Separation of Test-Free Propositional Dynamic Logics over Context-Free Languages

Markus Latte*
Department of Computer Science
University of Munich, Germany

For a class $\mathcal{L}$ of languages let PDL[$\mathcal{L}$] be an extension of Propositional Dynamic Logic which allows programs to be in a language of $\mathcal{L}$ rather than just to be regular. If $\mathcal{L}$ contains a non-regular language, PDL[$\mathcal{L}$] can express non-regular properties, in contrast to pure PDL.

For regular, visibly pushdown and deterministic context-free languages, the separation of the respective PDLs can be proven by automata-theoretic techniques. However, these techniques introduce non-determinism on the automata side. As non-determinism is also the difference between DCFL and CFL, these techniques seem to be inappropriate to separate PDL[DCFL] from PDL[CFL]. Nevertheless, this separation is shown but for programs without test operators.

1 Introduction

Propositional Dynamic Logic (PDL) \[9\] is a logical formalism to specify and verify programs \[12, 16, 11\]. These tasks rely on the satisfiability and model-checking problems. Applications in the field are supported by their relatively low complexities: EXPTIME- and PTIME-complete, respectively \[9\].

Formulas in PDL are interpreted over labeled transition systems. For instance, the formula $\langle p \rangle \varphi$ means that after executing the program $p$ the formula $\varphi$ shall hold. In this context, programs and formulas are defined mutually inductively. This mixture allows programs to test whether or not a formula holds at the current state. Additionally, programs are required to be regular over the set of atomic programs and test operations. For instance, the program while (b) do $p$; can be rendered as $\langle (b?:p)^*;!b \rangle \varphi$ to ensure that the loop is finite and that $\varphi$ holds when the loop terminates \[9\].

The small model property of PDL \[9\] cuts both ways. First, it admits a decision procedure for satisfiability, but secondly it restricts the expressivity to regular properties. As a consequence counting properties and, in particular, the nature of execution stacks cannot be expressed. The last consequence runs contrary to the verification of recursive programs.

A natural way to enhance the expressivity is to relax the regularity requirement. For a class $\mathcal{L}$ of languages let PDL[$\mathcal{L}$] denote the variation which requires that any program belongs to $\mathcal{L}$. For instance, we write a diamond as $\langle L \rangle \varphi$ for $L \in \mathcal{L}$. This leads to a hierarchy of logics. Obviously, PDL[$\mathcal{L}$] \leq PDL[$\mathcal{M}$] holds for $\mathcal{L} \subseteq \mathcal{M}$. Besides regular languages, we consider the variations for the class of visibly pushdown languages.

\[\text{DCFL} \subseteq \text{CFL} \subseteq \text{CFL} \subseteq \text{PDL}[\text{CFL}] \subseteq \text{PDL}[\text{DCFL}] \subseteq \text{PDL}[\mathcal{L}] \subseteq \text{PDL}[\mathcal{M}]\]

\[\text{For regular, visibly pushdown and deterministic context-free languages, the separation of the respective PDLs can be proven by automata-theoretic techniques. However, these techniques introduce non-determinism on the automata side. As non-determinism is also the difference between DCFL and CFL, these techniques seem to be inappropriate to separate PDL[DCFL] from PDL[CFL]. Nevertheless, this separation is shown but for programs without test operators.}\]

\[\text{A natural way to enhance the expressivity is to relax the regularity requirement. For a class } \mathcal{L} \text{ of languages let PDL[\mathcal{L}] denote the variation which requires that any program belongs to } \mathcal{L}. \text{ For instance, we write a diamond as } \langle L \rangle \varphi \text{ for } L \in \mathcal{L}. \text{ This leads to a hierarchy of logics. Obviously, PDL[\mathcal{L}] } \leq \text{ PDL[\mathcal{M}]} \text{ holds for } \mathcal{L} \subseteq \mathcal{M}. \text{ Besides regular languages, we consider the variations for the class of visibly pushdown languages.}\]

\[\text{DCFL} \subseteq \text{CFL} \subseteq \text{CFL} \subseteq \text{PDL}[\text{CFL}] \subseteq \text{PDL}[\text{DCFL}] \subseteq \text{PDL}[\mathcal{L}] \subseteq \text{PDL}[\mathcal{M}]\]
languages \[1\], VPL, the class of deterministic context-free languages, DCFL, and context-free languages, CFL. The inclusion order continues on the logics’ side.

\[
PDL = PDL[REG] \leq PDL[VPL] \leq PDL[DCFL] \leq PDL[CFL].
\]  

(1)

Harel et al. discussed the effect of adding single (deterministic) context-free programs to PDL \([13, 14, 12]\). The logic PDL[VPL] were introduced by Löding et al. \([17]\).

To handle the respective decision problems, the languages are represented by a machine model for the respective class. For each of these logics, any of its formula \(\varphi\) can be translated into an \(\omega\)-tree-automaton which recognizes exactly all tree-like models of \(\varphi\) where the out-degree of any node is globally bounded. Such a model exists iff \(\varphi\) is satisfiable. For PDL and PDL[REG] these tree-automata are finite-state \([23]\), for PDL[VPL] they are visibly pushdown tree-automata \([14, 17]\) and for PDL[DCFL] and PDL[CFL] they are tree-automata with unbounded number of stacks. The last notion is rather artificial. However, the stacks are used, first, to accumulate unfulfilled eventualities and to simulate the complementation of programs given as pushdown automata. Note that in the setting of visibly pushdown automata, only one stack suffices as \(\omega\)-VPLs are closed under complementation \([1]\) and under determinisation (for stair-parity conditions) \([18]\).

The first two inequalities in (1) are strict. In this paragraph we sketch the proofs for the first two inequalities. Consider the language \(L := \{c^nr^n \mid n \in \mathbb{N}\}\) over an alphabet \(\Sigma \supseteq \{c, r\}\). Hence, we have \(L \in VPL\) if we take \(c\) for a call and \(r\) for a return in a visibly pushdown alphabet for \(\Sigma\). Now, we claim that \(\vartheta := \langle L \rangle p\) is not expressible in PDL[REG] where \(p\) is a proposition. For the sake of contradiction, assume that there were such a formula. Restricted to linear models, the previous translation leads to a finite-state Büchi-automaton \(\mathcal{A}\) which recognizes those models. Let \(\mathcal{N}\) be sufficiently large—which depends on the pumping length and the, here omitted, encoding. Consider the following model of \(\vartheta\) for \(\mathcal{N}' = \mathcal{N}\).

![Diagram](image)

As \(\mathcal{A}\) accepts this model, it also accepts this transition system for \(\mathcal{N}' < \mathcal{N}\) due to the pumping lemma. However, this structure is not a model of \(\vartheta\). The separation of PDL[VPL] and PDL[DCFL] can be achieved in similar fashion. Take as program \(L := \{w^rh^R \mid w \in (\Sigma \setminus \{\varepsilon\})^*\} \in DCFL\) over an alphabet \(\Sigma \supseteq \{\varepsilon\}\). For any visibly pushdown alphabet for \(\Sigma\) its return-part is not empty in general. Using such a letter for the \(w\)-part in \(L\), an assumed visibly pushdown automaton for \(\langle L \rangle p\) operates on that part like a finite-state automaton. The same argumentation applies as for the first separation.

