Complexity of Subtype Satisfiability over Posets

Joachim Niehren, Tim Priesnitz, Zhendong Su

Complexity of Subtype Satisfiability over Posets

Joachim Niehren[1], Tim Priesnitz[2], and Zhendong Su[3]

[1] INRIA Futurs, Lille, France
[2] Programming Systems Lab, Saarland University, Saarbrücken, Germany
[3] 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" [17], 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 [18]. 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 $t_1 \leq t_2$, to capture that the type t_1 is a *subtype* of t_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. Besides of type constants, subtype constraints may contain type variables and type constructors, such as the constraint $int \times x \leq x \times real$ that is equivalent to $int \leq x \leq real$.

Type variables are typically interpreted as trees built from type constants and type constructors. 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, 5, 9, 13, 14, 18, 23, 26, 32, 33], entailment [7, 11, 12, 19, 20, 24, 25, 28, 34], and first-order validity [16, 31]. In this paper, we close a number of problems on satisfiability.

If the type constants form a lattice then subtype satisfiability is well understood [14, 18, 23]. For general partially-ordered sets (posets), however, there exist only

	structural	non-structural
finite types	*PSPACE* (Frey, 1997 [8]) / *PSPACE*-hard (Tiuryn, 1992 [32])	*PSPACE*-complete ($\star$)
recursive types	*DEXPTIME* (Tiuryn and Wand, 1993 [33]) / *DEXPTIME*-hard ($\star$)	*DEXPTIME*-complete ($\star$)

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

partial answers. Tiuryn and Wand show that recursive structural satisfiability is in *DEXPTIME* [33]. Tiuryn shows that finite structural satisfiability is *PSPACE*-hard [32], and subsequently Frey shows that it is in *PSPACE* and thus *PSPACE*-complete [8]. Decidability and complexity of non-structural subtype satisfiability are open, for both finite and recursive types.

We summarize here the main contributions of 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, 4, 6], which is subsumed by the monadic second-order logic SnS of the complete infinite n-ary tree [27]. 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. New results of this paper are marked with "$\star$". In particular, 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. This settles a longstanding problem left open by Tiuryn and Wand in 1993 [33].

Due to space limitations, we omit some of the proofs. Interested readers can refer to the full paper [21] for more details.

2 Subtyping

In this section, we formally define satisfiability problems of subtype constraints.

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 and constants*). Each function symbol f has an associated $arity(f) \geq 0$, indicating the number of arguments that f expects. Symbols with arity zero are *type constants*. The signature fixes for all type constructors f and all positions $1 \leq i \leq arity(f)$ 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.

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.*: $\forall \pi \in dom(\tau) \forall i \in \mathbb{N} : \pi i \in dom(\tau) \leftrightarrow 1 \leq i \leq arity(\tau(\pi))$. 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:

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

For partial orders $\leq$, 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 with signatures Σ that provide the standard type constructors $\times$ and $\to$ and a poset $(B, \leq_B)$ of type constants, i.e., $\Sigma = B \cup \{\times, \to\}$. 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 $\to$ 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: $\Sigma = B \cup \{\times, \to\} \cup \{\bot, \top\}$. For the non-structural subtype order, besides the three structural rules earlier, there is an additional requirement: $\bot \leq \tau \leq \top$ for any $\tau \in tree_\Sigma$.

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 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 $pol(\pi)$ instead of $pol_\tau(\pi)$.

The signatures $\{\times\}$ and $\{\to\}$, for instance, are both uniform, while $\{\times,\to\}$ 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 $\tau_1,\tau_2 \in tree_\Sigma$:

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

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

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 $x,y,z \in V$. We assume w.l.o.g. that subtype constraints are *flat*, and subtype constraints φ over a signature Σ satisfy:

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

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 in V to trees over Σ. 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 $B \cup \{\times,\to\}$.

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 $B \cup \{\times,\to\} \cup \{\bot,\top\}$.

Uniform subtype satisfiability is 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.

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

Fig. 1. Syntax of PDL_n.

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 [27].

