Non-Structural Subtype Entailment in Automata Theory

Joachim Niehren and Tim Priesnitz

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

Summary. Decidability of non-structural subtype entailment is a long standing open problem in programming language theory. In this paper, we apply automata theoretic methods to characterize the problem equivalently by using regular expressions and word equations. This characterization induces new results on non-structural subtype entailment, constitutes a promising starting point for further investigations on decidability, and explains for the first time why the problem is so difficult. The difficulty is caused by implicit word equations that we make explicit.

1 Introduction

Subtyping is a common concept of many programming languages (including C++ and Java). A subtype relation $\tau' \leq \tau$ means that all functions in a program that expect an argument of type τ are sufficiently polymorphic so that they can also be applied to values of the subtype τ'. Thus, one can safely replace values of a type τ by values of subtype τ'.

Subtype constraints are systems of inequations $t \leq t'$ that talk about the subtype relation. Terms t and t' in subtype constraints are built from type variables and type constructors. Two logical operations on subtype constraints were investigated: satisfiability and entailment [1, 4, 6, 11, 16, 18]. *Subtype satisfiability* can be checked in cubic time for many type languages. A quadratic time algorithm for the variable free case is presented in [9].

Interest in *subtype entailment* was first raised by practical questions on type inference engines with subtyping [5, 17, 19]. The efficiency of such systems relies on the existence of powerful simplification algorithms for typings. Such operations can be formulated on the basis of subtype entailment algorithms.

It then turned out that subtype entailment is a quite complex problem, even for unexpressive type languages where types are ordinary trees. Rehof and Henglein [20] clarified the situation for structural subtyping. This is a tree ordering that relates trees of the same shape only. It is induced by lifting an ordering on constants. If trees are built over the signature $\{int, real, \times, \rightarrow\}$, for instance, then structural subtyping is induced by the usual axiom $int \leq real$ which says that every integer is a real number. Rehof and Henglein showed that structural subtype entailment is coNP-complete [7] for finite trees (simple types) and PSPACE-complete [8] for possibly infinite trees (recursive types).

Subtyping becomes *non-structural* if the constants $\perp$ and $\top$ are admitted that stand for the least and greatest type. Now, trees of different shapes can be related since all trees τ satisfy $\perp \leq \tau \leq \top$. Several cases are to be distinguished: one can consider only *finite* trees or admit *infinite* trees, or one may assume that all function symbols are *co-variant* (such as $\times$) or that some are *contra-variant* (as the function type constructor $\rightarrow$ in its first argument). One can also vary the number of type constructors of each *arity*.

Decidability of non-structural subtype entailment (NSSE) is a prominent open problem in programming language theory. Only a PSPACE lower bound is known which holds in both cases, for finite trees and for infinite trees [8]. The signature $\{\bot, f, \top\}$ is enough to prove PSPACE hardness if f is a type constructor of arity at least 2. But this result does not explain why finding a decision procedure for NSSE is so difficult[1]. On the other hand, only a fragment of NSSE could be proved decidable [14] (and PSPACE-complete).

The idea behind the approach is to first reformulate Rehof and Henglein's approach for structural subtype entailment in automata theory and second to lift it to the non-structural case. We have carried out both steps successfully but report only on the second step. It leads us to a new characterization of NSSE that uses regular expressions and word equations [10, 15]. Word equations raise the real difficulty behind NSSE since they spoil the usual pumping arguments from automata theory. They also clarify why NSSE differs so significantly from seemingly similar entailment problems [12, 13].

We present the new characterization of NSSE in the remainder of this introduction. Let R_1 and R_2 range over regular expressions over a finite alphabet $\mathbf{A}$. We next introduce a new class of set expressions E over $\mathbf{A}$ that we call *cap set expressions.*

$$E ::= R_1 R_2^\circ \mid E_1 \cup E_2$$

Cap set expressions denote sets of words over $\mathbf{A}$ (that we call *cap sets*). They subsume regular expressions R through $R\varepsilon^\circ$. To interpret cap set expression, we must define a set S° for sets $S \subseteq \mathbf{A}^*$. Let pr be the prefix operator lifted to sets of words. We then set:

$$S^\circ \;=\; \{\pi \mid \pi \in pr(\mu^*),\ \mu \in S\}$$

A word π belongs to S° if π is a prefix of a power $\mu \dots \mu$ of some word $\mu \in S$. If S is infinite, then the $^\circ$ operator becomes problematic: there exists a regular expression R for which the cap set described by R° is neither regular nor context free. Consider for instance $(21^*)^\circ$ which denotes the set of all prefixes of words $21^n\, 21^n \dots 21^n$ where $n \geq 0$. Clearly this set is not context-free. We need to derive appropriate *restrictions* on cap set expressions such that following theorem becomes true.

