

# The Complexity of Subtype Entailment for Simple Types

Fritz Henglein and Jakob Rehof  
DIKU, Department of Computer Science  
University of Copenhagen  
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark  
{henglein, rehof}@diku.dk

## Abstract

A subtyping  $\tau \leq \tau'$  is entailed by a set of subtyping constraints  $C$ , written  $C \models \tau \leq \tau'$ , if every valuation (mapping of type variables to ground types) that satisfies  $C$  also satisfies  $\tau \leq \tau'$ .

We study the complexity of subtype entailment for simple types over lattices of base types. We show that:

- deciding  $C \models \tau \leq \tau'$  is coNP-complete.
- deciding  $C \models \alpha \leq \beta$  for consistent, atomic  $C$  and  $\alpha, \beta$  atomic can be done in linear time.

The structural lower (coNP-hardness) and upper (membership in coNP) bounds as well as the optimal algorithm for atomic entailment are new. The coNP-hardness result indicates that entailment is strictly harder than satisfiability, which is known to be in PTIME for lattices of base types. The proof of coNP-completeness gives an improved algorithm for deciding entailment and puts a precise complexity-theoretic marker on the intuitive “exponential explosion” in the algorithm.

Central to our results is a novel characterization of  $C \models \alpha \leq \beta$  for atomic, consistent  $C$ . This is the basis for correctness of the linear-time algorithm as well as a complete axiomatization of  $C \models \alpha \leq \beta$  for atomic  $C$  by extending the usual proof rules for subtype inference. It also incorporates the fundamental insight for understanding the structural complexity bounds in the general case.

## 1. Introduction

### 1.1. Subtype entailment: Relevance and related work

Subtyping is a fundamental concept in the theory of typed programming languages. Its basic principles are typically

studied in extensions of the simply typed lambda calculus. Following this line of research, the present paper studies a problem in the standard system of structural subtyping defined by Mitchell [15, 16] and subsequently studied extensively by many others (see references below.)

In structural subtyping a poset  $P$  of base types is lifted to an order (subtype) relation on structured types. This is done via a subtype logic  $\vdash_P$  which defines derivable judgements of the form  $C \vdash_P \tau \leq \tau'$ , where  $C$  is a finite set of subtype constraints between type expressions and  $\tau, \tau'$  are possibly structured types. The simply typed lambda calculus is then extended with a rule of *subsumption*, which allows any term having type  $\tau$  under the subtype assumptions  $C$  to have any supertype  $\tau'$  of  $\tau$ , i.e., any type  $\tau'$  such that  $C \vdash_P \tau \leq \tau'$ . The presence of subtype assumptions  $C$  in typings of lambda terms is necessitated by the desire to have *principal typings*, see [9, 16].

However, in contrast to, e.g., the simply typed lambda calculus and ML, subtyping systems typically suffer from the problem that the size of principal typings may get intractably large for natural programs of even moderate size, which makes typing information difficult to read and may seriously slow down type inference in polymorphic frameworks. The problem is now widely recognized and has generated a substantial amount of work that aims at *simplifying* typings generated by subtype inference algorithms (see, e.g., [9, 5, 12, 21, 6, 11, 18, 24, 20, 1, 7].) Recently, a number of researchers have independently suggested that the problem should be attacked by introducing stronger systems, based on more powerful, model-theoretic notions of *entailment* ( $\models$ ) rather than syntactic proof relations such as  $\vdash_P$ . Such works include [18, 24, 7, 1, 8]. The rationale is that a more powerful subsumption rule allows more typings to be considered equivalent in the system, and hence more succinct representations of principal typings can be found. Simplification algorithms exploiting this idea typically have to decide a predicate of the form  $C \models_P \alpha \leq \beta$  in order to verify that a simplification step is sound. However, even though several such algorithms have been suggested (see ref-

erences above), no study has so far addressed the problem of the inherent computational complexity of subtype entailment (see, however, work on set-constraint entailment, such as [8, 4].) Indeed no nontrivial complexity results — neither lower bounds nor interesting upper bounds — have been given for subtype entailment. The present paper aims at filling this gap for simple types over lattices.

