Model Checking Parse Trees

Anudhyan Boral[1] Sylvain Schmitz[2]

[1]CMI, Chennai, India
[2]LSV, ENS Cachan & CNRS, France

Abstract

Parse trees are fundamental syntactic structures in both computational linguistics and compilers construction. We argue in this paper that, in both fields, there are good incentives for model-checking sets of parse trees for some word according to a context-free grammar. We put forward the adequacy of propositional dynamic logic (PDL) on trees in these applications, and study as a sanity check the complexity of the corresponding model-checking problem: although complete for exponential time in the general case, we find natural restrictions on grammars for our applications and establish complexities ranging from nondeterministic polynomial time to polynomial space in the relevant cases.

1 Introduction

The parse trees of a sequence w are employed extensively in computational linguistics, where they represent constituent analyses of the natural language sentence w, and in compilers, where they provide the syntactic structure of the input program w. They are produced by a parsing algorithm on the basis of a grammar, for instance a context-free one, that typically required quite a bit of ingenuity in its conception: in each of the two communities, a subfield of *grammar engineering* has arisen (e.g. Klint et al., 2005; Kallmeyer et al., 2008), dedicated to the principled development of grammars. Part of the difficulty in this task stems from the limited expressiveness of the formalism: for instance, in a context-free grammar, a production $A \to B\, C$ only states that an 'A' node might have two children labeled 'B' and 'C' in this order. This means that orthogonal considerations, possibly involving distantly related tree nodes, have to be enforced manually, little by little in the local rules of the grammar, blowing up its size and quickly rendering it unmaintainable.

Model theoretic syntax provides an alternative approach to syntax specifications: rather than an 'enumerative' formalism that generates the desired parse trees, one can describe them as models of a logical formula. This point of view leads to interesting consequences for syntactic theory (Pullum and Scholz, 2001), but here we are mostly interested in the conciseness and ease of manipulation of formulæ. Several logical formalisms over trees have been employed to this end, notably first-order (FO) or monadic second-order (MSO) logics (Rogers, 1998), and *propositional dynamic logic on trees* ($\mathrm{PDL}_{\mathsf{tree}}$) (Kracht, 1995; Palm, 1999; Afanasiev et al., 2005). In practice however, model-theoretic approaches suffer from a prohibitively high complexity, as the known recognizing algorithms essentially amount to a *satisfiability* check (Cornell, 2000): given a formula φ and a sequence w to parse, build a formula φ_w that recognizes all the trees that yield w, and check $\varphi \wedge \varphi_w$ for satisfiability. As satisfiability is in general non-elementary for FO and MSO formulæ and ExpTime-complete for $\mathrm{PDL}_{\mathsf{tree}}$ ones, this seems like a serious impediment to a larger adoption of model-theoretic techniques.

In this paper, we introduce the *model-checking* problem for parse trees. Formally, given a sequence w, a grammar G, and a formula φ, we ask whether all the parse trees of w according to G verify φ. It turns out that checking sets of parse trees of a given w, i.e. *parse forests*, can be easier than other classes of tree languages, as could be defined by tree automata, document type definitions, etc.

This parse forest model-checking problem (PFMC) allows for a 'mixed' approach, where a context-free grammar is employed for a cursory syntax specification, alongside a logical formula for the fine-tuning. Because the logical languages we consider are closed under negation, under this viewpoint, the PFMC problem also answers the recognition problem for the 'conjunction' of the grammar G and the formula φ: is there a parse of w according to both G and φ? As φ might describe a non-local tree language, there is a slight expressive gain to be found in such conjunctions, but our interest lies more in the concision and clarity brought by refining a grammar with a $\mathrm{PDL}_{\mathsf{tree}}$ formula: it allows to capture long-distance dependencies that would often require a cumbersome and error-prone hard-wiring in the grammar, at the expense of an explosion of the number of nonterminal symbols.

We detail in Section 3 two applications of the PFMC; for these, we found it convenient to use $\mathrm{PDL}_{\mathsf{tree}}$ as the logical language for properties:

1. In computational linguistics, we advocate a mixed approach for model-theoretic syntax, with syntactic structures described by the conjunction of a grammar for localized specification together with a $\mathrm{PDL}_{\mathsf{tree}}$ constraint capturing long-distance syntactic phenomena;

2. In compilers construction, $\mathrm{PDL}_{\mathsf{tree}}$ formulæ provide a compelling means for parser disambiguation (Thorup, 1994; Klint and Visser, 1994; Kats et al., 2010) by allowing to express formally the informal disambiguation rules usually provided with grammars for programming languages.

We discuss the appropriateness of our formalizations in some depth, which allows us to motivate (1) practically relevant restrictions on the grammar G, and (2) considering the full logic $\mathrm{PDL}_{\mathsf{tree}}$ rather than some weaker fragments. We consider the two formalizations proposed in Section 3 as the initial steps of a larger research programme on the model-checking problem in syntax; we point for instance to several interesting open issues with the choice of finite labeled ordered trees as syntactic structures and $\mathrm{PDL}_{\mathsf{tree}}$ as logical formalism.

As a first usability check, we investigate the computational complexity of the PFMC and map the resulting complexity landscape for the problem in Section 4. Although the general case is ExpTime-complete like the $\mathrm{PDL}_{\mathsf{tree}}$ satisfiability problem, our restrictions on grammars lead to more affordable complexities

1. from NPTime-complete for our linguistic applications, where we can assume the grammar to be both ε-free and acyclic,

2. to PSpace-complete for our applications in ambiguity filtering, where we can only assume the grammar to be acyclic

(see Figure 3 for a summary). Our study also unearthed a somewhat surprising corollary for model-theoretic syntax (Cor. 1):

3. the *recognition problem* for $\mathrm{PDL}_{\mathsf{tree}}$, i.e. whether there exists a tree model with the input word as yield, is PSpace-complete if empty labels are forbidden—the best algorithms for this were only known to operate in exponential time (Cornell, 2000; Palm, 2004).

Interestingly, the PFMC is closely related to a prominent algorithmic problem studied by the XML community: there the formula φ is a *Core XPath* one—which is equivalent to a restricted fragment $\mathrm{PDL}_{\mathsf{core}}$ of $\mathrm{PDL}_{\mathsf{tree}}$—and the tree language L is generated by a *document type definition* (DTD), and the problem is accordingly referred to as 'satisfiability in presence of a DTD'. Benedikt et al. (2008) comprehensively investigate this topic, and in some restricted cases the problem becomes tractable (Montazerian et al., 2007; Ishihara et al., 2009). Our applications in computational linguistics and compilation lead however to a different setting, where L comes from a class of tree languages smaller than that of DTDs—and our grammar restrictions have no natural counterpoints in the XML literature—, but where φ requires more expressive power than that of $\mathrm{PDL}_{\mathsf{core}}$. Nevertheless, we will reuse several proof techniques developed in the XML setting and adapt them to ours, Prop. 6 being a prime example: it relies on an extension of the results of Benedikt et al. (2008) to the full logic $\mathrm{PDL}_{\mathsf{tree}}$ (see Prop. 4) and on an encoding of a restricted class of parse forests into non-recursive DTDs.

2 Propositional Dynamic Logic on Trees

Propositional dynamic logic (PDL, see (Fischer and Ladner, 1979)) is a modal logic where "programs"—in the form of regular expressions over the relations in a frame—are used as modal operators. Originally motivated by applications in computational linguistics (Kracht, 1995; Palm, 1999; Afanasiev et al., 2005), *propositional dynamic logic on trees* ($\mathrm{PDL}_{\mathsf{tree}}$) has also been extensively studied in the XML community (Marx, 2005; Benedikt et al., 2008; ten Cate and Segoufin, 2010), where it is better known as *Regular XPath*. It features two relations: the *child* relation $\downarrow$ between a parent node and any of its immediate children, and the *right-sibling* relation $\rightarrow$ between a node and its immediate right sibling.

2.1 Syntax and Semantics

Formally, a $\mathrm{PDL}_{\mathsf{tree}}$ formula φ is defined by the abstract syntax

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\pi\rangle\varphi \qquad \text{(node formulæ)}$$

$$\pi ::= \downarrow \mid \rightarrow \mid \pi\,;\pi \mid \pi + \pi \mid \pi^* \mid \pi^{-1} \mid \varphi? \qquad \text{(path formulæ)}$$

where p is an atomic proposition ranging over some countable set AP—because we only deal with satisfiability questions, we can actually assume AP to be finite. We enrich this syntax as usual by defining box modalities as duals $[\pi]\varphi \stackrel{\text{def}}{=} \neg\langle\pi\rangle\neg\varphi$ of the diamond ones, inverses to the atomic path formulæ as $\uparrow \stackrel{\text{def}}{=} \downarrow^{-1}$ and $\leftarrow \stackrel{\text{def}}{=} \rightarrow^{-1}$, and boolean connectives $\bot \stackrel{\text{def}}{=} \neg\top$, $\varphi_1 \vee \varphi_2 \stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \Rightarrow \varphi_2 \stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2$, and $\varphi_1 \equiv \varphi_2 \stackrel{\text{def}}{=} (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$.

Formulæ are interpreted over *finite ordered trees* t with nodes labeled by propositions in AP. Such a tree t is a partial function from the set $\mathbb{N}^*$ of finite sequences of natural numbers, i.e. the set of tree nodes, to AP, s.t. its *domain* $\mathsf{dom}\, t$ is (1) *finite*, (2) *prefix closed*, i.e. uv in $\mathsf{dom}\, t$ for some u, v in $\mathbb{N}^*$ implies that u is also in $\mathsf{dom}\, t$, and (3) *predecessor closed*, i.e. if ui is in $\mathsf{dom}\, t$ for some u in $\mathbb{N}^*$ and i in $\mathbb{N}$, then uj is also in $\mathsf{dom}\, t$ for all $j < i$ in $\mathbb{N}$. Such a tree can be seen as a structure $\mathfrak{M}_t = \langle \mathsf{dom}\, t, \downarrow_t, \rightarrow_t, t\rangle$ with

$$\downarrow_t \stackrel{\text{def}}{=} \{(u, ui) \mid ui \in \mathsf{dom}\, t\}$$
$$\rightarrow_t \stackrel{\text{def}}{=} \{(ui, u(i+1)) \mid u(i+1) \in \mathsf{dom}\, t\}\ .$$

We define the *interpretations* of $\mathrm{PDL}_{\mathsf{tree}}$ formulæ over t inductively by

$$\begin{aligned}
[\![p]\!]_t &\stackrel{\text{def}}{=} \{u \in \mathsf{dom}\, t \mid p = t(u)\} & [\![\top]\!]_t &\stackrel{\text{def}}{=} \mathsf{dom}\, t\\
[\![\varphi_1 \wedge \varphi_2]\!]_t &\stackrel{\text{def}}{=} [\![\varphi_1]\!]_t \cap [\![\varphi_2]\!]_t & [\![\neg\varphi]\!]_t &\stackrel{\text{def}}{=} \mathsf{dom}\, t \setminus [\![\varphi]\!]_t\\
[\![\varphi?]\!]_t &\stackrel{\text{def}}{=} \{(u, u) \mid u \in [\![\varphi]\!]_t\} & [\![\langle\pi\rangle\varphi]\!]_t &\stackrel{\text{def}}{=} [\![\pi]\!]_t^{-1}([\![\varphi]\!]_t)\\
[\![\downarrow]\!]_t &\stackrel{\text{def}}{=} \downarrow_t & [\![\rightarrow]\!]_t &\stackrel{\text{def}}{=} \rightarrow_t\\
[\![\pi_1 ; \pi_2]\!]_t &\stackrel{\text{def}}{=} [\![\pi_1]\!]_t \mathbin{⨾} [\![\pi_2]\!]_t & [\![\pi^*]\!]_t &\stackrel{\text{def}}{=} [\![\pi]\!]_t^*\\
[\![\pi_1 + \pi_2]\!]_t &\stackrel{\text{def}}{=} [\![\pi_1]\!]_t \cup [\![\pi_2]\!]_t & [\![\pi^{-1}]\!]_t &\stackrel{\text{def}}{=} [\![\pi]\!]_t^{-1}
\end{aligned}$$

Observe that these are sets of nodes included in $\mathsf{dom}\, t$ in the case of node formulæ, but binary relations included in $\mathsf{dom}\, t \times \mathsf{dom}\, t$ in the case of path formulæ; thus $[\![\pi]\!]_t^*$ denotes the reflexive transitive closure of $[\![\pi]\!]_t$ and $[\![\pi]\!]_t^{-1}$ its inverse, while $[\![\pi_1]\!]_t \mathbin{⨾} [\![\pi_2]\!]_t$ denotes the composition of the two relations $[\![\pi_1]\!]_t$ and $[\![\pi_2]\!]_t$. A node u in $\mathsf{dom}\, t$ *satisfies* φ, noted $t, u \models \varphi$, if u is in $[\![\varphi]\!]_t$. A tree t satisfies φ, noted $t \models \varphi$, if its root ε satisfies φ; we let $[\![\varphi]\!] \stackrel{\text{def}}{=} \{t \mid t \models \varphi\}$ be the set of models of φ.

Example 1 (Basic Navigation)**.** Several simple formulæ helping navigation can be defined: $\mathsf{root} \stackrel{\text{def}}{=} \neg\langle\uparrow\rangle\top$ holds only at the root, $\mathsf{leaf} \stackrel{\text{def}}{=} \neg\langle\downarrow\rangle\top$ only at a leaf node, $\mathsf{first} \stackrel{\text{def}}{=} \neg\langle\leftarrow\rangle\top$ at a leftmost one, and $\mathsf{last} \stackrel{\text{def}}{=} \neg\langle\rightarrow\rangle\top$ at a rightmost one.

