Subtype Inequalities

Jerzy Tiuryn*
Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw
Poland
e-mail: tiuryn@mimuw.edu.pl

Abstract

In this paper we study the complexity of the satisfiability problem for subtype inequalities in simple types. The naive algorithm which solves this problem runs in non-deterministic exponential time for every predefined poset of atomic subtypings. In this paper we show that over certain finite posets of atomic subtypings the satisfiability problem for subtype inequalities is PSPACE-hard. On the other hand we prove that if the poset of atomic subtypings is a disjoint union of lattices, then the satisfiability problem for subtype inequalities is solvable in PTIME. This result covers the important special case of the unification problem which can be obtained when the atomic subtype relation is equality (in this case the poset is a union of one-element lattices).

1 Introduction

The area of type reconstruction in the presence of subtyping is a special case of the broader area which has been recently a subject of intensive research. In many cases it is now a folklore result to reduce some kind of a type reconstruction problem to a certain algebraic problem formulated as the problem of satisfiability of constraints. In the case of type reconstruction for simple types with subtyping the constraints are term inequalities which represent the relation of subtyping (see [4]). The relation of subtyping is obtained by first taking a finite poset C which represents the set of atomic subtypings and then extending it to the set of all simple types $\mathcal{T}_C$ by making it anti-monotone with respect to the left argument and monotone with respect to the right argument of $\rightarrow$. This relation on simple types is what we call in this paper the *subtype partial order*. The problem of satisfiability of subtype inequalities is: given a finite system Σ of inequalities, is it satisfiable in $\mathcal{T}_C$? There are actually two versions of this problem, as we discuss it later, depending on how C is treated. Thus, as we mentioned above, the type reconstruction problem for simply typed lambda calculus with subtyping (abbreviated TRS) is polynomial-time reducible to the problem of satisfiability of subtype inequalities (abbreviated SSI).

We actually belive that the problem of SSI is mathematically more fundamental than the original motivating problem of TRS. So far it is not known whether TRS and SSI are polynomial-time equivalent. From the complexity point of view the latter problem may turn out to be strictly more powerful than the former.

A recent research has entirely concentrated on the problem of TRS ([6,2,3]) leaving the general problem of SSI untouched. Despite this activity not very much is known about the complexity of TRS in general. [6] establishes NP-lower bound for a certain extension of TRS. This result has been recently improved in [3] so that it is now a genuine NP-hardness result for one of the versions of TRS. The situation with upper bounds is even worse. The naive algorithm (the same as for SSI) runs in non-deterministic exponential time. [3] contains an observation which says that if C has a top element then a certain fragment of TRS is PTIME-solvable for the instances where all constants have only atomic types.

For some reasons the authors ([6,2,3]) stick to one version of TRS without acknowledging existence of another which is in our opinion even more important. The first variant of TRS, the one studied in the literature, is where an instance of the problem is not only an untyped term M and a typing of constants but also a poset C of atomic subtypings is a part of the in-

*This work was partly supported by NSF grants CCR-8901647, CCR-9002253 and by Polish KBN grant No. 2 1192 91 01.

0-8186-2735-2/92 $3.00 © 1992 IEEE

put. This corresponds to the situation of *user-defined subtypings*, i.e. subtypings which can be set by a programmer. We refer to this version of the problem as *uniform* TRS. However, we belive that the problem of *built-in subtypings* is more important for applications. It is also more interesting from the complexity point of view, as we explain it later. Built-in subtypings are represented on the level of TRS by fixing the poset C of atomic subtypings. C comes with the definition of the language and cannot be changed by a programmer. Hence, in this case we are dealing with a family of problems C-TRS, one for each finite poset C, rather than with one problem as it is the case with uniform TRS. Actually, uniform TRS can be represented as C-TRS for an infinite poset C which is a union of finite posets, one for each isomorphism class.

Since C-TRS is polynomial-time reducible to uniform TRS, it follows that lower bounds for C-TRS carry over to the uniform case and uper bounds for the uniform case carry over to each C-TRS. Hence, it is more interesting to establish good lower bounds for some C-TRS's and efficient algorithms for C-TRS which are applicable to a class of finite posets C. Should such an algorithm run uniformly over the class — it then gives a rise to an upper bound for the uniform case. This is why we belive that non-uniform problems are more important than the uniform one. The same comments fully apply to the case of SSI.

In this paper we present two results. We prove that over every poset C which is a *crown*, a notion which we explain in the paper, C-SSI is PSPACE-hard. The proof of this result is built on top of a proof of another result from the forthcoming paper [5] which establishes that the retraction problem[1] over every crown is NP-complete.