Whereas the complexity of subtype entailment is largely unstudied, the complexity of subtype *satisfiability* is rather well-understood by now (see [22, 23, 2, 3]). In particular, Tiurny [22] has shown that the satisfiability problem for simple types<sup>1</sup> is PSPACE-hard in general but in PTIME if  $P$  is a lattice (see also [2, 3, 19] for various generalizations.) Since the *entailment* predicate  $C \models_P \tau \leq \tau'$  (does every valuation satisfying  $C$  also satisfy  $\tau \leq \tau'$ ?) is at least as hard to decide as satisfiability in any non-trivial poset<sup>2</sup>, the problem is certainly PSPACE-hard in general, and a naive algorithm will give only a NEXPTIME upper bound.<sup>3</sup> For two reasons, however, it is of particular interest to know the complexity of entailment over *lattices* of base types. Firstly, it is not clear that the problem is harder or just as easy as satisfiability; in particular, it is a *prima facie* possibility that it might be in PTIME.<sup>4</sup> Secondly, virtually all recent proposals for model-theoretic subtyping systems use lattices as models. Thus the lower bounds for subtype satisfiability are inapplicable. Hence, we restrict ourselves in this paper to consideration of predicates of the form  $C \models_L \alpha \leq \beta$  where  $L$  is a lattice.

In addition to the references given above, related work on entailment problems in various set constraint theories should be mentioned; recent papers include [8, 4].

## 1.2. Main results

The main results of this paper are as follows:

1. We prove a characterization theorem for entailment with atomic types (type variables and type constants) over lattices, leading to a complete axiomatization of atomic entailment and a linear time algorithm for deciding  $C \models_L \alpha \leq \beta$  where  $C$  is consistent.

<sup>1</sup>The satisfiability problem is: given a constraint set  $C$  of inequalities between simple types, determine whether there exists a valuation assigning ground types to variables in  $C$  which makes all inequalities in  $C$  true in the ground ordering?

<sup>2</sup>I.e. if the poset  $P$  is not the singleton set, then there are two distinct elements  $b$  and  $b'$  in  $P$  such that  $b \not\leq_P b'$  and the problem  $C \models_P b \leq b'$  is then equivalent to the nonsatisfiability, the complement of the satisfiability problem, for  $C$ .

<sup>3</sup>The problem  $C \models_P \alpha \leq \beta$  can be reduced to a problem with only *atomic* subtype assumptions (i.e., with no constructed types) modulo an exponential expansion of the size of  $C$ .

<sup>4</sup>Note that whereas logical systems with negation sign typically have entailment problems of equal complexity to that of the unsatisfiability problem (by standard reduction of the former to the latter via negation), the situation is different in the rather weak logics of subtyping under study here.

2. We prove that the general entailment problem for structural subtyping for simple types is coNP-complete over any non-trivial lattice. The result shows in which precise sense the intuitive “exponential” explosion incurred by a naive reduction of the general case to the atomic case is inherent. The resulting coNP-algorithm exploits the characterization given for atomic entailment. It contains ideas that could be useful for engineering algorithms to decide entailment. The coNP-hardness result indicates that entailment is harder than satisfiability for subtype inequalities.
3. The entailment problem is coNP-complete in a system with only a single binary *covariant* type constructor. It is also coNP-complete for simple types with additional type constructors, including the contravariant function type constructor. This shows that the presence or absence of contravariance has *no impact* on the complexity of the problem.

The paper is structured as follows. In Section 2 we review basic results on subtyping. These are primarily used to establish the central characterization of entailment for atomic types (Theorem 3.2); this is done in Section 3. In Section 4 we generalize the characterization to entailment for simple types. This is used in our proof that entailment is in coNP in Section 4.1 and in the coNP-hardness proof of entailment in Section 4.2.

## 2. Preliminaries

Fix a lattice  $(L, \leq_L)$  of base types. We assume a denumerable set  $\mathcal{V}$  of type variables. The set of *type expressions* over  $L$ , denoted by  $T_L(\mathcal{V})$ , is ranged over by  $\tau$ , defined by

$$\begin{aligned} \tau & ::= A \mid \tau * \tau' \mid \tau \rightarrow \tau' \\ A & ::= \alpha \mid b \end{aligned}$$

where  $\alpha$  ranges over  $\mathcal{V}$  and  $b$  ranges over the constants in  $L$ . Constants from  $L$  and variables are referred to collectively as *atoms* and are ranged over by  $A$ . If  $\tau \in T_L(\mathcal{V})$  and no variable occurs in  $\tau$ , then  $\tau$  is called a *ground type*. The set of ground types over  $L$  is denoted  $T_L$ . A *constraint set*  $C$  is a finite set of formal inequalities of the form  $\tau \leq \tau'$  with  $\tau, \tau' \in T_L(\mathcal{V})$ .  $C$  is called *atomic* if all inequalities in  $C$  have the form  $A \leq A'$ . We let  $\text{Var}(C)$  and  $\text{Cnst}(C)$  denote, respectively, the set of type variables appearing in  $C$  and the set of constants appearing in  $C$ .

