Entailment of Non-Structural Subtype Constraints

Joachim Niehren Tim Priesnitz

Programming Systems Lab, Universität des Saarlandes, Saarbrücken, Germany
www.ps.uni-sb.de/~{niehren,tim}

Abstract. Entailment of subtype constraints was introduced for constraint simplification in subtype inference systems. Designing an efficient algorithm for subtype entailment turned out to be surprisingly difficult. The situation was clarified by Rehof and Henglein who proved entailment of structural subtype constraints to be coNP-complete for simple types and PSPACE-complete for recursive types. For entailment of non-structural subtype constraints of both simple and recursive types they proved PSPACE-hardness and conjectured PSPACE-completeness but failed in finding a complete algorithm. In this paper, we investigate the source of complications and isolate a natural subproblem of non-structural subtype entailment that we prove PSPACE-complete. We conjecture (but this is left open) that the presented approach can be extended to the general case.

Keywords: subtyping, constraints, entailment, automata.

1 Introduction

Subtyping is a natural concept in programming. This observation has motivated the design of programming languages featuring a system for subtype inference [8, 11, 2, 6, 18]. Simplification of typings turned out to be the key issue in what concerns the complexity of subtype inference systems [7, 19, 17]. Several authors proposed to simplify typings based on algorithms for *subtype entailment*, i.e. entailment of *subtype constraints* [22, 16]. First approaches towards subtype entailment seem to presuppose [22, 16] that the problem could be solved efficiently. But finding an efficient algorithm for subtype entailment turned out to surprisingly difficult [9, 20, 10, 18]. And in fact, it still remains open whether subtype entailment is decidable, even if restricted to an inexpressive type languages. The most prominent open problem is the decidability of entailment between non-structural subtype constraints.

Types. A simple type is finite *tree* built from a signature Σ of function symbols (i.e. a ground term over Σ). A recursive type is an infinite tree over Σ. Most typically, Σ contains the constants *int* and *real* and the binary function symbol $\times$ for pairing. The type of a pair of integers, for instance, is the finite tree $\mathit{int}\times\mathit{int}$. The signature Σ typically also provides constants $\bot$ and $\top$ for the *least type* and the *greatest type*.

Many further types are of interest for programming: contra-variant function types $\tau\to\tau'$, record types $\{f_1{:}\tau_1, \ldots, f_n{:}\tau_n\}$, intersection and union types, and polymorphic types. These types fall out of the scope of the present paper. In order to keep subtype entailment simple, we restrict ourself to types that are finite or infinite trees built from a signature $\Sigma \subseteq \{\mathit{int}, \mathit{real}, \times, \bot, \top\}$.

Subtyping. When considering types as trees, subtyping becomes a partial order on trees. A typical subtype relationship is $\mathit{int} \leq \mathit{real}$ which states that every integer can be used as a real (its relevance is discussed in depth in [12]). The former subtype relationship induces $\mathit{int}\times\mathit{int} \leq \mathit{real}\times\mathit{real}$ which means that every pair of integers can

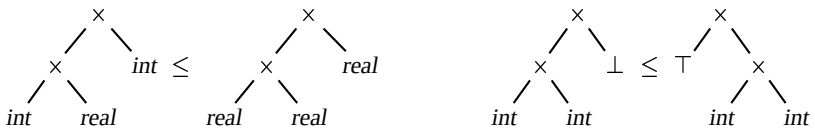


Fig. 1. Structural versus non-structural subtyping

be used as a pair of reals. Both relationships are *structural* in that they relate trees of the same shape. Subtyping becomes *non-structural* in the presence of a least type $\bot$ or greatest type $\top$, since $\bot \leq \tau$ and $\tau \leq \top$ hold for all types τ. The difference between structural and non-structural subtyping is illustrated in Figure 1.

Subtype Entailment. A subtype constraint is a logical description of types whose interpretation is based on the subtype relation. We assume a set of *type variables* ranged over by x, y, z. A *subtype constraint* ψ is a conjunction of ordering constraints $t \leq t'$ between terms t, t' built from variables and function symbols in Σ. *Subtype entailment* is the problem to decide whether an implication $\psi \rightarrow x \leq y$ is valid in the structure of trees, i.e. whether $\psi \models x \leq y$ holds. Four cases are to be distinguished: either we interpret over finite trees (simple types) or else over possibly infinite trees (recursive types); either we consider *non-structural subtyping* where $\bot, \top \in \Sigma$ or else *structural subtyping* where $\bot, \top \notin \Sigma$. The differences between these cases can be illustrated at the following example.

$$x \leq y \times y \wedge x \times x \leq y \;\models\; x \leq y$$

First, we consider structural subtyping with the signature $\Sigma = \{\mathit{int}, \mathit{real}, \times\}$. For finite trees, the left hand side is unsatisfiable and thus entailment holds. For infinite trees, there exists a unique solution where both x and y are mapped to the complete binary tree $\mu z.z \times z$; thus entailment holds again. Second, we consider non-structural subtyping with the signature $\Sigma = \{\top, \bot, \mathit{int}, \mathit{real}, \times\}$. There are many more solutions than for structural subtyping. For instance, the variable assignment mapping x to $\bot \times (\top \times \bot)$ and y to $\top \times (\bot \times \top)$ is a solution of $x \leq y \times y \wedge x \times x \leq y$ which contradicts entailment of $x \leq y$ for both finite and infinite trees.

Open Problem. Early algorithms for subtype entailment were incomplete [22, 16, 18]. The situation was clarified by Henglein and Rehof who determined the complexity of structural subtype entailment: for simple types, it is coNP-complete [9] and for recursive types it is PSPACE-complete [10]. However, the complexity of non-structural subtype entailment remains open; it is at least PSPACE-hard, both, for finite and infinite trees [20, 10]. It is even unknown whether non-structural subtype entailment is decidable. Nevertheless, Rehof conjectures PSPACE-completeness (see Conjecture 9.4.5 of [20]).

Contribution. In this paper, we investigate the source of complications underlying non-structural subtype entailment. To this purpose, we introduce an extension of finite automata that we call *P-automata* and illustrate the relevance of P-automata to non-structural subtype entailment at a sequence of examples. P-automata can recognize nonregular and even non-context-free languages, as we show. This fact yields new insights into the expressiveness of non-structural subtype entailment.

Based on the insight gained by P-automata, we isolate a fragment of non-structural subtype constraints for which we prove decidability of entailment. We consider the

signature $\{\bot, \top, \times\}$ and both cases, finite and possibly infinite trees respectively. The only restriction we require is that $\bot$ and $\top$ are not supported syntactically, i.e. that constraints such as $z \times \top \leq z$ are *not* cannot be written.

The algorithm we present is based on a polynomial time reduction to the universality problem of finite automata (which is PSPACE-complete). The idea is that more general P-automata are not needed for entailment of the restricted language. Our algorithm solves an entailment problem in PSPACE that was proved PSPACE-hard by Rehof and Henglein [10]. Its correctness proof is technically involved; it shows why nonregular sets of words – as recognized by P-automata – can be safely ignored.

Related Entailment Problems. Several entailment problems for constraint languages describing trees are considered in the literature. Two of them were shown PSPACE-complete in [14, 15]. The common property of these PSPACE-complete entailment problems is that entailment depends on properties of regular sets of words in the constraint graph. In contrast, nonregular sets have to be taken into account for non-structural subtype entailment.

In feature logics, several languages for describing feature trees (i.e. records types) have been investigated for entailment. Entailment for equality constraints over feature trees can be decided in quasi linear time [1, 21]. Ordering constraints over feature trees [5, 4] can be considered as record subtype constraints. Entailment of ordering constraints over feature trees can be solved in cubic time [13]. However, entailment with existential quantification is PSPACE-complete again [14].