The second result of our paper is a polynomial-time algorithm for C-SSI which is applicable to the class of all posets C which are disjoint union of lattices. Since our algorithm runs uniformly in polynomial time over these posets, it follows that uniform SSI over the class of posets which are disjoint union of lattices is in PTIME too. This result generalizes the well known case of the unification problem which is obtained when the poset relation is equality — such a poset, being a disjoint union of one-element lattices, is covered by our result. Our algorithm solves the decision problem without constructing a solution of the system of inequalities. We don't know whether there are succint ways of representing such solutions so that, at least some of these solutions may be computed in polynomial time, as this is the case with unification (see [1]). It follows that the type reconstruction problem for simply typed lambda calculus with subtyping is decidable in polynomial time for the class of atomic subtypings which form a disjoint union of lattices. This corollary can be viewed as a practical hint for language designers.

[1] The retraction problem over C is: given a poset $P \supseteq C$, does P retract to C, i.e. is there a monotonic and idempotent mapping $f : P \to C$?

2 The Problem

Let C be a finite poset. The elements of C are constant symbols of the signature which in addition contains a binary operation symbol $\to$. Let $\mathcal{T}_C$ be the term algebra over this signature. The elements of $\mathcal{T}_C$ are finite terms without variables (the reader should think of them as finite binary trees, whose leaves are labeled with constants from C). The carrier of $\mathcal{T}_C$ is partially ordered by extending the order from C to all trees by the following rule

$$(t_1 \to t_2) \leq (r_1 \to r_2) \quad \text{iff} \quad r_1 \leq t_1 \text{ and } t_2 \leq r_2$$

We call $\leq$ introduced above the *subtype partial order* induced by C. If the partial order on C is equality, then the induced partial order on $\mathcal{T}_C$ is equality too.

A *system* Σ of *inequalities* is a finite set of formulas of the following form $\Sigma = \{\tau_1 \leq \rho_1, \ldots, \tau_n \leq \rho_n\}$, where τ's and ρ's are terms over the above signature with variables from set V. Σ is said to be *flat* if every term in Σ is of size 1, i.e. it is either a constant symbol or a variable. Σ is said to be *satisfiable* in $\mathcal{T}_C$ if there is a valuation $v : V \to \mathcal{T}_C$ such that $\tau_i[v] \leq \rho_i[v]$ holds in $\mathcal{T}_C$ for all i.

We consider the following problems for satisfiability of subtype inequalities (abbreviated SSI).

(C-SSI) *Given a system Σ of inequalities, is it satisfiable in $\mathcal{T}_C$?*

(UNIFORM-SSI) *Given a finite poset C and a system Σ of inequalities, is Σ satisfiable in $\mathcal{T}_C$?*

(C-FLAT-SSI) *Given a flat system Σ of inequalities, is it satisfiable in $\mathcal{T}_C$?*

(UNIFORM-FLAT-SSI) *Given a finite poset C and a flat system Σ of inequalities, is Σ satisfiable in $\mathcal{T}_C$?*

The problems C-SSI and C-FLAT-SSI are parameterized with a finite poset C.

3 A Lower Bound

Flat problems are essentially about solving inequalities in C alone.[2] Clearly every flat problem is in NP.

For $N \geq 2$, let N-crown, C_N, be a poset with $2N$ elements $0, 1, \ldots, 2N-1$ ordered as follows. For $0 \leq i < N$,

$$2i \leq 2i+1$$

$$2i \leq 2i-1$$

In the above formulas counting is modulo $2N$, e.g. -1 denotes $2N-1$.

The following result has been proved in [5].

Theorem 1 *([5]) For every $N \geq 2$, N-crown-FLAT-SSI is NP-complete, hence UNIFORM-FLAT-SSI is NP-complete.*

In this paper we prove.

Theorem 2 *For every $N \geq 2$, N-crown-SSI is PSPACE-hard. Thus the uniform SSI problem is PSPACE-hard too.*

For $N \geq 3$, it is even PSPACE-hard for the systems of inequalities which are of one of the following forms: $i \leq x$, $x \leq y$, and $x = (y \rightarrow z)$, where i is an element of the crown and x, y, z are variables.

It is not known whether n-crown-SSI is in PSPACE. The best upper bound known for C-SSI problems in general is non-deterministic exponential time.[3]

For some posets C it is possible to find a finite set of possible 'obstacles' such that for a flat system Σ, Σ has a solution in C iff no obstacle is derivable from Σ. For example, for every lattice C the 'complete' set of obstacles is obtained by taking inequalities $c \leq c'$, for $c, c' \in C$ which do not hold in C. Let us observe that for a poset C for which there exists such a finite complete set of obtacles, C-FLAT-SSI is in PTIME, provided "derivability from Σ" can be checked in polynomial time. Theorem 1 implies that for no N-crown such a finite complete set of feasible obstacles is possible, unless P=NP.

Theorem 2 adds another impossibility result. Let us assume for a moment that we can introduce a succint way of denoting terms such that for every system Σ of subtype inequalities, if Σ is satisfiable in $\mathcal{T}_C$, then it has a solution in $\mathcal{T}_C$ whose succint notation is of size polynomial in $|\Sigma|$. Then it follows that for such C, C-SSI is in NP. For example, for C partially ordered by equality there is such a succint notation – direct acyclic graph (see [1]). It follows from Theorem 2 that, unless PSPACE=NP, there is no succint notation that would work for N-crowns.