The subtype logic  $\vdash_L$  is used in the definition of entailment below, and it is given in Figure 1. The subtype logic defines provable judgements of the form  $C \vdash_L \tau \leq \tau'$ , meaning that the inequality  $\tau \leq \tau'$  is a provable consequence of the subtype assumptions in constraint set  $C$ .

The following definition explains the main concepts we shall be interested in.

$$\begin{array}{l}
[\text{const}] \quad C \vdash_L b \leq b', \text{ provided } b \leq_L b' \\
[\text{ref}] \quad C \vdash_L \tau \leq \tau \\
[\text{hyp}] \quad C \cup \{\tau \leq \tau'\} \vdash_L \tau \leq \tau' \\
[\text{trans}] \quad \frac{C \vdash_L \tau \leq \tau' \quad C \vdash_L \tau' \leq \tau''}{C \vdash_L \tau \leq \tau''} \\
[\text{prod}] \quad \frac{C \vdash_L \tau_1 \leq \tau'_1 \quad C \vdash_L \tau_2 \leq \tau'_2}{C \vdash_L \tau_1 * \tau_2 \leq \tau'_1 * \tau'_2} \\
[\text{arrow}] \quad \frac{C \vdash_L \tau'_1 \leq \tau_1 \quad C \vdash_L \tau_2 \leq \tau'_2}{C \vdash_L \tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2}
\end{array}$$

**Figure 1. Subtype logic**

**DEFINITION 2.1** (Valuation, satisfaction, entailment) A substitution  $S$  is a function mapping type variables in  $\mathcal{V}$  to types in  $T_L(\mathcal{V})$ , and it is lifted homomorphically to types in the standard way, and it is extended to constraint sets by setting  $S(C) = \{S(\tau) \leq S(\tau') \mid \tau \leq \tau' \in C\}$ .

A ground substitution  $\rho : \mathcal{V} \rightarrow T_L$  is called a *valuation*.

We say that a valuation  $\rho$  *satisfies* an inequality  $\tau \leq \tau'$ , written  $\rho \models_L \tau \leq \tau'$ , if and only if  $\emptyset \vdash_L \rho(\tau) \leq \rho(\tau')$ . If  $C$  is a finite set of inequalities, then we say that  $\rho$  *satisfies*  $C$ , written  $\rho \models_L C$ , if and only if  $\rho \models_L \tau \leq \tau'$  for all  $\tau \leq \tau' \in C$ . We say that  $C$  *entails*  $\tau \leq \tau'$ , written  $C \models_L \tau \leq \tau'$ , if and only if we have

$$\forall \rho : \mathcal{V} \rightarrow T_L. \rho \models_L C \Rightarrow \rho \models_L \tau \leq \tau'$$

□

**DEFINITION 2.2** (Closure, consistency) We say that a constraint set  $C$  is *closed* if

- $\tau_1 \leq \tau' \in C$  and  $\tau' \leq \tau_2 \in C$  imply  $\tau_1 \leq \tau_2 \in C$
- $\tau_1 * \tau_2 \leq \tau'_1 * \tau'_2 \in C$  implies  $\tau_i \leq \tau'_i \in C$  and
- $\tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2 \in C$  implies  $\tau'_1 \leq \tau_1, \tau_2 \leq \tau'_2 \in C$ .

The *closure* of  $C$ , denoted  $cl(C)$ , is the least closed set containing  $C$ .

We say that a set of inequalities is *consistent*, if and only if we have  $b \leq_L b'$  whenever  $b \leq b' \in cl(C)$ . □

Note that, in case  $C$  is atomic, we have  $C$  consistent if and only if

$$\forall b, b' \in L. C \vdash_L b \leq b' \Rightarrow b \leq_L b'$$

In the remainder of this section we give some basic lemmas many of which are standard.