The separation for the last inequality in (1) is more cumbersome and intrinsic: For the satisfiability problem, the emptiness problem for finite-state and for visibly pushdown tree-automata is decidable \([19, 23, 22, 18]\). The emptiness problem for the tree-automata with an unbounded number of stacks can be considered as the halting problem for Büchi-Turing machines \([20]\). Indeed, the satisfiability problems for PDL[DCFL] and PDL[CFL] are \(\Sigma_1\)-complete \([13]\). Hence, both logics are not distinct by a “trivial” reason.

The standard translation \([23, 14, 17]\) from formulas to tree-automata bases on Hintikka-sets. For a fixed formula \(\vartheta\) and for every node of the given transition system the automaton for \(\vartheta\) guesses—among other things—the set of those subformulas of \(\vartheta\) which hold at that node. Informally speaking, the non-determinism is required to handle disjunctions in the given formula and to recognize the termination of a program in an expression such as \(\langle L \rangle \varphi\). Note that a language in DCFL might be not prefix-free. However, non-determinism is also the difference between DCFL and CFL. Hence, the translation seems not to suffice to separate PDL[DCFL] from PDL[CFL].
In this paper we make a step towards the separation of PDL[CFL] from PDL[DCFL]. For technical reasons we consider PDL[Ł] without the test operations like \( \varphi \)?—call the logic PDL\(_0[Ł]\)—and prove the separation of the corresponding logics. This restriction is proper as PDL\(_0\) is weaker than PDL\([5]\). Note that PDL\(_0[Ł]\) is exactly the EF/AG-fragment of CTL[Ł] \([2, 3]\). This logic is obtained from CTL by restricting the moments of until- and release-operations by languages in Ł. The separation of CTL[DCFL] and CTL[CFL] is unknown as well.

## 2 Preliminaries

Let Ł be an alphabet. For a finite word \( w \in \Sigma^* \) we write \(|w|\) for its length and \( w[i .. j] \) for its subword starting at index \( i \) and ending at index \( j \) where \( 0 \leq i \leq j < |w| \). Both indices are zero-based. For words \( u, v \in \Sigma^* \cup \Sigma^\omega \) their concatenation is written as \( uv \) and the reversal of \( u \) as \( u^R \). Concatenation is extended to sets in the usual way. The empty word is denoted by \( \epsilon \). A word \( u \in \Sigma^* \cup \Sigma^\omega \) is a (proper) prefix of \( w \in \Sigma^* \cup \Sigma^\omega \) iff there is \( v \in \Sigma^* \cup \Sigma^\omega \) such that \( uv = w \) (and \( v \neq \epsilon \)). The notation of a suffix is defined similarly. For two languages \( L_1 \) and \( L_2 \) their left quotient \( L_1 \setminus L_2 = \{ v \mid \exists u \in L_1. uv \in L_2 \} \). If one of both languages is a singleton we may replace the language by its single word. Standard notations are used \([15]\) for (deterministic) pushdown automata on finite words, DPDA and PDA, and (deterministic) context-free languages. Deterministic pushdown automata on \( \omega \)-words, \( \omega \)DPDA, are equipped with Büchi-acceptance conditions \([20]\).

Let \( \text{Prop} = \{p, q, \ldots\} \) be a set of propositions. A labeled transition system, LTS, is a triple \( \mathcal{T} = (\mathcal{I}, \rightarrow, \ell) \) consisting of a set of states \( \mathcal{I} \), of a labeled edge relation \( \rightarrow \subseteq \mathcal{I} \times \Sigma \times \mathcal{I} \) and of an evaluation function \( \ell : \mathcal{I} \rightarrow 2^{\text{Prop}} \). Write \( s \xrightarrow{a} t \) instead of \((s, a, t) \in \rightarrow\). A path is a sequence \( s_0, a_1, s_1, a_1, \ldots, a_{n-1}, s_n \) for some \( n \in \mathbb{N} \) such that \( s_i \xrightarrow{a_i} s_{i+1} \) for all \( i \in \{0, \ldots, n-1\} \). For such a path we may write \( s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n \). A structure is a pair \( \mathcal{M} = (\mathcal{I}, s) \) of an LTS and a state in it, called root. Previous notations for LTSS are also used for structures. A structure \( \mathcal{M} = ((\mathcal{I}', \rightarrow'), (\ell')') \) is an extension of \( \mathcal{M} \), written as \( \mathcal{M} \leq \mathcal{M}' \), iff \( \mathcal{I} \subseteq \mathcal{I}' \), \( \rightarrow \subseteq \rightarrow' \), \( \ell \) is the restriction of \( \ell' \) to \( \mathcal{I} \), and \( s = s' \).

Let Ł be a class of languages. We define the logic PDL\(_0[Ł]\) in negation normal form using a CTL-like syntax \([2]\)—that is, \( \text{EF}^L \varphi \) stands for the PDL-expression \( \langle L \rangle \varphi \) for instance. The formulas are given by the grammar

\[
\varphi ::= \; \begin{align*}
\text{ff} & \mid \text{tt} \mid p \mid \neg p \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid \text{EF}^L \varphi \mid \text{AG}^L \varphi
\end{align*}
\]

where \( p \in \text{Prop} \) and \( L \in \mathcal{L} \). Such formulas are denoted by \( \varphi, \psi, \varphi, \) and \( \delta \). The atoms \( \text{ff} \) and \( \text{tt} \) are called constants, and \( p \) and \( \neg \) are called literals. Implication and equivalence are definable. A formula \( \text{EF}^L \varphi \) is called EF-formula. An AG-formula is meant analogously. A formula is interpreted over a structure as follows.

\[
\mathcal{T}, s \models \text{ff} \quad \mathcal{T}, s \models \text{tt} \quad \mathcal{T}, s \models p \text{ iff } p \in \ell(s) \quad \mathcal{T}, s \models \neg p \text{ iff } p \not\in \ell(s)
\]

\[
\mathcal{T}, s \models \varphi_1 \lor \varphi_2 \text{ iff } \mathcal{T}, s \models \varphi_1 \text{ or } \mathcal{T}, s \models \varphi_2 \quad \mathcal{T}, s \models \varphi_1 \land \varphi_2 \text{ iff } \mathcal{T}, s \models \varphi_1 \text{ and } \mathcal{T}, s \models \varphi_2
\]

\[
\mathcal{T}, s \models \text{EF}^L \varphi \text{ iff there is path } s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n \text{ with } s = s_0, a_0 \cdots a_{n-1} \in L \text{ and } \mathcal{T}, s_n \models \varphi
\]

\[
\mathcal{T}, s \models \text{AG}^L \varphi \text{ iff for all paths } s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n \text{ with } s = s_0, a_0 \cdots a_{n-1} \in L : \mathcal{T}, s_n \models \varphi
\]

If \( \mathcal{T}, s \models \varphi \) then the structure \( (\mathcal{T}, s) \) is a model of \( \varphi \). A structure \( (\mathcal{T}, s) \) is tree-like iff \( \mathcal{T} \) forms a tree with root \( s \). Since PDL\(_0[Ł]\) is closed under bisimulation, every satisfiable formula has a tree-like structure as a model. A formula \( \varphi \) is a tautology, written as \( \models \varphi \), iff every structure is a model of \( \varphi \).
3 Outline of the Proof

For the following parts, fix an alphabet $\Sigma$ which at least contains 0 and 1 but not $\$, and set $\Sigma_0 := \Sigma \cup \{\}\$. The language of palindromes is denoted by $\text{Palindromes} := \{ w \in \Sigma^* \mid w = w^R \}$. We will show that there is no $\text{PDL}_0[\text{DCFL}]$-formula which is equivalent to the reference $\text{PDL}_0[\text{CFL}]$-formula $\text{EF}^\text{Palindromes}$. As the reference formula does not contain propositions we may assume that neither does any equivalent formula. Equivalently, we may assume that $\text{Prop} = \emptyset$.