We can also define the *first-child* relation $\swarrow \stackrel{\text{def}}{=} \downarrow; \mathsf{first}?$, and conversely express the child relation as $\downarrow \equiv \swarrow; \rightarrow^*$: this shows that we could work on binary tree models instead of the unranked ones we used in our definitions.

Example 2 (Parse Trees (Blackburn et al., 1993))**.** Recall that a *context-free grammar* (CFG) is a tuple $G = \langle N, \Sigma, P, S\rangle$ composed of a finite *nonterminal* alphabet N, a finite *terminal* alphabet Σ disjoint from N and forming a *vocabulary* $V \stackrel{\text{def}}{=} N \uplus \Sigma$, a finite set of *productions* $P \subseteq N \times V^*$, and an *axiom* $S \in N$. We denote the empty sequence by ε and write $\Sigma' \stackrel{\text{def}}{=} \Sigma \uplus \{\varepsilon\}$ and $V' \stackrel{\text{def}}{=} V \uplus \{\varepsilon\}$.

Given a context-free grammar G, its set of parse trees forms a local tree language, which can be expressed as $[\![\varphi_G]\!]$ for a $\mathrm{PDL}_{\mathsf{tree}}$ formula φ_G with V'

as set of atomic propositions. First define a path formula π_α that defines a sequence of sibling nodes labeled by α in V^*:

$$\pi_\alpha \stackrel{\text{def}}{=} \begin{cases} X?; \rightarrow; \pi_{\alpha'} & \text{if } \alpha = X\alpha', X \in V, \alpha' \neq \varepsilon\ , \\ X?; \mathsf{last}? & \text{if } \alpha = X \in V\ , \\ \varepsilon?; \mathsf{last}? & \text{otherwise, i.e. if } \alpha = \varepsilon\ . \end{cases}$$

$$\begin{aligned} \varphi_G \stackrel{\text{def}}{=} S & & \text{(the root is labeled by } S\text{)} \\ \wedge [\downarrow^*]\Big(\mathsf{leaf} \equiv \bigvee_{a \in \Sigma'} a & & \\ & & \text{(leaves are terminals and internal nodes nonterminals)} \\ \wedge \bigwedge_{A \in V} A \Rightarrow \bigvee_{A \to \alpha} \langle \swarrow; \pi_\alpha \rangle \top\Big)\ . & & \text{(productions are enforced)} \end{aligned}$$

2.2 The Conditional Fragment

We will consider in this paper several fragments of $\mathrm{PDL}_{\mathsf{tree}}$, most importantly the *conditional path* fragment $\mathrm{PDL}_{\mathsf{cp}}$ (Palm, 1999; Marx, 2005), with a restricted syntax on path formulæ

$$\begin{aligned} \pi &::= \alpha \mid \pi\, ;\pi \mid \pi + \pi \mid \varphi? \mid (\alpha; \varphi?)^* & & \text{(conditional paths)} \\ \alpha &::= \leftarrow \mid \rightarrow \mid \uparrow \mid \downarrow\ . & & \text{(atomic paths)} \end{aligned}$$

This fragment is of particular relevance, because it extends the *core language* $\mathrm{PDL}_{\mathsf{core}}$ (Blackburn et al., 1996; Gottlob and Koch, 2002) (which features α^* instead of $(\alpha; \varphi?)^*$) and captures exactly first-order logic over finite ordered trees with the two relations $\rightarrow^+$ and $\downarrow^+$ (Marx, 2005).

Example 3 (Depth-First Traversal)**.** Observe that the formulæ in examples 1 and 2 are actually in $\mathrm{PDL}_{\mathsf{core}}$. The *depth-first traversal* relation $\prec \stackrel{\text{def}}{=} (\mathsf{last}?;\uparrow)^*; \rightarrow; (\downarrow; \mathsf{first}?)^*$ is an example of a path that is not definable in $\mathrm{PDL}_{\mathsf{core}}$—this can be checked for instance using an Ehrenfeucht Fraïssé argument.

More generally, $\mathrm{PDL}_{\mathsf{cp}}$ allows to express relations akin to LTL's *until* and *since* modalities; see e.g. (Libkin and Sirangelo, 2010). We denote by $\mathrm{PDL}_{\mathsf{tree}}[\downarrow]$ (resp. $\mathrm{PDL}_{\mathsf{cp}}[\downarrow]$, $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$) the fragments with only downward navigation, i.e. without the $\rightarrow$, $\leftarrow$, and $\uparrow$ atomic paths.

3 Model-Checking Parse Trees

Many problems arising naturally with $\mathrm{PDL}_{\mathsf{tree}}$ are decidable, notably the

model-checking problem: given a tree t and a formula φ, does $t \models \varphi$? This is known to be in PTime even for larger fragments of PDL (Lange, 2006).

satisfiability problem: given a formula φ, does there exist a tree t s.t. $t \models \varphi$? This is known to be ExpTime-complete (Afanasiev et al., 2005).

In the context of XML processing and XPath, an intermediate question between model-checking and satisfiability also arises:

satisfiability in presence of a tree language: given a formula φ and a regular tree language L, does there exist a tree $t \in L$ s.t. $t \models \varphi$?

Due to its initial XML motivation, the basic case for this problem is that of a $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula (a downward Core XPath query) and of a local tree language (described by a DTD), but many variants exist (Benedikt et al., 2008; Montazerian et al., 2007; Ishihara et al., 2009)—in particular one where the tree language is the language of infinite trees of a two-way alternating parity tree automaton, which is used by Göller et al. (2009) to prove that the satisfiability problem for PDL with intersection and converse is 2ExpTime-complete.

Our own flavour is motivated by applications in computational linguistics and programming languages, where the tree language is the set of parse trees of a word w in Σ^* according to a CFG $G = \langle N, \Sigma, P, S\rangle$ verifying $V' \stackrel{\text{def}}{=} \Sigma \uplus N \uplus \{\varepsilon\} = \mathsf{AP}$. More precisely, following a well-known construction of Bar-Hillel et al. (1961), if $w = a_1 \cdots a_n$ is a word of length n, the set of parse trees or *parse forest* of a CFG G for w, written $L_{G,w}$, is the regular tree language recognized by a tree automaton $\mathcal{A}_{G,w}$ with state set

$$Q_{G,w} \stackrel{\text{def}}{=} \{(i, X, j) \mid 0 \leq i \leq j \leq n, X \in V'\} ,$$

alphabet V', initial state $(0, S, n)$, and rules

$$\begin{aligned}\delta_{G,w} \stackrel{\text{def}}{=} &\{(i_0, A, i_m) \to A((i_0, X_1, i_1) \cdots (i_{m-1}, X_m, i_m)) \\ &\quad \mid A \to X_1 \cdots X_m \in P \wedge 0 \leq i_0 \leq \cdots \leq i_m \leq n\} \\ &\cup \{(i, a_{i+1}, i+1) \to a_{i+1}() \mid 0 \leq i < n\} \\ &\cup \{(i, \varepsilon, i) \to \varepsilon() \mid 0 \leq i \leq n\} .\end{aligned}$$

Intuitively, a state (i, X, j) of this automaton recognizes the set of trees derivable in G from the symbol X and spanning the factor $a_{i+1} \cdots a_j$ of w. This automaton is in general not *trim*, in that many of its states and rules are never employed in any accepting configuration, but it can be trimmed in linear time if required.

Parse Forest Model-Checking Problem (PFMC)**.**

input a context free grammar G, a word w, and a $\mathrm{PDL}_{\mathsf{tree}}$ formula φ,

question does there exists $t \in L_{G,w}$ s.t. $t \models \varphi$? $\square$

Note that the automaton $\mathcal{A}_{G,w}$ has size $O(|G| \cdot |w|^{m+1})$ if m is the maximal length of a production rightpart in G; since the grammar can be put in quadratic form (corresponding to the binarization we would also perform on the formula), this typically results in size $O(|G| \cdot |w|^3)$. Therefore, although a tree automaton for the tree language is not part of the input, it can nevertheless be constructed in logarithmic space. The originality of the problem stems from considering parse forests, which form a rather restricted class of tree languages.

In Section 4, we will investigate the complexity of this problem, and focus on the influence of the acyclicity and ε-freeness of G: Define the *derivation* relation $\Rightarrow$ between sequences in V^* by $\beta A \gamma \Rightarrow \beta \alpha \gamma$ iff $A \to \alpha$ is a production of G and β, γ are arbitrary sequences in V^*. A CFG is *acyclic*, if none of its nonterminals

A allows $A \Rightarrow^+ A$. A CFG is *ε-free*, if none of its productions is of form $A \to \varepsilon$ for some nonterminal A.

In the remainder of this section, we motivate the problem by considering applications in computational linguistics (Section 3.1) and compilers construction (Section 3.2).

3.1 Application: Computational Linguistics

In contrast with many formal theories of syntax that describe natural language sentences through 'generative-enumerative means', Pullum and Scholz (2001) champion *model-theoretic syntax*, where the syntactic structures (typically, trees) of a natural language are the models of some logical formula. They point out interesting consequences on theories of syntax, but here we betray the spirit of their work in exchange for some practicality.[1]

Indeed, the usual approach to model-theoretic syntax would be to describe a language through a *huge* formula φ of $\mathrm{PDL}_{\mathsf{tree}}$ or monadic second-order logic (MSO) on trees. Checking whether a given sentence w can be assigned a structure then reduces to a recognition problem on a tree automaton $\mathcal{A}_\varphi$ of exponential (for $\mathrm{PDL}_{\mathsf{tree}}$) or non-elementary (for MSO) size (Cornell, 2000).

A Mixed Approach We consider a pragmatic approach, where

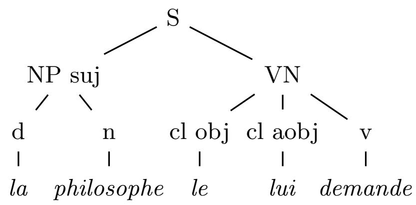
- a CFG describes the *local* aspects of syntax, e.g. that a canonical transitive French sentence can be decomposed into a noun phrase acting as subject followed by a verb kernel and an object noun phrase corresponds to a production $\mathrm{S} \to \mathrm{NP}\,\mathrm{VN}\,\mathrm{NP}$, while
- *long-distance* dependencies and more complex linguistic constraints are described through $\mathrm{PDL}_{\mathsf{tree}}$ formulæ.

Example 4 (French Clitics)**.** A toy grammar for French sentences with predicative verbs like 'dire' or 'demander' could look like (in an extended syntax where X? describes zero or one occurrences of symbol X):

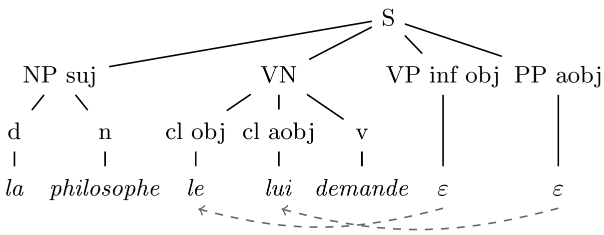
$$\begin{aligned}
\mathrm{S} &\to \mathrm{NPsuj?}\ \mathrm{VN}\ \mathrm{VPinfobj?}\ \mathrm{PPaobj?} \\
\mathrm{NPsuj} &\to \mathrm{d}\ \mathrm{n} \\
\mathrm{VN} &\to \mathrm{clsuj?}\ \mathrm{clobj?}\ \mathrm{claobj?}\ \mathrm{v} \\
\mathrm{VPinfobj} &\to \textit{de}\ \mathrm{VN} \\
\mathrm{PPaobj} &\to \textit{à}\ \mathrm{NP} \\
\mathrm{v} &\to \textit{demande} \mid \textit{réfléchir} & \mathrm{clsuj} &\to \textit{elle} \\
\mathrm{n} &\to \textit{philosophe} & \mathrm{clobj} &\to \textit{le} \\
\mathrm{d} &\to \textit{la} & \mathrm{claobj} &\to \textit{lui}
\end{aligned}$$

Such predicative verbs have a mandatory object and subject, and an optional indirect object. But all three canonical arguments can be replaced by *clitics* in the verb matrix VN. This grammar fragment generates reasonable sentences like

[1] In an ESSLLI 2013 lecture, Geoffrey Pullum famously explained that "model theoretic syntax is not generative enumerative syntax with constraints", the latter being exactly what we are proposing as a way of mitigating the complexity of model-theoretic techniques.



(a) Syntax tree according to Example 4.



(b) Analysis with moved constituents.

Figure 1: Syntax trees for "la philosophe le lui demande."

(1) La philosophe demande de réfléchir.
The philosopher asks to think.

(2) La philosophe le lui demande.
The philosopher it.ACC her.DAT asks.
"The philosopher asks it to her".

where the 'le' clitic acts as direct object and 'lui' as an indirect one (see Figure 1a for an example syntax tree). It also generates ungrammatical ones like

(3) * Elle le lui demande de réfléchir.
She it.ACC her.DAT asks to think.
"She asks it to her to think".

(4) * demande.
asks.

where there are duplicated or missing arguments.