**LEMMA 2.3** (*Substitution Lemma*)

1. If  $C \vdash_L \tau \leq \tau'$ , then  $S(C) \vdash_L S(\tau) \leq S(\tau')$ .
2. If  $C \models_L \tau \leq \tau'$ , then  $S(C) \models_L S(\tau) \leq S(\tau')$

**PROOF** As for 1, see [16]. As for 2, assume  $C \models_L \tau \leq \tau'$  and suppose that  $\rho \models_L S(C)$ , i.e.,  $\rho \circ S \models_L C$ , hence (by assumption)  $\rho \circ S \models_L \tau \leq \tau'$ , i.e.,  $\rho \models_L S(\tau) \leq S(\tau')$ . □

**DEFINITION 2.4** (Matching types, matching substitution, structural set) Two types are *matching* if they have the same shape; in more detail,  $\tau$  and  $\tau'$  match if they are both atoms (possibly distinct) or else  $\tau = \tau_1 * \tau_2$ ,  $\tau' = \tau'_1 * \tau'_2$  (or  $\tau = \tau_1 \rightarrow \tau_2$ ,  $\tau' = \tau'_1 \rightarrow \tau'_2$ ) with  $\tau_1$  matching  $\tau'_1$  and  $\tau_2$  matching  $\tau'_2$ .

A set  $C$  of inequalities is *matching* if it holds for all  $\tau \leq \tau' \in C$  that  $\tau$  matches  $\tau'$ .

A *matching substitution* for  $C$  is a substitution  $S$  such that  $S(C)$  is matching.

If there exists a matching substitution for  $C$ , then we say that  $C$  is *structural*; in that case there is a *most general* matching substitution for  $C$  (see [16] for details), denoted  $M_C$ , with the property that, whenever  $R$  is a matching substitution for  $C$ , then there exists a substitution  $V$  such that  $R = V \circ M_C$ ; here the equality is restricted to hold on variables that occur in  $C$ . □

**LEMMA 2.5** (*Match Lemma*)

1. If  $C$  is atomic, and  $C \vdash_L \tau \leq \tau'$ , then  $\tau$  and  $\tau'$  match.
2. Assume that  $C$  is structural. Then  $C \models_L \tau \leq \tau'$  if and only if  $M_C(C) \models_L M_C(\tau) \leq M_C(\tau')$ .

**PROOF** As for 1, see [16]. As for 2, the implication  $(\Rightarrow)$  follows from the Substitution Lemma. To see the implication  $(\Leftarrow)$ , assume  $M_C(C) \models_L M_C(\tau) \leq M_C(\tau')$  and suppose that  $\rho \models_L C$ . Then it follows from the Match Lemma that  $\rho(C)$  is matching, so there exists a substitution  $V$  such that  $\rho = V \circ M_C$ , and so we have by  $\rho \models_L C$  that  $V \models_L M_C(C)$ , and hence (by the assumption) we have  $V \models_L M_C(\tau) \leq M_C(\tau')$ , i.e.,  $V \circ M \models_L \tau \leq \tau'$ , i.e.,  $\rho \models_L \tau \leq \tau'$ . □

If  $C$  is satisfiable, then  $C$  must be structural, since if some  $\rho$  satisfies  $C$ , then  $\emptyset \vdash_L \rho(C)$  and hence, by the first part of the Match Lemma,  $\rho(C)$  must be matching.

**LEMMA 2.6** (*Decomposition Lemma*)

1. If  $C$  is atomic, then  $C \vdash_L \tau_1 * \tau_2 \leq \tau'_1 * \tau'_2$  if and only if  $C \vdash_L \{\tau_1 \leq \tau'_1, \tau_2 \leq \tau'_2\}$ , and  $C \vdash_L \tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2$  if and only if  $C \vdash_L \{\tau'_1 \leq \tau_1, \tau_2 \leq \tau'_2\}$

2.  $C \models_L \tau_1 * \tau_2 \leq \tau'_1 * \tau'_2$  if and only if  $C \models_L \{\tau_1 \leq \tau'_1, \tau_2 \leq \tau'_2\}$ , and  $C \models_L \tau_1 \rightarrow \tau_2 \leq \tau'_1 \rightarrow \tau'_2$  if and only if  $C \models_L \{\tau'_1 \leq \tau_1, \tau_2 \leq \tau'_2\}$