For the sake of contradiction, let $\vartheta \in \text{PDL}_0[\text{DCFL}]$ be a candidate formula which is assumed to be equivalent to $\text{EF}^\text{Palindromes}$. To illustrate the main problem about provoking a contradiction, we begin with a simpler setting in which $\vartheta$ does not contain any conjunctions or $\text{AG}$-formulas. As we have the equivalences

$$\text{EF}^{L, \emptyset} \leftrightarrow \emptyset, \quad \text{EF}^{L_1, \emptyset} \varphi \leftrightarrow \text{EF}^{L_1, \emptyset} \varphi, \quad \text{and} \quad \bigvee_i \text{EF}^{L_i} \varphi \leftrightarrow \text{EF}^{\bigcup_i L_i} \varphi$$

the formula $\vartheta$ can be rewritten as

$$\text{EF}^{\bigcup_i L_{i,1} \cdots L_{i,n_i}} \emptyset$$

where $L_{i,j}$ are DCFLs over $\Sigma_0$. In general, an equivalence $\text{EF}^\text{Palindromes} \emptyset \leftrightarrow \text{EF} \emptyset$ implies

$$\text{Palindromes} = \{ w \in \Sigma^* \mid w\$ is a prefix of a word in $L\}$$

for $L \subseteq \Sigma_0^+$. Therefore, we have that Palindromes would be expressible as a finite union over a finite concatenation over DCFLs over $\Sigma$. Some combinatorial argument shows that this is impossible.

Back to the real world, we are also faced with conjunctions and $\text{AG}$-formulas in $\vartheta$. A natural attempt is to eliminate these subformulas. Indeed, a conjunction seems not to support a statement which speaks about a single path only. Instead, it speaks about a bunch of paths. Similarly, an $\text{AG}$-formula is not monotone with respect to models but the reference formula is monotone. To turn off such formulas, one could saturate the considered structures with substructures which falsify $\text{AG}$-formulas and which do not affect the desired property $\text{EF}^\text{Palindromes} \emptyset$. However, on such a new structure, the attached substructures could be recognized by other $\text{EF}$-subformulas. But these subformulas need not to be concerned with palindromes in any reasonable way. Moreover, Bojańczyk proved [6]—for the dual setting—that such an elimination procedure is only possible if—in our setting—palindromes were expressible as a finite union of languages of the form $A_0^*a_1A_1^*a_2 \cdots A_{n-1}^*a_nA_n^*$ for $a_1, \ldots, a_n \in \Sigma$ and $A_0, \ldots, A_n \subseteq \Sigma$. Obviously, this is not the case.

Therefore, our strategy is different. First, we show that topmost $\text{AG}$-formulas and topmost conjunctions can be eliminated ($\S\ref{sec:elimination-ag}$ and $\S\ref{sec:elimination-conjunctions}$). This renders the candidate formula $\vartheta$ as $\bigvee_i \text{EF}^{L_i} \varphi_i$ for some $L_i \subseteq \Sigma_0^+$ and some formulas $\varphi_i$ with unknown structure. Secondly, if $L_i$ is not a singleton language then the formula $\text{EF}^{L_i} \varphi_i$ per se provides all the information required for a contradiction. Either it under- or over-approximates palindromes. And if $L_i$ is a singleton we proceed in a similar way with the left-quotient of $\vartheta$ with the only word in $L_i$. The whole procedure ($\S\ref{sec:combined-strategy}$) terminates through a sophisticated measure ($\S\ref{sec:saturation}$). The case that $L_i$ is not a singleton give rise to a characterization of languages which will bridge between the formula and the language part of the separation proof.

**Definition 1.** A language $L \subseteq \Sigma^*$ is good iff $L = \bigcup_{i \in I} L_i R_i$ such that $I$ is finite, and for each $i \in I$, the language $L_i$ is a DCFL, $|L_i| \geq 2$ and $R_i \subseteq \Sigma^*$.  

In the view of Bojańczyk’s result, our iterated elimination is non-uniform compared to the preferable approach in the previous paragraph. Finally, we show on the language-theoretical level that palindromes are not good ($\S\ref{sec:language-theoretical-level}$).
4 On Palindromes and DCFLs

In this section it is proven that the language of palindromes is not good. For this purpose we first show that this language is not expressible as a union of DCFLs (Theorem 3). Although it is known that the set of palindromes is not deterministic context-free, the standard proof [10, Cor. 1] does not seem to be adaptable because the applied min-operator does not commute with the union. As a second step, it is shown that if palindromes are underapproximated by a concatenation then the components of the concatenation follow a very simple pattern (Lemma 5).

Lemma 2 (Pumping lemma). Let \( u \in \Sigma^\omega \) be accepted by an \( \omega \)DPDA \( \mathcal{A} \). There are words \( u_0 \in \Sigma^* \), \( u_1 \in \Sigma^+ \) and \( u_2 \in \Sigma^\omega \) such that \( u_0u_1u_2 = u \), and \( u_0u_2 \) is accepted by \( \mathcal{A} \).

Proof. Firstly, we may assume that \( \mathcal{A} \) only erases or pushes symbols from or on the stack and never changes the topmost symbol. Indeed, an \( \omega \)DPDA can keep the topmost element of the stack in its control state [15, Sect. 10.1]. By this restriction, in any run the stacks of two consecutive configuration are comparable with respect to the prefix-order. Secondly, consider the infinitely many stair positions in the accepting run of \( \mathcal{A} \) on \( u \). By a stair position [18] we understand a position such that the current stack content is a prefix of all further stack contents in this run. As the set of states is finite, there are two different stair positions which name the same state. We may assume that a non-empty part of \( u \), say \( u_1 \) with \( u = u_0u_1u_2 \), fits into their gap. Hence, this part can be removed. By the definition of stairs, the obtained sequence of configurations is a run of \( \mathcal{A} \) on \( u_0u_2 \). As the modification affects a prefix of \( u \) only, \( \mathcal{A} \) also accepts \( u_0u_2 \).

Theorem 3. Let \( v \in \Sigma^* \), \( n \in \mathbb{N} \), and \( L_1, \ldots, L_n \) be DCFLs over \( \Sigma \). Then \( \bigcup_{i=1}^n L_i \neq v \setminus \text{Palindromes} \).

