Complexity of Subtype Satisfiability over Posets

Joachim Niehren
INRIA Futurs
Lille, France

Tim Priesnitz
Programming Systems Lab
Saarland University
Saarbrücken, Germany

Zhendong Su
Department of Computer Science
University of California
Davis, CA 95616 USA

Abstract

Subtype satisfiability is an important problem for designing advanced subtype systems and subtype-based program analysis algorithms. The problem is well understood if the atomic types form a lattice. However, little is known about subtype satisfiability over posets. In this paper, we investigate algorithms for and the complexity of subtype satisfiability over general posets. We present a uniform treatment of different flavors of subtyping: simple versus recursive types and structural versus non-structural subtype orders. Our results are established through a new connection of subtype constraints and modal logic. As a consequence, we settle a problem left open by Tiuryn and Wand in 1993.

1 Introduction

Many programming languages have some form of subtyping. The most common use is in the sub-classing mechanisms in object-oriented languages. Also common is the notion of "coercion" [19], for example automatic conversion from integers to floating point numbers.

Type checking and type inference for subtyping systems have been extensively studied since the original results of Mitchell [20]. The main motivations for investigating these systems today are more advanced designs for typed languages and program analysis algorithms based on subtyping.

Subtyping systems invariably involve *subtype constraints*, inequalities of the form $\tau_1 \leq \tau_2$, to capture that the type τ_1 is a *subtype* of τ_2. For example, the constraint $int \leq real$ means that at any place a floating point number is expected, an integer can be used instead.

Types are typically interpreted over trees over some base elements (drawn from a finite lattice or a partial order). The trees can be infinite if recursive types are allowed. There are two choices for the subtype relation. In a system with *structural subtyping* only types with the same shape are related. In a system with *non-structural subtyping*, there is a "least" type $\bot$ and a "largest" type $\top$ that can be related to types of arbitrary shape.

Three logical problems for subtype constraints are investigated in the literature: satisfiability [1, 6, 10, 15, 16, 20, 24, 27, 33, 34], entailment [8, 13, 14, 21, 22, 25, 26, 29, 35], and first-order validity [18, 32]. In this paper, we close a number of problems on satisfiability.

If the base constants form a lattice then subtype satisfiability is well understood [16, 20, 24]. For general partially-ordered sets (posets), however, there exist only partial answers. Tiuryn and Wand show that recursive structural satisfiability is in *DEXPTIME* [34]. Tiuryn shows that finite structural satisfiability is *PSPACE*-hard [33], and subsequently Frey shows that it is in *PSPACE* and thus *PSPACE*-complete [9]. Decidability and complexity of non-structural subtype satisfiability are open, for both finite and recursive types.

1.1 Main Contributions

In this paper, we close the open questions on subtype satisfiability over posets. We consider all combinations of finite versus recursive types, and structural versus non-structural orders.

We base our results on a new approach, connecting subtype constraints and modal logic. We introduce *uniform subtype constraints* and show that their satisfiability problem is polynomial time equivalent to that of a dialect of *propositional dynamic logic* [2, 5, 7], which is subsumed by the monadic second-order logic SnS of the complete infinite n-ary tree [28]. With this connection, we completely characterize the exact complexity of subtype satisfiability over posets in all cases.

Table 1 summarizes complexity results regarding subtype satisfiability over posets. We show in this paper, that recursive structural satisfiability is *DEXPTIME*-hard, finite non-structural satisfiability is *PSPACE*-complete, and recursive non-structural satisfiability is *DEXPTIME*-complete. In particular, this

	structural	non-structural
finite types	*PSPACE* (Frey, 1997 [9]) / *PSPACE*-hard (Tiuryn, 1992 [33])	*PSPACE*-complete (this paper)
infinite types	*DEXPTIME* (Tiuryn and Wand, 1993 [34]) / *DEXPTIME*-hard (this paper)	*DEXPTIME*-complete (this paper)

Table 1: Summary of complexity results on subtype satisfiability over posets.

settles a longstanding problem left open by Tiuryn and Wand in 1993 [34].

1.2 Plan of the Paper

We first recall *structural* and *non-structural* subtype orders, constraints, and satisfiability, and introduce so called *uniform* counterparts (Section 2). We present our dialect PDL_n of propositional dynamic logic (*PDL*) for infinite n-ary trees (Section 3) and determine the complexity of uniform subtype satisfiability, by establishing forth and back translations to PDL_n (Section 4). Next, we relate structural and non-structural subtype satisfiability to uniform subtype satisfiability, all interpreted over infinite trees (Section 5). We then consider subtype satisfiability interpreted over finite trees (Section 6) and conclude (Section 7). We include a proof of *DEXPTIME*-hardness for our variant of PDL_n in the appendix (Appendix A).

2 Subtyping

In this section, we review basic concepts on subtyping. We discuss the various choices of type expression languages, their signatures, subtype orders, and the notion of subtype constraints. We also introduce the main subject of this paper: subtype satisfiability over posets.

2.1 Types as Trees

Types can be viewed as trees over some ranked alphabet Σ, the *signature* of the given type language. A signature consists of a finite set of function symbols (a.k.a. *type constructors*). Each function symbol f has an associated $arity(f) \geq 0$, indicating the number of arguments that f expects, and for all $1 \leq i \leq arity(f)$ a polarity $pol(f,i) \in \{1,-1\}$. We call a position i of symbol f *covariant* if $pol(f,i) = 1$ and *contravariant* otherwise. Symbols with arity zero are *type constants*.

We identify *nodes* π of trees with relative addresses from the root of the tree, *i.e.*, with words in $(\mathbb{N}-\{0\})^*$. A word πi addresses the i-th child of node π, and $\pi\pi'$ the π' descendant of π. The root is represented by the empty word ε. We define a *tree* τ over Σ as a partial function:

$$\tau : (\mathbb{N} - \{0\})^* \to \Sigma$$

Tree domains $dom(\tau)$ are prefixed closed, non-empty, and arity consistent, *i.e.*: $\forall \pi \in dom(\tau) \forall i \in \mathbb{N} - \{0\} : \pi i \in dom(\tau) \leftrightarrow i \leq arity(\tau(\pi))$. A tree τ is *finite* if $dom(\tau)$ is a finite set, and *infinite* otherwise. We write $tree_\Sigma$ for the set of possibly infinite trees over Σ.

Given a function symbol f with $n = arity(f)$ and trees $\tau_1, \ldots, \tau_n \in tree_\Sigma$ we define $f(\tau_1, \ldots, \tau_n)$ as the unique tree τ with $f(\tau_1, \ldots, \tau_n)(\varepsilon) = f$ and $f(\tau_1, \ldots, \tau_n)(i\pi) = \tau_i(\pi)$. We define the *polarities* of nodes in trees as follows:

$$\begin{array}{lll} pol_\tau(\varepsilon) & =_{\mathrm{df}} & 1 \\ pol_{f(\tau_1,\ldots,\tau_n)}(i\pi) & =_{\mathrm{df}} & pol(f,i) * pol_{\tau_i}(\pi) \end{array}$$

For partial orders $\leq$, we let $\leq^1$ denote the order $\leq$ itself and $\leq^{-1}$ the reversed relation, $\geq$.

Subtype orders $\leq$ are partial orders on trees over some signature Σ. Two subtype orders arise naturally, *structural subtyping* and *non-structural* subtyping.

2.2 Structural Subtyping

We investigate structural subtyping over standard signatures with posets. These are parametrized by posets $(B, \leq_B)$ of constants and have the form:

$$\Sigma = B \cup \{\times, \to\}$$

The product type constructor $\times$ is a binary function symbol that is covariant in both positions ($pol(\times,1) = pol(\times,2) = 1$), while the function type constructor $\to$ is contravariant in its first and covariant in its second argument ($pol(\to,1) = -1$ and $pol(\to,2) = 1$).

Structural subtype orders $\leq$ are partial orders on trees over structural signatures Σ. They are obtained by lifting the ordering on constants $(B, \leq_B)$ in Σ to trees. More formally, $\leq$ is the smallest binary relation $\leq$ on $tree_\Sigma$ such that for all $b, b' \in B$ and types τ_1, τ_2, τ_1', τ_2' in $tree_\Sigma$:

- $b \leq b'$ iff $b \leq_B b'$;
- $\tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'$ iff $\tau_1 \leq \tau_1'$ and $\tau_2 \leq \tau_2'$;
- $\tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'$ iff $\tau_1' \leq \tau_1$ and $\tau_2 \leq \tau_2'$.

Notice that $\times$ is monotonic in both of its arguments while $\rightarrow$ is anti-monotonic in its first argument and monotonic in its second. For more general signatures, monotonic arguments are specified by covariant positions of function symbols, and anti-monotonic arguments by contravariant positions.

For structural subtyping, two types are related only if they have exactly the same shape, *i.e.*, tree domain. Notice that structural subtype orders are indeed partial orders. We do not restrict ourselves to lattices $(B, \leq_B)$ in contrast to most previous work.

2.3 Non-Structural Subtyping

In the *non-structural subtype order*, two distinguished constants are added to structural type languages, a *smallest type* $\bot$ and a *largest type* $\top$. The ordering is parametrized by a poset $(B, \leq_B)$ and has the signature:

$$\Sigma = B \cup \{\times, \rightarrow\} \cup \{\bot, \top\}$$

For the non-structural subtype order, besides the three structural rules earlier, there is an additional rule:

- $\bot \leq \tau \leq \top \quad$ for any $\tau \in \mathit{tree}_\Sigma$

2.4 Uniform Subtyping

We introduce *uniform subtyping* as an intermediate ordering for two reasons: (i) to capture both structural and non-structural subtyping effects and (ii) to use it as a bridge from uniform subtype constraints to modal logic.

We call a signature Σ *uniform* if all symbols in Σ have the same non-zero arity and the same polarities. All trees over Σ are complete infinite n-ary trees, where n is the arity common to all function symbols in Σ. Hence, all trees have the same shape. Furthermore, the polarities of nodes $\pi \in \{1, \ldots, n\}^*$ in trees τ over uniform signatures do not depend on τ. We therefore write $\mathit{pol}(\pi)$ instead of $\mathit{pol}_\tau(\pi)$.

The signatures $\{\times\}$ and $\{\rightarrow\}$, for instance, are both uniform, while $\{\times, \rightarrow\}$ or $\{\bot, \top, \times\}$ are not. The idea to model the non-structural signature $\{\bot, \top, \times\}$ uniformly is to raise the arities of $\bot$ and $\top$ to 2 and to order them by $\bot \leq_\Sigma \times \leq_\Sigma \top$.

A *uniform subtype order* $\leq$ is defined over a partially-ordered uniform signature $(\Sigma, \leq_\Sigma)$. It satisfies for all trees $\tau_1, \tau_2 \in \mathit{tree}_\Sigma$:

$$\tau_1 \leq \tau_2 \quad \text{iff} \quad \forall \pi \in \{1, \ldots, n\}^* : \ \tau_1(\pi) \leq_\Sigma^{\mathit{pol}(\pi)} \tau_2(\pi)$$

where n is the arity of the function symbols in Σ. For simplicity, we will often write $\leq_\Sigma^\pi$ instead of $\leq_\Sigma^{\mathit{pol}(\pi)}$.