Entailment has also been considered for set constraints (i.e. constraints for union and intersection types). Entailment of set constraints with intersection is proved DEXPTIME-complete in [3] for an infinite signature. Entailment of atomic set constraints [15] is proved PSPACE-complete in case of an infinite signature and DEXPTIME-hard for a finite signature.

2 Non-Structural Subtype Constraints

We assume a signature Σ which provides function symbols denoted by f each of which has a fixed arity $\mathsf{ar}(f) \geq 0$. We require that Σ contains the constants $\bot$ and $\top$, i.e. $\mathsf{ar}(\bot) = \mathsf{ar}(\top) = 0$. We also assume an infinite set of variables ranged over by x, y, z, u, v, w.

Paths and Trees. A *path* is a word of natural numbers $n \geq 1$ that we denote by π, o, ϱ, or σ. The *empty path* is denoted by ε and the free-monoid *concatenation* of paths π and π' by juxtaposition $\pi\pi'$, with the property that $\varepsilon\pi = \pi\varepsilon = \pi$. A *prefix* of a path π is a path π' for which there exists a path π'' such that $\pi = \pi'\pi''$. A *proper prefix* of π is a prefix of π but not π itself. If π' is a prefix of π then we write $\pi' \leq \pi$ and if π' is a proper prefix of π then we write $\pi' < \pi$. The *prefix closure* of a set of path Π is denoted as $pr(\Pi)$, i.e. $pr(\Pi) = \{\pi \mid \text{exists } \pi' \in \Pi : \pi \leq \pi'\}$ and its *proper prefix closure* with $pr_{\neq}(\Pi)$, i.e. $pr_{\neq}(\Pi) = \{\pi \mid \text{exists } \pi' \in \Pi : \pi < \pi'\}$.

A *tree* τ is a pair (D, L) where D is a tree domain, i.e. a non-empty prefixed-closed set of paths, and $L : D \to \Sigma$ a (total) function determining the labels of τ. We denote the tree domain of a tree τ by D_τ and its labeling function with L_τ. We require that trees τ are *arity consistent*: for all paths $\pi \in D_\tau$ and natural numbers i: $1 \leq i \leq \mathsf{ar}(L_\tau(\pi))$ iff $\pi i \in \mathcal{D}_\tau$. A tree is *finite* if its tree domain is finite and *infinite*

otherwise. We denote the set of all finite trees with $\mathit{Tree}_\Sigma^{\mathit{fin}}$ and the set of all trees with Tree_Σ.

Non-Structural Subtyping. Let $\leq_L$ be the least (reflexive) partial order on function symbols of Σ which satisfies for all $f \in \Sigma$:

$$\bot \ \leq_L \ f \ \leq_L \ \top$$

We define *non-structural subtyping* as a partial order $\leq$ on trees such that $\tau_1 \leq \tau_2$ holds for trees τ_1, τ_2 iff for all paths $\pi \in D_{\tau_1} \cap D_{\tau_2}$ it holds that $L_{\tau_1}(\pi) \leq_L L_{\tau_2}(\pi)$.

Let NS_Σ be the structure with signature $\Sigma \cup \{\leq\}$ whose domain is the set Tree_Σ. Function symbols in Σ are interpreted as tree constructors and the relation symbol $\leq$ as non-structural subtyping (which we also denote by $\leq$). The structure $\mathit{NS}_\Sigma^{\mathit{fin}}$ is the restriction of NS_Σ to the domain of finite trees $\mathit{Tree}_\Sigma^{\mathit{fin}}$.

A *term* t is either a variable or a construction $f(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are terms, $f \in \Sigma$, and $n = \mathsf{ar}(f)$. Of course, $\bot$ and $\top$ are terms since they are constants in Σ. A *non-structural subtype constraint* over Σ is a conjunction of ordering constraints $t_1 \leq t_2$. We consider two cases for their interpretation, either the structure NS_Σ or the structure $\mathit{NS}_\Sigma^{\mathit{fin}}$. We mostly use flattened constraints ψ of the following form:

$$\psi \ ::= \ x{=}f(x_1, \ldots, x_n) \ \mid \ x{\leq}y \ \mid \ \psi \wedge \psi' \qquad (f \in \Sigma,\ \mathsf{ar}(f) = n)$$

The omission of nested terms does not restrict the expressiveness of entailment. Terms on the left hand side of an entailment judgment can be flattened by introducing new variables for all subterms. Furthermore, $\psi \models t_1 \leq t_2$ is equivalent to $\psi \wedge t_1 \leq x \wedge y \leq t_2 \models x \leq y$ where x, y are fresh variables.

Satisfiability and Entailment. Let Φ denote a first-order formula built from ordering constraints with the usual first-order connectives and let $\mathcal{V}(\Phi)$ be the set of *free variables* in Φ. We write Φ' *in* Φ if there exists Φ'' such that $\Phi = \Phi' \wedge \Phi''$ up to associativity and commutativity of conjunction. Suppose that $\mathcal{A}$ is a structure with signature $\Sigma \cup \{\leq\}$. A *solution of* Φ in $\mathcal{A}$ is a variable assignment α into the domain of $\mathcal{A}$ such that Φ evaluates to true under $\mathcal{A}$ and α. We call Φ *satisfiable in* $\mathcal{A}$ if there exists a solution for Φ in $\mathcal{A}$. A formula Φ is *valid in* $\mathcal{A}$ if all variable assignments into the domain of $\mathcal{A}$ are solutions of Φ. A formula Φ *entails* Φ' in $\mathcal{A}$, written $\Phi \models_{\mathcal{A}} \Phi'$ if $\Phi \to \Phi'$ is valid in $\mathcal{A}$.

Restricted Language. Let Σ_2 be the signature $\{\bot, \top, \times\}$ where $\times$ is a binary function symbol. A *restricted subtype constraints* φ has the form:

$$\varphi \ ::= \ u{=}u_1{\times}u_2 \ \mid \ u_1{\leq}u_2 \ \mid \ \varphi_1 \wedge \varphi_2$$

The following restrictions are crucial for entailment as we will discuss later on: 1) The constraints $x{=}\bot$ and $x{=}\top$ are excluded. 2) The signature Σ_2 does not contain a unary function symbol. Nevertheless, the restricted entailment problem is not trivial. It is not difficult to see and proved by Rehof and Henglein [10] that entailment of the restricted language can express universality of non-deterministic finite automata; thus:

Proposition 1 (Hardness). *Non-structural subtype entailment for the restricted constraint language is PSPACE hard for both structures* NS_{Σ_2} *and* $\mathit{NS}_{\Sigma_2}^{\mathit{fin}}$.

S0	if $x \in \mathcal{V}(\psi)$ then $x \leq x$ *in* ψ
S1	if $x \leq y$ *in* ψ and $y \leq z$ *in* ψ then $x \leq z$ *in* ψ
S2	if $x = f(x_1, \ldots, x_n) \wedge x \leq y \wedge y = f(y_1, \ldots, y_n)$ *in* ψ then $\bigwedge_{i=1}^{n} x_i \leq y_i$ *in* ψ
S3	not $x = f_1(x_1, \ldots, x_n)$ *in* ψ, $x \leq y$ *in* ψ, $y = f_2(y_1, \ldots, y_n)$ *in* ψ, and $f_1 \not\leq_L f_2$
S4	not $\bigwedge_{i=1}^{n} x_i = f(\ldots, y_{i+1}, \ldots) \wedge y_{i+1} = x_{i+1}$ *in* ψ where $n \geq 1$ and $x_1 = x_{n+1}$

Table 1. Closure and Clash-freeness Properties: *S0-S3* for NS_Σ and *S0-S4* for NS_Σ^{fin}

We next recall a closure algorithm from [10] which decides the satisfiability of (unrestricted) non-structural subtype constraints ψ over an arbitrary signature Σ. In Table 1, a set of properties *S0-S4* is given. The properties for NS_Σ^{fin} and NS_Σ differ only in an additional occurs check for the case of finite trees (*S4*). Reflexivity and transitivity of subtype are required by (*S0*) and (*S1*). The decomposition property of subtyping is stated in (*S2*), and clash-freeness for labeling in (*S3*).