The proof of Theorem 2 is by reduction of the problem of satisfiability of quantified Boolean formulas (QBF) to N-crown-SSI.

We sketch the main steps of the proof of Theorem 2 for 3-crown. The proof for other N's is similar. Throughout this section satisfiability refers to $\mathcal{T}_{C_3}$, where C_3 is 3-crown.

Let φ be a Boolean formula in 3-CNF, and let $0 \leq k \leq n$ and let

$$\psi_k = \exists x_n \exists y_n \ldots \exists x_{k+1} \exists y_{k+1} \forall x_k \exists y_k \ldots \forall x_1 \exists y_1 \; \varphi$$

Let us assume that ψ_k has no free variables, i.e. every variable of φ is bound by a quantifier. We show how to construct a system of inequalities Σ_k such that

$$\psi_k \text{ holds} \qquad \text{iff} \qquad \Sigma_k \text{ is satisfiable}$$

The construction of Σ_k is by induction on k, the number of quantifier alternations in ψ_k.

Now we have to pause to recall some details from the construction of the proof of Theorem 1 which is given in [5]. [5] constructs a flat system Σ^φ of inequalities such that the Boolean formula φ is satisfiable iff Σ^φ is satisfiable. The system Σ^φ has three components:

- "double crowns" to simulate truth values
- "locking mechanism" to simulate negation
- a part which encodes clauses of φ

For each propositional variable u of φ there are two double crowns in Σ^φ. Each double crown is a flat system of inequalities such that it has exactly two solutions. These two solutions represent truth values. One double crown, call it *positive* for u, represents truth values assigned to u, while the other double crown, call it *negative* for u, represents truth values assigned to $\neg u$. The two double crowns must be *locked together* to ensure that **true** is assigned to u iff **false** is assigned to $\neg u$.

There is a special naming convention for the variables of double crowns. Variables $[u]^{i,i+1}$ and $[u]^{i,i+2}$, for $0 \leq i \leq 5$, occur[4] in the positive double crown for u and variables with transposed indices are used in the negative double crown for u. These variables are

[2] It is easy to prove that a flat system of inequalities is satisfiable in $\mathcal{T}_C$ iff it is satisfiable in C.

[3] The naive algorithm that trnsforms a system of inequalities into a flat sysyetm of inequalities produces a flat system of exponential size.

[4] Let us recall that counting is modulo 6.

arranged in such a way that the possible solutions for $[u]^{i,j}$ are in $\{i,j\}$. The mechanism which locks the two crowns can be expressed in many ways, one possiblity is

$$[u]^{0,4} \leq [u]^{1,3} \tag{1}$$

The constraint (1) has the following two solutions

$$[u]^{0,4} = 0,\ [u]^{1,3} = 1$$

and

$$[u]^{0,4} = 4,\ [u]^{1,3} = 3$$

As we mentioned earlier there are other, equivalent, ways of locking together positive and negative double rowns for the same propositional variable. For example, (1) can be eqivalently replaced by

$$[u]^{4,2} \leq [u]^{5,1} \tag{2}$$

Let us observe that (2) can be obtained from (1) by transposing indicies, adding 1 to each of them and reversing the inequality sign. We call the locking mechanism (2) obtained in such a way a *successor* of (1).

Finally, Σ^φ has a part that encodes the clauses of φ. For example, the clause $u \vee v \vee w$ can be expressed by the following inequalities

$$[u]^{1,0} \leq \tilde{u} \geq \tilde{w} \leq \tilde{v} \geq [v]^{0,5}$$

$$[w]^{3,1} \geq \tilde{w}$$

where $\tilde{u}, \tilde{v}, \tilde{w}$ are brand new variables, used only in this place of the system. The *successor* set of the above inequalities does the same job of encoding the same clause

$$[u]^{1,2} \geq \tilde{u} \leq \tilde{w} \geq \tilde{v} \leq [v]^{0,1}$$

$$[w]^{2,4} \leq \tilde{w}$$

Now we can resume the definition of Σ_k. In what follows we use a with sub- or super-scripts. These are new variables. We will be also using new variables $[u]^{i,j}_k$, where $0 \leq k \leq n$, $0 \leq i,j \leq 5$ and u is a propositional variable of φ. The variable $[u]^{i,j}_k$ is a version of $[u]^{i,j}$, lifted to level k. The variable a^i_k, which we use below, represents constant i lifted to level k.

Let us first define sets Δ_k, for $0 \leq k \leq n$.

$$\Delta_0 = \{a^{i,j}_{0,0} = a^j_0 \mid 0 \leq i,j \leq 5\} \cup \{a^i_0 = i \mid 0 \leq i \leq 5\}$$

For $k < n$, Δ_{k+1} is Δ_k plus the following equations[5] (3), (4), (5), (6), with i,j ranging over $\{0,\ldots,5\}$.