Theorem 1 (Characterization). *The decidability of NSSE for a signature $\{\bot, f, \top\}$ with a single function symbol of arity $n \geq 1$ is equivalent to the decidability of the universality problem for the class of restricted cap set expressions over the alphabet $\{1, \dots, n\}$.*

The theorem implies that all variants of NSSE with a single function symbol of arity $n \geq 2$ are equivalent. It does not even matter whether finite or infinite trees are considered.

The theorem can also be used to relate NSSE to word equations. The idea is to express membership in cap sets in the positive existential fragment of word equations with regular constraints [21]. The reduction can easily be based on the following lemma that is well known in the field of string unification.

Lemma 1. *For all words $\pi \in \mathbf{A}^*$ and nonempty words $\mu \in \mathbf{A}^+$ it holds that $\pi \in pr(\mu^*)$ if and only if $\pi \in pr(\mu\pi)$. Thus, all sets $S \subseteq \mathbf{A}^+$ of nonempty words satisfy:*

$$\pi \in S^\circ \leftrightarrow \exists\mu\exists\nu\ (\mu \in S \wedge \pi\nu{=}\mu\pi)$$

[1] A recent approach accepted for publication at the 2001 conference on Principles of Programming Languages failed and was retracted before publication (personal communication with the authors).

We can thus express the universality problem of cap set expressions E in the positive $\forall\exists^*$ fragment of the first-order theory of word equations with regular constraints. Unfortunately, even the positive $\forall\exists^3$ fragment of a single word equation is undecidable [3] except if the alphabet is infinite [2] or a singleton [22]. Therefore, it remains open whether NSSE is decidable or not. But it becomes clear that the difficulty is raised by word equations hidden behind cap set expressions R°, i.e. the equation $\pi\nu{=}\mu\pi$ in Lemma 1.

Theorem 1 constitutes a promising starting point to further investigate decidability of NSSE. For instance, we can infer a new decidability result for a special case where $n = 1$: NSSE is decidable for the signature $\{\bot, f, \top\}$ if f is a unary type constructor.

An extended version of this abstract is available at `www.ps.uni-sb.de/papers`.

2 Non-Structural Subtype Constraints

We investigate non-structural subtype constraints over signatures of function symbols $\Sigma = \{\bot, f, \top\}$ with a single non-constant function symbol f that is co-variant. We write ar_g for the *arity* of a function symbol $g \in \Sigma$, i.e $\mathsf{ar}_\bot = \mathsf{ar}_\top = 0$ and $\mathsf{ar}_f \geq 1$.

The choice of such signatures imposes two restrictions: first, we do not allow for contravariant type constructors. These could easily be covered in our framework but would impose an additional burden to the presentation. Second, we do not treat larger signature with more than one non-constant function symbol. This is a true restriction that cannot easily be circumvented.

2.1 Words, Trees, and Non-Structural Subtyping

A *word* over an alphabet $\mathbf{A}$ is a finite sequence of letters in $\mathbf{A}$. We denote words by π, ν, or μ and the set of words over $\mathbf{A}$ with $\mathbf{A}^*$. The *empty word* is written as ε and the free-monoid *concatenation* of words π and μ by juxtaposition $\pi\mu$, with the property that $\varepsilon\pi = \pi\varepsilon = \pi$. A *prefix* of a word π is a word μ for which there exists a word ν such that $\pi = \mu\nu$. If μ is a prefix of π then we write $\mu \leq \pi$ and if μ is a proper prefix of π then we write $\mu < \pi$.