We call a (flattened) constraint ψ *closed* it it satisfies *S0-S2*. Properties *S0-S2* can also be considered as a saturation algorithm which computes the *closure* of a (flattened) constraint ψ in cubic time. A constraint ψ is *clash-free* for NS_Σ it it satisfies *S3* and for NS_Σ^{fin} if it satisfies *S3-S4*.

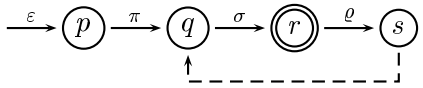
Proposition 2 (Satisfiability). *A constraint is satisfiable in NS_Σ (resp. NS_Σ^{fin}) if its closure is clash-free for NS_Σ (resp. NS_Σ^{fin}).*

3 P-Automata

We now present the notion of a *P-automaton* on which we will base our analysis of subtype entailment. A P-automaton is an extension of a finite automaton with a new kind of edges. Let $\mathcal{A} = (A, Q, I, F, \Delta)$ be a finite automaton with *alphabet* A, *states* Q, *initial states* I, *final states* F, and *transition edges* Δ. We write $\mathcal{A} \vdash p \xrightarrow{\pi} q$ for states $p, q \in Q$ and $\pi \in A^*$ if the automaton $\mathcal{A}$ permits a transition from p to q while reading π. Thus, $\mathcal{A}$ recognizes the language $\mathcal{L}(\mathcal{A}) = \{\pi \in A^* \mid \mathcal{A} \vdash p \xrightarrow{\pi} q,\ p \in I, q \in F\}$.

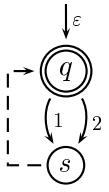
Definition 3. A *P-automaton* $\mathcal{P}$ is a pair $(\mathcal{A}, P)$ consisting of a finite automaton $\mathcal{A} = (A, Q, I, F, \Delta)$ and a set of *P-edges* $P \subseteq Q \times Q$ between the states of $\mathcal{A}$. The P-automaton $\mathcal{P}$ recognizes the language $\mathcal{L}(\mathcal{P}) \subseteq A^*$ given by:

$$\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{A}) \cup \bigcup \{\pi(\sigma\varrho)^*\sigma \mid \mathcal{A} \vdash p \xrightarrow{\pi} q \xrightarrow{\sigma} r \xrightarrow{\varrho} s,\ (s,q) \in P,\ p \in I,\ r \in F\}$$



A P-automaton recognizes all words in the language of the underlying finite automaton. In addition, it is permitted to use P-edges as ε-edges, except that the first usage of a P-edge determines the period of the remaining word to be read (the period is $\sigma\varrho$ in Definition 3). We draw P-edges as dashed lines.

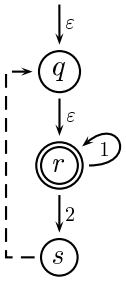
Example 4. Consider the P-automaton with alphabet $\{1, 2\}$, states $\{q, s\}$, initial and final state q, edges $q \xrightarrow{1} s$ and $q \xrightarrow{2} s$ and a single P-edge (s, q). The automaton can loop by using its P-edge multiply, but the first usage determines a period (the word 1 or 2) of the remaining word. Thus, the language recognized is $1^* \cup 2^*$ rather than $(1 \cup 2)^*$.



The length of a period fixed by a first usage of a P-edge needs not to be bounded by the number of states of the P-automaton. This fact raises the following problem.

Lemma 5 (Failure of context-freeness). *There exists a P-automaton whose language is not context-free (and thus not regular).*

Proof. We consider the P-automaton with alphabet $\{1, 2\}$, states $\{q, r, s\}$, initial states $\{q\}$, final states $\{r\}$, transition edges $q \xrightarrow{\varepsilon} r$, $r \xrightarrow{1} r$ and $r \xrightarrow{2} s$ and a single P-edge (r, q). This P-automaton is depicted to the right. It recognizes the language $\bigcup\{pr(\pi^*) \mid \pi \in 1^*2\}$. which is not context-free. Otherwise, the intersection with the regular language $1^*21^*21^*2$ would also be context-free. But this intersection is the language $\{1^n21^n21^n2 \mid n \geq 0\}$ which is clearly not context-free.



4 Path Constraints

We now introduce path constraints which express properties of subtrees at a given path. Path constraints are fundamental in understanding entailment for many languages of ordering constraints [9, 10, 20, 14, 13, 15].

If τ is a tree and $\pi \in D_\tau$ then we write $\tau.\pi$ for the *subtree of* τ *at* π, i.e. $D_{\tau.\pi} = \{\pi' \mid \pi\pi' \in D_\tau\}$ and $L_{\tau.\pi}(\pi') = L_\tau(\pi\pi')$ for all $\pi' \in D_{\tau.\pi}$. A *subtree constraint* $x.\pi{=}y$ requires that the domain of the denotation of x contains π and that its subtree at path π is the denotation of y.

Conditional path constraints of the first kind we use are of the form $x?\sigma \leq_L y?\pi$. The question mark indicates conditionality depending on the existence of a subtree. A path constraint $x?\sigma \leq_L y?\pi$ is solved by a variable assignment α if $\pi_1 \in D_{\alpha(x)}$ and $\pi_2 \in D_{\alpha(y)}$ implies $L_{\alpha(x)}(\pi_1) \leq_L L_{\alpha(y)}(\pi_2)$. We freely omit the conditionality $?\varepsilon$ since it ε path does always exist. We also write $x?\sigma \leq_L f$ instead of $\exists y(x?\sigma \leq_L y \wedge y{\leq}f(\top, \ldots, \top))$, and, symmetrically, $f \leq_L x?\sigma$ instead of $\exists y(f(\bot, \ldots, \bot){\leq}y \wedge y \leq_L x?\sigma)$.

Proposition 6 (Characterization of Entailment). *For all u, v the following equivalence is valid in NS_{Σ_n}:*

$$u{\leq}v \;\leftrightarrow\; \bigwedge\{u?\pi \leq_L v?\pi \mid \pi \textit{ a path }\}$$

We call a path π a *contradiction path for* $\psi \models x{\leq}y$ if and only if ψ does not entail $x?\pi \leq_L y?\pi$. In this terminology, Proposition 6 states that an entailment judgment $\psi \models x{\leq}y$ holds if and only if there exists no contradiction path for it.

We need further conditional path constraints of the form $x?\pi{\leq^{pr}}y$, $x{\leq^{pr}}y?o$, and $x?\pi{\leq^{pr}}y?o$ which do not only restrict x and y at the paths π and o but also at their prefixes. The semantics of these constraints is defined in Table 2. Note that the path constraint $x?\pi{\leq^{pr}}y$ entails $\exists z\,(x.\pi{=}z \to z{\leq}y)$ but not vice versa. The reason is $x?\pi{\leq^{pr}}y$ constrains x even if $x.\pi$ is not defined. For instance, the constraint $x{\leq}f(y)$ entails $x?1{\leq^{pr}}y$ which – if $x.1$ is not defined – requires $x{\leq}\bot$.

