INFORMATIK

Satisfiability of Functional+Record Subtype Constraints is NP-Hard

Sergei Vorobyov

MPI–I–98–2–004 January 1998

FORSCHUNGSBERICHT RESEARCH REPORT

MAX-PLANCK-INSTITUT FÜR INFORMATIK

Im Stadtwald 66123 Saarbrücken Germany

Author's Address

Sergei Vorobyov: Max-Planck Institut für Informatik, Im Stadtwald, D-66123, Saarbrücken, Germany, sv@mpi-sb.mpg.de, http://www.mpi-sb.mpg.de/~sv.

Publication Notes

January 21, 1998

Acknowledgements

Thanks to David McAllester for motivating this research, for fruitful discussions, careful proofreading, thoughtful and illuminating remarks.

Abstract

We show the NP-hardness of the satisfiability problem for subtype inequalities between object types built by using simultaneously both the functional and the record type constructors, without base types. Earlier research concentrated on the complexity of subtyping either solely functional, or solely record types. In both cases deterministic cubic time algorithms are known.

Keywords

Functional, record type constructors, partial, object types, type reconstruction problem, subtyping, subsumption, subtype constraints, satisfiability, complexity, NP, PSPACE, NEXPTIME.

Contents

1 Introduction

In this paper we address the inherent computational complexity of the subtyping constraints satisfaction problem for object types built by using simultaneously the functional and the record type constructors, but without base types and subtyping relation on them. The motivation for subtyping record+functional types comes from the Object-Oriented Programming, where an object (record) can be emulated by another object that has more more refined methods (functions). To handle this phenomenon, the so-called *subsumption* with record+functional types is needed. The primary importance of the problem stems from the fact that satisfiability of systems of subtype inequalities is known to be *polynomial time equivalent* to the *type reconstruction* problem for lambda and object terms (Kozen, Palsberg & Schwartzbach 1994, Hoang & Mitchell 1995, Palsberg 1995). The latter problem is in the core of efficient typechecking in functional and object-oriented programming.

The *Satisfiability of Subtype Constraints Problem* (further, the SSCP for short) is defined as follows.

> **SSCP:** *Given a finite set of subtype inequalities* $\left\{\sigma_i \leq \tau_i\right\}_{i=1}^{n}$ *between type expressions* σ_i*,* τ_i *containing free type variables, does there exist a substitution of type expressions for these variables making all inequalities simultaneously true?*

The SSCP is obviously parameterized by:

- the choice of type constructors involved in type expressions,
- the choice of subtyping rules,
- presence/absence of base types with a subtype relation on them.

The most common features involved in type construction considered in the literature are as follows:

1. the functional type constructor $\sigma \to \tau$,
2. the record type constructor $[l_1 : \tau_1 \ldots, l_n : \tau]$,
3. base types with a subtype relation on them, e.g., $int \leq real$.

The following cases of SSCP are well investigated in the literature:

1. Subtyping of *partial types* built with a single functional type constructor $\to$, the top type Ω, and no base types (Kozen et al. 1994).

2. Subtyping of *object types* built with a single record type constructor $[l_1 : \tau_1 \ldots, l_n : \tau]$ and no base types (Palsberg 1995).

3. Subtyping with a single type constructor $\rightarrow$ and base types with a subtype relation on them (Lincoln & Mitchell 1992, O'Keefe & Wand 1992, Tiuryn 1992, Benke 1993, Hoang & Mitchell 1995, Pratt & Tiuryn 96, Frey 1997).

> *In this paper we consider the SSCP for types built by using simultaneously the functional and the record type constructors, but without base types.*

We now briefly survey these results in more detail.