3.1 Other PDL Dialects

Propositional dynamic logic (*PDL*) over directed edge-labeled graphs goes back to Fischer and Ladner [6], 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, 10, 35] 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 [4, 15, 22], see [3] for a comparison. PDL over finite unranked ordered trees were proposed for computational linguistics applications [4] 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[4] A is given in Figure 1. Starting from some infinite set P of propositional variables $p \in P$, 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 $\dot{\vee}$, 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

[4] 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*$_n$.

$B ::= p_1 \wedge p_2 \mid \neg p \mid [i]p$ where $1 \leq i \leq n$
$C ::= p \mid [*](p \leftrightarrow B) \mid C_1 \wedge C_2$

Fig. 2. Syntax of flat core *PDL*$_n$.

formula in *PDL*$_n$ assigns Boolean values $0, 1$ to propositional variables in every node in $\{1, \ldots, n\}^*$, *i.e.*, $M : P \times \{1, \ldots, n\}^* \to \{0, 1\}$. 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: $M \models A$ 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: $\models A$ iff $\forall M.\, M \models A$. Two formulas A, A' are equivalent if $A \leftrightarrow A'$ is valid: $A \models\!\models A'$ iff $\models A \leftrightarrow A'$. For instance, $\langle i \rangle A \models\!\models [i]A$ holds for all $1 \leq i \leq n$ and all A, since all nodes of the n-ary tree have unique i successors.

Note that *PDL*$_n$ respects the substitution property: whenever $A_1 \models\!\models A_2$ then $A[A_1/A_2] \models\!\models A$. To see this note that if $A_1 \models\!\models A_2$ 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 theorem thus follows from the *DEXPTIME* upper bound for deterministic *PDL* [2, 10], which follows from 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 [35] that satisfiability of deterministic *PDL* is *DEXPTIME*-complete. This result clearly carries over to *PDL*$_n$.

An analysis of Spaan's proofs [30] reveals that nested $[*]$ modalities are not needed for *DEXPTIME*-hardness. But we can even do better, *i.e.*, restrict the language further.

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.*

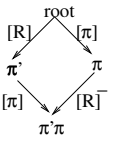
A proof is given in the full paper [21]. It is based on a new idea, by reduction to the emptiness of intersections of tree automata. This problem was shown *DEXPTIME*-hard by Seidl [29].

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

Table 3. Expressing uniform covariant subtype constraints in inverted *PDL*$_n$.

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.



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

Inverted flat core PDL$_n$ is defined in analogy to flat core *PDL*$_n$ except that all modalities are inverted.

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

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

We next 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 in polynomial time and preserve satisfiability (Propositions 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 interpreted over infinite n-ary trees into inverted *PDL*$_n$. The translation relies on ideas of Tiuryn and Wand [33], but it is simpler with modal logics as the target language. 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: $\mathit{wff}_V =_{\mathrm{df}} \bigwedge_{x \in V} [*] (\overset{+}{\bigvee}_{f \in \Sigma} p_{x=f})$. 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 $M_\alpha : P \times \{1, \ldots, n\}^* \to \Sigma$ 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 $p_1 \overset{+}{\vee} \ldots \overset{+}{\vee} p_n$ 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 $(p \wedge \neg p') \vee (\neg p \wedge p')$.

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 $\Sigma = \{\to\}$, where $\to$ 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 $[*](p \to [+]p')$ in inverted *PDL*$_n$, which states that whenever p holds at some node then p' holds in all 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 $[*](p_c \to [+]p)$ for all constants c, but this is impossible in inverted *PDL*$_n$.

This is not a problem for uniform signatures where every tree is completely n-ary, so that we do not need to express tree domains, as long as we are considering satisfiability. Unfortunately, however, the same technique does not extend to entailment and other fragments of first-order logic with negation.