For a restricted signature, the semantics of conditional path constraints is much less ad hoc than it might seem at first sight. This is shown by Lemma 8 for the signature $\Sigma_n = \{\bot, \top, g\}$ where g is a function symbol with $\mathsf{ar}(g) = n$.

$$
\begin{array}{rcl}
x?\pi{\leq^{pr}}y & \leftrightarrow & x.\pi{\leq}y \vee \bigvee_{\pi'<\pi} x.\pi'{\leq}\bot \\
x{\leq^{pr}}y?o & \leftrightarrow & x{\leq}y.o \vee \bigvee_{o'<o} \top{\leq}y.o' \\
x?\pi{\leq^{pr}}y?o & \leftrightarrow & \exists u(x?\pi{\leq^{pr}}u \wedge u{\leq^{pr}}y?o)
\end{array}
$$

Table 2. Semantics of conditional path constraints

Lemma 7. *For $n \geq 1$, signature $\Sigma_n = \{\bot, \top, g\}$, paths $\pi \in \{1, \ldots, n\}^*$ and $\pi' < \pi$: $u?\pi{\leq^{pr}}v \rightarrow u?\pi' \leq_L g$ and $u{\leq^{pr}}v?\pi \rightarrow g \leq_L v?\pi'$ are valid in NS_{Σ_n}.*

Lemma 8 (Subtree versus conditional path constraints). *For $n \geq 1$, paths $\pi \in \{1, \ldots, n\}^*$ and variables x, y the following equivalences hold in the structure NS_{Σ_n}:*

$$
\begin{array}{ll}
x?\pi{\leq^{pr}}y \leftrightarrow \exists z(x{\leq}z \wedge z.\pi{=}y), & \qquad x{\leq^{pr}}y?\pi \leftrightarrow \exists z(z{\leq}y \wedge z.\pi{=}x) \\
u.\pi{=}v \leftrightarrow u?\pi{\leq^{pr}}v \wedge v{\leq^{pr}}u?\pi &
\end{array}
$$

Proof. We only prove the implication from the right to the left in the third equivalence. Assume that $u?\pi{\leq^{pr}}v \wedge v{\leq^{pr}}u?\pi$. For arbitrary $\pi' < \pi$, Lemma 7 proves the validity of $u?\pi' \leq_L g$ and $g \leq_L u?\pi'$. Thus, $u.\pi$ must be defined, and hence, $u.\pi = v$.

Lemma 9 (Strange loops raising P-edges). *For all variable u, v, all paths $\sigma < \pi$, and $k \geq 0$ the following implication is valid in NS_{Σ_n} for all $k \geq 0$:*

$$
u?\pi{\leq^{pr}}v \wedge u{\leq^{pr}}v?\pi \quad \rightarrow \quad u?\pi^k\sigma \leq_L v?\pi^k\sigma
$$

Proof. By induction on k. Let $k = 0$. Since $\sigma < \pi$, Lemma 7 proves that $u?\pi{\leq^{pr}}v \wedge u{\leq^{pr}}v?\pi$ entails $u?\sigma \leq_L g \wedge g \leq_L v?\sigma$ which in turn validates $u?\sigma \leq_L v?\sigma$. Suppose $k > 0$ and that $u?\pi{\leq^{pr}}v \wedge u{\leq^{pr}}v?\pi$ is valid. By definition, $u?\pi{\leq^{pr}}v \leftrightarrow u.\pi{\leq}v \vee \bigvee_{\varrho<\pi} u.\varrho{\leq}\bot$ and $v{\leq^{pr}}v?\pi \leftrightarrow u{\leq}v.\pi \vee \bigvee_{\varrho<\pi} \top{\leq}u.\varrho$ hold. If there exists $\varrho < \pi$ such that $u.\varrho{\leq}\bot$ or $\top{\leq}u.\varrho$ then $u.\varrho{\leq}v.\varrho$ is entailed for some prefix of π. In this case, $u?\pi^k\sigma \leq_L v?\pi^k\sigma$ follows from Proposition 6. Otherwise, $u.\pi{\leq}v \wedge u{\leq}v.\pi$ is valid. Let u', v' be such that $u' = u.\pi$ and $v' = u.\pi$. Thus, $u'{\leq}v \wedge v.\pi{=}v'$ holds and entails $u'?\pi{\leq^{pr}}v'$ by Lemma 8. Symmetrically, $u'{\leq^{pr}}v'?\pi$ is entailed, too. The induction hypothesis yields $u'?\pi^{k-1}\sigma \leq_L v'?\pi^{k-1}\sigma$ and thus $u?\pi^k\sigma \leq_L v?\pi^k\sigma$.

5 Entailment and P-Automata

We continue with the signature $\Sigma_n = \{\bot, \top, g\}$ where $\mathsf{ar}(g) = n$. We fix two variables x, y globally and consider a constraint ψ with $x, y \in \mathcal{V}(\psi)$. In Table 3, we define a finite automaton $\mathcal{A}_\psi$ and a P-automaton $\mathcal{P}_\psi = (\mathcal{A}_\psi, P_\psi)$ for the judgment $\psi \models x{\leq}y$. Note that $\mathcal{A}_\psi$ and thus $\mathcal{P}_\psi$ depend on our global variables x and y.

The idea is that the P-automaton $\mathcal{P}_\psi$ recognizes all *safe* paths, i.e those paths that are not contradiction paths for $\psi \models x{\leq}y$. In fact, the definition of $\mathcal{P}_\psi$ does not always achieve this goal. This is not a problem for the purpose of this paper since our theory will be based exclusively on the regular approximation of $\mathcal{P}_\psi$ provided by the finite automaton $\mathcal{A}_\psi$. Even though the construction rules given in Table 3 apply without further restriction to ψ, an automaton $\mathcal{P}_\psi$ may well be useless if ψ is not closed and clash-free, or contains $\bot$ and $\top$.

Signature	$\Sigma_n = \{\bot, \top, g\}$	$\mathsf{ar}(g) = n$
Alphabet	$A_n = \{1, \ldots, n\}$	
States	$Q_\psi = \{(u, v) \mid u, v \in \mathcal{V}(\psi)\}$	
Intial States	$I_{xy} = \{(x, y)\}$	
Increase	$(u, v) \xrightarrow{\varepsilon} (u', v) \in \Delta_\psi$	if $u \leq u'$ *in* ψ
Decrease	$(u, v) \xrightarrow{\varepsilon} (u, v') \in \Delta_\psi$	if $v' \leq v$ *in* ψ
Decomposition	$\left.\begin{array}{l}(u, v) \xrightarrow{i} (u_i, v_i) \in \Delta_\psi \\ (u, v) \in F_\psi\end{array}\right\}$	if $\left\{\begin{array}{l} u = g(u_1, \ldots, u_n) \text{ in } \psi, \\ v = g(v_1, \ldots, v_n) \text{ in } \psi, \\ \text{and } i \in A_n\end{array}\right.$
Equality	$\left.\begin{array}{l}(u, u) \xrightarrow{i} (u, u) \in \Delta_\psi \\ (u, u) \in F_\psi\end{array}\right\}$	if $i \in A_n$
P−Edges	$((u, v), (v, u)) \in P_\psi$	if $u, v \in \mathcal{V}(\psi)$

Table 3. The finite automaton $\mathcal{A}_\psi = (A_n, Q_\psi, I_{xy}, F_\psi, \Delta_\psi)$ and P-automaton $(\mathcal{A}_\psi, P_\psi)$ for $\psi \models x \leq y$

Given a constraint ψ over the signature Σ_n, the automata $\mathcal{P}_\psi$ and $\mathcal{A}_\psi$ constructed in Table 3 recognize words over the alphabet $\{1, \ldots, n\}$. Its states are pairs of variables (u, v) in $\mathcal{V}(\psi)$. The initial state is (x, y), i.e. the pair of variables for which entailment is tested. Ordering constraints $u \leq v$ correspond to ε-transitions in the rules **Increase** and **Decrease**. The **Decomposition** rule permits transitions that read a natural number $i \in A_n$ and descend to the i-th child, in parallel for both variables in a state. States to which decomposition applies are final. The **Equality** rule requires that states (u, u) are final and permitted to loop into itself. The automaton $\mathcal{P}_\psi$ features **P-Edges** for switching the two variables in a state.