PROOF As for the first part, see [16]. As for the second part, the implications from right to left are obvious. As for the other implications, if  $C \models_L \tau_1 * \tau_2 \leq \tau'_1 * \tau'_2$ , and  $\rho \models_L C$ , then  $\emptyset \vdash_L \rho(\tau_1) * \rho(\tau_2) \leq \rho(\tau'_1) * \rho(\tau'_2)$ , and hence (since  $\emptyset$  is atomic) the first part of the lemma shows that  $\emptyset \vdash_L \{\rho(\tau_1) \leq \rho(\tau'_1), \rho(\tau_2) \leq \rho(\tau'_2)\}$ , i.e.,  $\rho \models_L \{\tau_1 \leq \tau'_1, \tau_2 \leq \tau'_2\}$ . The second claim follows by the same reasoning.  $\square$

Notice that the decomposition property does *not* hold for  $\vdash_L$  with non-atomic constraint sets.

For atomic constraint set  $C$  over  $L$ , define the sets

$$\begin{aligned}\uparrow_C(\alpha) &= \{b \in L \cap \text{Cnst}(C) \mid C \vdash_L \alpha \leq b\} \\ \downarrow_C(\alpha) &= \{b \in L \cap \text{Cnst}(C) \mid C \vdash_L b \leq \alpha\}\end{aligned}$$

Let the operations  $\vee$  and  $\wedge$  denote the least upper bound and the greatest lower bound in  $L$ .

LEMMA 2.7 (*Satisfiability*)

1. Let  $C$  be atomic. If  $C$  is structural and consistent, then

- (a) The valuation  $\rho_\wedge = \{\alpha \mapsto \bigwedge \uparrow_C(\alpha)\}_{\alpha \in \text{Var}(C)}$  is a solution to  $C$
- (b) The valuation  $\rho_\vee = \{\alpha \mapsto \bigvee \downarrow_C(\alpha)\}_{\alpha \in \text{Var}(C)}$  is a solution to  $C$

2. A general constraint set is satisfiable if and only if it is structural and consistent.

PROOF See [14, 22] for the first part; the second part is proven in [22].  $\square$

### 3. Atomic entailment

In this section, we consider the predicate  $C \models_L \alpha \leq \beta$  with  $C$  atomic, i.e., every inequality in  $C$  has the form  $A \leq A'$ . We aim at a complete axiomatization of entailment with atomic constraint sets. Note that, in case  $C$  is structural, the entailment problem  $C \models_L \tau \leq \tau'$  can be reduced, in linear time, to the problem  $C' \models_L \alpha \leq \beta$ , where  $\alpha$  and  $\beta$  are fresh variables and with  $C'$  the atomic decomposition (by Lemma 2.6) of the result of applying a matching substitution to the set  $C \cup \{\alpha = \tau, \beta = \tau'\}$ .<sup>5</sup> For the purposes of the following development it is therefore sufficient to consider just entailments of the form  $C \models_L \alpha \leq \beta$ .

<sup>5</sup>Here  $\tau_1 = \tau_2$  is a shorthand for the two inequalities  $\tau_1 \leq \tau_2$  and  $\tau_2 \leq \tau_1$

We begin with a technical lemma. Given atomic constraint set  $C$ , we can regard  $C \cup L$  as a digraph: there is an edge from  $A$  to  $A'$  for every inequality  $A \leq A'$  in  $C$ , and  $L$  defines a digraph by stipulating that there is an edge from  $b$  to  $b'$  whenever  $b \leq_L b'$ .

LEMMA 3.1 Assume  $C$  atomic, let  $b \in L$ , and let  $\alpha$  and  $\beta$  be two distinct variables. Assume that

- (i)  $b \not\leq_C(\beta)$  and
- (ii)  $C \not\vdash_L \alpha \leq \beta$

Then  $\downarrow_C(\beta) = \downarrow_{C[b/\alpha]}(\beta)$

PROOF (Sketch) By Lemma 2.3 we have (since  $\alpha \neq \beta$ ) that  $C \vdash_L b' \leq \beta$  implies  $C[b/\alpha] \vdash_L b' \leq \beta$ , for any  $b' \in L$ , which shows that  $\downarrow_C(\beta) \subseteq \downarrow_{C[b/\alpha]}(\beta)$ .