Instead of refining the grammar (which might prove impossible, for instance if it was automatically extracted from a treebank, i.e. a set of sentences annotated with syntactic trees), we can filter out the unwanted trees using a $\mathrm{PDL}_{\mathsf{tree}}$ formula. To improve readability, we take symbols like 'VPinfobj' or 'clsuj' to denote sets of atomic propositions, respectively $\{\mathrm{VP}, \mathrm{inf}, \mathrm{obj}\}$ and $\{\mathrm{cl}, \mathrm{suj}\}$ in this instance, and refine our grammar with the following formula:

$$\begin{aligned}
[\downarrow^*]\mathit{demande} \Rightarrow \big(&\langle(\uparrow;\uparrow;\rightarrow^+) + (\uparrow;\leftarrow^+;\mathrm{cl}?)\rangle\mathrm{obj} && \text{(at least one object)}\\
&\wedge \langle(\uparrow;\uparrow;\leftarrow) + (\uparrow;\leftarrow^+;\mathrm{cl}?)\rangle\mathrm{suj} && \text{(at least one subject)}\\
&\wedge \bigwedge_{f\in\{\mathrm{suj},\mathrm{obj},\mathrm{aobj}\}} \langle\uparrow;\leftarrow^+;\mathrm{cl}?\rangle f\\
&\qquad\qquad \Rightarrow \neg\langle\uparrow;\uparrow;(\leftarrow + \rightarrow^+)\rangle f\big)\\
&\text{(clitic arguments forbid the matching canonical arguments)}
\end{aligned}$$

Interestingly, such $\mathrm{PDL}_{\mathsf{tree}}$ constraints can easily be tested against tree corpora to check their validity; see (Lai and Bird, 2010) on using $\mathrm{PDL}_{\mathsf{tree}}$-like query languages to this end. We checked that the above $\mathrm{PDL}_{\mathsf{tree}}$ formula was satisfied by the trees in the Sequoia treebank (Candito and Seddah, 2012) using an XPath processor: note that our formula is indeed in $\mathrm{PDL}_{\mathsf{core}}$.

Discussion In this approach, the CFG can be a very permissive, over-generating one, like the probabilistic grammars extracted from treebanks,[2] since it is later refined by the $\mathrm{PDL}_{\mathsf{tree}}$ constraints. We are not aware of any linguistic rationale for cycles in CFGs; on the other hand, ε-productions are sometimes used as placeholders for *moved constituents*. However, in such analyses, the moved constituent and the placeholder are *coindexed*, i.e. related through an additional relation, which

- requires a richer class of models than mere trees over a finite alphabet if we want to make the coindexation explicit (see Figure 1b for an example)—one could consider *data trees* to this end (Bojańczyk and Lasota, 2012; Figueira, 2012)—, and
- can be simulated by a $\mathrm{PDL}_{\mathsf{tree}}$ formula, as seen with the connection we establish between a clitic and the corresponding missing argument in Example 4.

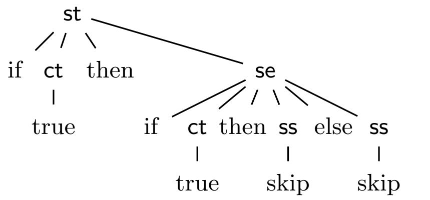
We therefore expect our grammars to be both acyclic and ε-free—and we could check that this was indeed the case on the three rather different CFGs proposed by Moore (2004) for natural language parsing benchmarks.

On the logical side, it seems necessary to be able to use e.g. depth-first traversals (recall Example 3). Palm (1999) and Lai and Bird (2010) study the question in much greater detail and argue that $\mathrm{PDL}_{\mathsf{cp}}$ provides an appropriate expressiveness for linguistic queries.

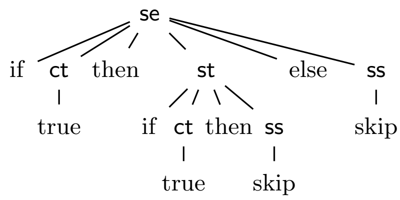
3.2 Application: Ambiguity Filtering

Ambiguities in context-free grammars describing the syntax of programming languages are a severe issue, as they might lead to different semantic interpretations, and complicate the use of deterministic parsers—they basically require manual fiddling. They are also quite useful, as they allow for more concise and more readable grammars, and it is actually uncommon to find a language reference proposing an unambiguous grammar.

[2] Moore (2004) finds an average of 7.2×10^{27} different parse trees per sentence with a grammar extracted from the Penn treebank!



(a) Parse when preferring shift over reduce.



(b) Parse when preferring reduce over shift.

Figure 2: Two parses for the ambiguous input "if true then if true then skip else skip" with the grammar of Example 5.

A nice way of dealing with ambiguities at parse time is to build a parse forest and *filter out* the unwanted trees (Klint and Visser, 1994). In contrast with tinkering with parsers, this allows to implement the 'side constraints' provided in the main text of language references as *declarative rules*, which, beyond readability and maintainability concerns (Kats et al., 2010), also enables some amount of static reasoning and optimization.

Example 5 (Dangling Else)**.** We propose to use $\mathrm{PDL}_{\mathsf{tree}}$ formulæ to filter out unwanted parses. Consider the following regular tree grammar for statements:[3]

$$\begin{aligned} S &\to \mathsf{st}(\text{if } C \text{ then } S) \mid \mathsf{se}(\text{if } C \text{ then } S \text{ else } S) \\ &\quad \mid \mathsf{sw}(\text{while } C\ S) \mid \mathsf{ss}(\text{skip}) \\ C &\to \mathsf{ct}(\text{true}) \mid \mathsf{cf}(\text{false}) \end{aligned}$$

Feeding this grammar to a LALR(1) parser generator like GNU/bison, we find a single shift/reduce conflict, where the parser has a choice on inputs like "if true then if true then skip else skip", upon reaching the 'else' symbol, between reading further (Figure 2a), and reducing first and leaving this else for later (Figure 2b). The usual convention in programming languages is a greedy one, where shift is always chosen. However, disambiguation by choosing between shift or reduce parsing actions is error-prone, and there are cases where both alternatives are incorrect on some inputs (see (Schmitz, 2010) for such an example in Standard ML).

A $\mathrm{PDL}_{\mathsf{tree}}$ formula that accepts the desired tree of Figure 2a but rejects the one of Figure 2b should check that no 'else' node can be next in a depth-first

[3]We use a regular tree grammar in a restricted way to label internal nodes differently depending on the chosen production; this allows for a simpler $\mathrm{PDL}_{\mathsf{tree}}$ formula but has otherwise no impact as the language remains local.

traversal (in the sense of Example 3) from an 'st' node:

$$\neg\langle\downarrow^*\rangle(\mathsf{st} \wedge \langle\prec\rangle\text{else})\,.$$

Observe that a depth-first traversal $\prec$ is really needed here, because the 'st' node can be at the end of an arbitrarily long sequence of 'sw' nodes from nested 'while' statements.

A very similar approach was proposed by Thorup (1994), who used simple tree patterns for similar purposes. Both tree patterns and $\mathrm{PDL}_{\mathsf{tree}}$ formulæ can be compiled into the grammar, so that only the desired trees can be generated, allowing to use deterministic parsers or ambiguity checking tools (Schmitz, 2010). $\mathrm{PDL}_{\mathsf{tree}}$ formulæ are strictly more expressive than patterns; the dangling else example required an involved extension of patterns in (Thorup, 1996).

Discussion The grammars used for programming languages are always acyclic—tools like GNU/bison will detect and reject cyclic grammars—but ε-productions are fairly common.

On the logic side, although the formula of Example 5 is in $\mathrm{PDL}_{\mathsf{cp}}$ and not in $\mathrm{PDL}_{\mathsf{core}}$, it uses depth-first traversals in a restricted manner, and they could be expressed in XPath 1.0 as `(descendant::*|following::*)[1]`, which selects the first node in document order among all descendants and right siblings. We expect $\mathrm{PDL}_{\mathsf{cp}}$ to be expressive enough for most tasks, but *layout sensitive* syntax would be beyond its grasp: in programming languages like Haskell or Python, the indentation level is used to delimit statement blocks—differentiating between possible parses then requires some limited counting capabilities, or infinite label sets with order as in a recent formalization by Adams (2013).

Excluding a tree considered individually is one approach among others to ambiguity filtering (Klint and Visser, 1994). A popular alternative considers the parse forest as a whole, i.e. the tree automaton $\mathcal{A}_{G,w}$ itself. The ambiguity resolution of Example 5 on the input "if true then if true then skip else skip" can be simply stated as a preference $\mathsf{st} > \mathsf{se}$ implying that the rule

$$(0,S,9) \to \mathsf{st}\big((0,\text{if},1)(1,C,2)(2,\text{then},3)(3,S,9)\big)$$

is preferred over the rule

$$(0,S,9) \to \mathsf{se}\big((0,\text{if},1)(1,C,2)(2,\text{then},3)(3,S,7)(7,\text{else},8)(8,S,9)\big)$$

in the automaton $\mathcal{A}_{G,w}$. Such disambiguation rules are easy to write, but they are also inherently *dynamic*: they cannot be compiled into the grammar, because whether the rule will be triggered depends on whether an ambiguity appears there—an undecidable problem.

4 Complexity Results

We investigate in this section the complexity of the parse forest model-checking problem. We obtain a classification of complexities depending on the properties of the grammar (see Figure 3). Interestingly, our hardness results always hold for a formula φ in the rather restricted fragment $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$, and generally hold already for fixed G and/or w. These bounds use logarithmic space reductions.

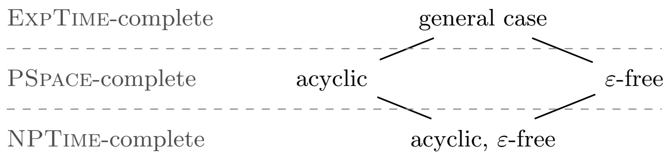


Figure 3: The complexity of the PFMC problem, depending on the grammar characteristics.

Turning first to the complexity in the general case, an immediate consequence of classical results in the field (e.g. Calvanese et al., 2009, Theorem 7) is that it lies in ExpTime.

Proposition 1. *PFMC is in* ExpTime.

Proof. One way to proceed is to *localize* the automaton $\mathcal{A}_{G,w}$ by replacing each rule $(i, X, j) \rightarrow X(\alpha)$ by $(i, X, j) \rightarrow (i, X, j)(\alpha)$. We can then apply the construction of Example 2 to the resulting local automaton, thereby obtaining a $\mathrm{PDL}_{\mathsf{tree}}$ formula $\varphi_{G,w}$ describing a relabeled parse forest of w according to G. It then suffices to apply the same relabeling to φ by interpreting each atomic proposition X as $\bigvee_{1\leq i\leq j\leq n}(i, X, j)$, yielding φ', and to use the ExpTime upper bound on $\mathrm{PDL}_{\mathsf{tree}}$ satisfiability of Afanasiev et al. (2005) to the conjunction $\varphi' \wedge \varphi_{G,w}$ to conclude.

An issue with this proof is that it yields an exponential complexity even if φ is fixed. We can improve on this using automata-based techniques: assume G to be in quadratic form and φ to work on binary trees that encode unranked trees with the $\swarrow$ and $\rightarrow$ relations from Example 1, as these transformations only incur a linear cost. Then, construct the tree automaton $\mathcal{A}_{G,w}$ of size $O(|G|\cdot|w|^3)$ that recognizes the set of parse trees of w in G and the tree automaton $\mathcal{A}_\varphi$ of size $2^{p(|\varphi|)}$ for a polynomial p that recognizes the models of φ: it suffices to test the emptiness of their product automaton, which can be performed in time linear in $|G| \cdot |w|^3 \cdot 2^{p(|\varphi|)}$ for a polynomial p. $\square$

An interesting consequence of the proof of Prop. 1 is that the PFMC problem is PTime-complete when the $\mathrm{PDL}_{\mathsf{tree}}$ formula is fixed, pleading for using small formulæ in practice.

Our proof for Prop. 1 does not benefit from the specificities of the PFMC problem: any satisfiability problem in presence of a tree language would use the same algorithm. Therefore, we might still hope for the existence of a more efficient solution, but adapting the proof of ExpTime-hardness for PDL satisfiability from (Blackburn et al., 2001), we obtain:

Proposition 2. *PFMC is* ExpTime*-hard, even for fixed G and w and for φ in* $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$.

Proof Idea. We go back to low-level arguments[4] and reduce from the *two-players corridor tiling game* of Chlebus (1986). We fix $w = \varepsilon$ and also fix G to generate

[4] One could attempt to reduce from the satisfiability problem for $\mathrm{PDL}_{\mathsf{tree}}[\downarrow]$—which is ExpTime-complete (Afanasiev et al., 2005)—, but it seems to us that such a proof would require changing the structure of the satisfaction witnesses by adding ε-leaves, and we do not see any straightforward way of handling this modification in the formula.

a parse forest encoding game trees; we use a $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula φ to check that there exists a winning strategy. See App. A for details. □

As can be seen from this proof idea, the fact that $w = \varepsilon$ and G is cyclic plays an important role, because the parse forest is essentially unconstrained. This is a good incentive to examine what happens when G is acyclic and/or ε-free, especially since those cases are most relevant for the applications we described in Section 3.

4.1 The Acyclic ε-free Case: Mixed Model-Theoretic Syntax

Let us therefore consider the other end of our spectrum, which we claimed was of particular relevance for the mixed approach to model-theoretic syntax we presented in Section 3.1: if G is acyclic and ε-free, then $\mathcal{A}_{G,w}$ is a non-recursive tree automaton generating a *finite* parse forest, albeit it might contain exponentially many trees. This yields an ExpTime algorithm that performs $\mathrm{PDL}_{\mathsf{tree}}$ model-checking (in Ptime (Lange, 2006)) on each tree individually. We can try to refine this first approach and resort to (Benedikt et al., 2008, Lemma 7.5), which entails that the problem for the $\mathrm{PDL}_{\mathsf{core}}$ fragment is in PSpace, but we can do better:

Proposition 3. *PFMC with acyclic and ε-free grammars is* NPTime*-complete; hardness holds even for fixed G and for φ in* $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$.