2.5 Subtype Constraints and Satisfiability

In a subtype system, *type variables* are used to denote unknown types. We assume that there are a denumerable set of type variables V. We assume w.l.o.g. that subtype constraints are *flat*, and subtype constraints φ over some signature Σ are given by the following grammar:

$$\varphi \ ::= \ x{=}f(x_1, \ldots, x_n) \mid x{\leq}y \mid \varphi \wedge \varphi$$

where n is the arity of $f \in \Sigma$. We call atomic constraints $x{=}f(x_1, \ldots, x_n)$ and $x{\leq}y$ the *literals*. The type variables in a constraint φ are called the *free variables* of φ, denoted by $V(\varphi)$.

We always consider two possible interpretations of subtype constraints, over possibly infinite tree over Σ, and over finite trees over Σ respectively. A variable assignment α is a function mapping type variables V to trees of the respective domain. A constraint φ is *satisfiable* over Σ if there is a variable assignment α such that $\alpha(\varphi)$ holds in Σ.

We distinguish three subtype satisfiability problems, each of which has two variants depending on interpretation over finite or possibly infinite trees.

Structural subtype satisfiability is the problem to decide whether a structural subtype constraint is satisfiable. The arguments of this problem are a posets $(B, \leq_B)$ and a constraint φ over the signature $B \cup \{\times, \rightarrow\}$.

Non-structural subtype satisfiability is the problem to decide whether a non-structural subtype constraint is satisfiable. The arguments are a poset $(B, \leq_B)$ and a constraint φ over signature $B \cup \{\times, \rightarrow\} \cup \{\bot, \top\}$.

Uniform subtype satisfiability is the the problem to decide whether a uniform subtype constraint is satisfiable. The arguments are a partially-ordered uniform signature $(\Sigma, \leq_\Sigma)$ and a subtype constraint φ over this signature.

3 Propositional Dynamic Logic over Trees

Propositional dynamic logic (*PDL*) is a modal logic that extends Boolean logic to directed graphs of possible worlds. The same proposition may hold in some node of the graph and be wrong in others. Nodes are connected by labeled edges, that can be talked about modal operators.

In this paper, we consider the modal logic PDL_n, the *PDL* language for the complete infinite n-ary tree. PDL_n is naturally subsumed by the monadic second-order logic SnS of the complete n-ary tree [28].

R	$::=$	$i \mid R \cup R' \mid RR' \mid R^*$	where $1 \leq i \leq n$
A	$::=$	$p \mid \neg A \mid A \wedge A' \mid [R]A$	

Figure 1: Syntax of PDL_n.

3.1 Other PDL Dialects

Propositional dynamic logic (*PDL*) over directed edge-labeled graphs goes back to Fischer and Ladner [7], who restricted Pratt's dynamic logic to the propositional fragment. It is well known that *PDL* has the *tree property*: every satisfiable *PDL* formula can be satisfied in a rooted edge-labeled tree. Deterministic *PDL* [2, 11, 36] restricts the model class to graphs whose edge labels are functional in that they determine successor nodes. Deterministic *PDL* with edge labels $\{1, \ldots, n\}$ is the closest relative to our language PDL_n, due to the tree property.

Besides of PDL_n, a large variety of PDL dialects with tree models were proposed in the literature. These differ in the classes of tree models, the permitted modal operators, and the logical connectives. Three different dialects of PDL over finite, binary, or n-ary trees were proposed in [5, 17, 23], see [4] for a comparison. PDL over finite unranked ordered trees were proposed for computational linguistics applications [5] and found recent interest for querying XML documents.

3.2 PDL_n and its Fragments

For every $n \geq 1$ we define a logic PDL_n as the PDL logic, for describing the complete infinite n-ary tree.

The syntax of PDL_n expressions[1] A is given in Figure 1. Starting from some infinite set Pr of propositional variables p, it extends the Boolean logic over these variables by universal modalities $[R]A$, where R is a regular expression over the alphabet $\{1, \ldots, n\}$.

We frequently use the modality $[*]$ as an abbreviation of $[\{1, \ldots, n\}^*]$, and sometimes $[+]$ as a shorthand for $[\{1, \ldots, n\}^+]$. We freely use definable logical connective for implication $\rightarrow$, equivalence $\leftrightarrow$, disjunction $\vee$, exclusive disjunction $\stackrel{+}{\vee}$, and the Boolean constants *true* and *false*. Furthermore, we can define existential modalities $\langle R \rangle A$ by $\neg[R]\neg A$.

We interpret formulas of PDL_n over the complete infinite n-ary trees. Tree nodes are labeled by the set of propositions that are valid there. Formally, a model M of a formula in PDL_n assigns Boolean values $0, 1$ to propositional variables in every node in $\{1, \ldots, n\}^*$:

$$M : Pr \times \{1, \ldots, n\}^* \rightarrow \{0, 1\}$$

[1] We could allow for test $?A$ in regular expressions, which frequently occur in PDL dialects but we will not need them.

$M, \pi \models p$	if $M(p, \pi) = 1$
$M, \pi \models A_1 \wedge A_2$	if $M, \pi \models A_1$ and $M, \pi \models A_2$
$M, \pi \models \neg A$	if not $M, \pi \models A$
$M, \pi \models [R]A$	if for all $\pi' \in L(R)$: $M, \pi\pi' \models A$

Table 2: Semantics of PDL_n.

Table 2 defines when a formula A holds in some node π of some model M, in formulas: $M, \pi \models A$. A formula $[R]A$ is valid for some node π of a tree M if A holds in all R descendants of π in M, *i.e.*, in all nodes $\pi\pi'$ where π' belongs to the language $L(R)$ of R.

Let us recall some logical notations. A formula A is *valid in a model* M if it holds in the root of M:

$$M \models A \text{ iff } M, \varepsilon \models A$$

A formula A is *satisfiable* if it is valid in some model; it is *valid* it is valid in all models:

$$\models A \text{ iff } \forall M.\, M \models A$$

Two formulas A, A' are equivalent if $A \leftrightarrow A'$ is valid:

$$A \models\!\mid A' \text{ iff } \models A \leftrightarrow A'$$

For instance, $\langle i \rangle A \models\!\mid [i]A$ holds for all $1 \leq i \leq n$ and all A, since nodes of the n-ary tree have unique i successors.

Note that PDL_n respects the substitution property: whenever $A_1 \models\!\mid A_2$ then $A[A_1/A_2] \models\!\mid A$. To see this note that if $A_1 \models\!\mid A_2$ then the equivalence $A \leftrightarrow A'$ is valid not only at the root of all models but also at all other nodes of all models. This is because all subtrees of complete n-ary trees are again complete n-ary trees.

Theorem 1 *Satisfiability of PDL_n formulas is in DEXPTIME.*

A PDL_n formula is satisfiable iff it can be satisfied by a deterministic rooted graph with edge labels in $\{1, \ldots, n\}$. The proposition thus follows from the *DEXPTIME* upper bound for deterministic *PDL* [2, 11], which is a corollary to the analogous result for *PDL*.

3.3 Flat Core PDL_n

We next investigate lower complexity bounds for PDL_n. It is known from Vardi and Wolper [36] that satisfiability of deterministic *PDL* is *DEXPTIME*-complete. This result clearly carries over to PDL_n.

An analysis of Spaan's proofs [31] reveals that nested $[*]$ modalities are not needed for *DEXPTIME*-hardness. But we can even do better, *i.e.*, restrict the language further.

$$
\begin{array}{lcll}
B & ::= & p_1 \wedge p_2 \mid \neg p \mid [i]p & \text{where } 1 \le i \le n \\
C & ::= & p \mid [*]\,(p \leftrightarrow B) \mid C_1 \wedge C_2 &
\end{array}
$$

Figure 2: Syntax of flat corePDL_n.

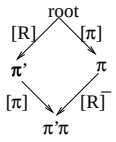
We define the fragment *flat core* PDL_n in Figure 2. A formula of flat core PDL_n is a conjunction of propositional variables and expressions of the form $[*]\,(p \leftrightarrow B)$. Note that $[*]$ modalities cannot be nested. Furthermore, all Boolean sub-formulas B are flat in that Boolean connectives only apply to variables.

Theorem 2 *Satisfiability of flat core PDL_n formulas is DEXPTIME-complete.*

We prove this theorem in Appendix A based on a new idea: we show how to express the emptiness of intersections of tree automata in PDL_n, a problem that was shown *DEXPTIME*-hard by Seidl [30].

3.4 Inversion

We now consider a variant of PDL_n with inverted modalities $[R]^-$, which address all nodes $\pi'\pi$ reached by prefixing some $\pi' \in L(R)$ to the actual node π.



$M, \pi \models [R^-]A$ if for all $\pi' \in L(R)$: $M, \pi'\pi \models A$

Inverted flat core PDL_n is defined in analogy to flat core PDL_n except that all modalities are inverted.

$$
\begin{array}{lcll}
B & ::= & p_1 \wedge p_2 \mid \neg p \mid [i]^- p & \text{for } 1 \le i \le n \\
C & ::= & p \mid [*]\,(p \leftrightarrow B) \mid C_1 \wedge C_2 &
\end{array}
$$

We will freely omit inversion for $[*]$ operators, as these are never nested below modalities. We can translate flat core PDL_n formulas C into formulas C^- of the inverted flat core, and vice versa, by replacing the operators $[i]$ through $[i]^-$. Models can be inverted too: $M^-(p, \pi) = M(p, \pi^{-1})$ where π^{-1} is the inversion of π.

Lemma 1 *$M \models C$ iff $M^- \models C^-$.*

4 Uniform Subtype Satisfiability

In this section, we investigate the complexity of uniform subtype satisfiability. We first show how to encode uniform subtype constraints into inverted PDL_n. We then give a translation from *inverted flat core PDL_n* back to uniform subtype satisfiability. Both translations are polynomial time and preserve satisfiability (Proposition 2 and 3). The complexity of PDL_n (Theorem 2) thus carries over to uniform subtype satisfiability.

Theorem 3 *Uniform subtype satisfiability over possibly infinite trees is DEXPTIME-complete.*

4.1 Uniform Subtype Constraints into PDL_n

We encode uniform subtype constraints over infinite trees into inverted PDL_n. The translation relies on ideas of Tiuryn and Wand [34], but it is simpler with modal logics as the target language, rather than infinite sets of regular path constraints. We first present our translation for covariant uniform signatures and then sketch the contravariant case.

Let Σ be a uniform covariant signature and $n > 1$ the arity of its function symbols. We fix a finite set of type variables V and consider subtype constraints φ over Σ with $V(\varphi) \subseteq V$. For all $x \in V$ and $f \in \Sigma$ we introduce propositional variables $P_{x=f}$ that are true at all nodes $\pi \in \{1, \ldots, n\}^*$ where the label of x is f.