To prove the inclusion  $\downarrow_{C[b/\alpha]}(\beta) \subseteq \downarrow_C(\beta)$ , we use that, whenever  $b' \in \downarrow_{C[b/\alpha]}(\beta)$ , there must exist a path  $P_{b'}$  in  $C[b/\alpha] \cup L$  (regarded as a digraph) witnessing this fact. The inclusion then follows from the property

- (\*) For any  $b' \in L$  and any path  $P_{b'}$  in  $C[b/\alpha] \cup L$  witnessing  $b' \in \downarrow_{C[b/\alpha]}(\beta)$  there is a path  $P'$  in  $C \cup L$  witnessing  $b' \in \downarrow_C(\beta)$ .

The property (\*) is proven by induction on the length of a path  $P_{b'}$  in  $C[b/\alpha] \cup L$  of shortest length witnessing  $b' \in \downarrow_{C[b/\alpha]}(\beta)$ . The induction proof involves a case analysis over the form of the path  $P_{b'}$  which is tedious but not difficult, and we leave out further details.  $\square$

We can now prove the main result of this section:

THEOREM 3.2 Let  $C$  be atomic, structural and consistent,  $\alpha$  and  $\beta$  distinct variables in  $C$ . Then  $C \models_L \alpha \leq \beta$  if and only if one of the following conditions holds:

- (i)  $C \vdash_L \alpha \leq \beta$ , or
- (ii)  $\bigwedge \uparrow_C(\alpha) \leq_L \bigvee \downarrow_C(\beta)$

PROOF ( $\Rightarrow$ ). Assume  $C \models_L \alpha \leq \beta$  and  $C \not\vdash_L \alpha \leq \beta$  We must then show

$$\bigwedge \uparrow_C(\alpha) \leq_L \bigvee \downarrow_C(\beta) \quad (1)$$

We proceed by contradiction, assuming

$$\bigwedge \uparrow_C(\alpha) \not\leq_L \bigvee \downarrow_C(\beta) \quad (2)$$

Since  $C \models_L \alpha \leq \beta$ , we have by substitutivity of  $\models_L$  (Lemma 2.3) and  $\alpha \neq \beta$  that

$$C[\bigwedge \uparrow_C(\alpha)/\alpha] \models_L \bigwedge \uparrow_C(\alpha) \leq \beta \quad (3)$$

Let  $C^\dagger = C[\bigwedge \uparrow_C(\alpha)/\alpha]$ . By the Satisfiability Lemma (Lemma 2.7) we know, since  $C$  is consistent, that the map

$$\{\alpha \mapsto \bigwedge \uparrow_C(\alpha)\}$$

can be extended to a satisfying valuation for  $C$ . It follows that  $C^\dagger$  is consistent. Then, by the Satisfiability Lemma again, we get that the valuation

$$\{\gamma \mapsto \bigvee \downarrow_{C^\dagger}(\gamma)\}_{\gamma \in \text{Var}(C^\dagger)}$$

satisfies  $C^\dagger$ . By (3) we then have

$$\bigwedge \uparrow_C(\alpha) \leq_L \bigvee \downarrow_{C^\dagger}(\beta) \quad (4)$$

If  $C \models_L \bigwedge \uparrow_C(\alpha) \leq \beta$ , then the Satisfiability Lemma entails  $\bigwedge \uparrow_C(\alpha) \leq_L \bigvee \downarrow_C(\beta)$  via the satisfying map  $\{\gamma \mapsto \bigvee \downarrow_C(\gamma)\}_{\gamma \in \text{Var}(C)}$ , contradicting (2). We must therefore conclude that  $C \not\models_L \bigwedge \uparrow_C(\alpha) \leq \beta$ . Hence,  $\bigwedge \uparrow_C(\alpha) \not\leq \downarrow_C(\beta)$ . This together with the assumption  $C \not\models_L \alpha \leq \beta$  allows us to apply Lemma 3.1, which shows that  $\downarrow_C(\beta) = \downarrow_{C^\dagger}(\beta)$ , and so by (4) we have

$$\bigwedge \uparrow_C(\alpha) \leq_L \bigvee \downarrow_C(\beta) \quad (5)$$

But (5) contradicts (2), thereby proving (1) and hence the implication ( $\Rightarrow$ ).