[5] An equation $\tau = \rho$ is expressed in this formalism by two inequalities $\tau \leq \rho$, $\rho \leq \tau$.

$$a^i_{k+1} = a^{i+1}_k \rightarrow a^i_k \tag{3}$$

For $k+1 < p \leq n$ and $z_p \in \{x_p, y_p\}$,

$$[z_p]^{i,j}_{k+1} = [z_p]^{j+1,i+1}_k \rightarrow [z_p]^{i,j}_k \tag{4}$$

For $1 \leq p \leq k$,

$$a^{i,j}_{p,k+1} = a^{j+1,i+1}_{p,k} \rightarrow a^{i,j}_{p,k} \tag{5}$$

$$a^{i,j}_{k+1,k+1} = a^j_{k+1} \tag{6}$$

For every $k \geq 0$, let Σ^φ_k be the system of inequalities obtained from Σ^φ by replacing every variable $[u]^{i,j}$ of Σ^φ by $[u]^{i,j}_k$, and replacing the constant $0 \leq i \leq 5$ by a (new) variable a^i_k. Hence, there are no constants in Σ^φ_k.

Finally we set $\Sigma_{k+1} = \Delta_{k+1} \cup \Sigma^\varphi_{k+1}$ plus the equation (7) with i,j ranging over $\{0,\ldots,5\}$ and $1 \leq p \leq k+1$.

$$[x_p]^{i,j}_{k+1} = a^{i,j}_{p,k+1} \tag{7}$$

Let $V_k = \{x_{k+1}, y_{k+1}, \ldots, x_n, y_n\}$. Now we can state the main technical lemma.

Proposition 3 *For all $k \geq 0$, and for every function $\xi : V_k \rightarrow \{0,1\}$, $\Sigma_{k+1} \cup \{[v]^{0,1}_k = a^{\xi(v)}_k \mid v \in V_{k+1}\}$ is satisfiable iff for every $i \in \{0,1\}$, $\Sigma_k \cup \{[v]^{0,1}_k = a^{\xi(v)}_k \mid v \in V_k\} \cup \{[x_{k+1}]^{0,1}_k = a^i_k\}$ is satisfiable.*

Proof: Take Σ_{k+1}. Let u be any propositional variable of φ. The inequalities in Σ^φ_{k+1} compare $[u]^{i,j}_{k+1}$ with some a^l_{k+1}, hence by (3), $[u]^{i,j}_{k+1}$ has to be expanded introducing two new variables. We use a special naming convention for the new variables introduced by this expansion, so that it will be easier to follow the proof. Let us choose the substitution

$$[u]^{i,j}_{k+1} = [u_0]^{j+1,i+1}_k \rightarrow [u_1]^{i,j}_k \tag{8}$$

Let us observe that if we take a double crown for positive $[u]_{k+1}$ in Σ^φ_{k+1}, perform the substitutions (3), (8) and simplify, then, due to the very special arrangement of variables in (3) and the naming convention of (8), we get a pair of double crowns, one for positive $[u_1]_k$ and the other for negative $[u_0]_k$. These double crowns are using of course the lower-level variables a^l_k. Similarly, a double crown for negative $[u]_{k+1}$ translates into a double crown for positive $[u_0]_k$ and a double crown for negative $[u_1]_k$.

By the same argument if we take a locking condition that locks $[u]^{0,4}_{k+1}$ and $[u]^{1,3}_{k+1}$, as described above, then

we get a pair of locking conditions: one locking $[u_1]_k^{0,4}$ and $[u_1]_k^{1,3}$, and the other, the successor of the former one which locks $[u_0]_k^{4,2}$ and $[u_0]_k^{5,1}$.

Similarly, the conditions expressing $[u]_{k+1} \vee [v]_{k+1} \vee [w]_{k+1}$ get translated into a pair of conditions, one expressing $[u_0]_k \vee [v_0]_k \vee [w_0]_k$ and the other expressing $[u_1]_k \vee [v_1]_k \vee [w_1]_k$.

Hence, what we have shown so far is that Σ_{k+1}^{φ} is equivalent to two copies of Σ_k^{φ}, one drawn for u_0's and the other for u_1's.

Let $k+1 < p \leq n$, let's take $z_p = x_p$ in (4) and use (8). We get then

$$[(x_p)_1]_k^{i,j} = [x_p]_k^{i,j} \tag{9}$$

and

$$[(x_p)_1]_k^{j+1,i+1} = [x_p]_k^{j+1,i+1} \tag{10}$$

Since i and j are arbitrary in (9) and (10), it follows that the variables $(x_p)_0$ and $(x_p)_1$ are equated for $k+1 < p \leq n$ and we can assume that we are dealing just with one copy x_p. A similar statement holds for $y_{k+2}, \ldots, y_n$.