$\mathit{all}\text{-}c(x)$	$=_{\mathrm{df}}$	$x{=}c(x,\ldots,x)$ for some $c \in \Sigma(n)$
$\mathit{all}\text{-}\mathit{bool}(x)$	$=_{\mathrm{df}}$	$\exists y \exists z.\ \mathit{all}\text{-}0(x) \wedge x{\leq}y{\leq}z \wedge \mathit{all}\text{-}1(z)$
$\mathit{all}\text{-}\overline{\mathit{bool}}(x)$	$=_{\mathrm{df}}$	$\exists y \exists z.\ \mathit{all}\text{-}\bar{1}(x) \wedge x{\leq}y{\leq}z \wedge \mathit{all}\text{-}\bar{0}(z)$
$\mathit{upper}(x,y)$	$=_{\mathrm{df}}$	$\exists z.\ x{\leq}z \wedge y{\leq}z$
$\mathit{lower}(x,y)$	$=_{\mathrm{df}}$	$\exists z.\ z{\leq}x \wedge z{\leq}y$
$y{=}\overline{x}$	$=_{\mathrm{df}}$	$\mathit{all}\text{-}\mathit{bool}(x) \wedge \mathit{all}\text{-}\overline{\mathit{bool}}(y) \wedge \mathit{upper}(x,y) \wedge \mathit{lower}(x,y)$
$\mathit{all}(p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4)$	$=_{\mathrm{df}}$	$\exists z.\ \bigwedge_{1\leq i\leq 4} \mathit{all}\text{-}\mathit{bool}(X_{p_i})$
		$\wedge \mathit{lower}(z, X_{p_1}) \wedge \mathit{upper}(z, \overline{X_{p_2}})$
		$\wedge \mathit{lower}(z, \overline{X_{p_3}}) \wedge \mathit{upper}(z, X_{p_4})$
$\mathit{all}(p_1 \vee p_2)$	$=_{\mathrm{df}}$	$\exists X_q.\ \mathit{all}(p_1 \vee p_2 \vee \neg q \vee \neg q) \wedge \mathit{all}\text{-}1(X_q)$

Table 4. Boolean operations expressed by subtype constraints.

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 [32], while the idea to lift this encoding to *PDL*$_n$ is new.

Let C be a formula of inverted flat core *PDL*$_n$. We aim to find a subtype constraints $[\![C]\!]^{-1}$ which preserves satisfiability. The critical point is how to translate *PDL*$_n$'s negation since it is absent in our target language of uniform subtype constraints.
We work around by constructing a uniform subtype constraints with function symbols ordered in a crown: $\Sigma(n) = \{0, \bar{0}, 1, \bar{1}\}$.

All function symbols have arity n and satisfy $x \leq_{\Sigma(n)} y$ for all $x \in \{0, \bar{1}\}$, $y \in \{1, \bar{0}\}$. The symbols 0 and 1 model *PDL*$_n$'s underlying boolean lattice $\mathit{bool} = \{\mathit{0}, \mathit{1}\}$; the additional two symbols are introduced to define negation by $\mathit{neg}(c) = \overline{c}$ for $c \in \mathit{bool}$.

Next, Table 4 shows how to define *neg* by a subtype constraint. For every propositional variable p we introduce a new type variables X_p in the subtype constraint we are constructing to.

The subtype constraint $\mathit{all}\text{-}c(x)$ holds for the unique trees that is completely labeled by some $c \in \Sigma(n)$. The subtype constraint $\mathit{all}\text{-}\mathit{bool}(x)$ holds for trees that are labeled in bool. The constraints $\mathit{lower}(x,y)$ and $\mathit{upper}(x,y)$ require the existence of lower and upper bounds respectively for trees x and y. These bounds are used to define the diagonal pairs $y{=}\overline{x}$ in the crown.

Lemma 2. $y = \overline{x} \models \forall \pi.\ (x(\pi) = 0 \wedge y(\pi) = \bar{0}) \vee (x(\pi) = 1 \wedge y(\pi) = \bar{1})$.

Proof. Since x is a tree labeled in bool, all nodes π satisfy $\alpha(x)(\pi){=}0$ or $\alpha(x)(\pi){=}1$. In the first case (the second is analogous) the constraint $\mathit{lower}(x,y)$ entails $\alpha(y)(\pi){\neq}\bar{1}$. Since y is a $\overline{\mathit{bool}}$ tree, $\alpha(y)(\pi){=}\bar{0}$.