Proposition 10 (Soundness). *Given a constraint ψ with $x, y \in \mathcal{V}(\psi)$ and signature Σ_n where $n \geq 1$, no word recognized by the P-automaton $\mathcal{P}_\psi$ is a contradiction path for $\psi \models x \leq y$.*

Proof. We first show that $\pi \in \mathcal{L}(\mathcal{A}_\psi)$ implies entailment $\psi \models x?\pi \leq_L y?\pi$ to hold. Clearly, if $\mathcal{A}_\psi \vdash (u, v) \xrightarrow{\pi} (u', v')$ then ψ entails $u?\pi \leq^{pr} u'$ and $v' \leq^{pr} v?\pi$. If $\pi \in \mathcal{L}(\mathcal{A}_\psi)$ due to a transition $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\pi} (u, v) \in F_\psi$ which ends in a final state (u, v) created by the **Decomposition** rule, then $\pi i \in \mathcal{L}(\mathcal{A}_\psi)$ for all $i \in A_n$. Thus, ψ entails $x?\pi i \leq^{pr} u$ and $v \leq^{pr} y?\pi i$ for some variables u, v which in turn entails $x?\pi \leq_L y?\pi$ (Lemma 7). If $\pi \in \mathcal{L}(\mathcal{A}_\psi)$ because a transition $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\pi} (u, u) \in F_\psi$ ends in a final state (u, u) contributed the **Equality** rule then ψ entails $x?\pi \leq^{pr} y?\pi$ and thus $x?\pi \leq_L y?\pi$.

It remains to verify that P-edges cannot contribute a contradiction path. If a path is contributed by a P-edge to $\mathcal{L}(\mathcal{P}_\psi)$ then it has the form $\pi(\sigma\varrho)^k\sigma$ such that $\mathcal{A}_\psi \vdash (x, y) \xrightarrow{\pi} (u, v) \xrightarrow{\sigma\tau} (v, u)$ for some $u, v \in \mathcal{V}(\psi)$ (see Definition 3 and the **P-Edges** rule in Table 3). From $\mathcal{A}_\psi \vdash (u, v) \xrightarrow{\sigma\tau} (v, u)$ it follows that ψ entails $u?\sigma\varrho \leq^{pr} v \wedge u \leq^{pr} v?\sigma\varrho$. Thus, Lemma 9 on strange loops implies that ψ entails $u?(\sigma\varrho)^k\sigma \leq_L v?(\sigma\varrho)^k\sigma$. Since $\mathcal{A}_\psi \vdash (x, y) \xrightarrow{\pi} (u, v)$ it follows that ψ entails $x?\pi(\sigma\varrho)^k\sigma \leq_L y?\pi(\sigma\varrho)^k\sigma$, too.

Example 11. For the signature Σ_2 the judgment φ_2: $x \leq y \times y \wedge x \times x \leq y \models x \leq y$ does not hold if x and y are distinct variables. Entailment is contradicted by the solution of

$x \leq y \times y \wedge x \times x \leq y \not\models x \leq y$

Entailment can be
contradicted at paths $12, 21, \ldots$

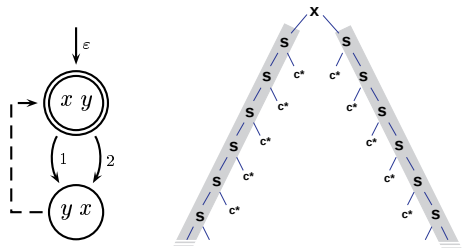


Fig. 2. The finite automaton and P-automaton for Example 11 and their languages

$x \leq g(y) \wedge g(x) \leq y \models x \leq y$

Entailment holds.

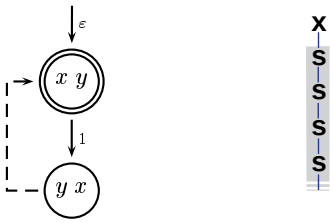


Fig. 3. The finite automaton and P-automaton for Example 12 and their languages

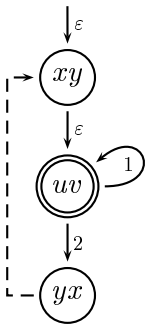
φ_2 which maps x to $\bot \times (\top \times \bot)$ and y to $\top \times (\bot \times \top)$. The P-automaton $\mathcal{P}_{\varphi_2}$ illustrated in Figure 2 explains what happens. The finite automaton $\mathcal{A}_{\varphi_2}$ recognizes the language $\{\varepsilon\}$ only but $\mathcal{P}_{\varphi_2}$ has an additional P-edge from (y, x) to (x, y) by which it can also recognize the words in $1^+ \cup 2^+$. Since P-edges are not normal ε-edges, the P-automaton does not recognize the words 12 nor 21 which are in fact contradiction paths.

In Figures 2 and 3, we depict the language recognized by an P-automaton over the alphabet $\{1, \ldots, n\}$ as an n-ary tree: a word recognized by the underlying finite automaton corresponds to a node labeled by **x**, a word recognized by the additional P-edges only is indicated by a node labeled with **s** (for strange loop). All other words correspond to a node labeled with **c** (for contradiction).

Example 12. For the signature $\Sigma_1 = \{\bot, \top, g\}$ with $\mathsf{ar}(g) = 1$ the entailment judgment $\varphi_1 : x \leq g(y) \wedge g(x) \leq y \models x \leq y$ holds. This might seem surprising since the only difference to Example 11 seems to be the choice of a unary versus a binary function symbol. The situation is again clarified when considering the P-automaton. The automaton $\mathcal{P}_{\varphi_1}$ is given in Figure 3. In contrast to $\mathcal{P}_{\varphi_2}$ in Figure 2, the alphabet of $\mathcal{P}_{\varphi_1}$ is the singleton $\{1\}$. Thus, its language $\mathcal{L}(\mathcal{P}_{\varphi_1}) = \{1\}^*$ is universal. Hence, there cannot be any contradiction path for $\varphi_1 \models x \leq y$, i.e. entailment holds.

Examples 11 and 12 illustrate that P-edges have less effect on entailment in absence of unary function symbols. In fact, we show in this paper that P-edges do not have any effect on entailment for the restricted language. Even more importantly, this property depends on the restriction that constraints $u = \bot$ or $u = \top$ are not supported.

The context freeness failure for languages of P-automata has a counterpart for non-structural subtype entailment, even for the restricted language. This is illustrated by the judgment: $\varphi_3 : x{\leq}u \wedge v{\leq}y \wedge u{=}u{\times}y \wedge v{=}v{\times}x \models x{\leq}y$. The language $\mathcal{L}(\mathcal{P}_{\varphi_3})$ is not context-free since $\mathcal{P}_{\varphi_3}$ is exactly the P-automaton considered in the proof of Lemma 5. On the other hand side, the non-context free part of $\mathcal{L}(\mathcal{P}_{\varphi_3})$ does not force entailment to hold.