1.1 Partial Types

(Kozen et al. 1994) consider the so-called *partial types* introduced by (Thatte 1988), built from a single base type Ω by using the single functional type constructor $\rightarrow$, i.e., defined by the grammar

$$\tau ::= \Omega \mid \tau_1 \rightarrow \tau_2$$

and the subtype relation $\leq$ defined by

$$\begin{array}{rl} \tau \leq \Omega & \text{for any type } \tau, \\ \sigma \rightarrow \tau \leq \sigma' \rightarrow \tau' & \text{iff } \sigma' \leq \sigma \text{ and } \tau \leq \tau'. \end{array} \tag{1}$$

(The latter is the standard subtyping rule for functional types.)

By using a nice automata-theoretic technique (Kozen et al. 1994) prove that the SSCP for these partial types is solvable in *deterministic time* $O(n^3)$. (O'Keefe & Wand 1992) were the first to show decidability of the problem by giving an exponential algorithm.

1.2 Object Types

(Palsberg 1995) considers the so-called *object types* built by using the single record type constructor, i.e., defined by the grammar

$$\tau \ ::= \ [l_1 : \tau_1, \ldots, l_n : \tau_n] \quad \text{for } n \geq 0,$$

where the field labels l_i's are drawn from an infinite set. (Note that in case $n = 0$ we get the empty record type $[\,]$. Obviously, $\tau \leq [\,]$ for each type τ.) The subtype relation on these object types is defined by

$$\sigma \leq \tau \text{ iff } \Big(\tau \text{ has field } l : \rho \Rightarrow \sigma \text{ has field } l : \rho\Big).$$

Note that this subtyping rule is different from a more well-known rule $\sigma \leq \tau$ iff $\Big(\tau$ has field $l : \tau' \Rightarrow \sigma$ has field $l : \sigma'$ such that $\sigma' \leq \tau'\Big)$. (Palsberg 1995) shows, also by using the automata-theoretic techniques similar to (Kozen et al. 1994), that the SSCP for the object types is decidable in deterministic time $O(n^3)$, and proves that the problem is PTIME-complete.

1.3 Functional Types with Base Subtyping

The SSCP for functional types defined starting from a set of *base types* with a *given subtype relation* on them extended to the set of all functional types by the standard subtyping rule (1) attracted most attention in the literature (Lincoln & Mitchell 1992, Tiuryn 1992, Pratt & Tiuryn 96, Benke 1993, Hoang & Mitchell 1995, Tiuryn 1997, Frey 1997). When the subtyping relation on base types is identity, the whole subtype relation is also identity, and the SSCP becomes the unification problem for simple types, known to be PTIME-complete (Dwork, Kanellakis & Mitchell 1984). (Lincoln & Mitchell 1992), improving (Wand & O'Keefe 1989), demonstrated that for some fixed ordering of base types the SSCP is NP-hard. (Tiuryn 1992, Pratt & Tiuryn 96) improved it to PSPACE-hardness for some simple fixed orderings of base types called *crowns*. (Lincoln & Mitchell 1992, Tiuryn 1992) gave the NEXPTIME upper bound for the problem. Recently (Frey 1997) improved it to PSPACE. Thus the SSCP for functional types with base types is PSPACE-complete, in general. When the subtype relation on base types is a disjoint union of lattices (Tiuryn 1992) or a tree-like partial order (Benke 1993), then the SSCP is in PTIME, i.e., becomes tractable.

It is interesting to note that the partial types subtyping of (Kozen et al. 1994) forms a lattice and the object types subtyping of (Palsberg 1995) forms a tree-like structure. But the relation between the PTIME results of (Tiuryn 1992, Benke 1993) on the one hand and the results of (Kozen et al. 1994, Palsberg 1995) on the other, seemingly, remains unexplored.

1.4 Contribution of This Paper

The complexity of the SSCP for types built by using *simultaneously* the functional $\rightarrow$ and the record $[\,]$ type constructors (both in presence or absence of base types) remained unknown, although type systems combining functions and records are quite natural in object-oriented programming. The papers cited above concentrate either solely on functional or only on record types, leaving open what happens when both features are combined together. The main result of this paper is that *even without any base types the SSCP for functional+record types is NP-hard.*

Thus, the absence of subtyping on base types does not lead to tractability, as contrasted to PSPACE/PTIME complexity for functional types with/without base subtyping, recall (Tiuryn 1992, Lincoln & Mitchell 1992, Tiuryn 1997).