Solutions of subtype constraints are variable assignments $\alpha : P \to \{1,\ldots,n\}^* \to \Sigma(n)$. For variable assignments α into trees over Booleans, we define corresponding *PDL*$_n$-models $M_\alpha : P \times \{1,\ldots,n\}^* \to \mathit{bool}$ by $M_\alpha(p,\pi) = \alpha(X_p)(\pi)$.

$$
\begin{array}{rcl}
[\![p]\!]^{-1} & =_{\mathrm{df}} & \exists x_1 \ldots \exists x_m.\ \mathit{all\text{-}bool}(X_p) \wedge X_p{=}1(x_1, \ldots, x_m) \\
[\![[*]\,(p \leftrightarrow [i]^- q)]\!]^{-1} & =_{\mathrm{df}} & \mathit{all\text{-}bool}(X_p) \wedge \mathit{all\text{-}bool}(X_q) \\
 & & \wedge\, \exists x_1 \ldots \exists x_m.(0(x_1, ..., x_m){\leq}X_q{\leq}1(x_1, ..., x_m) \wedge X_p{=}x_i) \\
[\![[*]\,(p \leftrightarrow \neg q)]\!]^{-1} & =_{\mathrm{df}} & \mathit{all}(p \vee q) \wedge \mathit{all}(\neg p \vee \neg q) \\
[\![[*]\,(p \leftrightarrow (q_1 \wedge q_2))]\!]^{-1} & =_{\mathrm{df}} & \mathit{all}(\neg p \vee q_1) \wedge \mathit{all}(\neg p \vee q_2) \wedge \mathit{all}(p \vee \neg q_1 \vee \neg q_2) \\
[\![C_1 \wedge C_2]\!]^{-1} & =_{\mathrm{df}} & [\![C_1]\!]^{-1} \wedge [\![C_2]\!]^{-1}
\end{array}
$$

Table 5. Inverted core flat PDL_n in subtype constraints.

Lemma 3. *Let A be the Boolean formula $p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4$. For all variable assignments α to trees over $\Sigma(n)$, $\alpha \models \mathit{all}(A)$ if and only if M_α is defined and $M_\alpha \models [*]A$.*

The lemma relies on a non-trivial property of the crown poset. For all $p_1, p_2, p_3, p_4 \in \mathit{bool}$:

$$
p_1 \vee p_2 \vee \neg p_3 \vee \neg p_4 \ \models\!\mid\ \exists z \in \{0, 1, \bar{0}, \bar{1}\}.\ \mathit{lower}(z, p_1) \wedge \mathit{upper}(z, \overline{p_2}) \wedge \mathit{lower}(z, \overline{p_3}) \wedge \mathit{upper}(z, p_4)
$$

We illustrate the claim for $p_3 = p_4 = 1$ where the left hand side is equivalent to $p_1 \vee p_2$. The conjunction of the last two literals becomes $\mathit{lower}(z, \bar{1}) \wedge \mathit{upper}(z, 1)$ which is equivalent to $z \in \{1, \bar{1}\}$. The first two literals with $p_1 = p_2 = 0$ yield:

$$
\mathit{lower}(z, 0) \Rightarrow z \neq \bar{1} \quad \text{and} \quad \mathit{upper}(z, \bar{0}) \Rightarrow z \neq 1
$$

Thus, the complete conjunction is unsatisfiable with $p_1 = p_2 = 0$. Conversely, if $p_1 = 1$ then we can choose $z = \bar{1}$ since $\mathit{upper}(\bar{1}, \overline{p_2})$ holds for all $p_2 \in \mathit{bool}$. Similarly, if $p_2 = 1$ then we can choose $z = 1$ since $\mathit{lower}(1, p_1)$ for all $p_1 \in \mathit{bool}$.

The back translation $[\![C]\!]^{-1}$ of inverted flat core PDL_n into subtype constraints is shown in Table 5. All Boolean formulas used there can be expressed by $p_1 \wedge p_2 \wedge \neg p_3 \wedge \neg p_4$ which we know how to encode.

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

For $n = 0$, subtype constraints become ordering constraints for a poset, 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 posets [26].

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 constraints for subtype orders with a single nonconstants type constructor that we call 1-subtype orders.