( $\Leftarrow$ ). If (i) is the case, the result follows immediately, and if (ii) is the case, the result follows because we evidently have

$$C \models_L \alpha \leq \bigwedge \uparrow_C(\alpha) \text{ and } C \models_L \bigvee \downarrow_C(\beta) \leq \beta$$

□

Theorem 3.2 shows that we get a sound and complete axiomatization of the relation  $\models_L$  on *atomic* constraint sets by adding the following rules to  $\vdash_L$ :

$$\frac{C \vdash_L A \leq b_1 \quad C \vdash_L A \leq b_2}{C \vdash_L A \leq b_1 \wedge_L b_2} \quad \frac{C \vdash_L b_1 \leq A \quad C \vdash_L b_2 \leq A}{C \vdash_L b_1 \vee_L b_2 \leq A}$$

$$\frac{C \vdash_L b_1 \leq b_2 \quad b_1 \not\leq_L b_2}{C \vdash_L \tau \leq \tau'}$$

The theorem also shows that the predicate  $C \models_L \alpha \leq \beta$ , where  $C$  is atomic, can be decided very efficiently:

**COROLLARY 3.3** *Assuming  $\vee_L, \wedge_L$  and  $\leq_L$  are constant-time operations, given consistent constraint set  $C$  of size  $n$  (number of symbols in  $C$ ) it can be decided in linear time whether  $C \models_L \alpha \leq \beta$ .*

**PROOF** Note that  $C$ , viewed as a directed graph, has  $O(n)$  vertices and  $O(n)$  edges. Criterion (i) can be decided in time  $O(n)$  by computing the set of nodes reachable from  $\alpha$  in  $C$ . As for criterion (ii), both  $\bigwedge \uparrow_C(\alpha)$  and  $\bigvee \downarrow_C(\beta)$  can be computed in time  $O(n)$ , again by computing the set of nodes reachable from  $\alpha$  in  $C$  and by computing the set of nodes reachable from  $\beta$  along the reverse edges in  $C$ . □

## 4. Entailment with simple types

We have seen in the previous section that entailment has complexity of the same order as the provability relation  $\vdash_L$  when restricted to *atomic* constraint sets. It turns out, however, that this situation changes when constructed types are allowed in constraint sets. Here, the provability relation is still decidable in PTIME (we can reduce the problem to closure of a constraint graph, involving essentially transitive closure), whereas we show below that the entailment problem becomes coNP-complete in the presence of compound types.

As in [13, 17, 18] we consider a type  $\tau$  as a function from strings (called *addresses*) in  $\{f, s, d, r\}^*$  ( $\Lambda$  denotes the empty string) to labels in  $\mathcal{L} = \mathcal{V} \cup L \cup \{\rightarrow, *\}$ . Here  $f, s, d, r$  denote, respectively, first component and second component of a pair and domain and range of a function type. For instance,  $b[p]$  (resp.  $\alpha[p]$ ) is defined and equal to  $b$  (resp.  $\alpha$ ) if and only if  $p = \Lambda$ ,  $\tau[fp]$  is defined if and only if  $\tau = \tau_1 * \tau_2$  and  $\tau_1[p]$  is defined; the value of  $\tau[fp]$  is then the variable, constant or type constructor which can be reached by traversing  $\tau_1$  along the path described recursively by  $p$ , and we identify  $\tau$  with the partial function  $\lambda p. \tau[p]$  with domain  $\text{dom}(\tau)$ . The *parity* of an address  $p$ , denoted  $\pi p$ , is 0 if the number of  $d$ 's in  $p$  is even and 1 otherwise.

### 4.1. Upper bounds

The problem of deciding  $C \models_L \alpha \leq \beta$  with a general constraint set  $C$  is reducible to the atomic case in a straight forward way: expand  $C$  by a most general matching substitution (if  $C$  is not structural, the entailment holds trivially), and decompose the resulting set, using Lemma 2.6, into a set of atomic constraints. This process may result in an exponential expansion of the set.