Proof. Define the sequence \( (w_i)_{i \in \mathbb{N}} \) of strictly prefix-ordered words as follows.

\[
\begin{align*}
    w_0 &= v^R, \\
    w_{i+1} &= w_i 10^i 1 w_i^R v^R \quad (i \in \mathbb{N})
\end{align*}
\]

For all \( i \in \mathbb{N} \) we have \( w_i \in v \setminus \text{Palindromes} \). For the sake of contradiction, assume that

\[
\bigcup_{i=1}^n L_i = v \setminus \text{Palindromes}.
\]

We sample the candidate on the left of Eq. 2 with the words \( \{w_i\}_{i \in \mathbb{N}} \). Since the union is finite, there is an infinite \( I \subseteq \mathbb{N} \) and an \( i \in \{1, \ldots, n\} \) such that the words \( \{w_i\}_{i \in I} \) belong to \( L_i \). Let \( \mathcal{A} \) be a DPDA for \( L_i \). Additionally, we consider \( \mathcal{A} \) as an \( \omega \)DPDA where the final states are the B"uchi-states. Hence, as \( \mathcal{A} \) is a deterministic device it accepts

\[
w := \lim_{i \to \infty} w_i = \lim_{i \to \ell} w_i \in \Sigma^\omega.
\]

Apply Lemma 2 to \( \mathcal{A} \) and \( w \). Let \( u_0, u_1, u_2 \) be the obtained factors. We run \( \mathcal{A} \) on \( w \) for at least \( |u_0u_1| \) steps until it processes some subword \( 10^k 1 \) for the first time. Note that the function which maps \( i \in \mathbb{N} \) to the first occurrence of \( 10^k 1 \) in \( w \) is unbounded. Let \( \ell \) be the first index in \( w \) after that subword. So far, \( \mathcal{A} \) has seen the first \( \ell \) letters in \( w \). We keep \( \mathcal{A} \) running for at least another \( \ell + |v| \) steps until it reaches a final state. Such a run is always possible as \( \mathcal{A} \) accepts infinitely many prefixes of \( w \). Let \( u' \) be the word constructed in this way. Hence, \( u' \in v \setminus \text{Palindromes} \) as \( \mathcal{A} \) accepts \( u' \).
Let \( u'' \) be the word \( u' \) where the \( u_1 \)-block is removed. That is \( u'' := u'[0 .. |u_0| - 1] u'_[|u_0| u_1 .. |u'| - 1] \).

Again by construction and Lemma 2, \( \mathcal{A} \) accepts \( u'' \). Thus, \( u'' \in v \backslash \text{Palindromes} \). Let \( \hat{u} \) be the word between \( u_1 \) and the block \( 10^k 1 \), that is \( \hat{u} = w[|u_0 u_1| .. \ell - 3 - k] \). As \( vu' \) is a palindrome, it ends in the word \((vu_0 u_1 \hat{u} 10^k)^R \) of length \( \ell + |v| \). The modification leading to \( u'' \) affects at most the first \( \ell \) positions only. Hence, as \( |u'| \geq 2 \ell + |v| \), \( u'' \) also ends in \((vu_0 u_1 \hat{u} 10^k)^R \). As \( vu'' \) is also a palindrome, \( u_0 u_1 \hat{u} 10^k \) is a prefix of \( u_0 u_1 \hat{u} 10^k 1 \). Since \( u_1 \) is not the empty word, this is a contradiction to the choice of \( 10^k 1 \).

**Lemma 4.** If \( LR \subseteq \text{Palindromes} \) and \( R \) is infinite then \( L \) is prefix-ordered.

**Proof.** Let \( \ell_0, \ell_1 \in L \) with \( |\ell_0| \leq |\ell_1| \). Take \( r \in R \) such that \(|r| \geq |\ell_1| \). This is possible as \( R \) is infinite. Since \( \ell_0 r \) and \( \ell_1 r \) are palindromes, \( \ell_0^R \) and \( \ell_1^R \) are suffixes of \( r \). Therefore, \( \ell_0 \) is a prefix of \( \ell_1 \).

**Lemma 5.** Suppose \( LR \subseteq \text{Palindromes} \), \( |L| \geq 2 \) and \( R \) is infinite. Then

\[ R \subseteq \hat{u}^* \hat{U} \]

for some word \( \hat{u} \in \Sigma^* \) and a finite language \( \hat{U} \subset \Sigma^* \).

**Proof.** Let \( u_0, u_1 \) be two distinct words in \( L \). By the Lemma 4, we may assume that \( u_0 \) is a proper prefix of \( u_1 \). Define \( \hat{u} := u_0 \backslash u_1 \). Note that \( u_0 \hat{u} = u_1 \).

**Claim 5-1.** For \( w \in R \) and \( n \in \mathbb{N} \) we have

\( (i) \) \( \hat{u}^n \) is a prefix of \( w \), and

\( (ii) \) \( (\hat{u}^R)^n u_0^R \) is a suffix of \( w \)

if \( n|\hat{u}| + |u_0| \leq |w| \).

**Proof of claim.** By induction on \( n \) for a fixed \( w \in R \). If \( n = 0 \), \( u_0^R \) is a suffix of \( w \) as \( u_0 w \) is a palindrome. For the step case from \( n \) to \( n + 1 \) assume that

\[ (n + 1)|\hat{u}| + |u_0| \leq |w|. \] (4)

The word \( v := u_0 \hat{u}^{n+1} = u_1 \hat{u}^n \) is prefix of \( u_1 w \) by Lemma 4. As \( u_1 w \) is palindrome, \( v^R \) is a suffix of \( w \) because of (4). This proves the second item. Since \( u_0 w \) is also a palindrome, it has \( v \) as a prefix. Hence \( \hat{u}^{n+1} \) is a prefix of \( w \)—this is the first item.

Let \( w \in R \). For \( N_w := [(|w| - |u_0|)/|\hat{u}|] \), the claim yields \( w = \hat{u}^N w \) where \( \hat{w} \) are the \( r_w := |w| - N_w |\hat{u}| \) last letters of \( w \). Since \( r_w \leq |u_0| + |\hat{u}| \) is bounded independently of \( w \), there is a finite set \( \hat{U} \) such that \( R \subseteq \hat{u}^* \hat{U} \).

**Lemma 6.** Let

\[ L := \bigcup_{i \in I} u_{i,0} u_{i,1} u_{i,2} u_{i,3} \] (5)

for \( I \) finite and \( u_{i,j} \in \Sigma^* \) for all suitable indices. Then there is a word \( w \in \Sigma^* \) which is not a prefix of any word in \( L \).
Proof. Consider the tree $\Sigma^ω$. For each $i \in I$, the word $u_{i,0}^ω u_{i,1}^ω$ defines a (finite or infinite) path in the tree. As $I$ is finite, there is a $w_0 \in \Sigma^*$ which is not on these paths. There are at most $|w_0| \cdot |I|$ paths of the form $u_{i,0}^ω u_{i,1}^ω$ for $i \in I$ and $j \in \mathbb{N}$ which pass $w_0$. By the same argument, we get a word $w_1$ which extends $w_0$ and cannot be reached by these paths. A final application to $w_1$ and $u_{i,0}^ω u_{i,1}^ω u_{i,2}^ω u_{i,3}$ for $i \in I$ and $j, k \in \mathbb{N}$ yields the claimed word $w$.  

Corollary 7. The set Palindromes is not good.

Proof. For the sake of contradiction, assume the contrary, that is

$$\text{Palindromes} = \bigcup_{i \in I} L_i R_i$$

(6)

where $I$ is finite, and for any $i \in I$ the language $L_i$ is a DCFL, and $|L_i| \geq 2$. Set $I^+ := \{i \in I \mid R_i \text{ is finite}\}$, and $I^- := I \setminus I^+$. For any $i \in I^+$, we have that $L_i R_i$ is a DCFL [10 Thm. 3.3] as $R_i$ is finite in particular.