By (7) (for $p = k+1$), (8), (6) and (3) we obtain

$$[(x_{k+1})_1]_k^{i,j} = a_k^j \tag{11}$$

and

$$[(x_{k+1})_0]_k^{j+1,i+1} = a_k^{j+1} \tag{12}$$

Substituting $i = 0$, $j = 1$ in (11) yields $[(x_{k+1})_1]_k^{0,1} = a_k^1$. Similarly, substituting $i = 0$, $j = 5$ in (12) we get $[(x_{k+1})_0]_k^{0,1} = a_k^0$. Putting these two together we obtain for $i = 0, 1$,

$$[(x_{k+1})_i]_k^{0,1} = a_k^i \tag{13}$$

By (7), (5) and (8) we obtain for $1 \leq p \leq k$,

$$[(x_p)_1]_k^{i,j} = a_{p,k}^{i,j}$$

and

$$[(x_p)_0]_k^{j+1,i+1} = a_{p,k}^{j+1,i+1}$$

Since i and j are arbitrary in the above equations, we can conclude that for $l = 0, 1$, $1 \leq p \leq k$ and $i, j \leq 5$,

$$[(x_p)_l]_k^{i,j} = a_{p,k}^{i,j}$$

Thus we have shown that Σ_{k+1} is equivalent to (13) plus two copies of Σ_k, one copy in which for every $1 \leq p \leq k$, every x_p and every y_p has been replaced by $(x_p)_0$ and $(y_p)_0$, respectively; and the other in which x_p and every y_p has been replaced by $(x_p)_1$ and $(y_p)_1$. This proves the result. ■

For $0 \leq k \leq n$ let

$$\varphi_k = \forall x_k \exists y_k \ldots \forall x_1 \exists y_1 \; \varphi$$

Hence, free variables of φ_k are among $V_k = \{x_{k+1}, y_{k+1}, \ldots, x_n, y_n\}$. The following result shows correctness of the choice of Σ_k.

Proposition 4 *For every $0 \leq k \leq n$ and for every valuation $\xi : V_k \to \{0,1\}$, ξ satisfies φ_k iff $\Sigma_k \cup \{[v]_k^{0,1} = a_k^{\xi(v)} \mid v \in V_k\}$ is satisfiable.*

Proof: The proof is by induction on k. For $k = 0$, it is enough to observe that φ_0 is φ and the statement follows from the proof of NP-hardness of the flat case.

Now, in order to complete the proof let us take any truth assignment $\xi : V_{k+1} \to \{0,1\}$, and for $i, j \in \{0,1\}$ let $\xi_{i,j} : V_k \to \{0,1\}$ be an extension of ξ such that $\xi_{i,j}(x_{k+1}) = i$ and $\xi_{i,j}(y_{k+1}) = j$. Then we have

$$\xi \text{ satisfies } \varphi_{k+1} \tag{14}$$

iff

$$\forall i \in \{0,1\} \exists j \in \{0,1\} \xi_{i,j} \text{ satisfies } \varphi_k \tag{15}$$

iff $\forall i \in \{0,1\} \exists j \in \{0,1\}$

$$\begin{aligned} \Sigma_k \;\cup\; & \{[v]_k^{0,1} = a_k^{\xi(v)} \mid v \in V_k\} \\ \cup\; & \{[x_{k+1}]_k^{0,1} = a_k^i \;,\; [y_{k+1}]_k^{0,1} = a_k^j\} \\ & \text{is satisfiable} \end{aligned} \tag{16}$$

iff $\forall i \in \{0,1\}$

$$\begin{aligned} \Sigma_k \;\cup\; & \{[v]_k^{0,1} = a_k^{\xi(v)} \mid v \in V_k\} \\ \cup\; & \{[x_{k+1}]_k^{0,1} = a_k^i\} \text{ is satisfiable} \end{aligned} \tag{17}$$

iff

$$\Sigma_{k+1} \cup \{[v]_k^{0,1} = a_k^{\xi(v)} \mid v \in V_{k+1}\} \text{ is satisfiable} \tag{18}$$

Equivalence of (15) and (16) follows from the induction assumpition. Equivalence of (17) and (18) follows from Proposition 3. ■

Theorem 2 now follows immediately from Proposition 4 except the special form of inequalities. To get the desired form change Δ_0 in the above construction to

$$\begin{aligned} \Delta_0' \;=\; & \{1 \leq a_0^1,\; 3 \leq a_0^3,\; 5 \leq a_0^5\} \cup \\ & \{a_0^0 \leq a_0^1,\; a_0^0 \leq a_0^5,\; a_0^2 \leq a_0^1,\; a_0^2 \leq a_0^3, \\ & a_0^4 \leq a_0^3,\; a_0^4 \leq a_0^5\} \cup \\ & \{a_{0,0}^{i,j} = a_0^j \mid 0 \leq i, j \leq 5\} \end{aligned}$$

4 A Polynomial Time Algorithm

Although Theorem 2 has negative flavor, as far as the complexity of the general satisfiability problem is concerned, it is possible that over certain posets or even over certain classes of posets the problem has more feasible decision algorithms. One important class of posets for which there has been known a polynomial-time algorithm is the class of posets with the order being equality. The satisfiability problem for this class is the well known unification problem (see [1]). A result which we present in this section substantially generalizes the above theorem.