The *well-formedness formula* wff_V states that all nodes of tree values of all $x \in V$ carry a unique label f:

$$\mathit{wff}_V \;=_{\mathrm{df}}\; \textstyle\bigwedge_{x \in V} [*]\,(\dot{\bigvee}_{f \in \Sigma} P_{x=f})$$

A polynomial time encoding of subtype constraints is presented in Table 3. Inverted modalities $[i]^-$ are needed to translate $x{=}f(x_1, \ldots, x_n)$ since $\alpha \models x{=}f(x_1, \ldots, x_n)$ if and only if $\alpha(x)(\varepsilon) = f$ and $\alpha(x)(i\pi) = \alpha(x_i)(\pi)$ for all words $i\pi \in \{1, \ldots, n\}^*$.

Proposition 1 *A uniform subtype constraint φ over a covariant signature Σ with $V(\varphi) \subseteq V$ is satisfiable if and only if $\mathit{wff}_V \wedge [\![\varphi]\!]$ is satisfiable.*

Proof. A solution of φ is a function $\alpha : V \to \mathit{tree}_\Sigma$. Let n be the arity of function symbols in Σ, so that all trees in tree_Σ are complete n-ary trees with nodes labeled in Σ, i.e., total functions of type $\{1, \ldots, n\}^* \to \Sigma$. A variable assignment α thus defines a PDL_n model $M_\alpha : Pr \times \{1, \ldots, n\}^* \to \Sigma$ that satisfies for all $x \in V$ and $\pi \in \{1, \ldots, n\}^*$:

$$M_\alpha(P_{x=f}, \pi) \;\leftrightarrow\; \alpha(x)(\pi) = f$$

We can now show by induction on the structure of φ that $\alpha \models \varphi$ iff $M_\alpha, \varepsilon \models \mathit{wff}_V \wedge [\![\varphi]\!]$. □

Proposition 2 *Uniform subtype satisfiability with covariant signatures over possibly infinite trees is in DEXPTIME.*

Proof. It remains to show that our reduction is in polynomial time. This might seem obvious, but it needs some care. Exclusive disjunctions of the form $p_1 \dot{\vee} \ldots \dot{\vee} p_n$ as used in the well-formedness formula can be encoded in quadratic time through $\bigvee_{i=1}^n (p_i \wedge \bigwedge_{1 \le j \ne i \le n} \neg p_j)$. Equivalences $p \leftrightarrow \neg p'$ as used can be encoded in linear time by $(p \wedge \neg p') \vee (\neg p \wedge p')$. □

$$
\begin{array}{rcl}
[\![x{=}f(x_1,\ldots,x_n)]\!] & =_{\mathrm{df}} & P_{x=f} \ \wedge \ \bigwedge_{g\in\Sigma}\bigwedge_{1\le i\le n}[*]\,(P_{x_i=g} \leftrightarrow [i]^{-}P_{x=g}) \\
[\![x{\le}y]\!] & =_{\mathrm{df}} & [*]\bigvee_{f\le_\Sigma g}(P_{x=f}\wedge P_{y=g}) \\
[\![\varphi_1\wedge\varphi_2]\!] & =_{\mathrm{df}} & [\![\varphi_1]\!]\wedge[\![\varphi_2]\!]
\end{array}
$$

Table 3: Expressing uniform covariant subtype constraints in inverted PDL_n.

Contravariance. Our approach smoothly extends to uniform subtyping with contravariant signatures. The key idea is that we can express polarities in *inverted* flat core PDL_n by using a new propositional variable p_{pol}. For example, consider the uniform signature $\Sigma = \{\rightarrow\}$, where $\rightarrow$ is the usual function type constructor. The variable p_{pol} is true in nodes with polarity 1 and false otherwise:

$$p_{pol} \wedge [*]\,(p_{pol} \leftrightarrow [1]^{-}\neg p_{pol}) \ \wedge \ [*]\,(p_{pol} \leftrightarrow [2]^{-}p_{pol}).$$

Limitation due to Inversion. Inversion is crucial to our translation and has a number of consequences. Most importantly, we cannot express the formula $[*](p \rightarrow [+]p')$ in inverted PDL_n, which states that whenever p holds at some node then it holds for all its proper descendants.

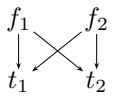
As a consequence, we cannot directly translate subtype constraints over standard signatures into PDL_n (which we consider in Sections 5). The difficulty is to encode tree domains in the presence of leafs. Suppose we want to define that p holds for all nodes outside the tree domain. We could do so by imposing $[*](p_c \rightarrow [+]p)$ for all constants c, but this is impossible in inverted PDL_n.

This is not a problem for uniform signatures, because every tree there is an infinite, complete n-ary tree. This shows that we do not need to express tree domains when considering satisfiability. Unfortunately, the same technique does not extend to entailment and other first-order fragments that require negations.

4.2 Back Translation

To prove *DEXPTIME*-hardness of uniform subtype satisfiability, we show how to express inverted flat core PDL_n by uniform subtype constraints, indeed only with covariant signatures. Our encoding of Boolean logic is inspired by Tiuryn [33], while the idea to lift this encoding to PDL_n is new.

Let C be a formula of inverted flat core PDL_n and $m \ge 0$ be the maximal i for which $[i]$ occurs in C. We construct uniform subtype constraints with function symbols ordered in a crown: $\Sigma(m) = \{t_1, t_2, f_1, f_2\}$. All function symbols have arity m and satisfy $t_i \le_{\Sigma(m)} f_j$ for all $i, j \in \{1, 2\}$. We use propositional variables p, q, and r as type variables in the subtype constraint we are constructing too, *i.e.*, $V = Pr$. Let $bool_i = \{t_i, f_i\}$ for $i \in \{1, 2\}$ be two Boolean lattices.



In Table 4, we encode Boolean formulas so that they hold in all nodes of a tree. The subtype constraints *all-*$t_i(p)$ and *all-*$f_i(p)$ hold for the unique trees that are completely labeled by t_i and f_i respectively. The subtype constraint *all-bool*$_i(p)$ holds for trees that are labeled in $bool_i$. The constraints *lower*(p, q) and *upper*(p, q) require the existence of lower and upper bounds respectively for trees p and q. These bounds are used to define the diagonal pairs *diag*(p, q) in the crown.

Lemma 2 *diag*$(p, q) \models\!\models \forall\pi.\ (p(\pi) = t_1 \wedge q(\pi) = f_2) \vee (p(\pi) = f_1 \wedge q(\pi) = t_2)$.

Proof. Since p is a tree labeled in $bool_1$, all nodes π satisfy $\alpha(p)(\pi){=}t_1$ or $\alpha(p)(\pi){=}f_1$. In the first case (the second is analogous) the constraint *upper*(p, q) entails $\alpha(q)(\pi){\neq}t_2$. Since q is a $bool_2$ tree, $\alpha(q)(\pi){=}f_2$. □

The subtype constraint *all*$(p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4)$ expresses a universally valid Boolean clause. In its definition, we use the diagonal operator in functional syntax to increase readability. Solutions of such formulas are variable assignments $\alpha : Pr \rightarrow \{1, \ldots, n\}^* \rightarrow \Sigma(m)$. We freely identify $t = t_1$ and $f = f_1$, and thus the Booleans $bool$ with $bool_1$. For variable assignments α into trees over Booleans, we define PDL_n-models $M_\alpha : Pr \times \{1, \ldots, n\}^* \rightarrow bool$ by Currying:

$$M_\alpha(p, \pi) = \alpha(p)(\pi)$$

Lemma 3 *For all variable assignments* α, $\alpha \models$ *all*$(p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4)$ *if and only if* M_α *is defined and* $M_\alpha \models [*](p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4)$.

Proof. Let $A = p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$. We assume $\alpha \models$ *all*(A). The model M_α exists since $A \models \bigwedge_{1\le i\le 4}$ *all-bool*$_1(p_i)$. We show $M_\alpha \models [*]A$ by contradicting the existence of a path π with:

$$\alpha \models p_1(\pi){=}f_1 \wedge p_2(\pi){=}f_1 \wedge p_3(\pi){=}t_1 \wedge p_4(\pi){=}t_1.$$

Each of these four disjuncts forbids one of four possible values for $\alpha(q)(\pi)$, as we argue below, where q is the existentially quantified variable in *all*(A). This is clearly impossible.

$$
\begin{array}{rcl}
\mathit{all\text{-}t_i}(p) & =_{\mathrm{df}} & \exists q.\ q{=}t_i(q,\ldots,q) \\
\mathit{all\text{-}f_i}(p) & =_{\mathrm{df}} & \exists q.\ q{=}f_i(q,\ldots,q) \\
\mathit{all\text{-}bool_i}(p) & =_{\mathrm{df}} & \exists q_1 \exists q_2.\ (\mathit{all\text{-}f_i}(q_1) \wedge q_1{\leq}p{\leq}q_2 \wedge \mathit{all\text{-}t_i}(q_2)) \\
\mathit{upper}(p_1,p_2) & =_{\mathrm{df}} & \exists q.\ p_1{\leq}q \wedge p_2{\leq}q \\
\mathit{lower}(p_1,p_2) & =_{\mathrm{df}} & \exists q.\ q{\leq}p_1 \wedge q{\leq}p_2 \\
\mathit{diag}(p,q) & =_{\mathrm{df}} & \mathit{all\text{-}bool_1}(p) \wedge \mathit{all\text{-}bool_2}(q) \wedge \mathit{upper}(p,q) \wedge \mathit{lower}(p,q) \\
\mathit{all}(p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4) & =_{\mathrm{df}} & \exists q. \bigwedge_{1\leq i\leq 4}\ \mathit{all\text{-}bool_1}(p_i) \\
 & & \wedge \mathit{upper}(q,p_4) \wedge \mathit{upper}(q,\mathit{diag}(p_2)) \\
 & & \wedge \mathit{lower}(q,p_1) \wedge \mathit{lower}(q,\mathit{diag}(p_3)) \\
\mathit{all}(p_1 \vee p_2) & =_{\mathrm{df}} & \exists q.\ \mathit{all}(p_1 \vee p_2 \vee \neg q \vee \neg q) \wedge \mathit{all\text{-}f_1}(q)
\end{array}
$$

Table 4: Boolean operations expressed by subtype constraints.

1. if $\alpha(p_1)(\pi){=}f_1$ then $\alpha(q)(\pi) \neq f_2$ since $\alpha \models \mathit{lower}(q,p_1)$;
2. if $\alpha(p_2)(\pi){=}f_1$ then $\alpha \models \mathit{diag}(p_2)(\pi){=}t_2$ (Lemma 2). This implies $\alpha(q)(\pi) \neq t_1$ since $\alpha \models \mathit{upper}(q,\mathit{diag}(p_2))$.
3. if $\alpha(p_3)(\pi){=}t_1$ then $\alpha \models \mathit{diag}(p_3)(\pi){=}f_2$ (Lemma 2). This implies $\alpha(q)(\pi) \neq f_1$ since $\alpha \models \mathit{lower}(q,\mathit{diag}(p_3))$.
4. if $\alpha(p_4)(\pi){=}t_1$ then $\alpha(q)(\pi) \neq t_2$ since $\alpha \models \mathit{upper}(q,p_4)$. □

Encoding clauses with two positive and two negative literals is sufficient to encode all other clauses needed in the back translation $[\![C]\!]^{-1}$ shown in Table 5.