Moreover, functions+records without base types, seemingly, do not allow to model the *crown structures* of (Tiuryn 1992, Pratt & Tiuryn 96), proved to be a useful uniform tool in showing NP and PSPACE lower bounds. Our proofs are not done by reduction from results of (Tiuryn 1992, Pratt & Tiuryn 96), we use a different method.

As compared to (Kozen et al. 1994, Palsberg 1995), our result shows that subtyping of functional types, as in (Kozen et al. 1994) cannot be straightforwardly combined with record subtyping of (Palsberg 1995) (both deterministic cubic time) to yield a polynomial deterministic algorithm. Thus, this is the intrinsic interplay between functional and record type subtyping (without base subtyping) that adds to the computational complexity of the SSCP. This shows that some generalizations are necessary to deal successfully with subtyping in type systems combining functions and records.

The remainder of the paper is organized as follows. In Section 2 we introduce the type system, and in Section 3 prove the main theorem. In Section 4 we discuss it and further results. We conclude in Section 5.

2 Functional+Record Types

Let $V = \{\alpha, \beta, \gamma, \ldots\}$ be an infinite set of *type variables* and $L = \{l_1, \ldots, l_n, \ldots\}$ be an infinite set of *field labels*.

Definition 2.1 (Types) are defined by the grammar

$$\tau \ ::= \ V \mid \tau' \to \tau'' \mid [l_1 : \tau_1, \ldots, l_n : \tau_n], \tag{2}$$

where $n > 0$ and l_i's are different labels from L. □

That is, types are constructed inductively, starting from type variables, by using the functional $\to$ and the record $[\ldots]$ type constructors. The subexpressions $l_i : \tau_i$ in the record construction are called *record fields*. Note that in the record construction the set of fields is always non-empty, i.e., we exclude the empty record. The reasons and consequences of this choice will be considered later.

Notational Conventions. Types will be denoted by Greek letters τ, σ, ρ, ... To stress the outermost type constructor in a type we will sometimes superscript a type by the corresponding outermost constructor, like $\tau^{\to}$ or $\tau^{[\,]}$. □

Definition 2.2 (Subtype Relation) is defined by the following standard rules:

Reflexivity: $\tau \leq \tau$,

Transitivity: $\sigma \leq \tau$ and $\tau \leq \rho$ imply $\sigma \leq \rho$,

Functional: $\sigma \to \tau \leq \sigma' \to \tau'$ iff $\sigma' \leq \sigma$ and $\tau \leq \tau'$,

Record: $\sigma^{[\,]} \leq \tau^{[\,]}$ iff for every field $l : \tau'$ in $\tau^{[\,]}$ there is a field $l : \sigma'$ in $\sigma^{[\,]}$ such that $\sigma' \leq \tau'$.

A subtype judgment $\sigma \leq \tau$ is *true* iff it is derivable by these rules. □

Remark. Note that a functional type is never a subtype of a record type, nor vice versa. All provable subtyping judgments are either $\alpha \leq \alpha$ for a type variable α, or of the form $\sigma^{\to} \leq \tau^{\to}$, or of the form $\sigma^{[\,]} \leq \tau^{[\,]}$, i.e., the subtyping relation is strictly structural. □

We are now ready to state precisely the problem we are interested in.

Satisfiability of Subtype Constraints Problem (SSCP).

> Given a finite system of subtype inequalities
>
> $$\left\{ \sigma_i \leq \tau_i \right\}_{i=1}^{n},$$
>
> does there exist a substitution of types for variables in σ_i's, τ_i's making all inequalities simultaneously true? $\square$

Our main result is as follows.

Theorem 2.3 *The SSCP for functional+record types is NP-hard.* $\square$

The proof of this theorem will be given in the next section.