Let $i \in I^-$. Since $|L_i| \geq 2$ and $R_i$ is infinite, Lemma 8 shows that $R_i \subseteq r_i^* \hat{R}_i$ for some $r_i \in \Sigma^*$ and for a finite language $\hat{R}_i \subset \Sigma^*$. Depending on the size of $L_i$ we can bound $L_i R_i$. If $L_i$ is infinite then the very same lemma shows by reversal that $L_i \subseteq \hat{L}_i \ell_i^*$ for some $\ell_i \in \Sigma^*$ and a finite $\hat{L}_i \subset \Sigma^*$. Hence,

$$L_i R_i \subseteq \hat{L}_i \ell_i^* r_i^* \hat{R}_i = \bigcup_{x, y \in \hat{R}_i} x \ell_i^* r_i^* y.$$ 

In the other case—$|L_i|$ is finite—one obtains

$$L_i R_i \subseteq \bigcup_{x \in L_i, y \in \hat{R}_i} x r_i^* y.$$ 

In both cases, the unions are finite. All in all, we have

$$\bigcup_{i \in I} L_i R_i = \bigcup_{i \in I^+} L_i R_i \cup \text{DCFL} \cup Q'$$

where $Q' \subseteq Q := \bigcup_{i \in I} u_{i,0}^ω u_{i,1}^ω u_{i,2}^ω u_{i,3}$ for some finite set $J$, and some words $u_{i,0}, \ldots, u_{i,3} \in \Sigma^*$. By Lemma 6 there is a finite word $w$ which is not a prefix of any word in $Q$. Using (6), we get

$$w \setminus \text{Palindromes} = \bigcup_{i \in I^+} w \setminus (L_i R_i).$$

The left quotient with a single word $w$ is the inverse of the gsm mapping which sends a word $u$ to $wu$. As DCFLs are closed under the inverse of gsm mappings [10 Thm. 3.2], the language $w \setminus (L_i R_i)$ is a DCFL for $i \in I^+$. But this a contradiction to Theorem 3.  

5 A Measure for the Extraction

Informally, the measure of a formula is a set of vectors. Each vector measures the languages annotated to EF-subformulas along a path from the root of the formula to its atoms. For the measure of a language, the size of its only word is considered if the language is a singleton.
Definition 8. Let $\mathbb{M}$ be the set of all finite subsets of $(\omega + 1)^*$ where $\omega + 1 = \{0, 1, 2, \ldots, \omega\}$. The second argument of the cons-operator $\cdot \cdot \cdot$ on $(\omega + 1)^*$ is extended to sets. The empty list is written as nil. The measure of a formula is defined by

$$
\mu(\ell) := \{\text{nil}\} \quad \text{for } \ell \text{ a literal or a constant}
$$

$$
\mu(\phi_0 \circ \phi_1) := \mu(\phi_0) \cup \mu(\phi_1) \quad \text{for } \circ \in \{\land, \lor\}
$$

$$
\mu(Q^\ell \phi) := ||L|| \quad \text{for } Q \in \{\text{EF, AG}\}
$$

where

$$
||L|| = \begin{cases} 
|w| & \text{if } L = \{w\} \text{ for some } w \in \Sigma^*, \\
\omega & \text{otherwise.}
\end{cases}
$$

Lemma 9. The lexicographic order [4, Sect. 2.4], $>_{\text{lex}}$, on $(\omega + 1)^*$ is defined by

$$(\omega + 1)^n \ni (u_1, \ldots, u_n) >_{\text{lex}} (v_1, \ldots, v_m) \in (\omega + 1)^m$$

iff $n > m \lor (n = m \land \exists k < n. u_k = v_k \land \forall i < k. u_i > v_i)$, where $>$ is the natural order on $\omega + 1$, that is $\omega > \ldots > 1 > 0$.

Definition 10. The binary relation $>_{\mathbb{M}}$ on $\mathbb{M}$ is defined as follows.

$M >_{\mathbb{M}} N$ iff there are $X, Y \in \mathbb{M}$ such that $\emptyset \not= X \subseteq M$, $N = (M \setminus X) \cup Y$, and $\forall y \in Y \exists x \in X \cdot x >_{\text{lex}} y$.

Lemma 11. The relation $>_{\mathbb{M}}$ is a strict and terminating order.

Proof. We follow Baader and Nipkow [4]. The natural order on $\omega + 1$ is strict and terminating. Hence, so is $>_{\text{lex}}$ [4, Lemma 2.4.3]. Therefore, the multiset order on $(\omega + 1)^*$ is also strict and terminating [4, Lemma 2.5.4 and Theorem 2.5.5]. Due to the natural embedding of $\mathbb{M}$ into the set of finite multisets on $(\omega + 1)^*$, the relation $>_{\mathbb{M}}$ is dominated by the multiset order. Hence $>_{\mathbb{M}}$ is terminating. Thanks to the same embedding, $>_{\mathbb{M}}$ is a strict order $\Box$

We write $\geq_{\mathbb{M}}$ for the reflexive closure of $>_{\mathbb{M}}$. Similarly, $\leq_{\mathbb{M}}$ and $<_{\mathbb{M}}$ are meant.

### 6 $\varepsilon$-Free Formulas

Formulas like $\text{EF}^L \psi$ and $\text{AG}^L \psi$ can speak about the current state if $\varepsilon \in L$. We intend to combine structures at their roots—in the proof to Thm. [17] and [18], such that formulas should not realize this modification. Nonetheless, formulas can be transformed accordingly.

Definition 12. The property being $\varepsilon$-free is inductively defined on $\text{PDL}_0[\cdot]$-formulas.

(i) Any literal is $\varepsilon$-free.

(ii) A conjunction and a disjunction is $\varepsilon$-free if both conjuncts or both disjuncts, respectively, are $\varepsilon$-free.

(iii) $\text{EF}^L \phi$ and $\text{AG}^L \phi$ are $\varepsilon$-free iff $\varepsilon \notin L$ and $\phi$ is $\varepsilon$-free.

Definition 13. The function $^\varepsilon$ is defined on $\text{PDL}_0[\cdot]$-formulas

$$
^\varepsilon \ell := \ell \quad \text{where } \ell \text{ literal or a constant}
$$

$$
^\varepsilon \phi_0 \circ ^\varepsilon \phi_1 := ^\varepsilon \phi_0 \circ ^\varepsilon \phi_1 \quad \text{for } \circ \in \{\land, \lor\}
$$

$$
^\varepsilon Q^\ell \phi := \begin{cases} 
Q^\ell \phi & \text{if } \varepsilon \notin L \\
\phi^\varepsilon \lor Q^\ell\setminus\{\varepsilon\} \phi^\varepsilon & \text{otherwise if } Q = \text{EF} \\
\phi^\varepsilon \land Q^\ell\setminus\{\varepsilon\} \phi^\varepsilon & \text{otherwise if } Q = \text{AG}
\end{cases}
$$

for $Q \in \{\text{EF, AG}\}$. We write $^\varepsilon L$ and $^\varepsilon Q^\ell \phi$ if $\varepsilon \notin L$ and $\phi$ is $\varepsilon$-free.
Lemma 14. For every $PDL_0[\cdot]$-formula $\phi$ we have,

(i) $\phi$ and $\phi^{\varepsilon}$ are equivalent,

(ii) $\phi^{\varepsilon}$ is $\varepsilon$-free, and

(iii) $\mu(\phi^{\varepsilon}) \leq M \mu(\phi)$.