Proposition 3 *Let C be a flat core inverted PDL_n formula. For all variable assignments α to trees over $\Sigma(m)$, $\alpha \models [\![C]\!]^{-1}$ if and only if M_α is defined and $M_\alpha \models C$.*

For $n = 0$, subtype constraints become ordering constraints for some given ordering, while PDL_0 satisfiability becomes a Boolean satisfiability problem that is well-known to be NP-complete. We thus obtain a new NP-completeness proof for ordering constraints interpreted over a given ordering relation [27].

5 Equivalence of Subtype Problems

We next show the equivalence of uniform subtype satisfiability with structural and non-structural subtype satisfiabilities over possibly infinite trees. Subtype satisfiability over finite trees will be treated in Section 6.

Theorem 4 *Structural, non-structural, and uniform subtype satisfiability over possibly infinite trees are equivalent and DEXPTIME-complete.*

The proof relies on so called 1-subtype orders which are subtype orders over signatures with a single non-constant, and the corresponding constraints.

1-subtype satisfiability is the satisfiability problem of subtype constraints over 1-subtype orders. This problem is parametric in the arities and polarities of the non-constant, the partial order on constants $(B, B_{\leq})$, and whether or not $\{\bot, \top\}$ is included in the signature.

We present the proof in four steps. We first show how to reduce structural subtype satisfiability to 1-subtype satisfiability (Section 5.1) and then do the same for the non-structural case (Section 5.2). Next, we reduce 1-subtype satisfiability to uniform subtype satisfiability (Section 5.3). Finally, we translate uniform subtype satisfiability back to both structural and non-structural subtype satisfiability (Section 5.4).

5.1 Structural to 1-Subtype Satisfiability

In this part, we show how to reduce structural to 1-subtype satisfiability. We first use a standard technique to characterize the shapes of solutions to a structural subtype constraints. We consider type expressions as terms over Σ.

Definition 1 *Let φ be a constraint over Σ and $\star$ be an arbitrary, fixed constant. For any type expression t, $t^\star$ denotes the same type expression as t except all constants in t are replaced with $\star$. Define the* shape constraint $sh(\varphi)$ *as:*

$$
\begin{array}{rcl}
sh(t_1 = t_2) & =_{\mathrm{df}} & t_1^\star = t_2^\star \\
sh(x{\leq}y) & =_{\mathrm{df}} & x = y \\
sh(\varphi_1 \wedge \varphi_2) & =_{\mathrm{df}} & sh(\varphi_1) \wedge sh(\varphi_2)
\end{array}
$$

The constraint φ is called weakly unifiable *iff $sh(\varphi)$ is unifiable.*

$$
\begin{array}{rcl}
[\![p]\!]^{-1} & =_{\mathrm{df}} & \exists p_1 \ldots \exists p_n.\ \mathit{all\text{-}bool}_1(p) \wedge p{=}t_1(p_1, \ldots, p_m) \\
[\![[*]\,(p \leftrightarrow [i]^{-}q)]\!]^{-1} & =_{\mathrm{df}} & \mathit{all\text{-}bool}_1(p) \wedge \mathit{all\text{-}bool}_1(q) \\
 & & \wedge\ \exists q_1 \ldots \exists q_n.(t_1(q_1, \ldots q_m){\leq}q{\leq}f_1(q_1, \ldots q_m) \wedge p{=}q_i) \\
[\![[*]\,(p \leftrightarrow \neg q)]\!]^{-1} & =_{\mathrm{df}} & \mathit{all}(p \vee q) \wedge \mathit{all}(\neg p \vee \neg q) \\
[\![[*]\,(p \leftrightarrow (q_1 \wedge q_2))]\!]^{-1} & =_{\mathrm{df}} & \mathit{all}(\neg p \vee q_1) \wedge \mathit{all}(\neg p \vee q_2) \wedge \mathit{all}(p \vee \neg q_1 \vee \neg q_2) \\
[\![C_1 \wedge C_2]\!]^{-1} & =_{\mathrm{df}} & [\![C_1]\!]^{-1} \wedge [\![C_2]\!]^{-1}
\end{array}
$$

Table 5: Inverted core flat PDL_n in subtype constraints.

Consider a signature $\Sigma = B \cup \{\times, \rightarrow\}$. We construct a signature $s(\Sigma) =_{\mathrm{df}} B \cup \{f, c\}$, where f is function symbol of arity four and c is a fresh constant. Our approach is to use f to capture both $\times$ and $\rightarrow$, *i.e.*, all the non-constant function symbols in Σ. The first two arguments of f are used to model the two arguments of $\times$ and the next two to model the two arguments of $\rightarrow$. Thus, f is co-variant in all arguments except the third one.

Given a constraint φ over Σ, we construct $s(\varphi)$ over $s(\Sigma)$:

$$
\begin{array}{rcll}
s(x = y \times z) & =_{\mathrm{df}} & x = f(y, z, c, c) & \\
s(x = y \rightarrow z) & =_{\mathrm{df}} & x = f(c, c, y, z) & \\
s(x = b) & =_{\mathrm{df}} & x = b & \forall b \in B \\
s(x{\leq}y) & =_{\mathrm{df}} & x{\leq}y & \\
s(\varphi_1 \wedge \varphi_2) & =_{\mathrm{df}} & s(\varphi_1) \wedge s(\varphi_2) &
\end{array}
$$

Lemma 4 *If φ is weakly unifiable, then φ is satisfiable over Σ iff $s(\varphi)$ is satisfiable over $s(\Sigma)$.*

The proof of the above lemma requires the following result. Let φ be a constraint over a structural signature Σ. We have the following result due to Frey [9] that relates the shape of a solution of φ to that of a solution of $sh(\varphi)$.

Lemma 5 (Frey [9]) *If φ is satisfiable, let α be a solution of $sh(\varphi)$. Then φ has a solution β that is of the same shape as α,* i.e.*, for all $x \in V(\varphi) = V(sh(\varphi))$, $sh(\alpha(x) = \beta(x))$ is unifiable.*

5.2 Non-Structural to 1-Subtype Satisfiability

We handle non-structural signatures $\Sigma = B \cup \{\bot, \top, \times, \rightarrow\}$, similarly. The new signature is defined in exactly the same way as for the structural case by $s(\Sigma) = B \cup \{\bot, \top, f, c\}$. Constraints are also transformed in the same way, except including two extra rules for $\bot$ and $\top$:

$$
\begin{array}{rcl}
s(x = \bot) & =_{\mathrm{df}} & x = \bot \\
s(x = \top) & =_{\mathrm{df}} & x = \top
\end{array}
$$

However, weak unifiability is not sufficient for the initial satisfiability check. To see that, consider, for example, $x{\leq}y \times z \wedge x{\leq}u \rightarrow v$, which is satisfiable, but not weakly unifiable. To address this problem, we introduce a notion of *weak satisfiability*. It is similar to weak unfiability, except subtype ordering is also retained.

Definition 2 *Let φ be a constraint over Σ, and $\star$ be an arbitrary and fixed constant. We define $t^{\star}$ as before, except $\bot^{\star} =_{\mathrm{df}} \bot$ and $\top^{\star} =_{\mathrm{df}} \top$. Define the* weak satisfiability constraint *$ws(\varphi)$ as:*

$$
\begin{array}{rcl}
ws(t_1 = t_2) & =_{\mathrm{df}} & t_1^{\star} = t_2^{\star} \\
ws(x{\leq}y) & =_{\mathrm{df}} & x{\leq}y \\
ws(\varphi_1 \wedge \varphi_2) & =_{\mathrm{df}} & ws(\varphi_1) \wedge ws(\varphi_2)
\end{array}
$$

The constraint φ is called weakly satisfiable *iff $ws(\varphi)$ is satisfiable.*

Lemma 6 *If φ is weakly satisfiable, then φ is satisfiable over Σ iff $s(\varphi)$ is satisfiable over $s(\Sigma)$.*

The proof of this lemma requires the following result. Let φ be a constraint over a non-structural signature Σ. If $ws(\varphi)$ is satisfiable, then $ws(\varphi)$ has a minimum shape solution α by a simple extension of a theorem of Palsberg, Wand and OKeefe on non-structural subtype satisfiability over lattices [24]. We claim that if φ is satisfiable, then φ also has a minimum shape solution that is of the same shape as α.

Lemma 7 *If φ is satisfiable over Σ, let α be a minimum shape solution for $ws(\varphi)$, and in addition, α is such a solution with the least number of leaves assigned $\star$. Then φ has a solution β that is of the same shape as α,* i.e.*, for all $x \in V(\varphi) = V(ws(\varphi))$, $sh(\alpha(x) = \beta(x))$ is unifiable. Furthermore, β is a minimum shape solution of φ.*

Proof. Let γ be a solution for φ. We construct a variable assignment β for φ from α and γ:

$$
\beta(x)(\pi) =_{\mathrm{df}} \begin{cases} \gamma(x)(\pi) & \text{if } \alpha(x)(\varphi) = \star \\ \alpha(x)(\pi) & \text{otherwise.} \end{cases}
$$

One can show that $dom(\alpha) = dom(\beta)$, because if $\alpha(x)(\pi) = \star$, $\gamma(x)(\pi)$ must be a constant. We verify that $\beta \models \varphi$:

- Consider a literal of the form $x = f(x_1, \ldots, x_n)$ for $n > 0$. Both $\alpha(x) = f(\alpha(x_1), \ldots, \alpha(x_n))$ and $\gamma(x) = f(\gamma(x_1), \ldots, \gamma(x_n))$. Thus, if $\alpha(x)(\pi) = \star = (f(\alpha(x_1), \ldots, \alpha(x_n))(\pi)$, $\beta(x)(\pi) = \gamma(x)(\pi) = f(\gamma(x_1), \ldots, \gamma(x_n))(\pi) = f(\beta(x_1), \ldots, \beta(x_n))(\pi)$. Otherwise, $\beta(x)(\pi) = \alpha(x)(\pi) = f(\alpha(x_1), \ldots, \alpha(x_n))(\pi) = f(\beta(x_1), \ldots, \beta(x_n))(\pi)$.
- Consider a literal of the form $x = b$ for $b \in B \cup \{\bot, \top\}$. Clearly, $\beta(x) = b$.
- Consider a literal of the form $x \leq y$. We use a simple case analysis on the possible values of $\alpha(x)(\pi)$ and $\alpha(y)(\pi)$ for each address π. □

Lemma 5 and Lemma 7 together imply the following corollary, which is used next in Section 6 to treat subtype satisfiability interpreted over finite trees.

Corollary 1 *A subtype constraint φ is satisfiable over finite trees if and only if φ is satisfiable over finite trees of height bounded by $|\varphi|$. This holds for both structural and non-structural signatures.*

5.3 1-Subtype to Uniform Satisfiability

In this part, we give a reduction from 1-subtype to uniform subtype satisfiability. This reduction is uniform for subtyping with and without $\bot$ and $\top$.