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 unique type constructor, the partial order on constants $(B, \leq_B)$, 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. Given a constraint φ over Σ, we construct the *shape constraint* of φ, $sh(\varphi)$, by replacing each constant in φ with an arbitrary, fixed constant $\star \in \Sigma$, and each inequality with an equality:

$$\begin{array}{rlrl} sh(x{=}f(x_1,x_2)) &=_{\mathrm{df}} x{=}f(x_1,x_2), & sh(x{\leq}y) &=_{\mathrm{df}} x{=}y, \\ sh(\varphi_1{\wedge}\varphi_2) &=_{\mathrm{df}} sh(\varphi_1){\wedge}sh(\varphi_2), & sh(x{=}c) &=_{\mathrm{df}} x{=}\star \end{array}$$

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

Next, we handle contravariance. Consider a signature $\Sigma = B \cup \{\times, \to\}$. 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 $\to$, *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 $\to$. Thus, f is co-variant in all arguments except the third one.

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

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

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 [8] that relates the shape of a solution of φ to that of a solution of $sh(\varphi)$.

Lemma 5 (Frey [8]). *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 $\top$:

$$s(x{=}\bot) =_{\mathrm{df}} x{=}\bot, \quad s(x{=}\top) =_{\mathrm{df}} x{=}\top$$

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 1. *Let φ be a constraint over Σ, and c be an arbitrary and fixed constant. Define the* weak satisfiability constraint $ws(\varphi)$ *as:*

$$\begin{array}{lll} ws(x{=}f(x_1,x_2)) =_{\mathrm{df}} x{=}f(x_1,x_2), & ws(x{\leq}y) =_{\mathrm{df}} x{\leq}y, & ws(x{=}\bot) =_{\mathrm{df}} x{=}\bot, \\ ws(\varphi_1{\wedge}\varphi_2) =_{\mathrm{df}} ws(\varphi_1){\wedge}ws(\varphi_2), & ws(x{=}b) =_{\mathrm{df}} x{=}c, & ws(x{=}\top) =_{\mathrm{df}} x{=}\top \end{array}$$

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 [23]. 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.*

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: (1) $s(\Sigma) =_{\mathrm{df}} \Sigma$; (2) $\forall f \in s(\Sigma).arity_{s(\Sigma)}(f) =_{\mathrm{df}} \max$; and (3) $\leq_{s(\Sigma)} =_{\mathrm{df}} \leq_{\Sigma}$, where $\max$ is the maximal arity of Σ.

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

$$
\begin{array}{ll}
s(x{=}f(x_1,\ldots,x_{\max})) =_{\mathrm{df}} x{=}f(x_1,\ldots,x_{\max}), & s(x{=}b) =_{\mathrm{df}} x{=}b(y_1,\ldots,y_{\max}),\\
s(\varphi_1{\wedge}\varphi_2) =_{\mathrm{df}} s(\varphi_1){\wedge}s(\varphi_2), & s(x_1{\leq}x_2) =_{\mathrm{df}} x_1{\leq}x_2,\\
s(x{=}\bot) =_{\mathrm{df}} x{=}\bot(u_1,\ldots,u_{\max}), & s(x{=}\top) =_{\mathrm{df}} x{=}\top(v_1,\ldots,v_{\max})
\end{array}
$$

where the y_i's, u_i's, and v_i's are fresh variables, and the last two rules 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 $s(\Sigma)$.*

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 satisfiability to structural and non-structural 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 $\overline{\overline{s}}$ 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 $\overline{s}(\Sigma)$ by including symbols in Σ as constants and adding $\rightarrow$: (1) $\overline{s}(\Sigma) =_{\mathrm{df}} \Sigma \cup \{\rightarrow\}$; (2) $\forall g \in \Sigma.arity_{\overline{s}(\Sigma)}(g) =_{\mathrm{df}} 0$; (3) $arity_{\overline{s}(\Sigma)}(\rightarrow) =_{\mathrm{df}} 2$; and (4) $\leq_{\overline{s}(\Sigma)} =_{\mathrm{df}} \leq_{\Sigma}$. We now translate a subtype constraint φ over Σ to a constraint $\overline{s}(\varphi)$ over $\overline{s}(\Sigma)$:

$$
\begin{array}{rl}
\overline{s}(x{=}g(x_1,x_2,x_3)) =_{\mathrm{df}} & x{=}(x_3 \rightarrow x_2) \rightarrow (x_1 \rightarrow g)\\
\overline{s}(x_1 \leq x_2) =_{\mathrm{df}} & x_1 \leq x_2\\
\overline{s}(\varphi_1 \wedge \varphi_2) =_{\mathrm{df}} & \overline{s}(\varphi_1) \wedge \overline{s}(\varphi_2)
\end{array}
$$

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 $\overline{s}(x{=}g(x_1,x_2,x_3))$. Thus, $\overline{s}$ preserves all polarities.

In our second step, we force every variable to be mapped to a fixed, infinite shape. We extend $\overline{s}(\Sigma)$ to $\overline{\overline{s}}(\Sigma)$ with four new constants a_1, a_2, a_3, and a_4 with the following ordering: $a_1 \leq c \leq a_3 \ \wedge \ a_2 \leq c \leq a_4$, for all constants $c \in \overline{s}(\Sigma)$. We define $\overline{\overline{s}}(\varphi)$ as the conjunction of $\overline{s}(\Sigma)$ 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 $x \in V(\overline{s}(\Sigma))$;
(2) $\bigwedge_{i=1,2,3,4} u_i{=}(u_i \rightarrow u_i) \rightarrow (u_i \rightarrow a_i)$

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 9. *If $\bar{\bar{s}}(\varphi)$ is interpreted over any (non-)structural signature $\bar{\bar{s}}(\Sigma)$ or $\bar{\bar{s}}(\Sigma) \cup \{\bot, \top\}$, any variable assignment $\alpha \models \bar{\bar{s}}(\varphi)$ satisfies for all paths $\pi \in (1(1 \cup 2) \cup 21)^*$:*

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

Lemma 10. *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 $map : tree_\Sigma \rightarrow tree_{\bar{\bar{s}}(\Sigma)}$ on trees for all $g \in \Sigma$:

$$\begin{aligned} map(g(\tau_1, \tau_2, \tau_3)) =_{\mathrm{df}} \quad & (map(\tau_3) \rightarrow map(\tau_2)) \\ & \rightarrow (map(\tau_1) \rightarrow g) \end{aligned}$$

With that it can be easily verified that if there exists a solution $\alpha \models \varphi$ over an uniform signature Σ then $map(\alpha) \models \bar{\bar{s}}(\varphi)$ 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 $\beta = map^{-1}(\alpha)$ according to the shape of any solution of $\bar{\bar{s}}(\varphi)$ stated in Lemma 9. 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 9).

6 Finite Subtype Satisfiability over Posets

Finite structural subtype satisfiability was shown *PSPACE*-complete by Tiuryn [32] and Frey [8]. Here, we establish the same complexity for the non-structural case.

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

The analogous result for the structural case was shown by Tiuryn [32]). To lift this result, we show how to reduce non-structural to structural subtype satisfiability.

Lemma 11. *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: $s(\Sigma) =_{\mathrm{df}} \Sigma \cup \{\bot, \top, a_1, a_2, a_3, a_4\}$ with the a_i's being four new constants. In addition, $\leq_{s(\Sigma)} =_{\mathrm{df}} \leq_\Sigma \cup \{(a_1, c), (a_2, c), (c, a_3), (c, a_4) \mid c \in \Sigma_0\}$.

Let φ be a constraint over Σ. We construct $s(\varphi)$ over $s(\Sigma)$. Consider φ's shape constraint $sh(\varphi)$ (see Section 5.1). If $sh(\varphi)$ is not unifiable, we simply let $s(\varphi) =_{\mathrm{df}} \top \leq \bot$. 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 11 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 $s(\Sigma)$.

By adapting the proof of Frey [8], we can show membership in *PSPACE*, and thus we have the following theorem. For an alternative proof of using *K-normal modal logic*, please refer to the full paper [21].

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

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 [31], were a fragment of S2S, which is decidable [27].