Proof. Each item can be proven by induction on $\phi$. We detail on the last item for the second case of $Q^L \phi^{\varepsilon}$. As the IH yields $\mu(\phi^{\varepsilon}) \leq M \mu(Q^L \phi^{\varepsilon})$, we have $\mu(Q^L \phi^{\varepsilon}) \leq M \mu(Q^L \phi)$.

Hence, the claim follows by $\mu(Q^L \phi^{\varepsilon}) \leq M \mu(Q^L \phi)$.

7 Elimination of Outermost $AG$-Formulas and Conjunctions

Although it is impossible to eliminate conjuncts and $AG$-formulas in general, the topmost ones can be removed (Thm. 17 and 18). Hence, if $\psi$ is equivalent to $EF_Ltt$ for some language $L$ then $\psi$ can be rearranged to a disjunction of $EF$-formulas only. However, these $EF$-formulas might contain conjunctions and $AG$-formulas in turn.

Definition 15. A formula $\psi$ is in disjunctive normal form (DNF for short) iff it has the shape

$$
\bigvee_{i \in I} \left( \bigwedge_{j \in J_i^A} \alpha_{i,j} \land \bigwedge_{j \in J_i^E} \varepsilon_{i,j} \right)
$$

where $I$, $J_i^A$ and $J_i^E$ are finite sets, $\alpha_{i,j}$ is an $\varepsilon$-free $AG$-formula, and $\varepsilon_{i,j}$ is an $\varepsilon$-free $EF$-formula (for all suitable indices). The completion of $\psi$ is

$$
\psi^* := \psi \lor \bigvee_{\Psi' \vdash \psi^* \rightarrow \psi} \bigwedge \Psi'
$$

where $\Psi := \{ \varepsilon_{i,j} \mid i \in I, j \in J_i^E \}$. A formula $\psi'$ is complete iff it is $\psi^*$ for some $\psi$. The term “DNF” and “complete” shall be applied up to associativity and commutativity of the Boolean connectives.

Lemma 16. For any $\varepsilon$-free formula $\psi$ we have

(i) an equivalent formula $\psi'$ in DNF such that $\mu(\psi') \leq M \mu(\psi)$, and

(ii) that $\psi^*$ is a DNF, $\psi^*$ and $\psi$ are equivalent, and $\mu(\psi^*) \leq M \mu(\psi)$.

Proof. To get a DNF, the distributive law is applied where $AG$- and $EF$-formulas are taken as atoms. This application might rearranging (positive) Boolean connectives and might duplicate atoms. However, the measure is defined in terms of unions for these cases.

For the second item, the implication to $\psi$ follows from the definition of the additional disjuncts. The other direction is weakening. As the additional terms are build only of top-level $EF$-formulas in $\psi$, their measure is already subsumed in $\mu(\psi)$. Note that $\mu$ is just the union in the case of the (positive) Boolean connectives.

\[\text{Note that this is well-defined when the measure } \mu \text{ is taken.}\]
For two structures $\mathcal{M}_1$ and $\mathcal{M}_2$ we define $\mathcal{M}_1 \oplus \mathcal{M}_2$ as the disjoint sum of both structures but with the root shared. The evaluation of the root is fixed as $\text{Prop} = \emptyset$ for our purposes. The notation is extended to sequences of structures, say $(\mathcal{M}_i)_{i \in I}$, in the usual way, written as $\oplus_{i \in I} \mathcal{M}_i$.

A formula $\psi$ is structurally monotone iff for any model of $\psi$ any of its extension is also a model of $\psi$. An example is $\text{EF}^L$ for any language $L$.

**Theorem 17** (Elimination of $\text{AG}$-formulas). Let

$$\psi := \bigvee_{i \in I} \left( \alpha_i \land \bigwedge_{j \in J_i} \varepsilon_{i,j} \right) =: \tau_i$$

be complete where $I, J_i$ are finite, each $\alpha_i$ is a (possibly empty) conjunction of $\varepsilon$-free $\text{AG}$-formulas, and each $\varepsilon_{i,j}$ is a $\varepsilon$-free $\text{EF}$-formula. If $\psi$ is structurally monotone, then $\psi$ is equivalent to

$$\psi' := \bigvee_{i \in I} \left( \bigwedge_{j \in J_i} \varepsilon_{i,j} \right) =: \tau_i'$$

Note that $\mu(\psi') \leq \mu(\psi)$.

**Proof.** $\models \psi \rightarrow \psi$ is obvious. As the considered logic is closed under bisimulation, we consider tree-like structures in the following only. For the other direction, let $\mathcal{M}$ be a model of $\psi$. We have to show that $\mathcal{M}$ is also a model of $\psi'$. If there is an $i \in I$ such that $\models \alpha_i$ and $\mathcal{M} \models \tau_i$ then $\mathcal{M} \models \tau_i'$ and we are done. Otherwise, there is an $i_0 \in I$ such that $\not\models \alpha_{i_0}$ and $\mathcal{M} \models \bigwedge_{j \in J_{i_0}} \varepsilon_{i,j}$, as $\mathcal{M} \models \psi$. For $i \in I$ define

$$J_i^+ := \{ j \in J_i \mid \mathcal{M} \models \varepsilon_{i,j} \} \quad \text{and} \quad J_i^- := J_i \setminus J_i^+.$$

There are two cases. Either

$$\bigvee_{i \in I} \alpha_i \land \bigwedge_{j \in J_i^-} \varepsilon_{i,j}$$

is a tautology or not. If (11) is a tautology then so is

$$\bigwedge_{i \in I} \bigwedge_{j \in J_i^+} \varepsilon_{i,j} \rightarrow \psi$$

as a simple case distinction on (11) shows. Indeed, let $\mathcal{N}$ be a model of the left side of (12). Then there is an $i \in I$ such that $\mathcal{N} \models \alpha_i \land \bigwedge_{j \in J_i^-} \varepsilon_{i,j}$. Both together lead to $\mathcal{N} \models \alpha_i \land \bigwedge_{j \in J_i^+} \varepsilon_{i,j}$ and finally to $\mathcal{N} \models \psi$. Hence, the left hand side of (12) is a term in $\psi$ as the latter is complete. But, by definition, this term is modeled by $\mathcal{N}$.

Otherwise (11) is not a tautology. So there is a structure $\mathcal{M}'$ with

$$\mathcal{M}' \not\models \alpha_i \land \bigwedge_{j \in J_i^-} \varepsilon_{i,j}$$

(13)
for all $i \in I$. We will exclude this situation. As $J^i_0 = \emptyset$ we have $\mathcal{M}' \not\models \alpha_i$ in particular. Now let $\mathcal{M}'' := \mathcal{M} \oplus \mathcal{M}'$. We claim that $\mathcal{M}'' \not\models \psi$ which is a contradiction to the assumption that $\psi$ is structurally monotone. For the sake of contradiction, suppose that there is an $i \in I$ such that $\mathcal{M}'' \models \tau_i$. Among the EF-formulas only those indexed by $J^i_+ \uparrow$ are already fulfilled in $\mathcal{M}$. Hence, $\alpha_i \land \bigwedge_{j \in J} \varepsilon_{i,j}$ must be satisfied by $\mathcal{M}'$. This is a contradiction to the choice of $\mathcal{M}'$. Note that we used implicitly that EF-formulas are $\varepsilon$-free.