Proof of the Upper Bound. We show that the parse trees in $L_{G,w} = L(\mathcal{A}_{G,w})$ are of polynomial size in $|G|$ and $|w|$. The NPTime algorithm then guesses a tree in $L_{G,w}$ and checks that it is a model in polynomial time (Lange, 2006).

Claim 1. Let $G = \langle N, \Sigma, P, S\rangle$ be an acyclic and ε-free CFG. Let $w \in \Sigma^*$. Any parse tree t in $L_{G,w}$ has at most $|N|(|w| - 1) + |w|$ nodes.

Consider the run of $A_{G,w}$ on t: each node of t is labeled by a state (i, A, j) describing two positions $0 \le i \le j \le n$ in w and a nonterminal A in N. Because G is ε-free, we know that $i < j$. We claim that the set of nodes labelled with positions (i, j) forms a connected chain.

To see this, suppose two nodes a and b are both labelled with positions (i, j). Suppose first that neither a nor b is an ancestor of the other. Let then $c \notin \{a, b\}$ be their least common ancestor (lca): c must have at least two children, and its children will be labelled with non-overlapping positions—recall that $i < j$. Only one of these non-overlapping intervals can contain the interval (i, j). The child corresponding to that interval would then be the lca of a and b, in contradiction with c being their lca: hence one of a or b is the lca of a and b.

Suppose now without loss of generality that a is an ancestor of b. Observe that a descendant of a would be labelled with a sub-interval of (i, j), and an ancestor of b would be labelled with a super-interval of (i, j). This forces every node in the path from a to b to be labelled with (i, j). Hence, the nodes labelled with (i, j) form a connected chain.

Since G is acyclic, each chain of nodes $(i, A_1, j), (i, A_2, j) \cdots (i, A_p, j)$ having the same positions (i, j) cannot have a non-terminal A_k occuring twice, or the grammar would allow a cycle. Therefore, each such chain will have at most $|N|$ nodes. We can 'collapse' these chains to form a tree where each (i, j)

pair appears at most once, and every node (except the leaves) has at least two children. Since there are exactly $|w|$ leaves (G is ε-free), there can be at most $|w|-1$ internal nodes in such a tree. We obtain that there were at most $|N|(|w|-1)$ internal nodes in the original parse tree, i.e. at most $|N|(|w|-1)+|w|$ nodes in the full parse tree. $\square$

Proof Idea for the Lower Bound. We reduce from 3SAT with a fixed grammar G and a $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula φ; see App. B for details. $\square$

4.2 Non-Recursive DTDs

Let us turn now to the more involved cases where G is either acyclic or ε-free: we rely in both cases for the upper bounds on the same result that extends Lemma 7.5 of Benedikt et al. (2008) to handle $\mathrm{PDL}_{\mathsf{tree}}$ instead of $\mathrm{PDL}_{\mathsf{core}}$:

Proposition 4. *Satisfiability of* $\mathrm{PDL}_{\mathsf{tree}}$ *in presence of a non-recursive DTD is* PSPACE*-complete.*

In this proposition, a *document type definition* (DTD) is a generalized CFG $D = \langle N, P, S\rangle$ where P is a mapping from N to *content models* in $\mathrm{Reg}(N^*)$ the set of regular languages over N—we will assume these content models to be described by finite automata (NFA). Given D, the derivation relation $\Rightarrow$ relates $\beta A\gamma$ to $\beta\alpha\gamma$ iff α is in $P(A)$; a DTD is *non-recursive* if no nonterminal has a derivation $A \Rightarrow^+ \beta A \gamma$ for some β, γ in N^*. Note that a non-recursive DTD might still generate an infinite tree language, but that all its trees will have a depth bounded by $|N|$.

Proof Idea for Prop. 4. The hardness part is proven by Benedikt et al. (2008) in their Prop. 5.1.

For the upper bound, we reduce to the emptiness problem of a 2-way alternating parity *word* automaton, which is in PSPACE (Serre, 2006). The key idea, found in Benedikt et al.'s work, is to encode trees of bounded depth as XML strings (i.e. with opening and closing tags): both the DTD D and the formula φ can then be encoded as alternating parity word automata $\mathcal{A}_D$ and $\mathcal{A}_\varphi$ of polynomial size. Because we handle the full $\mathrm{PDL}_{\mathsf{tree}}$ instead of only $\mathrm{PDL}_{\mathsf{core}}$, our construction for $\mathcal{A}_\varphi$ has to extend that of Benedikt et al.—for instance, we cannot assume $\mathcal{A}_\varphi$ to be loop-free. See App. C for details. $\square$

4.3 Acyclic Case: Ambiguity Filtering

We are now ready to attack the case of acyclic grammars. This restriction is enough to ensure that the parse forest is finite, and, more importantly, $\mathcal{A}_{G,w}$ is trivially non-recursive, thus Prop. 4 immediately yields a PSPACE upper bound. In fact, this is optimal:

Proposition 5. *PFMC with acyclic grammars is* PSPACE*-complete; hardness holds even for fixed w and for φ in* $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$.

Proof. Because G is acyclic, for any w, the trimmed version of $\mathcal{A}_{G,w}$ is a non-recursive tree automaton. Indeed, in this automaton, if a state (i_0, A, i_k) of $\mathcal{A}_{G,w}$ rewrites in n steps into a tree t with leaves labeled by $(i_0, X_1, i_1)\cdots(i_{k-1}, X_k, i_k)$, then by induction on n, $A \Rightarrow^n X_1 \cdots X_k$ in G. If the automaton is trim, then the existence of a state (i, B, j) implies that B derives the factor $a_{i+1} \cdots a_j$ of w. Thus, if (i, A, j) were to rewrite in at least one step into a tree $C[(i, A, j)]$ in the trimmed $\mathcal{A}_{G,w}$, then the tree $C[\varepsilon]$ has ε as yield, and there would be a cycle $A \Rightarrow^+ A$ in G, a contradiction. It remains to localize $\mathcal{A}_{G,w}$ by relabelling the rules $(i, A, j) \to A(q_1 \cdots q_m)$ as $(i, A, j) \to (i, A, j)(q_1 \cdots q_m)$ to obtain a *local* non-recursive tree automaton, which is just a particular case of a non-recursive DTD, and interpret the propositions p in $V' = \mathsf{AP}$ as $\bigvee_{0 \leq i \leq j \leq n} (i, p, j)$ over $Q_{G,w}$ in φ to apply Prop. 4 and obtain the upper bound.

The lower bound holds for the $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ satisfiability problem in presence of non-recursive and no-star DTDs (Benedikt et al., 2008, Prop. 5.1), which is easy to reduce to our problem by simply adding ε-leaves in the DTD; this lower bound thus already holds for a fixed $w = \varepsilon$. □

4.4 ε-Free Case: $\mathrm{PDL}_{\mathsf{tree}}$ Recognition

A key question if model-theoretic syntax is to be used in practice for natural language processing is the following *recognition problem*:

$\mathrm{PDL}_{\mathsf{tree}}$ Recognition Problem.

input a $\mathrm{PDL}_{\mathsf{tree}}$ formula φ, a word w in AP^*, and a distinguished proposition s in AP,

question does there exist a tree t with yield w and root label s s.t. $t \models \varphi$? □

Note in particular that the statement of the problem excludes ε-labeled leaves, which would require a different formulation and would yield an ExpTime-complete problem.

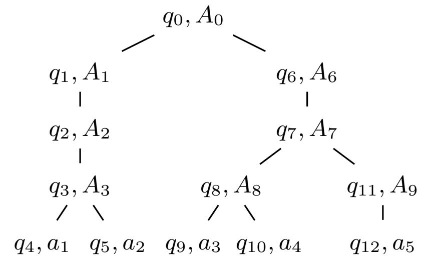
The recognition problem motivates the last case of our study: Due to cycles, an ε-free grammar G can have infinitely many parses for a given input string w, and its parse trees unbounded depth. Nevertheless, recursions in a parse forest of an ε-free grammar display a particular shape: they are *chains* of unit rules $q_i \to A_i(q_{i+1})$. The key idea here is that such chains define regular languages of single-strand branches, which can be encoded in a non-recursive DTD by 'rotating' them, i.e. seeing the chain as a siblings sequence instead of a parents sequence, taking advantage of the DTD's ability to describe trees of unbounded rank. This hints at a reduction to the PSpace algorithm of Prop. 4.

4.4.1 PFMC in the ε-free Case

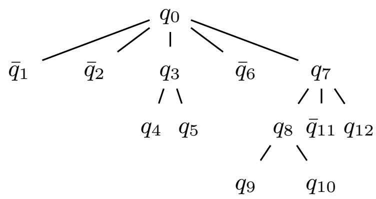
We want to reduce the problem to the satisfiability problem for $\mathrm{PDL}_{\mathsf{tree}}$ in presence of a non-recursive DTD and use Prop. 4. Our algorithm starts by constructing $\mathcal{A}_{G,w}$ in polynomial time on binarized trees. As in the proof of Prop. 5, we consider 'localized' rules $q \to q(q_1\, q_2)$ of $\mathcal{A}_{G,w}$, and replace them by productions of the form $q \to \mathsf{chains}(q_1)\mathsf{chains}(q_2)$ where the $\mathsf{chains}(q_i)$ are the languages of single chains out of q_i. By suitably labeling our trees, we can interpret φ over those transformed trees.

Proposition 6. *PFMC with ε-free grammars is in* PSpace.

Proof. Let $G = \langle N, \Sigma, P, S \rangle$, w be a string in Σ^*, and φ be a $\mathrm{PDL}_{\mathsf{tree}}$ formula. Without loss of generality, we assume G to have productions with right-parts of



(a) A run of $\mathcal{A}_{Q,w}$ on a tree in a parse forest of w according to G.



(b) The corresponding tree in $L(D)$.

Figure 4: The tree transformation for the proof of Prop. 6.

length at most 2; since G is ε-free, these right-parts have length at least one. We want to construct a non-recursive DTD $D = \langle N', P', S' \rangle$ and a $\text{PDL}_{\mathsf{tree}}$ formula φ' s.t. the parse forest model checking problem on G, w, and φ has a solution iff φ is satisfiable in presence of D, thereby reducing our instance to an instance of a problem in PSPACE by Prop. 4.

We build D from the polynomial-sized automaton $\mathcal{A}_{G,w}$ by removing chains of *unit* rules $q \to A(q')$ of $\mathcal{A}_{G,w}$; recall that this automaton uses states of form $q = (i, X, j)$ where $0 \leq i < j \leq |w|$ and X is in V. Let for this $\bar{Q}_{G,w}$ be a disjoint copy of $Q_{G,w}$ and define $N' \stackrel{\text{def}}{=} \bar{Q}_{G,w} \uplus Q_{G,w}$.

Chain Sequences For each q in $Q_{G,w}$, we consider the set of sequences of successive states $q = q_0, q_1, \ldots, q_n$ we can visit using only unit rules $q_i \to A_i(q_{i+1})$ of $\delta_{G,w}$ and such that q_n has a binary rule $q_n \to A_n(q'\, q'')$ or a nullary rule $q_n \to a()$ in $\delta_{G,w}$. More precisely, we are interested in the relabeled sequence $\bar{q} = \bar{q}_0, \bar{q}_1, \ldots, \bar{q}_{n-1}, q_n$ of copies $\bar{q}_i$ of q_i, except on the very last position. We call $\mathsf{chains}(q)$ the language of such sequences. Formally, $\mathsf{chains}(q)$ is a regular language over N' that we can define thanks to a NFA $\mathcal{A}_q \stackrel{\text{def}}{=} \langle N', N', \delta_q, \{\bar{q}\}, Q_{G,w} \rangle$ with state space N' where

$$\begin{aligned}\delta_q \stackrel{\text{def}}{=} &\{(\bar{p}, \bar{p}, \bar{p}') \mid \exists A \in N, p \to A(p') \in \delta_{G,w}\}\\ &\cup \{(\bar{p}, p, p) \mid \exists A \in N, \exists p_1, p_2 \in Q_{G,w}, p \to A(p_1 p_2) \in \delta_{G,w}\}\\ &\cup \{(\bar{p}, p, p) \mid \exists a \in \Sigma, p \to a() \in \delta_{G,w}\}\ .\end{aligned}$$

Note that $\mathcal{A}_q$ has a size linear in that of $\mathcal{A}_{G,w}$. We can see chains as a regular substitution from $Q_{G,w}^*$ to N'^* by setting $\mathsf{chains}(\varepsilon) \stackrel{\text{def}}{=} \varepsilon$ and $\mathsf{chains}(uv) \stackrel{\text{def}}{=} \mathsf{chains}(u)\mathsf{chains}(v)$ for all u, v in $Q_{G,w}^*$.

The DTD We can now express the productions P' of D:

$$P(\bar{q}) \stackrel{\text{def}}{=} \varepsilon \qquad\qquad P(q) \stackrel{\text{def}}{=} \bigcup_{q \to A(q_1\, q_2) \in \delta_{G,w}} \mathsf{chains}(q_1\, q_2) \cup \bigcup_{q \to a() \in \delta_{G,w}} \varepsilon\ .$$

Thus, the symbols in $\bar{Q}_{G,w}$ are leaves and only employed to represent a chain sequence that has been transformed into a sequence of siblings in the DTD. Also, because any word in $\mathsf{chains}(q)$ for some q is of form up with u in $\bar{Q}^*_{G,w}$ and p in $Q_{G,w}$, any internal node in a tree of D has exactly two children labeled by states in $Q_{G,w}$. Therefore, and because G is ε-free, we get that D is non-recursive. See Figure 4 for an illustration of the tree transformation we operated.