Theorem 5 *For every poset C which is a disjoint union of lattices, C-SSI is polynomial-time decidable. Moreover the decision algorithm is uniformly polynomial-time over the class of such posets, i.e. the following problem is in PTIME: given a finite poset C which a disjoint union of lattices and a system Σ of inequalities, is Σ satisfiable in $\mathcal{T}_C$?*

Hence the type reconstruction problem for simply typed lambda calculus with subtyping is uniformly in PTIME over the class of atomic subtypings which form a disjoint union of lattices.

The polynomial-time algorithm which we present in this paper is a purely decision algorithm, i.e. it gives an answer without constructing a solution. It is based on a technical result for which we need a few definitions.

Let $C = C_1 \cup \ldots \cup C_m$ be a disjoint union of lattices C_i, for $i = 1, \ldots, m$. Let $\mathcal{T}_*$ be the set of terms without variables over the set of constants $\{*_1, \ldots, *_m\}$ and the binary operation symbol $\rightarrow$. The elements of $\mathcal{T}_*$ we call *shapes*. We use a canonical map $()_* : \mathcal{T}_C(X) \rightarrow \mathcal{T}_*(X)$ which erases some information about constants leaving only the information from which component C_i a given constant comes. We call $(\tau)_*$ a shape of τ. It is defined as follows.

$$(c)_* = *_i, \qquad \text{if } c \in C_i$$

$$(x)_* = x$$

$$(\tau \rightarrow \rho)_* = (\tau)_* \rightarrow (\rho)_*$$

For $\sigma \in \mathcal{T}_*$ let

$$\mathcal{T}_\sigma = \{\rho \in \mathcal{T}_C \mid \rho_* = \sigma\}$$

It is easy to show that $\mathcal{T}_\sigma$ is a lattice, for every shape σ.

A system of subtype inequalities $\Sigma = \{\tau_1 \leq \rho_1, \ldots, \tau_n \leq \rho_n\}$ is said to be *weakly satisfiable* if $\Sigma_* = \{(\tau_1)_* = (\rho_1)_*, \ldots, (\tau_n)_* = (\rho_n)_*\}$ is satisfiable in $\mathcal{T}_*$. Weak satisfiability is clearly a necessary condition for satisfiability. It is decidable in polynomial time since it is an instance of the unification problem.

Next we introduce a very weak deduction system, $\vdash_s$, for deriving inequalities. It has one axiom and the following two rules.

$$(\textit{axiom}) \quad \Sigma \vdash_s \tau \leq \sigma \quad (\text{for } \tau \leq \rho \text{ in } \Sigma)$$

$$(\textit{transitivity}) \quad \frac{\Sigma \vdash_s \tau \leq \rho \ , \ \Sigma \vdash_s \rho \leq \sigma}{\Sigma \vdash_s \tau \leq \sigma}$$

$$(\textit{subterm}) \quad \frac{\Sigma \vdash_s \tau_1 \rightarrow \tau_2 \leq \rho_1 \rightarrow \rho_2}{\Sigma \vdash_s \rho_1 \leq \tau_1 \ , \ \Sigma \vdash_s \tau_2 \leq \rho_2}$$

Proposition 6 *Derivability in $\vdash_s$ is decidable in polynomial time, i.e. the problem: Given Σ and $\tau \leq \rho$, is $\Sigma \vdash_s \tau \leq \rho$ derivable, is decidable in time which is polynomial in $|\Sigma|, |\tau|$, and $|\rho|$.*

Proof: Let us observe that If $\Sigma \vdash_s \tau \leq \rho$, then τ and ρ are subterms of terms in Σ. A routine proof by induction on the length of proof of $\Sigma \vdash_s \tau \leq \rho$ is left for the reader. The set of all subterms of terms of Σ is of size $O(|\Sigma|)$. One can systematically write in polynomial time an $|\Sigma| \times |\Sigma|$ Boolean array such that at the τ-th row and ρ-th column there is 1 iff $\Sigma \vdash_s \tau \leq \rho$ holds. ∎

Call Σ *ground consistent* if for all $c, c' \in C$, if $\Sigma \vdash_s c \leq c'$, then $c \leq c'$ holds in C. By Proposition 6, ground consistency is decidable in polynomial time.

For a system Σ of equations, $\Sigma \models \tau = \rho$ stands for the statement: "every solution of Σ satisfies $\tau = \rho$". We first prove an auxiliary result which will be used later.

Proposition 7 *Let C be a poset which is a disjoint union of lattices and let Σ be a flat system of inequalities. Σ is satisfiable in C iff it is weakly satisfiable and ground consistent.*

Proof: Let $C = C_1 \cup \ldots \cup C_m$ be a disjoint union of lattices C_i, for $i = 1, \ldots, m$. We use the above introduced notation concerning shapes. Assume that Σ is weakly satisfiable and ground consistent. Let $1 \leq i \leq m$ and

$$V_i = \{x \in var(\Sigma) \mid \Sigma_* \models x = *_i\}$$

It follows from weak satisfiability of Σ that every variable of Σ belongs to at most one V_i. Let

$$\Sigma_i = \{\xi \leq \xi' \in \Sigma \mid \xi \in V_i \text{ or } \xi' \in V_i\}$$

Again, it follows from weak satisfiability of Σ that it has a solution in C iff every Σ_i has a solution in C_i, for $i = 1, \ldots, m$.