3 Proof of the Main Theorem

Our proof is by reduction from the well-known NP-complete problem:

> **SATISFIABILITY.** Given a Boolean formula in Conjunctive Normal Form (CNF), does there exist an assignment of truth values to variables making the formula true?

We therefore proceed to representing truth values, clauses, and satisfiability in terms of types and subtyping.

3.1 Truth Values

Fix a type variable γ. Define the *truth values* $\mathbf{t}$ (true) and $\mathbf{f}$ (false) as types

$$\begin{aligned} \mathbf{t} &\equiv_{df} [1 : \gamma], \\ \mathbf{f} &\equiv_{df} \gamma \to \gamma, \end{aligned}$$

where 1 is an arbitrary label from L.

Clearly, neither $\mathbf{f} \leq \mathbf{t}$, nor $\mathbf{t} \leq \mathbf{f}$, because one is functional and the other is record (recall that the subtyping is strictly structural). Note that by definition of $\mathbf{t}$, $\mathbf{f}$, and $\leq$, for any type τ one has

$$\mathbf{t} \leq \tau \quad \Rightarrow \quad \tau = \mathbf{t}, \tag{3}$$

$$\mathbf{f} \leq \tau \quad \Rightarrow \quad \tau = \mathbf{f}. \tag{4}$$

This is because the empty record [ ] is excluded from consideration by Definition 2.1.

3.2 Representing Clauses

A clause is a disjunction of literals. A literal is either a propositional variable, or a negation of a propositional variable. Propositional variables $A_1, \ldots, A_k$ are represented by labels $1, \ldots, k$. A clause (i.e., a disjunction of literals)

$$C \equiv A_{i_1} \vee \ldots \vee A_{i_m} \vee \neg A_{j_1} \vee \ldots \vee \neg A_{j_n},$$

where $\{i_1, \ldots, i_m\} \cap \{j_1, \ldots, j_n\} = \emptyset$ and $\{i_1, \ldots, i_m\} \cup \{j_1, \ldots, j_n\} \subseteq \{1, \ldots, k\}$, is represented as a type

$$C^* \equiv_{df} [\, i_1 : \mathbf{t}, \ldots, i_m : \mathbf{t},\ j_1 : \mathbf{f}, \ldots, j_n : \mathbf{f}\,].$$

Proposition 3.1 *A truth assignment $\nu : \{A_1, \ldots, A_k\} \longrightarrow \{\boldsymbol{t}, \boldsymbol{f}\}$ satisfies a clause C iff $i : \nu(A_i)$ occurs in C^* for some $i \in \{1, \ldots, k\}$.*

Proof. ν satisfies C iff for some propositional variable A_i one has:

- either $\nu(A_i) = \mathbf{t}$ and $C = \ldots \vee A_i \vee \ldots$; by definition of C^*, $C = \ldots \vee A_i \vee \ldots$ iff $i : \mathbf{t}$ occurs in C^*,
- or $\nu(A_i) = \mathbf{f}$ and $C = \ldots \vee \neg A_i \vee \ldots$; by definition of C^*, $C = \ldots \vee \neg A_i \vee \ldots$ iff $i : \mathbf{f}$ occurs in C^*. $\square$

Proposition 3.2 *Let $C^* \leq \tau$ for some type τ. Then $\tau \subseteq C^*$, where $[i : \sigma_i]_{i=1}^m \subseteq [j : \tau_j]_{j=1}^n$ iff $\{i : \sigma_i\}_{i=1}^m \subseteq \{j : \tau_j\}_{j=1}^n$.*

(Recall that $\tau \neq [\ ]$, by Definition 2.1.)