The Formula It remains to define a formula φ' that will be interpreted on the transformed trees of D. For this, we need to interpret the atomic propositions in $\mathsf{AP} = V$ over the new set of labels N', and to interpret the child $\downarrow$ and sibling $\rightarrow$ relations.

Regarding the atomic propositions, we can interpret a label X in V as

$$\bigvee_{0 \leq i < j \leq |w|} (i, X, j) \vee \overline{(i, X, j)} \qquad \text{(interpretation of } X\text{)}$$

over N'.

Regarding the relations, we first define $\mathsf{bar} \stackrel{\text{def}}{=} \bigvee_{\bar{q} \in \bar{Q}_{G,w}} \bar{q}$ to help us differentiate between 'rotated' nodes and preserved ones. We then interpret $\downarrow$ as a disjunction of paths depending on whether we are on a rotated node, where the test $[\leftarrow]\neg\mathsf{bar}$ allows to check that the current node is an 'original' child:

$$(\neg\mathsf{bar}?; \downarrow; [\leftarrow]\neg\mathsf{bar}?) + (\mathsf{bar}?; \rightarrow) \qquad \text{(interpretation of } \downarrow\text{)}$$

For $\rightarrow$, we set:

$$([\leftarrow]\neg\mathsf{bar}?); (\mathsf{bar}?; \rightarrow)^*; \neg\mathsf{bar}?; \rightarrow\ . \qquad \text{(interpretation of } \rightarrow\text{)}$$

The initial test prevents the nodes taken from a chain from having a right sibling; then the test sequence advances to the end of the chain before we make the actual move to the original right sibling.

We can conclude by first noting that both D and φ' can be computed in logarithmic space in the size of the input, and then by invoking Prop. 4. $\square$

Prop. 6 is optimal:

Proposition 7. *PFMC with ε-free grammars is* PSPACE*-hard, even for fixed G and w and for φ in* $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$.

Proof Idea. The proof is by reduction from membership in a linear bounded automaton. We fix $w = a$ for some symbol a of Σ, and also fix the CFG G to basically generate any single-strand tree with a root S and a leaf a over a fixed alphabet. A $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula of polynomial size then checks that this tree encodes an accepting run of the LBA. See App. D for details. $\square$

4.4.2 $\mathrm{PDL}_{\mathsf{tree}}$ Recognition

The previous approaches to the recognition problem have used tree automata techniques (e.g. Cornell, 2000) or tableau-like techniques (Palm, 2004). In both cases, exponential time upper bounds were reported by the authors—to be fair, these algorithms solve the *parsing* problem and find a representation of *all* the parses for w compatible with φ—, but we can improve on this thanks to Prop. 6:

Corollary 1. $\mathrm{PDL}_{\mathsf{tree}}$ *recognition is* PSPACE*-complete; hardness holds even for fixed* w *and for* φ *in* $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$.

Proof Sketch. The lower bound stems from an easy reduction from Prop. 7: we can encode the grammar G into a $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula φ_G as in Example 2 and reduce to the recognition problem for w and $\varphi \wedge \varphi_G$.

For the upper bound, we can assume as usual φ to work on a binary encoding of trees. The idea is to reduce to the PFMC problem with a 'universal' CFG that accepts all the trees of rank at most 2 over AP. A smallish issue is that we need to separate between nonterminal and terminal labels, but we can create a disjoint copy $N \stackrel{\text{def}}{=} \{P \mid p \in \mathsf{AP}\}$ of AP and interpret φ as a formula over $N \uplus \mathsf{AP}$ with $P \vee p$ as the interpretation of p. This grammar has then S as axiom and productions $A \to X\ Y$ and $A \to X$ for all A in N and X, Y in V, and we can resort to Prop. 6 to conclude. $\square$

5 Conclusion

Because $\mathrm{PDL}_{\mathsf{tree}}$ formulæ can freely navigate in trees, properties that rely on long-distance relations are convenient to express, in contrast with the higly local view provided by a grammar production. However, this expressiveness comes at a steep price, as recognition problems using $\mathrm{PDL}_{\mathsf{tree}}$ are EXPTIME-complete instead of PTIME-complete on CFGs.

The $\mathrm{PDL}_{\mathsf{tree}}$ model-checking of the parse trees of a CFG allows to mix the two approaches, using a grammar for the bulk work of describing trees and using more sparingly a $\mathrm{PDL}_{\mathsf{tree}}$ formula for the fine work. We argue that this trade-off finds natural applications in computational linguistics and compilers construction, where sensible restrictions on the grammar lower the complexity to NPTIME or PSPACE.

An additional consequence is that the recognition problem for $\mathrm{PDL}_{\mathsf{tree}}$ is in PSPACE. This is a central problem in model-theoretic syntax, and this lower complexity suggests that 'lazy' approaches, in the spirit of the tableau construction of Palm (2004), should perform significantly better than the automata constructions of Cornell (2000).

More broadly, we think that our initial investigations of model-checking parse trees open the way to a new range of applications of model-checking techniques on parse structures and grammars. In particular, as pointed out in our examples, both in computational linguistics and in ambiguity filtering for programming languages, there are incentives to look at classes of models that generalize finite trees over a finite label set—we expect furthermore these generalizations to differ from the data tree model found in the XML literature.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments; in particular for pointing out that the $\text{PDL}_{\textsf{tree}}$ formula in Example 5 could be expressed in XPath 1.0.

References

Adams, M.D., 2013. Principled parsing for indentation-sensitive languages: revisiting Landin's offside rule. In *POPL 2013, 40th Annual Symposium on Principles of Programming Languages*, pages 511–522. doi:10.1145/2429069.2429129.

Afanasiev, L., Blackburn, P., Dimitriou, I., Gaiffe, B., Goris, E., Marx, M., and de Rijke, M., 2005. PDL for ordered trees. *Journal of Applied Non-Classical Logics*, 15 (2):115–135. doi:10.3166/jancl.15.115-135.

Bar-Hillel, Y., Perles, M., and Shamir, E., 1961. On formal properties of simple phrase-structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft, und Kommunikationsforschung*, 14:143–172.

Benedikt, M., Fan, W., and Geerts, F., 2008. XPath satisfiability in the presence of DTDs. *Journal of the ACM*, 55(2:8). doi:10.1145/1346330.1346333.

Blackburn, P., Gardent, C., and Meyer-Viol, W., 1993. Talking about trees. In *EACL '93*, pages 21–29. ACL Press. doi:10.3115/976744.976748.

Blackburn, P., Meyer-Viol, W., and de Rijke, M., 1996. A proof system for finite trees. In *CSL '95, 9th EACSL Annual Conference on Computer Science Logic*, volume 1092 of *Lecture Notes in Computer Science*, pages 86–105. Springer. doi:10.1007/3-540-61377-3_33.

Blackburn, P., de Rijke, M., and Venema, Y., 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Bojańczyk, M. and Lasota, S., 2012. An extension of data automata that captures XPath. *Logical Methods in Computer Science*, 8(1):5. doi:10.2168/LMCS-8(1:5)2012.

Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M., 2009. An automata-theoretic approach to Regular XPath. In *DBPL 2009, 12th Biennial Symposium on Database Programming Languages*, volume 5708 of *Lecture Notes in Computer Science*, pages 18–35. Springer. doi:10.1007/978-3-642-03793-1_2.

Candito, M.H. and Seddah, D., 2012. Le corpus sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical. In *TALN 2012, Traitement automatique des langues naturelles*. https://gforge.inria.fr/projects/sequoiabank/.

Chlebus, B.S., 1986. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374–392. doi:10.1016/0022-0000(86)90036-X.

Cornell, T., 2000. Parsing and grammar engineering with tree automata. In *AMiLP 2000, Algebraic Methods in Language Processing*, pages 267–274.

Figueira, D., 2012. Alternating register automata on finite words and trees. *Logical Methods in Computer Science*, 8(1):22. doi:10.2168/LMCS-8(1:22)2012.

Fischer, M.J. and Ladner, R.E., 1979. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211. doi:10.1016/0022-0000(79)90046-1.

Göller, S., Lohrey, M., and Lutz, C., 2009. PDL with intersection and converse: satisfiability and infinite-state model checking. *Journal of Symbolic Logic*, 74(1):279–314. doi:10.2178/jsl/1231082313.

Gottlob, G. and Koch, C., 2002. Monadic queries over tree-structured data. In *LICS 2002*, *17th Annual IEEE Symposium on Logic in Computer Science*, pages 189–202. doi:10.1109/LICS.2002.1029828.

Ishihara, Y., Morimoto, T., Shimizu, S., Hashimoto, K., and Fujiwara, T., 2009. A tractable subclass of DTDs for XPath satisfiability with sibling axes. In Gardner, P. and Geerts, F., editors, *DBPL 2009*, *12th Biennial Symposium on Database Programming Languages*, volume 5708 of *Lecture Notes in Computer Science*, pages 68–83. Springer. doi:10.1007/978-3-642-03793-1_5.

Kallmeyer, L., Lichte, T., Maier, W., Parmentier, Y., Dellert, J., and Evang, K., 2008. TuLiPA: towards a multi-formalism parsing environment for grammar engineering. In *GEAF 2008*, *Workshop on Grammar Engineering Across Frameworks*, pages 1–8. ACL Press.

Kats, L.C., Visser, E., and Wachsmuth, G., 2010. Pure and declarative syntax definition: paradise lost and regained. In *OOPSLA 2010*, *ACM international conference on Object oriented programming systems languages and applications*, pages 918–932. ACM. doi:10.1145/1869459.1869535.

Klint, P. and Visser, E., 1994. Using filters for the disambiguation of context-free grammars. In Pighizzini, G. and San Pietro, P., editors, *ASMICS Workshop on Parsing Theory*, Technical Report 126-1994, pages 89–100. Università di Milano.

Klint, P., Lämmel, R., and Verhoef, C., 2005. Toward an engineering discipline for grammarware. *ACM Transactions on Software Engineering and Methodology*, 14 (3):331–380. doi:10.1145/1072997.1073000.

Kracht, M., 1995. Syntactic codes and grammar refinement. *Journal of Logic, Language, and Information*, 4(1):41–60. doi:10.1007/BF01048404.

Lai, C. and Bird, S., 2010. Querying linguistic trees. *Journal of Logic, Language, and Information*, 19(1):53–73. doi:10.1007/s10849-009-9086-9.

Lange, M., 2006. Model checking propositional dynamic logic with all extras. *Journal of Applied Logic*, 4(1):39–49. doi:10.1016/j.jal.2005.08.002.

Libkin, L. and Sirangelo, C., 2010. Reasoning about XML with temporal logics and automata. *Journal of Applied Logic*, 8(2):210–232. doi:10.1016/j.jal.2009.09.005.

Marx, M., 2005. Conditional XPath. *ACM Transactions on Database Systems*, 30(4): 929–959. doi:10.1145/1114244.1114247.

Montazerian, M., Wood, P., and Mousavi, S., 2007. XPath query satisfiability is in PTIME for real-world DTDs. In Barbosa, D., Bonifati, A., Bellahsène, Z., Hunt, E., and Unland, R., editors, *XSym 2007*, *5th International XML Database Symposium*, volume 4704 of *Lecture Notes in Computer Science*, pages 17–30. Springer. doi: 10.1007/978-3-540-75288-2_3.

Moore, R.C., 2004. Improved left-corner chart parsing for large context-free grammars. In *New Developments in Parsing Technology*, pages 185–201. Springer. doi:10.1007/1-4020-2295-6_9. See http://www.informatics.sussex.ac.uk/research/groups/nlp/carroll/cfg-resources/.

Palm, A., 1999. Propositional tense logic of finite trees. In *MOL 6*, *6th Biennial Conference on Mathematics of Language*.

Palm, A., 2004. Model theoretic syntax and parsing: An application to temporal logic. In *FG-MOL 2001*, *Joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language*, volume 53 of *Electronic Notes in Theoretical Computer Science*, pages 261–273. Elsevier. doi:10.1016/S1571-0661(05)82588-5.

Pullum, G.K. and Scholz, B.C., 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *LACL 2001*, *4th International Conference on Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture*

Notes in Computer Science, pages 17–43. Springer. doi:10.1007/3-540-48199-0_2.

Rogers, J., 1998. *A Descriptive Approach to Language Theoretic Complexity*. CSLI Publications.

Schmitz, S., 2010. An experimental ambiguity detection tool. *Science of Computer Programming*, 75(1–2):71–84. doi:10.1016/j.scico.2009.07.002.

Serre, O., 2006. Parity games played on transition graphs of one-counter processes. In Aceto, L. and Ingólfsdóttir, A., editors, *FoSSaCS 2006, 9th International Conference on Foundations of Software Science and Computational Structures*, volume 3921 of *Lecture Notes in Computer Science*, pages 337–351. Springer. doi:10.1007/11690634_23.

ten Cate, B. and Segoufin, L., 2010. Transitive closure logic, nested tree walking automata, and XPath. *Journal of the ACM*, 57(3):18:1–18:41. doi:10.1145/1706591.1706598.