6 Deciding Entailment in PSPACE

We now show how to decide entailment for the restricted entailment problem with signature Σ_2. Our algorithm requires polynomial space and applies to both structures NS_{Σ_2} or $NS^{fin}_{\Sigma_2}$ respectively. The only difference is hidden in the satisfiability test used. Let *NS* be either of the two structures.

Proposition 13 (Characterization). *Let φ be a closed (restricted) constraint with $x, y \in \mathcal{V}(\varphi)$ which is clash-free with respect to NS. Then the entailment judgment $\varphi \models x{\leq}y$ holds in NS if and only if the set $\mathcal{L}(\mathcal{A}_\varphi)$ is universal, i.e. $\mathcal{L}(\mathcal{A}_\varphi) = \{1,2\}^*$.*

Proof. If $\mathcal{L}(\mathcal{A}_\varphi) = \{1,2\}^*$ then no contradiction path for $\varphi \models x{\leq}y$ exists (Proposition 10) and hence $\varphi \models x{\leq}y$ holds (Proposition 6). Proving the converse (completeness) is much more involved. This proof is sketched in Section 8.

Theorem 14 (Decidability and Complexity). *Non-structural subtype entailment in the restricted language is PSPACE-complete for both structures NS_{Σ_2} and $NS^{fin}_{\Sigma_2}$.*

Proof. Proposition 1 claims that entailment is PSPACE-hard. For deciding $\varphi \models x{\leq}y$, we compute the closure of φ in polynomial time and check whether it is clash-free with respect to NS_{Σ_2} or $NS^{fin}_{\Sigma_2}$ respectively (Proposition 2). For closed and clash-free φ, entailment holds if and only if $\mathcal{L}(\mathcal{A}_\varphi)$ is universal (Proposition 13). This can be checked in PSPACE since $\mathcal{A}_\varphi$ is a finite automaton which can be constructed from φ in (deterministic) polynomial time.

7 Completeness Proof

We prove the completeness of the characterization of entailment in Proposition 13. For a constraint φ of the restricted language, the idea is that we can freely extend the P-automaton $(\mathcal{A}_\varphi, P_\varphi)$ with additional P-edges without affecting universality. This motives to consideration of a language $Trace_\varphi$ which is recognized by the P-automaton $(\mathcal{A}_\varphi, Q_\varphi{\times}Q_\varphi)$ where Q_φ is the set of all states of $\mathcal{A}_\varphi$.

Definition 15. We define the set $Base_\varphi$ of *bases* and $Trace_\varphi$ of *traces* of $\varphi \models x{\leq}y$ by:

$$Base_\varphi = \{\pi \mid \exists u, v:\ \mathcal{A}_\varphi \vdash (x,y) \xrightarrow{\pi} (u,v)\}$$
$$Trace_\varphi = \bigcup\{pr(o\pi^*) \mid o\pi \in Base_\varphi\}$$

Lemma 16. *The set $pr_{\neq}(Base_\varphi)$ is equal to the set $\mathcal{L}(\mathcal{A}_\varphi)$.*

$\psi \vdash x?\varepsilon \leq^{pr} y$ if $x \leq y$ *in* ψ

$\psi \vdash x?\pi i \leq^{pr} y$ if $\psi \vdash x?\pi \leq^{pr} z$ and $z = f(z_1, \ldots z_i \ldots, z_n),\ z_i \leq y$ *in* ψ

$\psi \vdash x \leq^{pr} y?\varepsilon$ if $x \leq y$ *in* ψ

$\psi \vdash x \leq^{pr} y?\pi i$ if $\psi \vdash z \leq^{pr} y?\pi$ and $z = f(z_1, \ldots z_i \ldots, z_n), x \leq z_i$ *in* ψ

$\psi \vdash x?\pi \leq^{pr} y?\pi'$ if $\exists z : \psi \vdash x?\pi \leq^{pr} z$ and $\psi \vdash z \leq^{pr} y?\pi'$

Table 4. Syntactic Support

Proof. Showing that $\varepsilon \in \mathcal{L}(\mathcal{A}_\varphi)$ implies $\varepsilon \in \mathit{pr}_{\neq}(\mathit{Base}_\varphi)$ is left to the reader. If $\pi i \in \mathcal{L}(\mathcal{A}_\varphi)$ then $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\pi i} (u, v)$ for some (final) state (u, v). Thus, $\pi i \in \mathit{Base}$, i.e. $\pi \in \mathit{pr}_{\neq}(\mathit{Base}_\varphi)$. For the converse, assume $\pi \in \mathit{pr}_{\neq}(\mathit{Base}_\varphi)$. Hence, $\pi i \in \mathit{Base}_\varphi$ for some i. There exists transitions $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\pi} (u, v) \xrightarrow{i} (u', v')$ with a final step done by the **Decomposition** rule in Table 3. Thus, $(u, v) \in F_\varphi$, i.e. $\pi \in \mathcal{L}(\mathcal{A}_\varphi)$.

Lemma 16 implies that $\mathcal{L}(\mathcal{P}_\varphi) \subseteq \mathit{Trace}_\varphi$. The next proposition states that if $\mathcal{L}(\mathcal{A}_\varphi)$ is not universal then neither Trace_φ nor $\mathcal{L}(\mathcal{P}_\varphi)$ are universal.

Proposition 17 (Escape). *If $\sigma \notin \mathcal{L}(\mathcal{A}_\varphi)$ then there is a path $\varrho \notin \mathit{Trace}_\varphi$ with $\sigma \leq \varrho$.*

Proof. We assume $\sigma \notin \mathcal{L}(\mathcal{A}_\varphi)$ and define $\varrho := \sigma 1^{|\sigma|} 2$ where $|\sigma|$ denotes the length of σ and 1^n a word which consists of exactly n letters 1. We prove $\varrho \notin \mathit{Trace}_\varphi$ by contradiction. Suppose that $\varrho \in \mathit{Trace}_\varphi$. By definition there exists paths o, π such that $o\pi \in \mathit{Base}_\varphi$ and $\varrho \in \mathit{pr}(o\pi^*)$. Hence $\sigma \in \mathit{pr}(o\pi^*)$ such that either $\sigma < o\pi$ or $o\pi \leq \sigma$. It it not possible that $\sigma < o\pi$ since otherwise, $\sigma \in \mathit{pr}_{\neq}(o\pi) \subseteq \mathit{pr}_{\neq}(\mathit{Base}_\varphi)$ which by Lemma 16 contradicts $\sigma \notin \mathcal{L}(\mathcal{A}_\varphi)$. Hence $o\pi \leq \sigma$ such that $\sigma = o\pi\sigma_0$ for some path σ_0. In combination with $\varrho = \sigma 1^{|\sigma|} 2 \in \mathit{pr}(o\pi^*)$ this yields $\sigma_0 1^{|\sigma|} 2 \in \mathit{pr}(\pi^*)$. Furthermore, $|\pi| \leq |o\pi| \leq |\sigma|$. The key point comes now: $\sigma_0 1^{|\sigma|} 2 \in \mathit{pr}(\pi^*)$ and $|\pi| \leq |\sigma|$ imply $\pi \in 1^*$ which is impossible since π must contain the letter 2. Hence, $\varrho \notin \mathit{Trace}_\varphi$.

Lemma 18 (Contradiction). *Let φ be closed and clash-free, $o \notin \mathcal{L}(\mathcal{A}_\varphi)$, and $\varrho \notin \mathit{Trace}_\varphi$: if $o \leq \varrho$ then ϱ is a contradiction path for $\varphi \models x \leq y$ in NS.*

Proof of Proposition 13 continued (Completeness). If $\mathcal{L}(\mathcal{A}_\varphi)$ is not universal then there exists a path $o \leq \varrho$ such that $o \notin \mathcal{L}(\mathcal{A}_\varphi)$ and $\varrho \notin \mathit{Trace}_\varphi$ according to the Escape Proposition 17. By Lemma 18, there exists a contradiction path which proves that entailment $\varphi \models x \leq y$ cannot hold.