Proposition 4 *Over possibly infinite trees, 1-subtype satisfiability is linear time reducible to uniform subtype satisfiability.*

Proof. Let Σ be a 1-subtype signature. We define a uniform signature $s(\Sigma)$ by extending the arities of all function symbols to the maximal arity of Σ (*i.e.*, the arity of the only non-trivial function symbol), such that:

- $s(\Sigma) =_{\text{df}} \Sigma$;
- $\forall f \in s(\Sigma).arity_{s(\Sigma)}(f) =_{\text{df}} \max$;
- $\leq_{s(\Sigma)} =_{\text{df}} \leq_\Sigma$

where max is the maximal arity of Σ.

We next translate a subtype constraint φ over Σ to a constraint $s(\varphi)$ over $s(\Sigma)$:

$$
\begin{aligned}
s(x{=}f(x_1, \ldots, x_{\max})) &=_{\text{df}} x{=}f(x_1, \ldots, x_{\max}) && (1)\\
s(x{=}b) &=_{\text{df}} x{=}b(y_1, \ldots, y_{\max}) && (2)\\
s(x_1 \leq x_2) &=_{\text{df}} x_1 \leq x_2 && (3)\\
s(\varphi_1 \wedge \varphi_2) &=_{\text{df}} s(\varphi_1) \wedge s(\varphi_2) && (4)\\
s(x{=}\bot) &=_{\text{df}} x{=}\bot(u_1, \ldots, u_{\max}) && (5)\\
s(x{=}\top) &=_{\text{df}} x{=}\top(v_1, \ldots, v_{\max}) && (6)
\end{aligned}
$$

where the y_i's, u_i's, and v_i's are fresh variables, and rules (5) and (6) are additional ones for a non-structural signature.

Lemma 8 *A subtype constraint φ over a standard signature Σ is satisfiable if and only if $s(\varphi)$ is satisfiable over the uniform signature $s(\Sigma)$.*

Proof of ($\Leftarrow$). For this implication, we define a transformation of $cut : tree_{s(\Sigma)} \to tree_\Sigma$:

- $cut(f(\tau_1, \ldots, \tau_{\max})) =_{\text{df}} f$, if $f \in \Sigma_0$;
- $cut(f(\tau_1, \ldots, \tau_{\max})) =_{\text{df}} f(cut(\tau_1), \ldots, cut(\tau_{\max}))$, otherwise.

We fix a solution α of $s(\varphi)$ and show that the variable assignment $cut \circ \alpha$ satisfies φ over Σ. We need to verify that $cut \circ \alpha$ satisfies all literals of φ:

1. Consider a literal of the form $x{=}f(x_1, \ldots, x_{\max})$ in φ (for $arity(f) = \max$). We know that $\alpha \models x = f(x_1, \ldots, x_{\max})$. In addition, the following sequence of implications holds:

$$
\begin{aligned}
& \alpha \models x{=}f(x_1, \ldots, x_{\max})\\
\Leftrightarrow\ & \alpha(x) = f(\alpha(x_1), \ldots, \alpha(x_{\max}))\\
\Rightarrow\ & cut(\alpha(x)) = f(cut(\alpha(x_1)), \ldots, cut(\alpha(x_{\max})))\\
\Leftrightarrow\ & cut \circ \alpha \models x{=}f(x_1, \ldots, x_{\max})
\end{aligned}
$$

2. Consider a literal of the form $x{=}b$ in φ. We know $\alpha \models x = b(y_1, \ldots, y_{\max})$ for some fresh variables y_i. Similarly, we have the following sequence of implications:

$$
\begin{aligned}
& \alpha \models x{=}b(y_1, \ldots, y_{\max})\\
\Leftrightarrow\ & \alpha(x) = b(\alpha(y_1), \ldots, \alpha(y_{\max}))\\
\Rightarrow\ & cut(\alpha(x)) = b\\
\Leftrightarrow\ & cut \circ \alpha \models x{=}b
\end{aligned}
$$

3. Consider a literal of the form $x \leq y$ in φ. We know $D_2 = dom(cut(\alpha(x))) \cap dom(cut(\alpha(y))) \subseteq dom(\alpha(x)) \cap dom(\alpha(y)) = D_1$. We thus have:

$$
\begin{aligned}
& \alpha \models x \leq y\\
\Leftrightarrow\ & \alpha(x) \leq \alpha(y)\\
\Leftrightarrow\ & \forall \pi \in D_1.\alpha(x)(\pi) \leq_\Sigma \alpha(y)(\pi)\\
\Rightarrow\ & \forall \pi \in D_2.cut(\alpha(x)(\pi)) \leq_\Sigma cut(\alpha(y)(\pi))\\
\Leftrightarrow\ & cut \circ \alpha \models x \leq y
\end{aligned}
$$

4. For a non-structural signature, literals of the form $x{=}\bot$ and $x{=}\top$ are treated similarly as $x{=}b$ in the second case.

($\Rightarrow$) We now consider the inverse implication. We first define a mapping $ext : tree_\Sigma \to tree_{s(\Sigma)}$:

- $ext(f(\tau_1, \ldots, \tau_{\max})) =_{\text{df}} f(ext(\tau_1), \ldots, ext(\tau_{\max}))$

- $ext(b) =_{\text{df}} b(\star^\infty, \ldots, \star^\infty)$, where $\star$ is an arbitrary, fixed constant in B of Σ, and $\star^\infty$ denotes the complete, infinite tree where each node is labeled $\star$, *i.e.*, the unique solution to the equation $x = \star(x, \ldots, x)$.
- For a non-structural signature, $ext(\bot)$ and $ext(\top)$ are defined respectively as the smallest tree and the greatest tree (over the two max-ary symbols $\bot$ and $\top$ in $s(\Sigma)$). They are defined mutually recursively and are unique solutions to the following equations:

$$\begin{aligned} x &= \bot(x_1, \ldots, x_{\max}) \\ y &= \top(y_1, \ldots, y_{\max}) \end{aligned}$$

  where $x_i = x$ and $y_i = y$ if the i-th argument is co-variant; and $x_i = y$ and $y_i = x$ otherwise.

  As an example, for a standard signature with the function type constructor $\rightarrow$, $ext(\bot)$ and $ext(\top)$ give the unique solution to the equations $x = \bot(y, x)$ and $y = \top(x, y)$.

We first state and prove the following lemma regarding *ext*.

Lemma 9 *If* $\tau_1 \leq \tau_2$, *then* $ext(\tau_1) \leq ext(\tau_2)$.

Proof. We use a proof by contradiction. We prove for a non-structural signature. For structural signatures, the proof is exactly the same, except discarding all cases involving $\bot$ or $\top$.

For $\tau_1 \leq \tau_2$, assume that $ext(\tau_1) \not\leq ext(\tau_2)$. Then there is a *shortest* path π such that $ext(\tau_1)(\pi) \not\leq^\pi ext(\tau_2)(\pi)$. Clearly, $\pi \neq \varepsilon$, and we let $\pi = \pi'.i$ for some π' and $i \in \{1, \ldots, \max\}$. We have a few cases:

1. When $\pi \in dom(\tau_1) \wedge \pi \in dom(\tau_2)$: This case is impossible because it contradicts the assumption that $\tau_1 \leq \tau_2$.
2. When $\pi \in dom(\tau_1) \wedge \pi \notin dom(\tau_2)$: $ext(\tau_2)(\pi')$ must be $\top$ if $par(\pi') = 1$ or $\bot$ if $par(\pi') = -1$. In either case, it is clear that $ext(\tau_1)(\pi) \leq^\pi ext(\tau_2)(\pi)$, a contradiction.
3. When $\pi \notin dom(\tau_1) \wedge \pi \in dom(\tau_2)$: This case is symmetric to the previous one.
4. When $\pi \notin dom(\tau_1) \wedge \pi \notin dom(\tau_2)$: We know $ext(\tau_1)(\pi')$ and $ext(\tau_2)(\pi')$ must be constants. In addition, $ext(\tau_1)(\pi') \leq^{\pi'} ext(\tau_2)(\pi')$, because π is the shortest path such that $ext(\tau_1)(\pi) \not\leq^\pi ext(\tau_2)(\pi)$. This would, however, imply that $ext(\tau_1)(\pi) \leq^\pi ext(\tau_2)(\pi)$, a contradiction. □

We can now finish the proof of Lemma 8. Let $\alpha \models \varphi$. We construct a variable assignment β for $s(\varphi)$:

- $\beta(x) =_{\text{df}} ext(\alpha(x))$ for all variables $x \in V(\varphi)$;
- $\beta(x) =_{\text{df}} \star^\infty$ for the y_i's;
- $\beta(x) =_{\text{df}} ext(\bot)$ for a co-variant u_i or contra-variant v_i;
- $\beta(x) =_{\text{df}} ext(\top)$ for a contra-variant u_i or co-variant v_i.

We need to show that $\beta \models s(\varphi)$. There are a few kinds of literals in $s(\varphi)$:

1. Consider a literal of the form $x = f(x_1, \ldots, x_{\max})$ of $s(\varphi)$, derived from $x = f(x_1, \ldots, x_{\max})$ of φ. We know that $\alpha \models x = f(x_1, \ldots, x_{\max})$, *i.e.*, $\alpha(x) = f(\alpha(x_1), \ldots, \alpha(x_{\max}))$. Thus, we have:

$$\begin{aligned} \beta(x) &= ext(f(\alpha(x_1), \ldots, \alpha(x_{\max}))) \\ &= f(ext(\alpha(x_1)), \ldots, ext(\alpha(x_{\max}))) \\ &= f(\beta(x_1), \ldots, \beta(x_{\max})) \\ &= \beta(f(x_1, \ldots, x_{\max})) \end{aligned}$$

   Hence, $\beta \models x = f(x_1, \ldots, x_{\max})$.

2. Consider a literal of the form $x = b(y_1, \ldots, y_{\max})$, derived from $x = b$ in φ. We know that $\alpha \models x = b$, *i.e.*, $\alpha(x) = b$. We thus have:

$$\begin{aligned} \beta(x) &= ext(\alpha(x)) \\ &= ext(b) \\ &= b(\beta(y_1), \ldots, \beta(y_{\max})) \\ &= \beta(b(y_1, \ldots, y_{\max})) \end{aligned}$$

   Hence, $\beta \models x = b(y_1, \ldots, y_{\max})$.

3. Consider a literal of the form $x \leq y$, derived from $x \leq y$ in φ. We know $\alpha \models x \leq y$, *i.e.*, $\alpha(x) \leq \alpha(y)$. By Lemma 9, we have $\beta(x) = ext(\alpha(x)) \leq ext(\alpha(y)) = \beta(y)$. Thus, $\beta \models x \leq y$.
4. For a non-structural signature, we have literals of the form $x{=}\bot(u_1, \ldots, u_{\max})$ and $x{=}\top(v_1, \ldots, v_{\max})$. For the case with $\bot$, we know $\alpha \models x = \bot$. We thus have:

$$\begin{aligned} \beta(x) &= ext(\alpha(x)) \\ &= ext(\bot) \\ &= \bot(\beta(u_1), \ldots, \beta(u_{\max})) \\ &= \beta(\bot(u_1, \ldots, u_{\max})) \end{aligned}$$

   The case for $\top$ is similar. □

With Lemma 8, we have finished the proof of Proposition 4.