Thorup, M., 1994. Controlled grammatic ambiguity. *ACM Transactions on Programming Languages and Systems*, 16(3):1024–1050. doi:10.1145/177492.177759.

Thorup, M., 1996. Disambiguating grammars by exclusion of sub-parse trees. *Acta Informatica*, 33(5):511–522. doi:10.1007/BF03036460.

A General Case

PDL satisfiability is known to be ExpTime-complete in general (Fischer and Ladner, 1979). The general case of the parse forest model-checking problem, i.e. when G is an arbitrary grammar, is also ExpTime-complete. The upper bound follows from classical techniques (Calvanese et al., 2009)—see the proof sketch of Proposition 1.

The lower bound could be proven by a reduction from PDL satisfiability using a 'universal' CFG as in the proof of Corollary 1. However, this proof does not lend itself very easily to the restricted case we want to consider, where w and G are fixed and φ is a downward $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula. We present in this section a reduction from the *two-player corridor game*, which is known to be ExpTime-hard (Chlebus, 1986), adapted from a similar proof for the hardness of PDL satisfiability by Blackburn et al. (2001, Theorem 6.52).

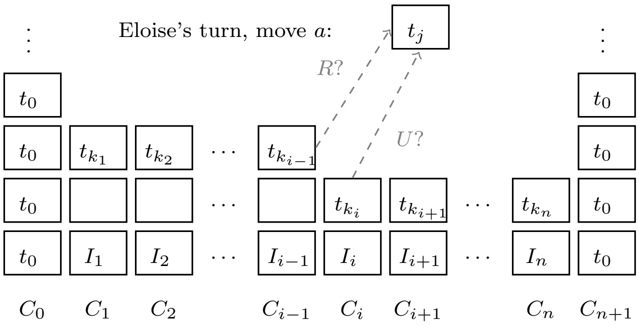
Two Player Corridor Game sees two players, Eloise and Abelard, compete by tiling a corridor. The tiles are squares decorated by $s+2$ different patterns $T = \{t_0, \ldots, t_{s+1}\}$; two binary relations U and R over T tell if a tile can be placed on top of the other and to the right of the other. Two tiles are distinguished: t_0 is called the *white* tile and t_{s+1} the *winning* tile. The corridor is made of $n+2$ columns of infinite height, with the first and last columns filled with white tiles t_0 and delimiting n columns for the play. The initial bottom row is tiled by a sequence $I_1 \cdots I_n$ of tiles, which is assumed to be correct, i.e. to respect the R relation.

The players alternate and choose a next tile in T and place it in the next position, which is the lowest leftmost free one—thus the chosen tile should match the tile to its left (using R) and the tile below (using U)—; see Figure 5a. Eloise starts the game and wins if after a finite number of rounds, the winning tile t_{s+1} is put in column 1. Given an instance of the 2-players corridor tiling game, i.e. $\langle s+2, I_1 \cdots I_n, R, U\rangle$, deciding whether Eloise has a *winning strategy*, i.e. a way of winning no matter what Abelard plays, is ExpTime-complete.

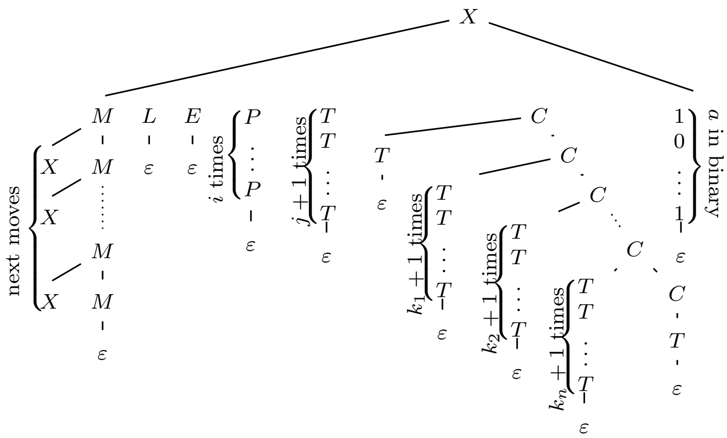
Notation We represent strategy trees as parse trees. Our $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula φ will ensure that the parse tree is indeed a valid game tree, and that it encodes a winning strategy for Eloise.

A game turn is encoded locally by an X-labeled node and its immediate children, with the next reachable configurations reachable through a path of M-labeled nodes. More precisely, each X node has the following children (see Figure 5b):

- a node labeled either W or L, stating that the configuration is winning or not for Eloise,
- a node labeled either E or A, stating whether it is Eloise's or Abelard's turn to move,
- a chain of i P-labeled nodes, stating that the current playing column is C_i,
- a chain of $j+1$ T-labeled nodes, stating that the chosen tile at this turn is t_j,



(a) One of Eloise's turns in the game.



(b) The tree encoding of the turn.

Figure 5: A turn of the 2-players corridor game and its tree encoding.

- a comb-shaped subtree of $n+2$ nodes labeled by C, describing the contents of the top layer in the corridor (which in general spans two rows), with a strand of $k+1$ T-labeled nodes telling for each column that tile t_k is on top,
- a chain of m nodes labeled 0 or 1, encoding in binary the number of the moves made so far; this chain does not need to be longer than $m \stackrel{\text{def}}{=} \lceil \log(2n^{s+3}) \rceil$—or some move would have been repeated.

The Grammar We *fix* $w \stackrel{\text{def}}{=} \varepsilon$ and $G \stackrel{\text{def}}{=} \langle N, \emptyset, P, X \rangle$ over the nonterminal alphabet $N = \{X, M, W, L, E, A, P, T, C, 0, 1\}$ with productions (with a slightly extended syntax with alternatives built-in the productions right-hand sides):

$$
\begin{array}{lll}
X \to M\,(W \mid L)\,(E \mid A)\,P\,T\,C\,(0 \mid 1) & M \to X\,M \mid \varepsilon & C \to T \mid T\,C \\
P \to P \mid \varepsilon & T \to T \mid \varepsilon & W \to \varepsilon \\
L \to \varepsilon & E \to \varepsilon & A \to \varepsilon \\
0 \to 0 \mid 1 \mid \varepsilon & 1 \to 0 \mid 1 \mid \varepsilon &
\end{array}
$$

Observe that all the tree encodings of strategies are generated by G, but that not all the trees of G encode a strategy: for instance, the number of C's might be different from $n+2$, or the described tiling might not respect the placement constraints, etc. The formula will check these conditions.

Game Structure and Mechanics We use the non-terminal labels and ε as atomic propositions in our $\mathrm{PDL}_{\mathsf{tree}}$ formula. Because there are at most $s+2$ choices of tiles at each turn, we can define the path

$$\mathsf{move} \stackrel{\text{def}}{=} \sum_{i=1}^{s+2} (\downarrow; M?)^i; \downarrow; X?$$

that relates two successive configurations. Further define the following formulæ for $0 \leq i \leq n+1$, $0 \leq j \leq s+1$, $1 \leq a \leq m$ and b in $\{0,1\}$:

$$\mathsf{p}(i) \stackrel{\text{def}}{=} \langle (\downarrow; P?)^i; \downarrow \rangle \varepsilon \qquad \mathsf{t}(j) \stackrel{\text{def}}{=} \langle (\downarrow; T?)^{j+1}; \downarrow \rangle \varepsilon$$
$$\mathsf{q}(a,b) \stackrel{\text{def}}{=} \langle (\downarrow; (0+1)?)^a \rangle b \qquad \mathsf{c}(i,j) \stackrel{\text{def}}{=} \langle (\downarrow; C?)^{i+1}; (\downarrow; T?)^{j+1}; \downarrow \rangle \varepsilon$$

In an X node, $\mathsf{p}(i)$ holds if the current column is C_i, $\mathsf{t}(j)$ if the chosen tile is T_j, $\mathsf{c}(i,k_i)$ if tile T_{k_i} is on top of column C_i, and $\mathsf{q}(m,b)$ if the binary encoding of the current move number has bit a set to b.

We ensure some preliminary structure on the game tree: At the start of the play, the current player must be Eloise, and the referee should have placed the initial tiles in the first row. The counter must be initialized to zero.

$$\varphi_1 \stackrel{\text{def}}{=} X \wedge (\langle \downarrow \rangle E) \wedge \mathsf{p}(1) \wedge \bigwedge_{i=1}^{n} \mathsf{c}(i, I_i) \wedge \bigwedge_{a=1}^{m} \mathsf{q}(a,0) \ .$$

In every state, the tiles in columns 0 and $n+1$ should be the white tile t_0.

$$\varphi_2 \stackrel{\text{def}}{=} [\downarrow^*]X \Rightarrow \mathsf{c}(0,0) \wedge \mathsf{c}(n+1,0) \ .$$

In every state with current column i, the next move should be at position $(i \bmod n)+1$.

$$\varphi_3 \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} [\downarrow^*](X \wedge \mathsf{p}(i)) \Rightarrow [\mathsf{move}]\mathsf{p}((i \bmod n)+1) \ .$$

The columns are updated with the correct tiles.

$$\varphi_4 \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} \bigwedge_{j=0}^{s+1} [\downarrow^*](X \wedge \mathsf{p}(i) \wedge \mathsf{t}(j)) \Rightarrow [\mathsf{move}]\mathsf{c}(i,j)$$
$$\wedge \bigwedge_{i \neq j=1}^{n} \bigwedge_{k=0}^{s+1} (X \wedge \mathsf{p}(i) \wedge \mathsf{c}(j,k)) \Rightarrow [\mathsf{move}]\mathsf{c}(j,k) \ .$$

Players alternate.

$$\varphi_5 \stackrel{\text{def}}{=} [\downarrow^*](X \wedge \langle \downarrow \rangle E \equiv [\mathsf{move}]\langle \downarrow \rangle A) \ .$$

The chosen tiles verify the adjacency constraints: define for this the Boolean:

$$\mathsf{adj}(i,j,k,\ell) \stackrel{\text{def}}{=} t_\ell \, U \, t_j \wedge (i > 0 \Rightarrow t_k \, R \, t_j) \wedge (i = 0 \Rightarrow t_0 \, R \, t_j) \wedge (i = n \Rightarrow t_j \, R \, t_0)$$

$$\varphi_6 \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} \bigwedge_{j,k,\ell=0}^{s+1} [\downarrow^*](X \wedge \mathsf{p}(i) \wedge \mathsf{t}(j) \wedge \mathsf{c}(i-1,k) \wedge \mathsf{c}(i,\ell)) \Rightarrow \mathsf{adj}(i,j,k,\ell) \; .$$

The counter is incremented.

$$\varphi_7 \stackrel{\text{def}}{=} \bigwedge_{d=1}^{m} \bigwedge_{a=1}^{d-1} \bigwedge_{b\in\{0,1\}} [\downarrow^*]\big(X \wedge \mathsf{q}(a,b) \wedge \mathsf{q}(d,0) \wedge \bigwedge_{e=d+1}^{m} \mathsf{q}(e,1)\big)$$
$$\Rightarrow [\mathsf{move}]\big(\mathsf{q}(a,b) \wedge \mathsf{q}(d,1) \wedge \bigwedge_{e=d+1}^{m} \mathsf{q}(e,0)\big) \; .$$

Winning Strategy The previous formulæ were making sure that the tree would be a proper game tree. We want now to check that it describes a winning strategy for Eloise: We should check that all the possible moves of Abelard are tested:

$$\varphi_8 \stackrel{\text{def}}{=} \bigwedge_{i=1}^{n} \bigwedge_{j,k,\ell=0}^{s+1} [\downarrow^*](X \wedge \mathsf{p}(i) \wedge \langle\downarrow\rangle E \wedge \mathsf{t}(k) \wedge \mathsf{c}((i \bmod n)+1,\ell) \wedge \mathsf{adj}(i,j,k,\ell)) \Rightarrow \langle\mathsf{move}\rangle \mathsf{t}(j) \; .$$

Finally, the winning condition should be met:

$$\varphi_9 \stackrel{\text{def}}{=} (\langle\downarrow\rangle W) \wedge [\downarrow^*](X \wedge \langle\downarrow\rangle W) \Rightarrow \big(\mathsf{c}(1,s+1)$$
(the game is immediately winning)
$$\vee\, ((\langle\downarrow\rangle E) \wedge (\langle\mathsf{move};\downarrow\rangle W))$$
(Eloise can win later)
$$\vee\, (\langle\downarrow\rangle A) \wedge (\langle\mathsf{move}\rangle\top) \wedge [\mathsf{move}]\langle\downarrow\rangle W)\big) \; .$$
(None of Abelard's moves can prevent Eloise from winning)

Finally, our final $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula is $\varphi \stackrel{\text{def}}{=} \bigwedge_{i=1}^{9} \varphi_i$. Because G and w are fixed and φ can be computed in space logarithmic in the size of the game instance, we have therefore shown the general PFMC problem to be ExpTime-hard.

B Acyclic and ε-Free Case: Proposition 3

We prove here the lower bound part of Proposition 3: the PFMC problem is NPTime-hard for acyclic and ε-free grammars.

Proof. We reduce 3SAT to our problem.

Fix the grammar $G \stackrel{\text{def}}{=} \langle \{S,F,T\}, \{a\}, P, S\rangle$ with productions:

$$S \to S\,F \mid S\,T \mid F \mid T \qquad\qquad F \to a \qquad\qquad T \to a$$

and consider an instance $\psi = \bigwedge_{i=1}^{m} C_i$ of 3SAT where each C_i is a disjunction of literals over n variables $\{x_1,\ldots,x_n\}$. Define $w \stackrel{\text{def}}{=} a^n$.