The theorem requires a syntactical presence of formulas called $\alpha_i$. Note that minor changes make the proof also working if not all such parts are present. On the other hand, inserting such an empty conjunction does not increase the measure as atoms—such as $\tt$—have the lowest measure anyway.

**Theorem 18** (Elimination of $\land$ EF-formulas). Suppose

$$\EF^L \tt = \delta \lor \bigwedge_{i \in I} \EF^i \psi_i \tag{14}$$

where $\varepsilon \notin L_i$ for all $i \in I$. If $I \neq \emptyset$ then there is an $i \in I$ such that

$$\EF^L \tt = \delta \lor \EF^i \psi_i. \tag{15}$$

Note that the measure of (15) is bounded by that of (14), trivially.

**Proof.** For any $i \in I$, (14) implies (15). If there is an $i \in I$ with $\models \EF^L \psi_i \to \EF^L \tt$, this $i$ suffices for the other direction. To exclude the other case, assume that we have tree-like structures $\mathcal{M}_i$ for all $i \in I$ such that

(i) $\mathcal{M}_i \models \EF^L \psi_i$ but

(ii) $\mathcal{M}_i \not\models \EF^L \tt$.

Let $w_i \in L_i$ be the witness for the first item. Set $\mathcal{M} := \oplus_{i \in I} \mathcal{M}_i$. The root of $\mathcal{M}$ might satisfy different formulas than the root of $\mathcal{M}_i$, but this change is invisible to $\EF^L \psi_i$ since $|w_i| > 0$. Hence, $\mathcal{M} \models \EF^L \psi_i$. For the sake for a contradiction, assume that $\mathcal{M} \models \EF^L \tt$. This property depends only on a path in $\mathcal{M}$. The path is inherited from some $\mathcal{M}_i$ for $i \in I$. Since $\EF^L \tt$ does not depend on the evaluation of the root, $\mathcal{M}_i \models \EF^L \tt$ which is a contradiction to the second property of $\mathcal{M}_i$. Therefore, $\mathcal{M} \not\models \EF^L \tt$. By construction we have $\mathcal{M} \models \bigwedge_{i \in I} \EF^L \psi_i$ but $\mathcal{M} \not\models \EF^L \tt$. This property contradicts (14). \qed

8 Extraction

In the proof of Theorem 22 we apply previous elimination techniques to show that the candidate formula is equivalent to $\lor_i \EF^L \psi_i$. In the case that $L_i$ is not a singleton set, we cannot decompose $\psi_i$ any further. Indeed, the proof relies on the property $w(w \setminus L) \subseteq L$ for any language $L$ and word $w$. However, this inequality is false when $w$ is replaced by a non-singleton language. Nevertheless if the term $\EF^L \psi_i$ accepts a linear structure then the term factorises the word on the structure. The left factor is $L_i$, surely, and the right one can be read off as follows.

**Definition 19.** Let $\varphi$ be a PDL$[\cdot \cdot]$-formula. Its language is $\mathcal{L}(\varphi) := \{w \in \Sigma^* \mid \pi_w \models \varphi\}$ where $\pi_{a_1 \ldots a_n}$ is a path labeled with $a_1$ to $a_n$ for $a_1, \ldots, a_n \in \Sigma$. The node reached after $n$ steps has no successor. In each node, no proposition holds.

**Lemma 20.** $\mathcal{L}(\EF^L \tt) = L$ for any $L \subseteq \Sigma^*$. 
Additionally, the measure of (17) is weakly bounded by that of (17).  

Lemma 21. Let $L_0 \subseteq \Sigma^*_\mathbf{s}$, $L \subseteq \Sigma^*$, let $\delta$ be a formula, and let $\psi$ be a satisfiable formula. Suppose 

$$EF^{L_0} \psi = \delta \lor EF^{L_0} \psi.$$  

(16) 

Define $L_1 = L_0 \cap \Sigma^*$ and $L_2 = \{w \in \Sigma^* \mid w \text{ is a prefix of a word in } L_0\}$. Then 

$$\delta \lor EF^{L_0} \psi = \delta \lor EF^{L_1} \psi \lor EF^{L_2} \psi.$$  

(17) 

Additionally, the measure of (17) is weakly bounded by that of (17).  

Proof. Case $\rightarrow$: Note that $\models EF^{L_0} \psi \rightarrow EF^{L_1} \psi \lor EF^{L_2} \psi$ since for every word $w \in L_0 \setminus L_1$ there is a prefix of $w$ in $L_2$. Case $\leftarrow$: Since $L_1 \subseteq L_0$, $\models EF^{L_1} \psi \rightarrow EF^{L_0} \psi$ holds. So, we assume a model $\mathcal{M}$ of $EF^{L_2} \psi$. Hence there is a path $\pi$ in $\mathcal{M}$ labeled with a word $uS \in L_2$. Let $v \in \Sigma^*_\mathbf{s}$ such that $uSv \in L_0$. At the end of the path, we attach a path labeled with $v$ and on that one a model of $\psi$—note that $\psi$ is assumed to be satisfiable. The new structure, say $\mathcal{M}'$, is a model of $EF^{L_0} \psi$. and also, by (16), of $EF^{L_0} \psi$.  

All (rooted) finite paths in $\mathcal{M}'$ which not yet occur in $\mathcal{M}$ passes the labels $uS$. For the sake of contradiction, assume that $\mathcal{M}'$ is not a model of $EF^{L_0} \psi$. Hence $uS$ is a prefix of a word in $L$. So, $u \in L$ because $L \subseteq \Sigma^*$. Contradiction.  

And as for the measure, 

$$\mu(EF^{L_1} \psi) = ||L_1|| : \mu(\psi) \leq \mu(EF^{L_0} \psi) \text{ and } \mu(EF^{L_2} \psi) = ||L_2|| : \mu(\psi) \leq \mu(EF^{L_0} \psi).$$  

(18) 

hold, and imply $\mu(EF^{L_1} \psi \lor EF^{L_2} \psi) \leq \mu(EF^{L_0} \psi).$  

Two remarks, to previous lemma: (1) If $\epsilon \notin L_0$ then it is neither in $L_1$ nor in $L_2$—but $S \notin L_2$ might be. (2) If $L_0$ is a DCFL then so are $L_1$ and $L_2$. 

Theorem 22. Let $P \subseteq \Sigma^*$, and let $\varphi$ a PDL$_0$[DCFL]-formula over $\Sigma_\mathbf{s}$. If $\varphi = EF^{P_S} tt$ then $\mathcal{L}(\varphi)$ is good.  

Proof. We apply to $\varphi$ several transformations in sequence. Each transformation leads to a formula which is equivalent to $\varphi$ and whose measure is weakly bounded by $\mu(\varphi)$ from above. The transformations are the following ones.  

- Make the formula $\varepsilon$-free by Definition 13 and Lemma 14. 
- Transform it into a DNF and complete the formula: Definition 15 and Lemma 16. 
- Eliminate the outermost $\forall$-quantifiers using Theorem 17. 
- Apply Theorem 18 to each term of the DNF gotten from the previous transformation. Note that the applied formula is still $\varepsilon$-free. 
- Apply Lemma 21 and its remarks to the outermost $\forall$-formulas. 