5.4 Uniform to (Non-)Structural Satisfiability

In this part, we prove the last step of the equivalence (Theorem 4), namely, how to reduce uniform subtype satisfiability to structural and non-structural subtype satisfiabilities.

Proposition 5 *Uniform subtype satisfiability is linear time reducible to structural and non-structural subtype satisfiability over possibly infinite trees.*

To simplify its proof we assume a uniform subtype problem where all function symbols have arity three with their first two arguments being contravariant and the last one covariant. This proof can be easily adapted to uniform signatures with other arities and polarities.

We construct a reverse translation $\bar{\bar{s}}$ of s (defined in Section 5.3) in two steps. Let Σ be a uniform signature with symbols of arity three. We first define a standard signature $\bar{s}(\Sigma)$ by including symbols in Σ as constants and adding $\to$:

- $\bar{s}(\Sigma) \;=_{\text{df}}\; \Sigma \cup \{\to\}$;
- $\forall g \in \Sigma.\mathit{arity}_{\bar{s}(\Sigma)}(g) \;=_{\text{df}}\; 0$;
- $\mathit{arity}_{\bar{s}(\Sigma)}(\to) \;=_{\text{df}}\; 2$;
- $\leq_{\bar{s}(\Sigma)} \;=_{\text{df}}\; \leq_{\Sigma}$;

We now translate a subtype constraint φ over Σ to a constraint $\bar{s}(\varphi)$ over $\bar{s}(\Sigma)$:

$$\begin{array}{rcl} \bar{s}(x{=}g(x_1,x_2,x_3)) & =_{\text{df}} & x{=}(x_3 \to x_2) \to (x_1 \to g) \\ \bar{s}(x_1 \leq x_2) & =_{\text{df}} & x_1 \leq x_2 \\ \bar{s}(\varphi_1 \wedge \varphi_2) & =_{\text{df}} & \bar{s}(\varphi_1) \wedge \bar{s}(\varphi_2) \end{array}$$

where we use a non-flat constraint in the first line for a simpler presentation. The arguments x_1, x_2 are again contravariant and x_3 is covariant in the constraint $\bar{s}(x{=}g(x_1,x_2,x_3))$. Thus, $\bar{s}$ preserves all polarities.

In our second step, we force every variable to be mapped to a fixed, infinite shape. We extend $\bar{s}(\Sigma)$ to $\bar{\bar{s}}(\Sigma)$ with four new constants a_1, a_2, a_3, and a_4 with the following ordering:

$$\begin{array}{rcl@{\qquad}rcl} a_1 & \leq & c & a_2 & \leq & c \\ c & \leq & a_3 & c & \leq & a_4 \end{array}$$

for all constants $c \in \bar{s}(\Sigma)$. We define $\bar{\bar{s}}(\varphi)$ as the conjunction of $\bar{s}(\Sigma)$ and the following constraints:

(1) $u_1 \leq x \wedge u_2 \leq x \wedge x \leq u_3 \wedge x \leq u_4$, for each variable $x \in V(\bar{s}(\Sigma))$; and

(2) $\bigwedge_{i=1,2,3,4}\; u_i{=}(u_i \to u_i) \to (u_i \to a_i)$

The constraints (1) and (2) in $\bar{\bar{s}}(\varphi)$ determine the shape of any variable $x \in V(\bar{s}(\varphi))$. We claim, in the following lemma, that any solution to $\bar{\bar{s}}(\varphi)$ must be of a particular shape and must also map variables $x \in V(\bar{s}(\varphi))$ to trees over $\bar{s}(\Sigma)$.

Lemma 10 *When the constraint $\bar{\bar{s}}(\varphi)$ is interpreted over any (non-)structural signature $\bar{\bar{s}}(\Sigma)$ or $\bar{\bar{s}}(\Sigma) \cup \{\bot, \top\}$, every variable assignment $\alpha \models \bar{\bar{s}}(\varphi)$ satisfies that for all paths $\pi \in (1(1{\cup}2) \cup 21)^*$:*

$$\begin{array}{rcll} \alpha(x)(\pi') & = & \to & \text{if } \pi' \text{ is a prefix of } \pi \\ \alpha(x)(\pi 22) & = & \begin{cases} a_i & \text{if } x = u_i \\ c \in \Sigma & \text{otherwise.} \end{cases} & \end{array}$$

Lemma 11 *A subtype constraint φ over a uniform signature Σ is satisfiable if and only if the constraint $\bar{\bar{s}}(\varphi)$ over $\bar{\bar{s}}(\Sigma)$ is satisfiable. This statement also holds if we replace the structural signature $\bar{\bar{s}}(\Sigma)$ by the non-structural signature $\bar{\bar{s}}(\Sigma) \cup \{\bot, \top\}$.*

Proof. We define a transformation of $\mathit{map} : \mathit{tree}_{\Sigma} \to \mathit{tree}_{\bar{\bar{s}}(\Sigma)}$ on trees for all $g \in \Sigma$:

$$\begin{array}{rcl} \mathit{map}(g(\tau_1,\tau_2,\tau_3)) & =_{\text{df}} & (\mathit{map}(\tau_3) \to \mathit{map}(\tau_2)) \\ & \to & (\mathit{map}(\tau_1) \to g) \end{array}$$

With that it can be easily verified that if there exists a solution $\alpha \models \varphi$ over an uniform signature Σ then $\mathit{map}(\alpha) \models \bar{\bar{s}}(\varphi)$ holds over $\bar{\bar{s}}(\Sigma)$. For the other direction we assume an assignment $\alpha \models \bar{\bar{s}}(\varphi)$. Then there also exists an assignment $\beta = \mathit{map}^{-1}(\alpha)$ according to the shape of any solution of $\bar{\bar{s}}(\varphi)$ stated in Lemma 10. Again, it can be easily verified that $\beta \models \Sigma$.

The proof also holds in the case where we add $\bot$ and $\top$ to $\bar{\bar{s}}(\Sigma)$ since both symbols cannot occur in any node of any solution of $\bar{\bar{s}}(\Sigma)$ (again Lemma 10). □

6 Finite Subtype Satisfiability over Posets

The complexity of finite structural subtype satisfiability was shown to be *PSPACE*-complete by Tiuryn [33] and Frey [9]. Here, we establish the same complexity for the non-structural case.

6.1 *PSPACE*-hardness

Proposition 6 *Non-structural subtype satisfiability over finite trees is PSPACE-hard.*

The analogous result for the structural case was shown by Tiuryn [33]). To lift this result, we show how to reduce non-structural to structural subtype satisfiability.

Lemma 12 *Structural subtype satisfiability is polynomial time reducible to non-structural subtype satisfiability (both for finite and infinite trees).*

Proof. Let Σ be a structural signature. We construct a non-structural signature:

$$s(\Sigma) \;=_{\text{df}}\; \Sigma \cup \{\bot, \top, a_1, a_2, a_3, a_4\}$$

with the a_i's four new constants. In addition,

$$\leq_{s(\Sigma)} =_{\text{df}} \leq_\Sigma \cup\{(a_1,c),(a_2,c),(c,a_3),(c,a_4) \mid c \in \Sigma_0\}$$

Let φ be a constraint over Σ. We construct $s(\varphi)$ over $s(\Sigma)$. Consider φ's shape constraint $sh(\varphi)$ (see Definition 1). If $sh(\varphi)$ is not unifiable, we simply let $s(\varphi) =_{\text{df}} \top\leq\bot$. Otherwise, consider the most general unifier (m.g.u.) γ of $sh(\varphi)$. We let $sh(\varphi)'$ be the same as $sh(\varphi)$ except each occurrence of $\star$ is replaced with a fresh variable. We make two copies of $sh(\varphi)'$, $sh(\varphi)'_L$ and $sh(\varphi)'_R$ (for left and right), where each variable x is distinguished as x_L and x_R respectively. For each variable $x \in V(\varphi)$, if $\gamma(x)$ is either $\star$ or belongs to $V(\varphi)$, we say x is atomic. For a variable x, let $force(x)$ denote the constraint:

$$a_1\leq x \wedge a_2\leq x \wedge x\leq a_3 \wedge x\leq a_4$$

Notice that Lemma 12 holds both for finite and infinite trees.

We can now construct $s(\varphi)$, which is the conjunction of the following components: (1) φ itself; (2) $sh(\varphi)'_L$; (3) $sh(\varphi)'_R$; (4) For each atomic $x \in V(\varphi)$, $force(x_L)$ and $force(x_R)$; (5) For each fresh variable x in $sh(\varphi)'_L$ and $sh(\varphi)'_R$, $force(x)$; and (6) For each variable $x \in V(\varphi)$, $x_L\leq x\leq x_R$. One can show that φ is satisfiable over Σ iff $s(\varphi)$ is satisfiable over $s(\Sigma)$. □

6.2 A *PSPACE* Algorithm

Theorem 5 *Finite non-structural subtype satisfiability is PSPACE-complete.*

It remains to prove membership in *PSPACE*. We present a proof based on *K-normal modal logic* which applies uniformly to the non-structural and the structural cases.

We assume a signature Σ that contains at least one constant c and non-constant symbol f. Satisfiability would be trivial otherwise. We adapt our reduction to uniform signature for the finite case. Let $s(\Sigma)$ be the uniform signature for Σ. With the following formulas we define an additional subtype ordering $\sqsubset_{s(\Sigma)}$ by $g\sqsubset_{s(\Sigma)}f$ for all $g \in s(\Sigma)$ and one fixed symbol f:

$$\begin{array}{lll} \mathit{finite}_n(x) & =_{\text{df}} & \exists y_1 ... \exists y_{n+1}.\ x\sqsubseteq y_1 \\ & \wedge & \wedge_{i=1}^{n} y_i{=}f(y_{i+1},...y_{i+1}) \\ & \wedge & y_{n+1}{=}c(y_{n+1},...y_{n+1}) \\ \mathit{finite}_n(V) & =_{\text{df}} & \wedge_{x\in V}\, \mathit{finite}_n(x) \end{array}$$

The following proposition holds by Theorem 4 and Corollary 1.

Proposition 7 *A subtype constraint φ is satisfiable over finite Σ-trees if and only if the constraint $s(\varphi) \wedge \mathit{finite}_{|\varphi|}(V(\varphi))$ is satisfiable over uniform signatures $s(\Sigma)$.*

We can easily adapt the translation of subtype constraints over uniform signatures into inverted core PDL_n from Table 3 to handle the formulas $\mathit{finite}_{|\varphi|}(V(\varphi))$ as well. This yields a satisfiability preserving encoding into inverted core PDL_n for the finite case. We finally alter this encoding to a translation into the following modal logic:

$$D \ ::= \ B \mid [\{1,\ldots,n\}^{|\varphi|}]B \mid D_1 \wedge D_2$$

Because all trees in finite solutions of φ have at most linear depth it is correct to replace all $[*]$ modalities by $[\{1,\ldots,n\}^{|\varphi|}]$, both, in the reduction of Table 3 and in the well-formedness property wff_V. This gives a translation into formulas D.