Proof. By definition of record subtyping, $C^* \leq \tau$ implies that $C^* \equiv [\ldots i_1 : \sigma_1, \ldots, i_p : \sigma_p, \ldots] \leq [i_1 : \tau_1, \ldots, i_p : \tau_p] \equiv \tau$ and $\sigma_j \leq \tau_j$, where $\sigma_j \in \{\mathbf{t}, \mathbf{f}\}$. The conclusion now follows immediately by (3), (4). $\square$

3.3 Satisfiability of a Propositional Formula

Definition 3.3 The *translation* of a propositional formula in CNF (i.e., a conjunction of clauses)

$$\Phi \equiv \bigwedge_{i=1}^{l} C_i$$

is a set of subtyping judgments

$$\Phi^* \equiv_{df} \left\{ C_i^* \leq \beta_i,\ \alpha \leq \beta_i \right\}_{i=1}^{l}, \tag{5}$$

where α, β_i are fresh pairwise distinct type variables. A *solution* to Φ^* is a substitution of types for free type variables making all subtyping judgments in Φ^* true. $\square$

The main technical result we need is as follows.

Lemma 3.4 *Φ is satisfiable if and only if Φ^* has a solution.*

Proof.

- $(\Rightarrow)$. Let an assignment $\nu(A_i) = v_i$ of truth values $v_i \in \{\mathbf{t}, \mathbf{f}\}$ satisfy Φ. Then, by Proposition 3.2, the substitution

$$\begin{array}{rcl} \alpha & \leftarrow & [1 : v_1, \ldots, i : v_i, \ldots, k : v_k], \\ \beta_i & \leftarrow & [i : v_i \mid \nu(A_i) = v_i \text{ and } i : v_i \in C_i^*] \end{array}$$

  is a solution to Φ^*.

- ($\Leftarrow$). Suppose, Φ^* has a solution $\alpha \leftarrow \tau_\alpha$, $\beta_i \leftarrow \tau_{\beta_i}$. Construct the truth assignment ν by defining for every $i = 1, \ldots, l$

  - $\nu(A_i) = \mathbf{t}$ if $i : \rho$ occurs in τ_α for some record type ρ,
  - $\nu(A_i) = \mathbf{f}$ if $i : \phi$ occurs in τ_α for some functional type ϕ,
  - $\nu(A_i) = \mathbf{t}$ if neither of the above.

  We claim that such a ν satisfies $\Phi \equiv \wedge_{i=1}^{l} C_i$, i.e., ν satisfies C_i for every $i = 1, \ldots, l$. Since $\alpha \leftarrow \tau_\alpha$, $\beta_i \leftarrow \tau_{\beta_i}$ is a solution to Φ^*, by Proposition 3.2,

  $$C_i^* \leq \tau_{\beta_i} \text{ implies } \tau_{\beta_i} \subseteq C_i^*.$$

  Consequently, by definition of C^*, τ_{β_i} is a nonempty record consisting of elements $j : v_j$ with $v_j \in \{\mathbf{t}, \mathbf{f}\}$ and with different j's.

  Let $j : v_j$ occur in τ_{β_i}. Since $\tau_\alpha \leq \tau_{\beta_i}$ and $j : v_j$ occurs in τ_{β_i}, the type τ_α should contain $j : \rho$ for some type $\rho \leq v_j$. Recall that v_j is either $\mathbf{t}$ or $\mathbf{f}$.

  - If $v_j = \mathbf{t}$, then ρ must be a record type, hence, by construction of ν, $\nu(A_j) = \mathbf{t}$. But $j : v_j = j : \mathbf{t} \in C_i^*$ means that $C_i = \ldots \vee A_i \vee \ldots$. Therefore, ν satisfies C_i.
  - If $v_j = \mathbf{f}$, then ρ must be a functional type, hence, by construction of ν, $\nu(A_j) = \mathbf{f}$. But $j : v_j = j : \mathbf{f} \in C_i^*$ means that $C_i = \ldots \vee \neg A_i \vee \ldots$. Therefore, ν satisfies C_i.