We next define finite and infinite trees over Σ. We consider trees as partial functions $\tau : \mathbb{N}^* \rightsquigarrow \Sigma$ which map words over natural numbers to function symbols. A tree τ is *finite* if its domain D_τ is finite and otherwise *infinite*. The words in $D_\tau \subseteq \mathbb{N}^*$ are called the *nodes* or *paths* of the tree. The idea is to identify a node with the path that addresses it relative to the root. We require that every tree has a root $\varepsilon \in D_\tau$ and that tree domains D_τ are always prefix closed and arity-consistent. The latter means for all trees τ, nodes $\pi \in D_\tau$, and naturals $i \in \mathbb{N}$ that $\pi i \in D_\tau$ if and only if $1 \leq i \leq \mathsf{ar}_{\tau(\pi)}$.

We will freely interpret function symbols in Σ as tree constructors. To make clear distinctions, we will write $=_\Sigma$ for equality of symbols in Σ and $=$ for equality of trees over Σ. Given $g \in \Sigma$ and trees $\tau_1, \ldots, \tau_{\mathsf{ar}_g}$ we define $\tau = g(\tau_1, \ldots, \tau_{\mathsf{ar}_g})$ by $\tau(\varepsilon) =_\Sigma g$ and $\tau(i\pi) =_\Sigma \tau_i(\pi)$ for all $\pi \in D_{\tau_i}$ and $1 \leq i \leq \mathsf{ar}_g$. We thus consider ground terms over Σ as (finite) trees, for instance $f(\bot, \top)$ or $\bot$. Thereby, we have overloaded our notation since a constant $a \in \Sigma$ can also be seen as tree $\varepsilon \mapsto a$. But this should never lead to confusion.

Let $<_\Sigma$ be the irreflexive partial order on Σ that satisfies $\bot <_\Sigma f <_\Sigma \top$ and $\leq_\Sigma$ its reflexive counterpart. We define *non-structural subtyping* to be the unique partial order on trees which satisfies for all trees τ_1, τ_2 over Σ:

$$\tau_1 \leq \tau_2 \quad \text{iff} \quad \tau_1(\pi) \leq_\Sigma \tau_2(\pi) \text{ for all } \pi \in D_{\tau_1} \cap D_{\tau_2}$$

2.2 Constraint Language

We assume an infinite set of tree valued variables that we denote by x, y, z, u, v, w. A *subtype constraint* φ is a conjunction of literals with the following abstract syntax where $n = \mathsf{ar}_f$:

$$\varphi, \varphi' ::= x{\leq}f(y_1, \ldots, y_n) \mid f(y_1, \ldots, y_n){\leq}x \mid x{=}\bot \mid x{=}\top \mid \varphi \wedge \varphi'$$

We interpret constraints φ in the structure of trees over Σ with non-structural subtyping. We distinguish two cases, the structure of finite trees or else of possibly infinite trees. We interpret function symbols in both cases as tree constructors and the predicate symbol $\leq$ by the non-structural subtype relation. Again, this overloads notation: we use the same symbol $\leq$ for the subtype relation on trees and the predicate symbol denoting the subtype relation in constraints. Again, this should not raise confusion.

Note that we do not allow for formulas $x{\leq}y$ in our constraint language. This choice will help us to simplify our presentation essentially. It is, however, irrelevant from the point of view of expressiveness. We can still express $x{\leq}y$ by using existential quantifiers:

$$x{\leq}y \leftrightarrow \exists z \exists u \ (f(x, u, \ldots, u){\leq}z \wedge z{\leq}f(y, u, \ldots, u))$$

As in this equivalence, we will sometimes use first-order formulas Φ built from constraints and the usual first-order connectives. We will write V_Φ for the set of free variables occurring in Φ. A solution of Φ is a variable assignment α into the set of finite (resp. possibly infinite) trees which satisfies the required subtype relations; we write $\alpha \models \Phi$ if α solves Φ and say that Φ is *satisfiable*.

Example 1. The constraint $x{\leq}f(x)$ is satisfiable, even when interpreted over finite trees. We can solve it by mapping x to $\bot$. In contrast, the equality constraint $x{\leq}f(x) \wedge f(x){\leq}x$ is unsatisfiable over finite trees. It can however be solved by mapping x to the infinite tree $f(f(f(\ldots)))$.