Let us fix $i \leq m$. We construct a solution of Σ_i. For $x \in V_i$, let

$$U_x = \{c \in C_i \mid \Sigma_i \vdash_s x \leq c\}$$

Let $v(x) = g.l.b.(U_x)$.

If $x \leq d \in \Sigma_i$, for some constant d, then $d \in U_x$ and $v(x) \leq d$. If $d \leq x \in \Sigma_i$, for some constant d then, by ground consistency of Σ, d is a lower bound of U_x, hence $d \leq v(x)$. If $x \leq y \in \Sigma_i$, then $U_y \subseteq U_x$ and thus $v(x) \leq v(y)$. Thus v is a solution of Σ_i. ■

The main technical result of this section says that one can lift Proposition 7 to arbitrary systems of inequalities.

Theorem 8 *Let C be a poset which is a disjoint union of lattices and let Σ be a system of inequalities. Σ is satisfiable in $\mathcal{T}_C$ iff it is weakly satisfiable and ground consistent.*

Proof: The left-to-right implication is obvious. The opposite implication is proved by induction on the number of equivalence classes of $\sim$ defined on $var(\Sigma)$ as follows

$$x \sim y \quad \text{iff} \quad \Sigma_* \models x = y$$

If the quotient set $var(\Sigma)/\sim$ has only one element, then we consider the set

$$\hat{\Sigma} = \{\xi \leq \xi' \mid \Sigma \vdash_s \xi \leq \xi' \text{ and } |\xi| = 1 \text{ or } |\xi'| = 1\}$$

It is easy to prove that

$$\Sigma \text{ is satisfiable iff } \hat{\Sigma} \text{ is satisfiable} \qquad (19)$$

In fact, the above two systems have the same solutions. It follows that if τ is a term of depth greater than 1 which occurs in $\hat{\Sigma}$, then it must be a term without variables. Thus $\hat{\Sigma}$ can be viewed as a flat system of inequalities. Since Σ is weakly satisfiable and ground consistent it follows that $\hat{\Sigma}$ is too. Hence, by Proposition 7, it has a solution, and by (19), Σ has a solution too.

Now let us assume that $var\Sigma/\sim$ has $n+1$ elements and let $y \in var(\Sigma)$ be such that for no $z \in var(\Sigma)$ there is a term τ with $|\tau| > 1$, $z \in var(\tau)$ and $\Sigma_* \models \tau = y$. If there is no such a y, then one easily finds a term τ such that $|\tau| > 1$, $z \in var(\tau)$ and $\Sigma_* \models \tau = z$. This would contradict weak satisfiability of Σ_*.

Let

$$[y] = \{z \in var(\Sigma) \mid z \sim y\} = \{y_1, \ldots, y_k\}$$

and consider the set

$$\hat{\Sigma} = \{\xi \leq \xi' \mid \Sigma \vdash_s \xi \leq \xi', \text{ and } \xi \in [y] \text{ or } \xi' \in [y]\}$$

It follows from the choice of y that if τ is a non-variable term which occurs in $\hat{\Sigma}$, then it contains no variables. Moreover all such terms must be of the same shape, say σ. If there is no such terms, then we choose the shape σ arbitrarily.

It follows that $\hat{\Sigma}$ can be viewed as a flat system of inequalities over $\mathcal{T}_\sigma$. It is weakly satisfiable and ground consistent. Let $\hat{v} : [y] \to \mathcal{T}_\sigma$ be a solution of $\hat{\Sigma}$ and let

$$\Sigma_1 = \hat{v}(\Sigma) = \{\hat{v}(\tau) \leq \hat{v}(\rho) \mid \tau \leq \rho \in \Sigma\}$$

In the above definition $\hat{v}$ acts as identity on variables other than those in $[y]$. Let $\sim_1$ be the equivalence relation associated with Σ_1. We claim

$$|var(\Sigma_1)/\sim_1| = n \qquad (20)$$

$$\Sigma_1 \text{ is weakly satisfiable} \qquad (21)$$

$$\Sigma_1 \text{ is ground consistent} \qquad (22)$$

(20) and (21) are easy to prove since $(\Sigma_1)_* = \Sigma_*[\sigma/y_1, \ldots, \sigma/y_k]$, and if $(\Sigma_1)_* \models \tau = \rho$, then $\Sigma_* \models \tau = \rho$.

The rest of the proof is devoted to showing (22). Let us observe that, by induction assumption, (20) – (22) imply existence of a solution v_1 of Σ_1. Then $\hat{v} \cup v_1$ is a solution of Σ, therby proving the theorem.