  Therefore the assignment ν satisfies C_i, and the proof is finished. □

Since the solvability of a system of subtyping judgments is equivalent to the solvability of a single judgment (by using records), and the translation * of propositional formulas into systems of subtype constraints (5) is obviously polynomial time computable, we thus proved our main result:

Theorem 3.5 *The problem of determining whether a given subtype judgment containing free variables is satisfiable by some type assignment to its free variables in the system of Definitions 2.1, 2.2 is NP-hard.* □

Remark. The proof above uses the number of record fields labels equal to the number of propositions in an input propositional formula. In fact, just two labels, say 0 and 1, suffice. Instead of flat records $[1 : v_1, \ldots, k : v_k]$ used in the proof we could have used *nested records* like

$$[0.[0.[0.u_{000}, 1.u_{001}], 1.[0.u_{010}, 1.u_{011}]], 1.[0.[0.u_{100}, 1.u_{101}], 1.[0.u_{110}, 1.u_{111}]]],$$
$$[0.[0.[0.v_{000}, 1.v_{001}], 1.[0.v_{010}, 1.v_{011}]], 1.[0.[0.v_{100}, 1.v_{101}], 1.[0.v_{110}, 1.v_{111}]]].$$

It is clear that two these record types are in the subtype relation iff for all $i, j, k \in \{0, 1\}$ one has $u_{ijk} \leq v_{ijk}$.

Thus the NP-hardness result holds already in case of a two-label set. In contrast, for just one label the SSCP is deterministic polynomial (even linear) time decidable, as we discuss it below. This follows from the linearity of typability of an untyped term by simple types. □

Remark. Another generalization comes from the fact that the particular case of *SATISFIABILITY*, *3-SATISFIABILITY*, restricted to formulas with at most three literals per clause is also NP-complete. It follows that the SSCP remains NP-hard for systems of constraints containing at most three fields per record. □

Remark. If the empty record type [ ] is excluded from the type system of (Palsberg 1995) (by adding a type constant or type variables to make the set of types non-empty), the SSCP becomes NP-hard. The proof is by simplification of the argument given above Indeed, it suffices to represent the truth values as $\mathbf{t} = [1 : \gamma]$ and $\mathbf{f} = [2 : \gamma]$ (i.e., without functional type constructor). Without the empty record type, the properties (3), (4) hold, and the remainder of the proof works without any substantial changes. It follows that deriving type information is *strictly more complicated* than deriving partial type information, although, both problems are important and deserve attention.

By the same argument, the NP-hardness result holds for the purely record types with two incomparable type constants (to model $\mathbf{t}$ and $\mathbf{t}$), and the empty record type [ ] excluded. □

4 Discussion

(Kozen et al. 1994, Hoang & Mitchell 1995, Palsberg 1995) show that satisfiability of systems of subtype inequalities is polynomial time equivalent to the type reconstruction problem (i.e., given a term can it be assigned some type?). By similar arguments we can prove that in any reasonable type system based on functional+record types with subtyping as defined in Definitions 2.1, 2.2 the type reconstruction problem is polynomial time equivalent to the SSCP, hence also is NP-hard.

Therefore, unlike the results of (Kozen et al. 1994, Palsberg 1995), which show deterministic cubic time tractability of the SSCP for either functional or record types (separately), our Main Theorem 2.3 shows that subtyping and type reconstruction in presence of *both functional and record* types is NP-hard, i.e., presumably intractable. It follows that the automata-theoretic decidability techniques of (Kozen et al. 1994, Palsberg 1995) (for the functional and record subtyping, separately) do not carry over straightforwardly to the combined case of functional+record types. Extra effort is necessary even to establish decidability of the SSCP for functional+record types. Here we just claim without a proof the following complexity result

Theorem 4.1 *The SSCP for functional+record types of Definitions 2.1, 2.2 is in NEXPTIME.* □

Recall that NEXPTIME is the class of problems solvable by nondeterministic Turing machines in time $O(2^{n^k})$ for some fixed k, where n denotes the length of input.

