic with Chop. In C. Meinel and S. Tison, editors, *STACS 99*, Lecture Notes in Computer Science, pages 510–520, Berlin, Heidelberg, 1999. Springer.

[9] G. Pighizzini, J. Shallit, and M.-w. Wang. Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds. *Journal of Computer and System Sciences*, 65(2):393–414, Sept. 2002.

[10] M. O. Rabin and D. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.