Finally, we obtain a formula 

$$\varphi' := \bigvee_{i \in I} EF^{L_i} \psi_i = \varphi$$  

(18) 

such that $\mu(\varphi') \leq \mu(\varphi)$ holds. In addition, $I$ is finite and for each $i \in I$ we have that
Corollary 23. Let $\varphi \in \text{PDL}_0[\text{DCFL}]$. If $\varphi = \text{EF}^{\text{Palindromes}}\top\top$ then the language Palindromes is good.

Proof. By Theorem 22 and Lemma 20.

Corollary 24. $\text{PDL}_0[\text{DCFL}] \subseteq \text{PDL}_0[\text{CFL}]$.

Proof. By Corollaries 23 and 7.
9 Conclusion and Further Work

We proved that PDL₀[DCFL] is distinct from PDL₀[CFL] by means of model and language theory. Similar results—such as CTL vs. Fairness [8], PDL₀ vs. PDL [5], and unary CTL vs. unary CTL⁺ [7]—uses two sequences of transition systems which are indistinguishable for the smaller logic. Their proofs are pretty compact. So, is it possible to reformulate our proof in a similar way? The main difficulty should be the incorporation of Theorem 3 into transition systems.

The considered logic is exactly the EF/-AG-fragment of the Extended Computation Tree Logic [2], say CTL[2]. This observation poses at least two questions. First, is it possible to extend the separation from the unary fragment to the binary EU/-AR-fragment? Here, the main challenge is the interpretation of $E(\psi_1 U^{\varepsilon} \psi_2)$ in the sense of Definition 19 as $\psi_1$ could prohibit linear models: take $E((p \land (EF^{\varepsilon} \neg p)) U^{\varepsilon} \psi_2)$ for instance. Secondly, one could go from one of these fragments to the whole logic to obtain a separation of CTL[DCFL] and CTL[CFL]. In addition to the mentioned difficulties, one is faced with the alternating quantifiers EG and AF. To achieve such a goal, note that the Theorems 17 and 18 also hold for arbitrary path quantifications as long as $\varepsilon$-freedom is guaranteed. An iteration of these tools along a given $\omega$-word could unravel an $\omega$-sequence of disjunctions of E-formulas. Such a sequence could be a subject for a pumping lemma similar to Lemma 2. The $\omega$-word could follow the lines of Theorem 3.

Finally, a separation of the full PDL (i.e. with tests) and of the $\Delta$-variants of PDL [21,17] could provide more insight into the difference between the non-determinism in CFLs and the non-determinism used in the translation of formulas into automata.

References

[1] Rajeev Alur & Parthasarathy Madhusudan (2004): Visibly pushdown languages. In: Proc. 36th Ann. ACM Symp. on Theory of Computing, STOC'04, pp. 202–211, doi:10.1145/1007352.1007390
[2] Roland Axelsson, Matthew Hague, Stephan Kreutzer, Martin Lange & Markus Latte (2010): Extended Computation Tree Logic. In Christian Fermüller & Andrei Voronkov, editors: Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2010, Yogyakarta, Indonesia), Lecture Notes in Computer Science 6397, Springer, pp. 67–81, doi:10.1007/978-3-642-16242-8_6
[3] Roland Axelsson, Matthew Hague, Stephan Kreutzer, Martin Lange & Markus Latte (2010): Extended Computation Tree Logic. arXiv.org Available at http://arxiv.org/abs/1006.3709
[4] Franz Baader & Tobias Nipkow (1998): Term Rewriting and All That. Cambridge University Press.
[5] Francine Berman & Michael Paterson (1981): Propositional dynamic logic is weaker without tests. Theoretical Computer Science 16(3), pp. 321–328, doi:10.1016/0304-3975(81)90102-X
[6] Mikołaj Bojańczyk (2008): The common fragment of ACTL and LTL. In: Proceedings of the Theory and practice of software, 11th international conference on Foundations of software science and computational structures, FOSSACS’08/ETAPS’08, Springer-Verlag, Berlin, Heidelberg, pp. 172–185, doi:10.1007/978-3-540-78499-9_13
[7] E. Allen Emerson & Joseph Y. Halpern (1985): Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. Journal of Computer and System Sciences 30, pp. 1–24, doi:10.1016/0022-0000(85)90001-7
[8] E. Allen Emerson & Joseph Y. Halpern (1986): “Sometimes” and “Not Never” Revisited: On Branching versus Linear Time Temporal Logic. Journal of the ACM 33, pp. 151–178, doi:10.1145/4904.4999
[9] Michael J. Fischer & Richard E. Ladner (1979): Propositional Dynamic Logic of Regular Programs. Journal of Computer and System Sciences 18(2), pp. 194–211.
[10] Seymour Ginsburg & Sheila Greibach (1966): *Deterministic Context Free Languages*. Information and Control 9(6), pp. 620–648, doi:10.1016/S0019-9958(66)80019-0.

[11] David Harel (1984): *Dynamic Logic*. In Dov M. Gabbay & F. Guenthner, editors: *Handbook of Philosophical Logic, Vol. II: Extensions of Classical Logic*, chapter 10, D. Reidel Publishing Company, Dordrecht, Holland, pp. 497–604.

[12] David Harel, Dexter Kozen & Jerzy Tiuryn (2000): *Dynamic Logic*. Foundations of Computing, MIT Press, Cambridge, Massachusetts.

[13] David Harel, Amir Pnueli & Jonathan Stavi (1983): *Propositional Dynamic Logic of Nonregular Programs*. Journal of Computer and System Sciences 26(2), pp. 222–243.

[14] David Harel & Danny Raz (1990): *Deciding properties of nonregular programs*. Foundations of Computer Science, Annual IEEE Symposium on , pp. 652–661.

[15] John E. Hopcroft & Jeffrey D. Ullman (1979): *Introduction to Automata Theory, Languages and Computation*. Addison Wesley.

[16] Dexter Kozen & Jerzy Tiuryn (1990): *Logics of Programs*. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, Elsevier and MIT Press, pp. 789–840.

[17] Christof Löding, Carsten Lutz & Olivier 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

[18] Christof Löding, Parthasarathy Madhusudan & Olivier 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.

[19] Michael O. Rabin (1970): *Weakly definable relations and special automata*. In Yehoshua Bar-Hillel, editor: *Mathematical Logic and Foundations of Set Theory*, North-Holland, Amsterdam, pp. 1–23.

[20] Ludwig Staiger (1997): *ω-languages*. In Grzegorz Rozenberg & Arto Salomaa, editors: *Handbook of formal languages*, chapter 6, 3: beyond words, Springer, New York, NY, USA, pp. 339–387.

[21] Robert S. Streett (1982): *Propositional Dynamic Logic of Looping and Converse is Elementarily Decidable*. Information and Control 54(1-2), pp. 121–141, doi:10.1016/S0019-9958(82)91258-X.

[22] Wolfgang Thomas (1990): *Automata On Infinite Objects*. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, Elsevier and MIT Press, pp. 133–191.

[23] Moshe Y. Vardi & Pierre Wolper (1986): *Automata-Theoretic Techniques for Modal Logics of Programs*. Journal of Computer and System Sciences 32(2), pp. 183–221, doi:10.1016/0022-0000(86)90026-7.