Let τ be a term such that it contains no constants, every variable of τ occurs exactly once, $var(\tau) \cap var(\Sigma) = \emptyset$, and for some function $\tilde{v} : var(\tau) \to \{*_1, \ldots, *_m\}$, $\tilde{v}(\tau) = \sigma$. Let $\tau_1, \ldots, \tau_k$ be terms obtained from τ by renaming its variables so that they have pairwise disjoint variables and disjoint from $var(\Sigma)$. Let $\Sigma_2 = \Sigma[\tau_1/y_1, \ldots, \tau_k/y_k]$ and let $\eta : \bigcup_{i=1}^{k} var(\tau_i) \to C$ be a function such that $\eta(\tau_i) = \hat{v}(y_i)$, for $i = 1, \ldots, k$. Such a function is uniquely determined by the above conditions.

In order to show ground consistency of Σ_1 we need one more definition. A sequence $\xi_1, \ldots, \xi_{2\ell}$ of terms in $\mathcal{T}_C(X)$ is called an *η-chain* if the following two conditions hold

$$\Sigma_2 \vdash_s \xi_i \leq \xi_{i+1}, \quad \text{for even } i < 2\ell \qquad (23)$$

$$\eta(\xi_i) = \eta(\xi_{i+1}), \quad \text{for odd } i < 2\ell \qquad (24)$$

Equality in (24) represents syntactic identity of terms. Again, we assume that in the above formula η acts as identity on variables other than those in

$\bigcup_{i=1}^{k} var(\tau_i)$. Now, ground consistency of Σ_1 follows immediately from the following two claims.

$$\text{For all terms } \tau, \rho, \text{ if } \Sigma_1 \vdash_s \tau \leq \rho, \quad \text{then there exists an } \eta\text{-chain from } \tau \text{ to } \rho \qquad (25)$$

$$\text{For all constants } a, b \in C, \text{ if there is an } \eta\text{-chain from } a \text{ to } b, \text{ then } a \leq b \text{ holds in } C \qquad (26)$$

The proof of (26) is by an obvious induction on the length of η-chain. We sketch the proof of (25). We use induction on the length of derivation of $\tau \leq \rho$ from Σ_1.

If $\tau \leq \rho$ is in Σ_1, then there is an inequality $\tau' \leq \rho'$ in Σ_2 such that $\eta(\tau') = \tau$ and $\eta(\rho') = \rho$. Thus τ, τ', ρ', ρ is an η-chain.

If the last rule in the derivation of $\tau \leq \rho$ was *transitivity*, then we have $\Sigma_1 \vdash_s \tau \leq \xi$ and $\Sigma_1 \vdash_s \xi \leq \rho$. We have two η-chains: one from τ to ξ and another from ξ to ρ. Their concatenation yields an η-chain from τ to ρ.

Finally, if the last rule was *subterm*, then $\tau \leq \rho$ must have been obtained from, say $\xi \leq \xi'$. Let $\xi_1, \ldots, \xi_{2\ell}$ be an η-chain from ξ to ξ' and assume that it is of a mimimal length. If all terms in that chain are of depth geater than 1, then by applying the rule *subterm* throughout the chain we get an η-chain from τ to ρ (or in the opposite direction). Otherwise, let ξ_j be a term of depth 1. It cannot be a constant nor a variable in $\bigcup_{i=1}^{k} var(\tau_i)$ since $|\xi| > 1$. Hence $\xi_j \in var(\Sigma) - [y]$. If j is odd then since $|\xi| > 1$ and $|\xi'| > 1$, it follows that $1 < j < 2\ell - 1$ and

$$\Sigma_2 \vdash_s \xi_{j-1} \leq \xi_j, \quad \text{and} \quad \Sigma_2 \vdash_s \xi_{j+1} \leq \xi_{j+2}$$

Moreover, $\eta(\xi_j) = \eta(\xi_{j+1})$. Thus ξ_j and ξ_{j+1} are identical variables and we obtain a contradiction since the η-chain can be shortened. The case when j is even is argued similarly. This proves that all terms in the η-chain must be of depth greater than 1, therby completing the proof of (25). ■

References

[1] C. Dwork, P. Kanellakis, and J.C. Mitchell. On the sequential nature of unification. *Journal of Logic Programming*, 1:35 – 50, 1984.

[2] Y. Fuh and P. Mishra. Type inference with subtypes. *Theoretical Computer Science*, 73:155 – 176, 1990.

[3] P. Lincoln and J. Mitchell. Algorithmic problems for type inference with subtypes. (extended abstract). In *to appear in Proc. ACM Symp. Principles of Programming Languages*, 1992.

[4] J.C. Mitchell. Coercion and type inference (summary). In *Proceedings of 11th ACM Symposium on Principles of Programming Languages*, pages 175 – 185, 1984.

[5] V. Pratt and J. Tiuryn. Satisfiability of inequalities in a poset. 1992. To appear.

[6] M. Wand and P. O'Keefe. On the complexity of type inference with coercions. In *Proc. ACM Conf. Functional Programming and Computer Architecture*, pages 293 – 298, 1989.