A formula Φ_1 *entails* Φ_2 (we write $\Phi_1 \models \Phi_2$) if all solutions $\alpha \models \Phi_1$ satisfy $\alpha \models \Phi_2$. We will consider entailment judgments that are triples of the form (φ, x, y) that we write as $\varphi \models^? x{\leq}y$. *Non-structural subtype entailment* (NSSE) for Σ is the problem to check whether entailment $\varphi \models x{\leq}y$ holds for a given entailment judgment $\varphi \models^? x{\leq}y$.

Example 2. The prototypical example where NSSE holds somehow surprisingly is:

$$x{\leq}f(y) \wedge f(x){\leq}y \models^? x{\leq}y \qquad \text{(yes)}$$

To see this, note that all finite trees in the unary case are of the form $f \ldots f(\bot)$ or $f \ldots f(\top)$. Thus, $x{\leq}y \vee y{<}x$ is valid in this case. Next let us contradict the assumption that there is a solution $\alpha \models y{<}x \wedge x{\leq}f(y) \wedge f(x){\leq}y$. Transitivity yields $\alpha(y){\leq}f(\alpha(y))$ and then also $f(\alpha(x)){\leq}f(\alpha(y))$. Hence $\alpha(x){\leq}\alpha(y)$ which contradicts $\alpha(y){<}\alpha(x)$.

Note that entailment judgments of the simple form $\varphi \models^? x{\leq}y$ can express more general entailment judgments, where both sides are conjunctions of inequations $t_1{\leq}t_2$ between nested terms or variables. The main trick is to replace a judgment $\varphi \models^? t_1{\leq}t_2$ with terms t_1 and t_2 by $\varphi \wedge x{=}t_1 \wedge y{=}t_2 \models^? x{\leq}y$ where x and y are fresh variables. Note also that the omission of formulas $u{\leq}v$ on the left hand side does not restrict the problem.

References

1. R. M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, September 1993.
2. F. Baader and K. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In *Journal of Symbolic Computation*, volume 21, pages 211–243, 1996.
3. V. Durnev. Unsolvability of positive $\forall\exists^3$-theory of free groups. In *Sibirsky mathematichesky jurnal*, volume 36(5), pages 1067–1080, 1995. In Russian, also exists in English translation.
4. 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.
5. J. Eifrig, S. Smith, and V. Trifonow. Type inference for recursively constrained types and its application to object-oriented programming. *Elec. Notes in Theoretical Comp. Science*, 1, 1995.
6. Y. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73, 1990.
7. 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, 1997.
8. F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *Proc. of the25^{th} Int. Conf. on Automata, Languages, & Programming*, LNCS, 1998.
9. D. Kozen, J. Palsberg, and M. I. Schwartzbach. Efficient recursive subtyping. In *In Proc. of the 20^{th} ACM Symp. on Principles of Programming Languages*, pages 419–428. ACM Press, 1993.
10. G. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32, 1977. (English translation).
11. J. C. Mitchell. Type inference with simple subtypes. *The Journal of Functional Programming*, 1(3):245–285, July 1991.
12. 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.
13. J. Niehren, M. Müller, and J.-M. Talbot. Entailment of atomic set constraints is PSPACE-complete. In *14^{th} IEEE Symposium on Logic in Computer Sience*, pages 285–294, 1999.
14. J. Niehren and T. Priesnitz. Entailment of non-structural subtype constraints. In *Asian Computing Science Conference*, LNCS, pages 251–265, Thailand, Dec. 1999. Springer-Verlag, Berlin.
15. W. Plandowski. Satisfiability of word equations with constants is in PSPACE. In *Proc. of the 40^{th} IEEE Symp. on Found. of Comp. Science*, pages 495–500. IEEE Comp. Society Press, 1999.
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, Copenhagen, 1998.
21. K. U. Schulz. Makanin's algorithm for word equations – two improvements and a generalization. In *Word Equations and Related Topics*, LNCS 572, pages 85–150. Springer-Verlag, 1991.
22. Y. Vazhenin and B. Rozenblat. Decidability of the positive theory of a free countably generated semigroup. In *Math. USSR Sbornik*, volume 44, pages 109–116, 1983.