Any parse tree t of w will have a 'comb' shape of length n with S-labeled nodes, each giving rise to one of F or T as a child. The parse forest is thus in bijection with the set of valuations of $\{x_1, \ldots, x_n\}$: if the value of variable x_i is 0, then in our encoding, the ith S node has a node with label F as a child; otherwise, it has a node with label T as a child.

Given such an encoded valuation, our formula φ must verify that each clause is satisfied. For a clause $C_i = \ell_{i,1} \vee \ell_{i,2} \vee \ell_{i,3}$ with $\ell_{i,j} = x_{k_j}$ or $\ell_{i,j} = \neg x_{k_j}$, define $\varphi_i \stackrel{\text{def}}{=} \bigvee_{j=1}^{3} \langle (S;\downarrow)^{k_j} \rangle \beta_{i,j}$ where $\beta_{i,j} = F$ if $\ell_{i,j} = \neg x_{k_j}$ and $\beta_{i,j} = T$ otherwise. Finally, let $\varphi \stackrel{\text{def}}{=} \bigwedge_{i=1}^{m} \varphi_i$. Then $t \models \varphi$ if and only if the corresponding assignment of the variables is a satisfying assignment. Because G is fixed and w and φ can be computed in space logarithmic in the size of the 3SAT instance, this shows the NPTime-hardness of the PFMC problem in the acyclic ε-free case. Note that φ is in $\text{PDL}_{\mathsf{core}}[\downarrow]$. □

C Model-Checking Non-Recursive DTDs: Proposition 4

We present in this section a proof of Proposition 4: the satisfiability of a $\text{PDL}_{\mathsf{tree}}$ formula φ in presence of a non-recursive DTD D is PSpace-complete.

The lower bound is proved as Proposition 5.1 by Benedikt et al. (2008), and we follow their general proof plan from Lemma 7.5 for the upper bound. As presented in the main text, the fact that we consider a non-recursive DTD means that the height of any tree of interest is bounded by $|N|$ the number of nonterminals of the DTD. The proof plan is then to consider XML word encodings of trees, and construct two *2-way alternating parity word automata* (2APWA) $\mathcal{A}_D$ and $\mathcal{A}_\varphi$ of polynomial size which will respectively recognize the XML encodings of the trees of D and of the models of φ of height bounded by $|N|$. Then, by taking the conjunction of the two automata, we reduce the initial satisfiability problem to a 2APWA emptiness problem, which is known to be in PSpace by the results of Serre (2006).

We can find a suitable construction for an automaton $\mathcal{A}_D$ for D as Claim 7.7 of (Benedikt et al., 2008), thus we will only present the construction of $\mathcal{A}_\varphi$.

XML Encoding Define the alphabet

$$\mathsf{XML}(N) \stackrel{\text{def}}{=} \{\langle X \rangle, \langle / X \rangle \mid X \in N\}$$

and choose a fresh root symbol r not in N. We encode our a tree t as $\langle \mathsf{r} \rangle \mathsf{stream}(t) \langle / \mathsf{r} \rangle$ where the *XML streaming* function is defined inductively on terms by

$$\mathsf{stream}(f(t_1 \cdots t_m)) \stackrel{\text{def}}{=} \langle f \rangle \mathsf{stream}(t_1) \cdots \mathsf{stream}(t_m) \langle / f \rangle \ .$$

2-Way Alternating Parity Word Automata A *positive boolean formula* f in $\mathbb{B}^+(X)$ over a set X of variables is defined by the syntax

$$f ::= \top \mid \bot \mid f \wedge f \mid f \vee f \ .$$

A subset $X' \subseteq X$ satisfies a formula f, written $X' \models f$, if the formula is satisfied by the valuation $x \mapsto \top$ whenever $x \in X'$ and $x \mapsto \bot$ if $x \in X' \setminus X$.

A *2-way alternating parity word automaton* is a tuple $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, c\rangle$ where Q is a finite set of states, Σ a finite alphabet, $q_0 \in Q$ an initial state, c a coloring from Q to a finite set of priorities $C \subseteq \mathbb{N}$, and δ a transition function from $Q \times \Sigma$ to $\mathbb{B}^+(Q \times \{-1, 0, 1\})$ that associates to a current state and current symbol boolean formulæ on pairs (q', d) of a new state q' and a direction d.

A *run* of a 2APWA on a finite word $w = a_1 \cdots a_n$ in Σ^* is a generally infinite tree with labels in $Q \times \{1, \ldots, n\}$ holding a current state and a current position in w, such that the root is labeled $(q_0, 1)$, and every node labeled (q, i) with has a children set $\{(q_1, i_1), \ldots, (q_m, i_m)\}$ that satisfies $\delta(q, a_i)$. A run is *accepting* iff for every branch, the smallest priority $c(q)$ that occurs infinitely often among the nodes (q, i) is even—this also means in particular that any finite run is accepting—, and w is accepted if there exists some accepting run for it.

Inductive Construction We construct $\mathcal{A}_\varphi \stackrel{\text{def}}{=} \langle Q_\varphi, \Sigma, \delta_\varphi, q_{0,\varphi}, c_\varphi\rangle$ by induction on the subterms of the formula φ. We work with the alphabet $\Sigma \stackrel{\text{def}}{=} \mathsf{XML}(N) \uplus \{\langle \mathsf{r}\rangle, \langle /\mathsf{r}\rangle\}$ and set $n \stackrel{\text{def}}{=} |N|$—which is the maximum height of any tree of D. The guiding principles in this construction is that our inductively constructed automaton will track their height relative to that of their starting position. Because we are working on trees of bounded depth, this can be achieved by considering states that combine a 'control' state with a height in $\{-n, \ldots, n\}$.

Let us start with the base cases for node formulæ: by convention, our automata for a node formulæ must check that their starting positions are labeled by opening tags:

$\boldsymbol{\mathcal{A}_p}$ The automaton checks if it starts at an opening node $\langle p\rangle$. It immediately goes into either an accepting or rejecting state. Formally, $Q_p \stackrel{\text{def}}{=} \{q_{p,0}\}$, the coloring c_p maps $q_{p,0}$ to 1, and $\delta(q_{p,0}, \langle p\rangle) \stackrel{\text{def}}{=} \top$ and $\delta_p(q_{p,0}, X) = \bot$ for all $X \neq \langle p\rangle$.

$\boldsymbol{\mathcal{A}_\top}$ The automaton immediately goes into an accepting state, unless it is at a closing node or the root node. Formally, $Q \stackrel{\text{def}}{=} \{q_{\top,0}\}$ and $c_\top$ is defined by $c_\top(q_{\top,0}) \stackrel{\text{def}}{=} 1$; $\delta_\top(q_0, X)$ is defined as $\top$ for $X = \langle p\rangle$ in $\mathsf{XML}(N)$ and as $\bot$ otherwise.

The automata $\mathcal{A}_\pi$ for π a path formula additionally carry a distinguished subset $C_\pi \subseteq Q_\pi$ of *continuation* states, such that there is a 'partial run' from some initial position with branches starting from their initial state, which are either infinite but verifying the parity condition, or are finite but end in a continuation state in a position related to the initial one through $[\![\pi]\!]$. Let us see this at work with the base cases of path formulæ:

$\boldsymbol{\mathcal{A}_\downarrow}$ The automaton moves right from the initial node while maintaining the depth relative to this initial node. It stops (goes into a dead state) if it reaches a node at the same or lesser depth than the initial node. All the visited nodes with a relative depth of 1 are direct children of the initial node, and therefore visited by continuation states. We set where $Q_\downarrow \stackrel{\text{def}}{=} \{q_0, q_1, \cdots, q_n\}$ with $q_{\downarrow,0} \stackrel{\text{def}}{=} q_0$; the coloring $c_\downarrow$ is identically 1 on

$Q_\downarrow$, $C_\downarrow = \{q_1\}$, and $\delta_\downarrow$ is defined by:

$$\begin{aligned}
&\delta_\downarrow(q_i, \langle p\rangle) \stackrel{\text{def}}{=} (1, q_{i+1}), \quad i < n, p \in N && \delta_\downarrow(q_n, \langle p\rangle) \stackrel{\text{def}}{=} \bot, && p \in N\\
&\delta_\downarrow(q_i, \langle /p\rangle) \stackrel{\text{def}}{=} (1, q_{i-1}), \quad i > 1, p \in N && \delta_\downarrow(q_0, \langle /p\rangle) \stackrel{\text{def}}{=} \delta_\downarrow(q_1, \langle /p\rangle) \stackrel{\text{def}}{=} \bot, && p \in N\\
&\delta_\downarrow(q_i, \langle /\mathsf{r}\rangle) \stackrel{\text{def}}{=} \bot, \quad i \in \{0, \ldots, n\}\ .
\end{aligned}$$

$\boldsymbol{\mathcal{A}_\rightarrow}$ Similarly to $\mathcal{A}_\downarrow$, the automaton moves right while maintaining the depth relative to the initial node. It fails if it reaches a node at a lesser depth that the initial node. Otherwise, it finds the next node at a same depth as the initial node. Formally, $Q_\rightarrow \stackrel{\text{def}}{=} \{q_0, q_1, \ldots, q_n, q_f\}$, $q_{\rightarrow,0} \stackrel{\text{def}}{=} q_0$, the coloring $c_\rightarrow$ is identically 1 on $Q_\downarrow$, there is a unique continuation state $C_\rightarrow \stackrel{\text{def}}{=} \{q_f\}$ when reaching the right sibling, and $\delta_\rightarrow$ is defined by

$$\begin{aligned}
&\delta_\rightarrow(q_i, \langle p\rangle) \stackrel{\text{def}}{=} (1, q_{i+1}), \quad i < n, p \in N && \delta_\rightarrow(q_n, \langle p\rangle) \stackrel{\text{def}}{=} \bot, && p \in N\\
&\delta_\rightarrow(q_i, \langle /p\rangle) \stackrel{\text{def}}{=} (1, q_{i-1}), \quad i > 1, p \in N && \delta_\rightarrow(q_1, \langle /p\rangle) \stackrel{\text{def}}{=} (1, q_f), && p \in N\\
&\delta_\rightarrow(q_0, \langle /p\rangle) \stackrel{\text{def}}{=} \bot, \quad p \in N && \delta_\rightarrow(q_f, X) \stackrel{\text{def}}{=} \bot, && X \in \Sigma\\
&\delta_\rightarrow(q_i, \langle /\mathsf{r}\rangle) \stackrel{\text{def}}{=} \bot, \quad i \in \{0, \ldots, n\}\ .
\end{aligned}$$

$\boldsymbol{\mathcal{A}_\uparrow}$ **,** $\boldsymbol{\mathcal{A}_\leftarrow}$ We define these automata similarly to $\mathcal{A}_\downarrow$ and $\mathcal{A}_\rightarrow$. Observe however that, because we always finish on opening brackets, it is not enough to exchange -1 and 1 in the directions of transitions.

Next, we consider the induction step for path formulæ:

$\boldsymbol{\mathcal{A}_{\pi_1;\pi_2}}$ We combine the automata $\mathcal{A}_{\pi_1}$ and $\mathcal{A}_{\pi_2}$. We add transitions from the continuation states of $\mathcal{A}_{\pi_1}$ at opening nodes to the initial state of $\mathcal{A}_{\pi_2}$. Formally, $Q_{\pi_1;\pi_2} \stackrel{\text{def}}{=} Q_{\pi_1} \uplus Q_{\pi_2}$, $q_{\pi_1;\pi_2} \stackrel{\text{def}}{=} q_{\pi_1,0}$, $c_{\pi_1;\pi_2}$ preserves the priorities of c_{π_1} and c_{π_2}, $C_{\pi_1;\pi_2} \stackrel{\text{def}}{=} C_{\pi_2}$ and $\delta_{\pi_1;\pi_2}$ is defined by

$$\begin{aligned}
\delta_{\pi_1;\pi_2}(q_1, X) &\stackrel{\text{def}}{=} \delta_{\pi_1}(q_1, X), && q_1 \in Q_{\pi_1} \setminus C_{\pi_1}\\
\delta_{\pi_1;\pi_2}(q_1, \langle p\rangle) &\stackrel{\text{def}}{=} \delta_{\pi_1}(q_1, \langle p\rangle) \vee (0, q_{\pi_2,0}), && q_1 \in C_{\pi_1}\\
\delta_{\pi_1;\pi_2}(q_1, \langle /p\rangle) &\stackrel{\text{def}}{=} \delta_{\pi_1}(q_1, \langle /p\rangle), && q_1 \in C_{\pi_1}\\
\delta_{\pi_1;\pi_2}(q_2, X) &\stackrel{\text{def}}{=} \delta_{\pi_2}(q_2, X), && q_2 \in Q_{\pi_2}
\end{aligned}$$

for X in Σ and p in N.

$\boldsymbol{\mathcal{A}_{\pi_1+\pi_2}}$ This is a straightforward union: We define $Q_{\pi_1+\pi_2} \stackrel{\text{def}}{=} Q_{\pi_1} \uplus Q_{\pi_2} \uplus \{q_{\pi_1+\pi_2,0}\}$, $C_{\pi_1+\pi_2} \stackrel{\text{def}}{=} C_{\pi_1} \cup C_{\pi_2}$, $c_{\pi_1+\pi_2} \stackrel{\text{def}}{=} c_{\pi_1} \cup c_{\pi_2} \cup \{(q_{\pi_1+\pi_2,0}, 1)\}$, $\delta_{\pi_1+\pi_2} \stackrel{\text{def}}{=} \delta_{\pi_1} \cup \delta_{\pi_2} \cup \{(q_{\pi_1+\pi_2,0}, X, (0, q_{\pi_1,0}) \vee (0, q_{\pi_2,0})) \mid X \in \Sigma\}$.