Proposition 8 *Satisfiability of inverted linearly depth-bounded PDL_n formulas D is in PSPACE.*

Proof. We translate the problem to satisfiability of *K-normal modal logic* over the complete infinite n-ary tree (which is known to be in *PSPACE* [31]). It is defined by the syntax of PDL_n formulas A restricted to the single modality $[\{1,2\}]$:

$$E \ ::= \ p \mid \neg E \mid E \wedge E' \mid E \leftrightarrow E' \mid \Box E$$

We denote i repetitions of $\Box$ by $\Box^i$. The only complication is to translate formulas $[i]B$. In the case of binary trees (other cases of n are analogous) we do so by translating $[1]B$ to $\Box(p \rightarrow B)$ and $[2]B$ to $\Box(\neg p \rightarrow B)$, where we use a new variable p that is true at all paths $\pi 1$ and false at paths $\pi 2$. Following [12] p can be axiomatized by:

$$\wedge_{i=1}^{m} \Box^{i-1}(\langle\{1,2\}\rangle\Box^{m-i}p \wedge \langle\{1,2\}\rangle\Box^{m-i}\neg p).$$

7 Conclusions

We have given a complete characterization of the complexity of subtype satisfiability over posets through a new connection of subtype satisfiability with modal logics, which have well understood satisfiability problems. Our technique yields a uniform and systematic treatment of different choices of subtype orderings: finite versus recursive types, structural versus non-structural subtyping, and considerations of symbols with co- and contra-variant arguments.

Our technique, however, does not extend beyond satisfiability to other first-order fragments that require negations, such as subtype entailment, whose decidability is a longstanding open problem over non-structural signatures. Negations can certainly be modeled by our modal logic, but only over uniform signatures. In fact,

there must not exist reductions from standard signatures to uniform ones that preserve subtype entailment, for example. Otherwise, such a reduction would have implied that the first-order theory of non-structural subtyping, which is undecidable [32], were a fragment of S2S, which is decidable [28].

References

[1] R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993.

[2] M. Ben-Ari, J. Y. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: Finite models, complexity, and completeness. *Journal of Computer and System Sciences*, 25(3):402–417, 1982.

[3] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cam. Uni. Press, 2001.

[4] P. Blackburn, B. Gaiffe, and M. Marx. Variable free reasoning on finite trees. In *Proceedings of Mathematics of Language*, pages 17–30, 2003.

[5] P. Blackburn and W. Meyer-Viol. Linguistics, logic, and finite trees. *Logic Journal of the IGPL*, 2:3–29, 1994.

[6] J. Eifrig, S. Smith, and V. Trifonov. Sound polymorphic type inference for objects. In *OOPSLA*, pages 169–184, 1995.

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

[8] C. Flanagan and M. Felleisen. Componential set-based analysis. In *PLDI*, 1997.

[9] A. Frey. Satisfying subtype inequalities in polynomial space. *Theoretical Computer Science*, 277:105–117, 2002.

[10] Y. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73(2):155–175, 1990.

[11] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, 2000.

[12] E. Hemaspaandra. The complexity of poor man's logic. In *TACS*, pages 230–242, 2000.

[13] F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *LICS*, pages 362–372, 1997.

[14] F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *ICALP*, pages 616–627, 1998.

[15] M. Hoang and J. Mitchell. Lower bounds on type inference with subtypes. In *POPL*, 176–185, 1995.

[16] D. Kozen, J. Palsberg, and M. I. Schwartzbach. Efficient inference of partial types. *Journal of Computer and System Sciences*, 49(2):306–324, 1994.

[17] M. Kracht. Inessential features. In *Logical Aspects of Computational Linguistics*, volume 1328, pages 43–62, 1997.

[18] V. Kuncak and M. Rinard. Structural subtyping of non-recursive types is decidable. In *LICS*, pages 96–107, 2003.

[19] J. Mitchell. Coercion and type inference. In *POPL*, pages 175–185, 1984.

[20] J. C. Mitchell. Type inference with simple subtypes. *The Journal of Functional Programming*, 1(3):245–285, 1991.

[21] J. Niehren and T. Priesnitz. Entailment of non-structural subtype constraints. In *Asian Computing Science Conference*, pages 251–265, 1999.

[22] J. Niehren and T. Priesnitz. Non-structural subtype entailment in automata theory. *Information and Computation*, 186(2):319–354, 2003.

[23] A. Palm. Propositional tense logic for trees. In *Proceedings of the Sixth Meeting on Mathematics of Language*, pages 74–87, 1999.

[24] J. Palsberg, M. Wand, and P. O'Keefe. Type Inference with Non-structural Subtyping. *Formal Aspects of Computing*, 9:49–67, 1997.

[25] F. Pottier. Simplifying subtyping constraints. In *ICFP*, pages 122–133, 1996.

[26] F. Pottier. *Type inference in the presence of subtyping: from theory to practice*. PhD thesis, INRIA, 1998.

[27] V. Pratt and J. Tiuryn. Satisfiability of inequalities in a poset. *Fundamenta Informaticae*, 28:165–182, 1996.

[28] M. O. Rabin. Decidability of second-order theories and automata on infinte trees. *Trans. Amer. Math. Soc.* , 141:1–35, 1969.

[29] J. Rehof. *The Complexity of Simple Subtyping Systems*. PhD thesis, DIKU, 1998.

[30] H. Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52(2):57–60, 1994.

[31] E. Spaan. *Complexity of Modal Logics*. PhD thesis, University of Amsterdam, 1993.

[32] Z. Su, A. Aiken, J. Niehren, T. Priesnitz, and R. Treinen. First-order theory of subtyping constraints. In *POPL*, pages 203–216, 2002.

[33] J. Tiuryn. Subtype inequalities. In *LICS*, pages 308–315, 1992.

[34] J. Tiuryn and M. Wand. Type reconstruction with recursive types and atomic subtyping. In *Theory and Practice of Software Development*, 686-701,'93.

[35] V. Trifonov and S. Smith. Subtyping constrained types. In *SAS*, pages 349–365, 1996.

[36] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal Computer & System Science*, 32(2):183–221, 1986.

A Proving *DEXPTIME*-hardness

We prove that satisfiability of flat core PDL_n is *DEXPTIME*-hard and thus *DEXPTIME*-complete (Theorem 2).

We proceeds in two steps. We first introduce a new dialect of PDL_n that we call the *core of* PDL_n , and express emptiness of intersections of tree automata in that language. This proofs *DEXPTIME* hardness [30] of core PDL_n. In the second step, we normalize core PDL_n into flat core PDL_n. This relies on a flattening procedure inspired by techniques of Spaan [31].

A.1 Core PDL_n

The core of PDL_n is a fragment of PDL_n that is slightly richer than flat core PDL_n. Formulas C of core PDL_n are conjunctions of propositional variables and expression $[*]B$, where B is an arbitrary, possibly non-flat Boolean expression.

$$\begin{array}{lcl} B & ::= & p \mid \neg B \mid B \wedge B' \mid B \leftrightarrow B' \mid [i]B \\ C & ::= & B \mid [*]B \mid C \wedge C \end{array}$$

The modalities are again restricted to immediate $[i]$ successors (where $1 \leq i \leq n$) and arbitrary $[*]$ descendants such that $[*]$ cannot be nested below other modalities.

In Table 6 we define a set of standard operators on Boolean expressions in core PDL_n formulas. Note that all operators affect the size of formulas linearly. Our

or.	$B_1 \vee B_2$	$=_{\mathrm{df}} \neg(\neg B_1 \wedge \neg B_2)$
implication.	$B_1 \rightarrow B_2$	$=_{\mathrm{df}} \neg B_1 \vee B_2$
exclusive or.	$B_1 \stackrel{+}{\vee} B_2$	$=_{\mathrm{df}} \neg(B_1 \leftrightarrow B_2)$
false value.	*false*	$=_{\mathrm{df}} p_0 \wedge \neg p_0$ for some $p_0 \in Pr$
true value.	*true*	$=_{\mathrm{df}} \neg$*false*

Table 6: Operators on Boolean expressions.

syntax provides for equivalences, to avoid the exponential blow up in the standard:

$$A_1 \leftrightarrow A_2 \models\!| A_1 \rightarrow A_2 \wedge A_2 \rightarrow A_1$$

But expressing nested equivalences through two-sided implications might blow up sizes exponentially.

Proposition 9 *Satisfiability of core PDL_n formulas is DEXPTIME-hard.*

Proof. This can be proved by a closer inspection of *DEXPTIME*-hardness proofs for *PDL* [3, 11, 31] or deterministic *PDL* [36]. Here, we give a new direct proof by encoding emptiness of intersections of tree automata.

Let Σ be a finite ranked signature. A *tree automaton* A over a signature Σ consists of a finite set $states(A)$ of *states*, a subset $final(A) \subseteq states(A)$ of *final states*, and a set $rules(A)$ of *transition rules* of the form $f(q_1, \ldots, q_n) \rightarrow q$ where $q_1, \ldots, q_n, q \in states(A_i)$ and $n = arity_\Sigma(f)$. The language of a tree automaton $L(A)$ contains all those ground terms over Σ that can be evaluated into a final state by rule application.

We first encode trees over Σ in PDL_n with max successors where max is the maximal arity of function symbols in Σ. We introduce fresh propositional variables p_f for every symbol $f \in \Sigma$ to represent f-labeled nodes, and a propositional variable p_{dom} to express tree domains. A model M encodes a tree τ if for all $\pi \in \{1, \ldots, max\}^*$:

$$\begin{array}{lcll} M, \pi & \models & p_{dom} \quad \text{iff} \quad \pi \in dom(\tau) & \text{and} \\ M, \pi & \models & p_f \quad \text{iff} \quad \tau(\pi) = f & \end{array}$$

Lemma 13 *There exists a formula $tree_\Sigma$ in the core of PDL_n whose models represent precisely the trees in $tree_\Sigma$.*

Proof. We use a couple of well-formedness conditions for representations of possibly infinite trees. Formula $label_\Sigma$ says that the root of every ground term belongs to its domain and every node of the domain is labeled in Σ.

$$label_\Sigma =_{\mathrm{df}} p_{dom} \wedge [*](p_{dom} \rightarrow \bigvee_{f \in \Sigma}^{+} p_f)$$

Condition $arity_\Sigma$ requires that every node of a tree fulfills the arity required by its label.

$$arity_\Sigma =_{\text{df}} [*](\bigwedge_{f\in\Sigma} p_f \rightarrow \ (\bigwedge_{i=1}^{arity_\Sigma(f)}[i]p_{dom} \wedge \bigwedge_{j=1+arity_\Sigma(f)}^{max}[j]\neg p_{dom}))$$

Property *prefix* restricts tree domains of ground terms to be prefixed-closed:

$$prefix =_{\text{df}} [*]p_{dom} \rightarrow \left(\bigwedge_{i=1}^{max}[i]\ p_{dom}\right)$$

Possibly infinite trees are now definable:

$$tree_\Sigma =_{\text{df}} label_\Sigma \wedge arity_\Sigma \wedge prefix \qquad \square$$

We next want to restrict models to representation ground terms, i.e., to finite trees over Σ, but unfortunately, finiteness cannot be expressed in PDL_n. Lemma 14 indicates a way out of this problem. It is sufficient to restrict the depth of terms exponentially, rather then to impose finiteness.