8 Proof of the Contradiction Lemma

In a first step, we refine the contradiction Lemma 18 into Lemma 21. This requires a notion of *syntactic support* that is given in Table 4. If μ is a path constraint then the judgment $\varphi \vdash \mu$ reads as 'φ supports μ syntactically'. Syntactic support for φ refines judgments performed by the finite automaton $\mathcal{A}_\varphi$. For instance, it holds for a closed and clash-free constraint φ that $\varphi \vdash x?\pi \leq^{pr} y?\pi$ iff $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\pi} (u, u)$. Judgments like $\varphi \vdash x?\pi \leq^{pr} y$ or $\varphi \vdash x?\pi \leq^{pr} y?\pi'$ cannot be expressed by $\mathcal{A}_\varphi$.

Lemma 19. *For all path constraints μ if $\varphi \vdash \mu$ then $\varphi \models \mu$ holds.*

Definition 20. We define two functions l_φ and r_φ for the judgment $\varphi \models x{\leq}y$.

$$l_\varphi(\sigma) = \max\{\pi \mid \pi{\leq}\sigma \wedge \exists u.\varphi \vdash x?\pi 1{\leq}^{pr} u\} \qquad \text{(left)}$$
$$r_\varphi(\sigma) = \max\{\pi \mid \pi{\leq}\sigma \wedge \exists v.\varphi \vdash v{\leq}^{pr} y?\pi 1\} \qquad \text{(right)}$$

Note that if $l_\varphi(\sigma) \leq r_\varphi(\sigma)$ then $l_\varphi(\sigma)$ is the maximal prefix of σ in $\mathcal{L}(\mathcal{A}_\varphi)$. Symmetrically, if $r_\varphi(\sigma) \leq l_\varphi(\sigma)$ then $r_\varphi(\sigma)$ is the maximal prefix of σ in $\mathcal{L}(\mathcal{A}_\varphi)$.

Lemma 21 (Contradiction refined). *Let φ be a closed and clash-free constraint and $o{\leq}\varrho$ paths such that $o \notin \mathcal{L}(\mathcal{A}_\varphi)$ and $\varrho \notin \mathit{Trace}_\varphi$.*

1. *if $l_\varphi(\varrho) \leq r_\varphi(\varrho)$ then $\varphi \wedge x.\varrho{=}\top \wedge y.\varrho{=}_L{\times}$ is satisfiable.*
2. *if $l_\varphi(\varrho) > r_\varphi(\varrho)$ then $\varphi \wedge x.\varrho{=}{\times} \wedge y.\varrho{=}_L\bot$ is satisfiable.*

Trivially, Lemma 21 subsumes the contradiction Lemma 18. The proof of Lemma 21 captures the rest of this section. Since both of its cases are symmetric we restrict ourself to the first one. We assume that φ is closed and clash-free and satisfies $x, y \in \mathcal{V}(\varphi)$. Given a fresh variable u we define a constraint $s(\varphi, \varrho)$ that is satisfaction equivalent to $\varphi \wedge x.\varrho{=}u \wedge y.\varrho{=}_L{\times}$ and in addition closed and clash-free.

Definition 22. We call a set $D \subseteq \{1,2\}^*$ *domain closed* if D is prefixed-closed and satisfies the following property for all $\pi \in \{1,2\}^*$: $\pi 1 \in D$ iff $\pi 2 \in D$. The *domain closure* $dc(D)$ is the least domain closed set containing D.

Definition 23 (Saturation). Let φ be a constraint, $x, y \in \mathcal{V}(\varphi)$, and $\varrho \in \{1,2\}^*$. For every $z \in \{x, y\}$ and $\pi \in dc(\{\varrho 1, \varrho 2\})$ let q^z_π be a fresh variable and $W(\varphi, \varrho)$ the collection of these fresh variables. The saturation $s(\varphi, \varrho)$ of φ at path ϱ is the constraint of minimal size satisfying properties a-f:

a. φ *in* $s(\varphi, \varrho)$
b. for all $q^z_\pi \in W(\varphi, \varrho)$: $q^z_\pi{=}q^z_{\pi 1}{\times}q^z_{\pi 2}$ *in* $s(\varphi, \varrho)$ if $\pi < \varrho$
c. $q^y_\varrho{=}q^y_{\varrho 1}{\times}q^y_{\varrho 2}$ *in* $s(\varphi, \varrho)$
d. for all $q^z_\pi \in W(\varphi, \varrho), u \in \mathcal{V}(\varphi)$: $q^z_\pi{\leq}u$ *in* $s(\varphi, \varrho)$ if $\varphi \vdash z?\pi{\leq}^{pr}u$
e. for all $q^z_\pi \in W(\varphi, \varrho), u \in \mathcal{V}(\varphi)$: $u{\leq}q^z_\pi$ *in* $s(\varphi, \varrho)$ if $\varphi \vdash u{\leq}^{pr}z?\pi$
f. for all $q^z_{o\pi}, q^{z'}_{o'\pi} \in W(\varphi, \varrho)$: $q^z_{o\pi}{\leq}q^{z'}_{o'\pi}$ *in* $s(\varphi, \varrho)$ if $\varphi \vdash z?o{\leq}^{pr}z'?o'$

Lemma 24. *If φ is closed and clash-free then $s(\varphi, \varrho)$ is also closed and clash-free.*

Lemma 24 would go wrong for unrestricted constraints containing $\bot$ or $\top$. Its proof is not difficult but tedious since it requires a lot of case distinctions. We omit it for lack of space. Instead we note the following lemma which despite of its simplicity will turn out to be essential.

Lemma 25. *Let σ and o be paths with $\sigma{\leq}o\sigma$. If $o{\neq}\varepsilon$ then $\sigma \in pr(o^*)$.*

The proof of Lemma 25 is simple and thus omitted. We can now approach the final step in which we show that $s(\varphi, \varrho) \wedge q^x_\varrho{=}\top$ is also closed and clash-free. Closedness follows trivially from Lemma 24 but clash-freeness requires work. The only clash rule

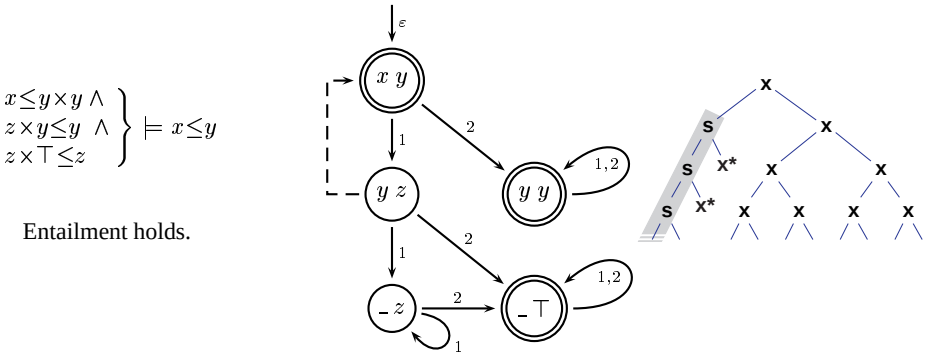


Fig. 4. An example for the general case

which might possibly apply is S3. Since $\bot$ does not occur in $s(\varphi, \varrho)$, S3 can only be applied with $q^x_\varrho{=}\top$ and $\top \not\leq_L \times$, i.e. if there are $w, w_1, w_2 \in \mathcal{V}(s(\varphi, \varrho))$ such that:

$$w{=}w_1{\times}w_2 \textit{ in } s(\varphi, \varrho) \text{ and } q^x_\varrho{\leq}w \textit{ in } s(\varphi, \varrho)$$

We have to distinguish all possible choices of $w \in \mathcal{V}(s(\varphi, \varrho))$ but restrict ourself to the more interesting cases where $w \in W(\varphi, \varrho)$. In this case, $q^x_\varrho{\leq}w$ was added to $s(\varphi, \varrho)$ by rule f in Definition 23. Since $w{=}w_1{\times}w_2$ *in* $s(\varphi, \varrho)$ and $w \in W(\varphi, \varrho)$ it follows that $w = q^y_\varrho$, or $w = q^z_\pi$ for some $\pi < \varrho$ and $z \in \{x, y\}$.