$\boldsymbol{\mathcal{A}_{\pi^*}}$ This case is similar to that of $\mathcal{A}_{\pi_1;\pi_2}$; we add transitions from the critical states of $\mathcal{A}_\pi$ to its own initial state. Define $Q_{\pi^*} \stackrel{\text{def}}{=} Q_\pi \uplus \{q_{\pi^*,0}\}$, $C_{\pi^*} \stackrel{\text{def}}{=} \{q_{\pi^*,0}\}$, $c_{\pi^*} \stackrel{\text{def}}{=} c_\pi \cup \{(q_{\pi^*,0}, 1)\}$, and

$$\begin{aligned}
&\delta_{\pi^*}(q_{\pi^*,0}, X) \stackrel{\text{def}}{=} (0, q_{\pi,0}), && \delta_{\pi^*}(q, \langle /p\rangle) \stackrel{\text{def}}{=} \delta_\pi(q, X), && q \in C_\pi\\
&\delta_{\pi^*}(q, X) \stackrel{\text{def}}{=} \delta_\pi(q, X), \quad q \in Q_\pi \setminus C_\pi && \delta_{\pi^*}(q, \langle p\rangle) \stackrel{\text{def}}{=} \delta_\pi(q, X) \vee (0, q_{\pi^*,0}), && q \in C_\pi
\end{aligned}$$

for X in Σ and p in N. Because we assigned an odd priority to $q_{\pi^*,0}$, the automaton cannot loop indefinitely in $q_{\pi^*,0}$ and must eventually continue.

$\boldsymbol{\mathcal{A}_{\psi?}}$ Define $Q_{\psi?} \stackrel{\text{def}}{=} Q_\psi \uplus \{q_{\psi?,0}, q_f\}$, $C_{\psi?} \stackrel{\text{def}}{=} \{q_f\}$, $c_{\psi?}$ extends c_ψ with $c_{\psi?}(q_{\psi?,0}) = c_{\psi?}(q_f) = 1$, and

$$\delta_{\psi?}(q_{\psi?,0}, X) \stackrel{\text{def}}{=} (0, q_f) \wedge (0, q_{\psi,0}), \quad \delta_{\psi?}(q_f, X) \stackrel{\text{def}}{=} \bot, \quad \delta_{\psi?}(q, X) \stackrel{\text{def}}{=} \delta_\psi(q, X),$$

for X in Σ and q in Q_ψ.

Finally, we consider the induction step for node formulæ:

$\boldsymbol{\mathcal{A}_{\langle\pi\rangle\psi}}$ We construct the automaton by joining the critical states of $\mathcal{A}_\pi$ with the initial state of $\mathcal{A}_\psi$. Define $Q_{\langle\pi\rangle\psi} \stackrel{\text{def}}{=} Q_\pi \uplus Q_\psi$, $q_{\langle\pi\rangle\psi,0} \stackrel{\text{def}}{=} q_{\pi,0}$, $c_{\langle\pi\rangle\psi} \stackrel{\text{def}}{=} c_\pi \cup c_\psi$, and $\delta_{\langle\pi\rangle\psi}$ by

$$\begin{aligned} \delta_{\langle\pi\rangle\psi}(p, X) &\stackrel{\text{def}}{=} \delta_\pi(p, X), & p \in Q_\pi \setminus C_\pi \quad \delta_{\langle\pi\rangle\psi}(q, X) \stackrel{\text{def}}{=} \delta_\psi(q, X), \quad q \in Q_\psi \\ \delta_{\langle\pi\rangle\psi}(p, X) &\stackrel{\text{def}}{=} \delta_\pi(p, X) \vee (0, q_{\psi,0}), & p \in C_\pi \end{aligned}$$

for X in Σ.

$\boldsymbol{\mathcal{A}_{\psi_1 \wedge \psi_2}}$ We do a simple conjunction of the automata $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$: define $Q_{\psi_1\wedge\psi_2} \stackrel{\text{def}}{=} Q_{\psi_1} \uplus Q_{\psi_2} \uplus \{q_{\psi_1\wedge\psi_2,0}\}$, $c_{\psi_1\wedge\psi_2} \stackrel{\text{def}}{=} c_{\psi_1} \cup c_{\psi_2} \cup \{(q_{\psi_1\wedge\psi_2,0}, 1)\}$, $\delta_{\psi_1\wedge\psi_2} \stackrel{\text{def}}{=} \delta_{\psi_1} \cup \delta_{\psi_2} \cup \{(q_{\psi_1\wedge\psi_2,0}, X, (0, q_{\psi_1,0}) \wedge (0, q_{\psi_2,0})) \mid X \in \Sigma\}$.

$\boldsymbol{\mathcal{A}_{\neg\psi}}$ We essentially construct the dual $\mathsf{dual}(\mathcal{A}_\psi)$ of $\mathcal{A}_\psi$: the latter accepts the complement of the language accepted by $\mathcal{A}_\psi$. However, we need to ensure that only opening nodes $\langle p\rangle$ are accepted, thus intersect with the automaton $\mathcal{A}_\top$ that only accepts opening nodes. Formally, $\mathsf{dual}(\mathcal{A}_\psi) = \langle Q_\psi, \Sigma, \delta_\psi, q_{\psi,0}, c_{\neg\psi}\rangle$ where $\delta_{\neg\psi}(q, X) \stackrel{\text{def}}{=} \mathsf{dual}(\delta_\psi(q, X))$ and $c_{\neg\psi}(q) \stackrel{\text{def}}{=} c_\psi(q) + 1$ for all $q \in Q$ and $X \in \Sigma$. Here, dual is a function from $\mathbb{B}^+(Q \times \{-1, 0, 1\})$ to itself that applies the usual DeMorgan's law. It is easy to check that $\mathsf{dual}(\mathcal{A}_\psi)$ accepts the complement of $L(\mathcal{A}_\psi)$.

D ε-Free Case

We prove in this section Proposition 7, thus showing that the PFMC problem is PSPACE-hard in this case.

Proof. We reduce from the membership problem of linear bounded automata (LBA). Suppose we are given an LBA $M = \langle Q, \Gamma, \Sigma, \delta, q_1, F\rangle$ with state set Q, tape alphabet Γ, input alphabet $\Sigma \subseteq \Gamma$, transition relation $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{-1, 0, 1\}$, initial state $q_1 \in Q$, and set of final states $F \subseteq Q$. Let $Q = \{q_1, \ldots, q_\ell\}$; we assume that $\Gamma = \{a_1, a_2, \ldots, a_m\}$ contains two endmarkers $a_1 = \triangleleft$ and $a_2 = \triangleright$ that surround the input and are never erased nor crossed during the run of the machine.

We are also given a string $x = b_1 b_2 \cdots b_n$ with each $b_i \in \Sigma$; $b_1 = \triangleleft$ and $b_n = \triangleright$. We have to decide whether x is accepted by M. We are going to construct a word w, a CFG G, and a $\text{PDL}_{\mathsf{core}}[\downarrow]$ formula φ, s.t. the PFMC problem has a solution for $\langle w, G, \varphi\rangle$ iff M accepts x.

Encoding as Linear Trees A configuration of M is a sequence of length n of form $\triangleleft\gamma q\gamma'\triangleright$ where q is the current state in Q, $\triangleleft\gamma\gamma'\triangleright$ is the current tape contents, and $|\triangleleft\gamma| = h$ indicates that the head is currently on the last symbol of $\triangleleft\gamma$, i.e. the hth symbol of the tape.

We encode such a configuration by a contiguous sequence α of nodes as follows:

- The first node is S and it is followed by a sequence of n nodes, among which one is labeled H and the others $\bar{H}$; the position of H in this sequence denotes the position of the head in the configuration of M.
- This sequence is followed by a sequence of ℓ nodes, one labeled C and the others $\bar{C}$, which together describe the current state as q_k if the occurrence of C is the kth symbol in the sequence.
- Then we encode the tape contents as n successive sequences each of length m of nodes, with each time one labeled A and the others $\bar{A}$. The ith such sequence encodes the contents of the ith cell of the tape of M with A occurring at the jth position indicating that this cell contains a_j.

Thus α is of length $1 + n + \ell + nm$.

String and Grammar We *fix* $w \stackrel{\text{def}}{=} a$ and $G \stackrel{\text{def}}{=} \langle\{S, H, \bar{H}, A, \bar{A}, C, \bar{C}\}, \{a\}, P, S\rangle$ with productions

$$\begin{array}{lll} S \to H \mid \bar{H} & & \\ H \to H \mid \bar{H} \mid C \mid \bar{C} & C \to C \mid \bar{C} \mid A \mid \bar{A} & A \to A \mid \bar{A} \mid S \mid a \\ \bar{H} \to H \mid \bar{H} \mid C \mid \bar{C} & \bar{C} \to C \mid \bar{C} \mid A \mid \bar{A} & \bar{A} \to A \mid \bar{A} \mid S \mid a\,. \end{array}$$

Therefore, the trees in the parse forest $L_{G,w}$ are essentially sequences over $N^* \cdot \{a\}$. Clearly, all the encodings of finite runs of any LBA M will be in this set; it will be the formula's task to look for an accepting run of our particular M on x among all these trees.

The Formula $\varphi_{M,x}$ Let us turn to the definition of our $\text{PDL}_{\mathsf{core}}[\downarrow]$ formula. We start by defining low-level formulæ useful for testing the properties of the current configuration: assume we are on an S-labeled node:

$$\mathsf{h}(h) \stackrel{\text{def}}{=} (\langle\downarrow^h\rangle H) \wedge \bigwedge_{i\in\{1,\dots,n\}\setminus\{h\}} \langle\downarrow^j\rangle\bar{H}$$

tests whether the head is at position h. In the same way,

$$\mathsf{q}(k) \stackrel{\text{def}}{=} \langle\downarrow^n\rangle\big((\langle\downarrow^k\rangle C) \wedge \bigwedge_{k'\in\{1,\dots,\ell\}\setminus\{k\}} \langle\downarrow^{k'}\rangle\bar{C}\big)$$

then tests whether the current state is q_k, and

$$\mathsf{p}(i,j) \stackrel{\text{def}}{=} \langle\downarrow^{n+\ell+im}\rangle\big((\langle\downarrow^j\rangle A) \wedge \bigwedge_{j'\in\{1,\dots,m\}\setminus\{j\}} \langle\downarrow^{j'}\rangle\bar{A}\big)$$

tests whether the i position on the tape is symbol a_j. Finally, we can go to the next configuration by the path

$$\mathsf{next} \stackrel{\text{def}}{=} \downarrow^{n+\ell+nm} \ .$$

We can now check that a parse tree of $L_{G,w}$ is really the encoding of an accepting run of M on x. First, at each S node, we should find a full configuration:

$$\varphi_{conf} \stackrel{\text{def}}{=} [\downarrow^*]S \Rightarrow \bigvee_{h=1}^{n} \mathsf{h}(h) \wedge \bigvee_{k=1}^{\ell} \mathsf{q}(k) \wedge \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} \mathsf{p}(i,j) \wedge \langle \mathsf{next} \rangle a \vee S \ .$$

The initial configuration should have its head on the initial position 1, be in the initial state q_1, and have $x = b_1 \cdots b_n$ as tape contents:

$$\varphi_{init} \stackrel{\text{def}}{=} S \wedge \mathsf{h}(1) \wedge \mathsf{q}(1) \wedge \bigwedge_{i=1}^{n} \mathsf{p}(i, b_i) \ .$$

The leaf of the tree should be reached in a final configuration:

$$\varphi_{final} \stackrel{\text{def}}{=} [\downarrow^*](S \wedge \langle \mathsf{next} \rangle a) \Rightarrow \bigvee_{q_k \in F} \mathsf{q}(k) \ .$$

Successive configurations should respect the transition relation:

$$\varphi_{trans} \stackrel{\text{def}}{=} [\downarrow^*](S \wedge \neg \langle \mathsf{next} \rangle a) \Rightarrow \bigvee_{h=1}^{n} \bigvee_{k=1}^{\ell} \bigvee_{c=1}^{m} \bigvee_{(q_k, a_c, q_{k'}, a_{c'}, d) \in \delta} \big(\mathsf{h}(h) \wedge \mathsf{q}(k) \wedge \mathsf{p}(h,c)$$
$$\wedge \Big(\bigwedge_{h \neq i=1}^{n} \bigvee_{j=1}^{m} \mathsf{p}(i,j) \wedge \langle \mathsf{next} \rangle \mathsf{p}(i,j) \Big) \wedge \langle \mathsf{next} \rangle \big(\mathsf{h}(h+d) \wedge \mathsf{q}(k') \wedge \mathsf{p}(h,c') \big) \big) \ .$$

We finally define our $\mathrm{PDL}_{\mathsf{core}}[\downarrow]$ formula as the conjunction of the previous formulæ:

$$\varphi_{M,x} \stackrel{\text{def}}{=} \varphi_{conf} \wedge \varphi_{init} \wedge \varphi_{final} \wedge \varphi_{trans} \ .$$

To conclude, we observe that a tree in $L_{G,w}$ is a model of $\varphi_{M,x}$ iff there is an accepting run of M on x. As G and w are fixed and $\varphi_{M,x}$ can be computed in space logarithmic in the size of $\langle M, x \rangle$, this proves the PSPACE-hardness of the PFMC problem in the ε-free case. $\square$