Lemma 14 *Let $(A_i)_{i=1}^n$ be a finite sequence of tree automata over the same signature. If the intersection $\cap_{i=1}^n L(A_i)$ is nonempty, then it contains some tree of depth bounded by $\prod_{i=1}^n |states(A_i)|$.*

Proof. We can construct a tree automaton for the intersection with at most $\prod_{i=1}^n |states(A_i)|$ and then apply the pumping lemma for regular tree languages. $\square$

It is thus sufficient to encode ground terms whose depth is bounded exponentially in the size of the given intersection of tree automata. This can be expressed by a PDL_n formula of polynomial size, which simulates a counter.

Lemma 15 *For every $n \geq 0$, there exists a formula $ground{-}term_\Sigma(n)$ in the core of PDL_n describing all finite trees over Σ with depth bounded by 2^n.*

Proof. Condition $counter(n)$ describes an n-bit counter that counts the depth of nodes starting from the root. We consider tree models with propositional variables $(p_i)_{i=1}^n$ that represent the n bits of the counter. Lets identify the Boolean values t with the digit 1 and f with 0. For every model M and node π the sequence $M(p_n,\pi)\ldots M(p_1,\pi)$ is the binary representation of the depth of node π in tree M, modulo 2^n.

$$\begin{array}{rcl}
all(n) & =_{\text{df}} & \bigwedge_{i=1}^n p_i \\
counter(0) & =_{\text{df}} & true \\
counter(n) & =_{\text{df}} & counter(n{-}1) \wedge \neg p_n \\
 & & \wedge[*]((\neg p_n \wedge \neg all(n{-}1)) \rightarrow \wedge_{i=1}^n [i]\neg p_n) \\
 & & \wedge[*]((\neg p_n \wedge all(n{-}1)) \rightarrow \wedge_{i=1}^n [i] p_n) \\
 & & \wedge[*]((p_n \wedge \neg all(n{-}1)) \rightarrow \wedge_{i=1}^n [i] p_n) \\
 & & \wedge[*]((p_n \wedge all(n{-}1)) \rightarrow \wedge_{i=1}^n [i]\neg p_n)
\end{array}$$

$$\begin{array}{rcl}
flat_1(p) & =_{\text{df}} & [*]\,(P_p \leftrightarrow p \wedge p) \\
flat_1(\neg B) & =_{\text{df}} & [*]\,(P_{\neg B} \leftrightarrow \neg P_B) \wedge flat_1(B) \\
flat_1(B \wedge B') & =_{\text{df}} & [*]\,(P_{B\wedge B'} \leftrightarrow (P_B \wedge P_{B'})) \\
 & & \wedge\, flat_1(B) \wedge flat_1(B') \\
flat_1([i]B) & =_{\text{df}} & [*]\,(P_{[i]B} \leftrightarrow [i]P_B) \wedge flat_1(B) \\
flat_1(B \leftrightarrow B') & =_{\text{df}} & [*]\,(P_{B\leftrightarrow B'} \leftrightarrow (P_{B\rightarrow B'} \wedge P_{B'\rightarrow B})) \\
 & & \wedge\, flat_1(B{\rightarrow}B') \wedge flat_1(B'{\rightarrow}B) \\
flat_2(B) & =_{\text{df}} & P_B \wedge flat_1(B) \\
flat_2([*]B) & =_{\text{df}} & [*]P_B \wedge flat_1(B) \\
flat_2(C \wedge C) & =_{\text{df}} & flat_2(C) \wedge flat_2(C')
\end{array}$$

Table 7: Flattening core PDL_n formulas.

The formula $depth(n)$ bounds the depth of nodes in the domain to 2^n.

$$depth(n) =_{\text{df}} counter(n) \wedge [*](all(n) \rightarrow \neg p_{dom})$$

We can now define ground terms:

$$ground{-}term_\Sigma(n) =_{\text{df}} tree_\Sigma \wedge depth(n) \qquad \square$$

Let $(A_i)_{i=1}^n$ be a sequence of tree automata over a signature Σ with disjunct state sets. We encode simultaneously accepting runs of all tree automata $(A_i)_{i=1}^n$. We use propositional variables p_q for all states $q \in \uplus_{i=1}^n states(A_i)$.

$$\begin{array}{rcl}
run(A_i) & =_{\text{df}} & [*]\ (\wedge_{q\in states(A)}\ ((p_q \wedge p_f) \rightarrow \\
 & & \quad \vee_{\ f(q_1,\ldots,q_n)\rightarrow q\in rules(A_i)}\ \wedge_{i=1}^n\ [i]p_{q_i})) \\
accept(A_i) & =_{\text{df}} & \vee_{q\in final(A_i)} p_q \wedge run(A_i)
\end{array}$$

Lemma 16 *The intersection $\cap_{i=1}^n L(A_i)$ is the set of ground terms that yield models of the following PDL_n formula for $k = \max_{i=1}^n |states(A_i)|$:*

$$ground{-}term(\lceil log(k)\rceil * n) \wedge \wedge_{i=1}^n accept(A_i)$$

A.2 Flattening

Proposition 10 *Every core PDL_n formula C is satisfaction equivalent to some flat core PDL_n formula, that can be computed in linear time.*

The idea of the proof is to introduce new propositional variables for all sub-term positions of a given PDL_n formula. We fix a finite set Pr_0 of propositional variables and an injective generator function:

$$P : PDL_n \rightarrow (Pr - Pr_0)$$

that maps a PDL_n formulas A to propositional variables P_A. Given this generator, Table 7 defines two flattening

$$
\begin{array}{rcll}
M'(p,\pi) & =_{\mathrm{df}} & M(p,\pi) & \text{if } p \in Pr_0 \\
M'(p,\pi) & =_{\mathrm{df}} & \text{arbitrary} & \text{if } p \notin Pr_0 \cup P(B) \\
M'(P_p,\pi) & =_{\mathrm{df}} & M'(p,\pi) & \\
M'(P_{\neg B},\pi) & =_{\mathrm{df}} & \neg M'(B,\pi) & \\
M'(P_{B_1 \wedge B_2},\pi) & =_{\mathrm{df}} & M'(B_1,\pi) \wedge M'(B_2,\pi) & \\
M'(P_{B_1 \leftrightarrow B_2},\pi) & =_{\mathrm{df}} & M'(B_1,\pi) \leftrightarrow M'(B_2,\pi) & \\
M'(P_{[i]B},\pi) & =_{\mathrm{df}} & \forall\, 1 \leq i \leq n. M'(B,\pi i) &
\end{array}
$$

Figure 3: Extending models to new variables P_B.

functions flat_1 and flat_2, for core PDL_n formulas of type B and C respectively.

Formulas $\mathit{flat}_1(B)$ and $\mathit{flat}_2(C)$ are clearly flat core PDL_n formulas for all core PDL_n formulas B and $\mathit{flat}_2(C)$, except for sub-formulas $[*]P_B$ which can be expressed through $[*](P_B \leftrightarrow \mathit{true})$ and thus by the flat formula: $[*](P_B \leftrightarrow P_{\mathit{true}} \wedge P_{\mathit{true}}) \wedge \mathit{flat}_1(\mathit{true})$.

The sizes of $\mathit{flat}_1(B)$ and $\mathit{flat}_2(C)$ remain linear in those of B and C respectively, when sharing common subconstraints $\mathit{flat}_1(B_1)$ and $\mathit{flat}_1(B_2)$ in translations of equivalences $\mathit{flat}_1(B_1 \leftrightarrow B_2)$, i.e., in $\mathit{flat}_1(B_1 \rightarrow B_2)$ and $\mathit{flat}_1(B_2 \rightarrow B_1)$.

Lemma 17 (Correctness) *For all core PDL_n formulas C with $Pr(C) \subseteq Pr_0$:*

$$C \models\!\!\models \exists Pr{-}Pr_0.\ \mathit{flat}_2(C)$$

The proof relies on the following two Lemmas.

Lemma 18 *For all core PDL_n formulas B with variables $Pr(B) \subseteq Pr_0$:*

$$\models \forall Pr_0\ \exists Pr{-}Pr_0.\ \mathit{flat}_1(B)$$

Proof. We fix a model $M : Pr \times \{1,\ldots n\}^* \rightarrow \{0,1\}$ of B and define a model $M' : Pr \times \{1,\ldots n\}^* \rightarrow \{0,1\}$ of $\mathit{flat}_1(B)$ in Figure 3. The definition of $M'(P_B,\pi)$ is by induction on the structure of terms B. Clearly, M' differs from M only on variables in $Pr - Pr_0$. We can show $M' \models \mathit{flat}_1(B)$ for all formulas B with variables $Pr(B) \subseteq Pr_0$ by induction on the structure of B. □

Lemma 19 $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow B)$ *for all core PDL_n formulas B.*

Proof. By induction on the structure of formulas B.

1. Let $B = p$ then $\mathit{flat}_1(p) \models [*]\,(P_p \leftrightarrow p)$ (Table 7).
2. Let $B = B_1 \wedge B_2$ (the remaining cases $B = \neg B'$ or $B = B_1 \leftrightarrow B_2$ are analogous). It holds that $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow (P_{B_1} \wedge P_{B_2})$ (Table 7). It further holds for $i \in \{1,2\}$ that $\mathit{flat}_1(B) \models [*]\,(P_{B_i} \leftrightarrow B_i)$ by induction on B_i. We conclude $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow B_1 \wedge B_2)$.
3. Case $B = [i]\,B'$. It hold that $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow [i]P_{B'})$ (Table 7) and $\mathit{flat}_1(B) \models [*]\,(P_{B'} \leftrightarrow B')$ by induction on B'. It follows that $\mathit{flat}_1(B) \models [*][i]\,(P_{B'} \leftrightarrow B')$ also holds. Again we conclude $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow [i]B')$. □

Proof. [of Correctness Lemma 17] By induction on C. Let $C = [*]B$ (the case $C = B$ will be subsumed). To prove is $[*]B \models\!\!\models \exists Pr{-}Pr_0\ ([*]P_B \wedge \mathit{flat}_1(B))$ for all core PDL_n formulas B with $Pr(B) \subseteq Pr_0$ (see Table 7). Lemma 19 yields $\mathit{flat}_1(B) \models [*]\,(P_B \leftrightarrow B)$ and thus $[*]\,P_B \wedge \mathit{flat}_1(B) \models [*]\,B$. Since $Pr(B) \subseteq Pr_0$ this is equivalent to $\exists Pr{-}Pr_0.[*]\,P_B \wedge \mathit{flat}_1(B) \models [*]\,B$. The converse follows from Lemma 18:

$$
\begin{array}{rcl}
[*]\,B & \models\!\!\models & [*]\,B \wedge \forall Pr_0.\ \exists Pr{-}Pr_0.\ \mathit{flat}_1(B) \\
& \models & [*]\,B \wedge \exists Pr{-}Pr_0.\ \mathit{flat}_1(B) \\
& \models & \exists Pr{-}Pr_0\ ([*]\,B \wedge \mathit{flat}_1(B))
\end{array}
$$