The proof of Theorem 4.1 proceeds by a tedious pumping argument showing that whenever an instance of SSCP of size n has a solution, then it necessarily has a solution of depth polynomial in n. Thus, given an SSCP instance, it suffices to nondeterministically guess a polynomially deep solution tree (forest), with the resulting tree size $O(2^{poly(n)})$, and to check that it is indeed a solution in deterministic exponential time. This proves the NEXPTIME membership. Of course, this is not a very efficient algorithm. It remains an open problem whether the SSCP for functional+record types is in PSPACE or NP. We believe that more sophisticated data structures, like DAGs and automata on DAGs, may lead to improvement of the above NEXPTIME upper bound to PSPACE, or even NP. It is also possible that the sophisticated techniques of (Tiuryn 1992) may raise the lower bound to PSPACE. This remains an intriguing subject for further investigations.

The Empty Record. (Palsberg 1995) admits the empty record [ ]. In his subtype system this comes naturally, since it allows for a very simple one-alternative definition of types: $\tau ::= [l_1 : \tau_1, \ldots, l_n : \tau_n]$, where the case $n = 0$ corresponds to the base case of the empty record [ ]. If the empty record is excluded, some type constant or variable is needed to make the set of types non-empty. Note that $\tau \leq [\ ]$, i.e., [ ] plays the top type role in Palsberg's object type system.

Different reasons for including or excluding the empty record type may be given. One reason for excluding it is as follows. In the case of just one-element label set $L = \{app\}$ the subtyping relation is *identity*, because there is just one label and no empty record. Consider the set of *simple types* constructed from type variables by using $\to$ only, and translate them into the purely record object types with the unique label *app* by:

$$\begin{array}{rcl} \alpha^{\#} & = & \alpha \quad \text{(i.e., type variables are fixed points)}, \\ (\sigma \to \tau)^{\#} & = & [app : \sigma^{\#} \to \tau^{\#}]. \end{array}$$

λ-terms can also be converted into first-order object terms (due to David McAllester, personal communication, July 1997) in such a way that the following *commutation property* holds: a λ-term t is typable with a simple type τ iff its object conversion $T[t]$ is typable in an appropriately defined associated object type system with type $\tau^{\#}$. In this sense the object type system with the unique label and without the empty record type is a precise counterpart of the type system for the simply typed lambda terms. As for the simple types, both the SSCP and the type reconstruction problem are in PTIME in this case[1]. Typability lies outside the scope of this paper, so precise definitions and proofs for the commutation property will appear elsewhere.

When the empty record is admitted, the precise correspondence with the simply typed terms via the object conversion is lost. Some "senseless" λ-terms untypable with simple types become typable, e.g., $\lambda x.xx$. Probably, the resulting object system becomes the precise counterpart of the system of partial types due to (Thatte 1988), but this needs further investigation. Recall that the top type Ω only carries the least possible information "well-typed", terms do not have "principal" or minimal types, sometimes partially typed terms need dynamic type checking. On the positive side, every λ-term in normal form has a partial type (Palsberg 1993), and each term typable with a partial type strongly normalizes[2]. The following intriguing problem

[1] Recall that the proof of the NP-lower bound of Theorem 2.3 requires at least two labels; here we have just one – *app*.

[2] (Palsberg 1993) refers to an unpublished manuscript of Wand & O'Keefe "Partially typed terms are SN" (December, 1991).

arises naturally. Does there exist a functional+record system of partial types, which: 1) has the PTIME SSCP and type reconstruction problem, 2) types all normal forms, 3) all typable terms are SN, 4) a typable term never goes wrong (appropriately defined)? Could PTIME decidability of such a system be obtained by a generalization of the automata-based decision procedures of (Kozen et al. 1994, Palsberg 1995)?