1. **Case** $w = \boldsymbol{q^y_\varrho}$**:** Rule f requires $\varphi \vdash x?\sigma{\leq}^{pr}y?\sigma$ for some prefix $\sigma{\leq}\varrho$. This is equivalent to $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\sigma} (u, u)$ for some u. The **Equality** rule in the automaton construction yields $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\varrho} (u, u)$. Thus, $\varrho \in \mathcal{L}(\mathcal{A}_\varphi)$ which contradicts $\varrho \notin \mathit{Trace}_\varphi$.
2. **Case** $w = \boldsymbol{q^z_\pi}$ **where** $\pi < \varrho$ **and** $z \in \{x, y\}$**:** Rule f requires the existence of σ, ϱ', π' such that $\varphi \vdash x?\varrho'{\leq}^{pr}z?\pi'$ where $\varrho = \varrho'\sigma$ and $\pi = \pi'\sigma$. From $\pi < \varrho$ it follows that $\pi'\sigma < \varrho'\sigma$ and thus $\pi' < \varrho'$. Let $o \neq \varepsilon$ be such that $\pi'o = \varrho'$. Thus, $\pi'\sigma < \pi'o\sigma$ which in turn yields $\sigma < o\sigma$. The key point comes now. We can apply Lemma 25 in order to deduce $\sigma \in \mathit{pr}(o^*)$. Hence $\varrho = \varrho'\sigma = \pi'o\sigma \in \pi'o\mathit{pr}(o^*) \subseteq \mathit{pr}(\pi'o^*)$. Since $\varphi \vdash x?\varrho'{\leq}^{pr}z?\pi'$ there exists u such that $\varphi \vdash x?\varrho'{\leq}^{pr}u$; together with our assumption $l_\varphi(\varrho) \leq r_\varphi(\varrho)$ it follows that $\mathcal{A}_\varphi \vdash (x, y) \xrightarrow{\varrho'} (u, v)$ for some u, v. Hence, $\varrho' \in \mathit{Base}_\varphi$, i.e. $\pi'o \in \mathit{Base}_\varphi$. Combined with $\varrho \in \mathit{pr}(\pi'o^*)$, we obtain $\varrho \in \mathit{Trace}_\varphi$ in contradiction to our assumption.

9 Conclusion and Future Work

We have solved the problem of non-structural subtype entailment over the signature $\{\bot, \top, \times\}$ for the restricted language where $\bot$ and $\top$ are not supported syntactically. We have proved PSPACE-completeness both for simple and recursive types. We have presented the notion of a P-automaton and illustrated its importance for understanding non-structural subtype entailment. Because of its P-edges a P-automaton can recognize non context-free languages. In what concerns non-structural subtype entailment for the restricted language, we have proved that non regularity can be safely ignored.

We believe that our methods can be extended to the full problem of non-structural subtype entailment. However, the full problem may well turn out to be more complex then PSPACE-complete. More research is needed to answer this question finally. The main problem in the general case is that we have to take P-edges into account. This is illustrated by the following example:

$$\varphi_4\colon x{\leq}y{\times}y \wedge z{\times}y{\leq}y \ \wedge z{\times}\top{\leq}z \models x{\leq}y$$

Entailment holds even though the language of finite automaton for φ_4 given in Figure 4 is not universal. The construction rules for this automaton are more involved than in Table 3 since $\top$ has to be accounted for. A P-edge from (x, y) to (y, z) has to be added even though only one of the two variables is switched.

References

1. H. Aït-Kaci, A. Podelski, and G. Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1–2):263–283, Jan. 1994.
2. R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, September 1993.
3. W. Charatonik and A. Podelski. Set constraints with intersection. In *Proceedings of the 12^{th} IEEE Symposium on Logic in Computer Science*, pages 352–361, Warsaw, Poland, 1997.
4. J. Dörre. Feature logics with weak subsumption constraints. In *Annual Meeting of the ACL (Association of Computational Logics)*, pages 256–263, 1991.
5. J. Dörre and W. C. Rounds. On subsumption and semiunification in feature algebras. In *Proceedings of the 5^{th} IEEE Symposium on Logic in Computer Science*, pages 300–310, 1990.
6. J. Eifrig, S. Smith, and V. Trifonow. Sound polymorphic type inference for objects. In *ACM Conference on Object-Oriented Programming: Systems, Languages, and Applications*, 1995.
7. J. Eifrig, S. Smith, and V. Trifonow. Type inference for recursively constrained types and its application to object-oriented programming. *Elec. Notes in Theoretical Computer Science*, 1, 1995.
8. Y. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73, 1990.
9. F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *Proceedings of the 12^{th} IEEE Symposium on Logic in Computer Science*, pages 362–372, Warsaw, Poland, 1997.
10. F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *Proceedings of the 25^{th} Int. Conf. on Automata, Languages, and Programming*, LNCS, 1998.
11. J. C. Mitchell. Type inference with simple subtypes. *The Journal of Functional Programming*, 1(3):245–285, July 1991.
12. J. C. Mitchell. *Foundations for Programming Languages*. The MIT Press, Cambridge, MA, 1996.
13. M. Müller, J. Niehren, and A. Podelski. Ordering constraints over feature trees. *Constraints, an International Journal, Special Issue on CP'97*, 5(1–2), Jan. 2000. To appear.
14. M. Müller, J. Niehren, and R. Treinen. The first-order theory of ordering constraints over feature trees. In *IEEE Symposium on Logic in Computer Science*, pages 432–443, 21-24 June 1998.
15. J. Niehren, M. Müller, and J.-M. Talbot. Entailment of atomic set constraints is PSPACE-complete. In *IEEE Symposium on Logic in Computer Sience*, 2–5, July 1999. to appear.
16. F. Pottier. Simplifying subtyping constraints. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pages 122–133. ACM Press, New York, May 1996.
17. F. Pottier. A framework for type inference with subtyping. In *Proceedings of the third ACM SIGPLAN International Conference on Functional Programming*, pages 228–238, Sept. 1998.
18. F. Pottier. *Type inference in the presence of subtyping: from theory to practice*. PhD thesis, Institut de Recherche d'Informatique et d'Automatique, 1998.
19. J. Rehof. Minimal typings in atomic subtyping. In *ACM Symposium on Principles of Programming Languages*. ACM Press, 1997.
20. J. Rehof. *The Complexity of Simple Subtyping Systems*. PhD thesis, DIKU, University of Copenhagen, 1998.
21. G. Smolka and R. Treinen. Records for logic programming. *Journal of Logic Programming*, 18(3):229–258, Apr. 1994.
22. V. Trifonov and S. Smith. Subtyping constrained types. In *Proceedings of the 3^{rd} International Static Analysis Symposium*, volume 1145 of *LNCS*, pages 349–365, Aachen, 1996.