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, B. Gaiffe, and M. Marx. Variable free reasoning on finite trees. In *Proceedings of Mathematics of Language*, pages 17–30, 2003.
4. P. Blackburn and W. Meyer-Viol. Linguistics, logic, and finite trees. *Logic Journal of the IGPL*, 2:3–29, 1994.
5. J. Eifrig, S. Smith, and V. Trifonov. Sound polymorphic type inference for objects. In *OOPSLA*, pages 169–184, 1995.
6. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

7. C. Flanagan and M. Felleisen. Componential set-based analysis. *ACM Transactions on Programming Languages and Systems*, 21(2):370–416, 1999.
8. A. Frey. Satisfying subtype inequalities in polynomial space. *Theoretical Computer Science*, 277:105–117, 2002.
9. Y. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73(2):155–175, 1990.
10. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. The MIT Press, 2000.
11. F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *LICS*, pages 362–372, 1997.
12. F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *ICALP*, pages 616–627, 1998.
13. M. Hoang and J. Mitchell. Lower bounds on type inference with subtypes. In *POPL*, 176–185, 1995.
14. D. Kozen, J. Palsberg, and M. I. Schwartzbach. Efficient inference of partial types. *Journal of Computer and System Sciences*, 49(2):306–324, 1994.
15. M. Kracht. Inessential features. In *Logical Aspects of Computational Linguistics*, volume 1328, pages 43–62, 1997.
16. V. Kuncak and M. Rinard. Structural subtyping of non-recursive types is decidable. In *LICS*, pages 96–107, 2003.
17. J. Mitchell. Coercion and type inference. In *POPL*, pages 175–185, 1984.
18. J. C. Mitchell. Type inference with simple subtypes. *The Journal of Functional Programming*, 1(3):245–285, 1991.
19. J. Niehren and T. Priesnitz. Entailment of non-structural subtype constraints. In *Asian Computing Science Conference*, pages 251–265, 1999.
20. J. Niehren and T. Priesnitz. Non-structural subtype entailment in automata theory. *Information and Computation*, 186(2):319–354, 2003.
21. J. Niehren, T. Priesnitz, and Z. Su. Complexity of subtype satisfiability over posets. Available at `http://www.cs.ucdavis.edu/~su/publications/poset.pdf`, 2004.
22. A. Palm. Propositional tense logic for trees. In *Proceedings of the Sixth Meeting on Mathematics of Language*, pages 74–87, 1999.
23. J. Palsberg, M. Wand, and P. O'Keefe. Type Inference with Non-structural Subtyping. *Formal Aspects of Computing*, 9:49–67, 1997.
24. F. Pottier. Simplifying subtyping constraints. In *ICFP*, pages 122–133, 1996.
25. F. Pottier. *Type inference in the presence of subtyping: from theory to practice*. PhD thesis, INRIA, 1998.
26. V. Pratt and J. Tiuryn. Satisfiability of inequalities in a poset. *Fundamenta Informaticae*, 28:165–182, 1996.
27. M. O. Rabin. Decidability of second-order theories and automata on infinte trees. *Trans. Amer. Math. Soc.* , 141:1–35, 1969.
28. J. Rehof. *The Complexity of Simple Subtyping Systems*. PhD thesis, DIKU, 1998.
29. H. Seidl. Haskell overloading is DEXPTIME-complete. *Information Processing Letters*, 52(2):57–60, 1994.
30. E. Spaan. *Complexity of Modal Logics*. PhD thesis, University of Amsterdam, 1993.
31. Z. Su, A. Aiken, J. Niehren, T. Priesnitz, and R. Treinen. First-order theory of subtyping constraints. In *POPL*, pages 203–216, 2002.
32. J. Tiuryn. Subtype inequalities. In *LICS*, pages 308–315, 1992.
33. J. Tiuryn and M. Wand. Type reconstruction with recursive types and atomic subtyping. In *Theory and Practice of Software Development*, 686-701,'93.
34. V. Trifonov and S. Smith. Subtyping constrained types. In *SAS*, pages 349–365, 1996.
35. M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal Computer & System Science*, 32(2):183–221, 1986.