5 Conclusions

We presented the NP-lower bound for the *Satisfiability of Subtype Constraints Problem* (SSCP) for the types built by using simultaneously the functional $\rightarrow$ and the record $[\ldots]$ type constructors. Earlier research concentrated exclusively either on the SSCP for purely functional, or for purely record types, but not for both simultaneously. Both in the case of the purely functional types (Lincoln & Mitchell 1992, Tiuryn 1992, Kozen et al. 1994) and purely record types (Palsberg 1995), deterministic polynomial time algorithms are possible. In the case of the purely functional types constructed from base types with nontrivial subtyping relation on them the SSCP, in general, is NP-hard (Lincoln & Mitchell 1992) and even PSPACE-hard (Tiuryn 1992) (and, in fact, PSPACE-complete (Frey 1997)). In contrast, our result shows that even without any base types, the SSCP for functional+record types is NP-hard. We give the NEXPTIME upper bound for the problem, but conjecture that by using more sophisticated data structures and more subtle arguments this upper bound can be improved to PSPACE (or even NP). It is also quite possible that the techniques of (Tiuryn 1992) may lead to the PSPACE lower bound. All these topics constitute an interesting and promising direction for future research, together with the investigation of the related partial type systems with possible simplification of the SSCP, as suggested by results of (Kozen et al. 1994, Palsberg 1995).

References

Benke, M. (1993), Efficient type reconstruction in presence of inheritance, *in* 'Mathematical Foundations of Computer Science'93', Vol. 711 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 272–280.

Dwork, C., Kanellakis, P. & Mitchell, J. (1984), 'On the sequential nature of unification', *J. Logic Programming* **1**, 35–50.

Frey, A. (1997), Satisfying subtype inequalities in polynomial space, *in* 'Static Analysis (SAS'97)', Vol. 1302 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 265–277.

Hoang, M. & Mitchell, J. C. (1995), Lower bounds on type inference with subtypes, *in* '22nd ACM Symp. on Principles of Programming Languages (POPL'95)', pp. 176–185.

Kozen, D., Palsberg, J. & Schwartzbach, M. I. (1994), 'Efficient inference of partial types', *J. Comput. Syst. Sci.* **49**, 306–324.

Lincoln, P. & Mitchell, J. C. (1992), Algorithmic aspects of type inference with subtypes, *in* '19th ACM Symp. on Principles of Programming Languages (POPL'92)', pp. 293–304.

O'Keefe, P. M. & Wand, M. (1992), Type inference for partial types is decidable, *in* 'European Symposium on Programming (ESOP'92)', Vol. 582 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 408–417.

Palsberg, J. (1993), 'Normal forms have partial types', *Information Processing Letters* **45**, 1–3.

Palsberg, J. (1995), 'Efficient inference of object types', *Information and Computation* **123**, 198–209. Preliminary version in LICS'94.

Pratt, V. & Tiuryn, J. (96), 'Satisfiability of inequalities in a poset', *Fundamenta Informaticæ* **28**, 165–182.

Thatte, S. (1988), Type inference with partial types, *in* 'International Colloquium on Automata, Languages, and Programming (ICALP'88)', Vol. 317 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 615–629.

Tiuryn, J. (1992), Subtype inequalities, *in* 'Symposium on Logic in Computer Science', pp. 308–315.

Tiuryn, J. (1997), Subtyping over a lattice (abstract), *in* '5th Kurt Gödel Colloquium (KGC'97)', Vol. 1289 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 84–88.

Wand, M. & O'Keefe, P. (1989), On the complexity of type inference with coercion, *in* 'Functional Programming Languages and Computer Architecture'89', pp. 293–298.

INFORMATIK

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Sybtype Constraints is NP-Hard
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda \rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics
MPI-I-97-2-001	D.A. Basin, S. Matthews, L. Viganò	Labelled modal logics: quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D.A. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification
MPI-I-96-2-008	S. Vorobyov	On the decision complexity of the bounded theories of trees
MPI-I-96-2-007	A. Herzig	SCAN and Systems of Conditional Logic
MPI-I-96-2-006	D.A. Basin, S. Matthews, L. Viganò	Natural Deduction for Non-Classical Logics
MPI-I-96-2-005	A. Nonnengart	Auxiliary Modal Operators and the Characterization of Modal Frames