In this section, we show that general entailment can be reduced to the atomic case *without* expanding the entire constraint set, since we can show that we need in fact only expand the variables  $\alpha$  and  $\beta$ . The reduction is accomplished by considering *p-entailment* (precisely defined below), which captures entailment relations that must hold between corresponding leaf variables, at address  $p$ , in the expansions of  $\alpha$  and  $\beta$ .

By exploiting characterizations of *p-entailment* we then proceed to show that the entailment problem is in coNP.

We first observe that the entailment relation  $\models_L$  restricted to atomic constraint sets is completely determined by *atomic valuations*, i.e., valuations mapping variables to constants in  $L$  (rather than trees in  $T_L$ ). When  $C$  is an *atomic* constraint set we can define *atomic entailment* (written  $C \models_L^{at} \tau \leq \tau'$ ) by requiring only that any *atomic* valuation, which satisfies all inequalities in  $C$ , also satisfies  $\tau \leq \tau'$ .

LEMMA 4.1 *If  $C$  is atomic, then  $C \models_L \alpha \leq \beta$  if and only if  $C \models_L^{at} \alpha \leq \beta$ .*

PROOF To see the implication  $(\Rightarrow)$ , if  $C \models_L \alpha \leq \beta$  and  $\rho \models_L^{at} C$ , then  $\rho \models_L C$ , hence  $\rho \models_L \alpha \leq \beta$ , hence (since  $\rho$  maps every variable to an atom)  $\rho \models_L^{at} \alpha \leq \beta$ .

To see the implication  $(\Leftarrow)$ , let  $\rho$  be a valuation in  $T_L$  and define the valuation  $\rho^b$  in  $L$  to be the modification of  $\rho$  such that  $\rho^b(\alpha) = \rho(\alpha)$  if  $\rho(\alpha)$  is an atom and  $\rho^b(\alpha) = b$  otherwise. One can show by simple case analysis that, if  $\rho \models_L C$  with  $C$  atomic, then  $\rho^b \models_L^{at} C$  for any fixed constant  $b \in L$ . Now suppose that  $C \models_L^{at} \alpha \leq \beta$  with  $C$  atomic, but  $C \not\models_L \alpha \leq \beta$ ; then we can derive a contradiction in all cases using the observation above. For instance, if  $\rho \models_L C$  but  $\rho(\alpha) \not\leq_L \rho(\beta)$  because  $\rho(\alpha)$  is not an atom and  $\rho(\beta)$  is an atom  $b$ , then we choose a constant  $b'$  such that  $b' \not\leq_L b$ ; we have  $\rho^{b'} \models_L^{at} C$ , hence  $\rho^{b'}(\alpha) = b' \leq_L \rho^{b'}(\beta) = b$ , which is a contradiction. The remaining possibilities for  $\rho(\alpha) \not\leq_L \rho(\beta)$  are handled similarly and are left to the reader.  $\square$

A *maximal address* in a type  $\tau$  is an address  $p \in \text{dom}(\tau)$  such that  $\tau[p]$  is a leaf (i.e.,  $p$  is not proper prefix of any address in  $\text{dom}(\tau)$ ). We say that  $p$  is a *maximal address for  $\alpha$  in  $C$*  if  $p$  is a maximal address in  $M_C(\alpha)$  (note that, by Lemma 2.5, any such address is in the domain of  $\rho(\alpha)$  for any valuation  $\rho$  satisfying  $C$ .)

DEFINITION 4.2 (*p-entailment*) If  $p$  is a maximal address for  $\alpha$  and  $\beta$  in  $C$ , then we say that  $C$  *p-entails*  $\alpha \leq \beta$ , written  $C \models_L^p \alpha \leq \beta$ , if and only if  $\rho(\alpha)[p] \leq_{\pi_p} \rho(\beta)[p]$  for every valuation  $\rho$  satisfying  $\rho \models_L C$  and  $\rho(\alpha)[p], \rho(\beta)[p] \in L$ . Here  $\leq_0$  is  $\leq_L$  and  $\leq_1$  its reverse.  $\square$

LEMMA 4.3  *$C \models_L \alpha \leq \beta$  if and only if  $C \models_L^p \alpha \leq \beta$  for every maximal address  $p$  for  $\alpha$  and  $\beta$  in  $C$ .*

PROOF By the Match Lemma, we have  $C \models_L \alpha \leq \beta$  if and only if  $M_C(C) \models_L M_C(\alpha) \leq M_C(\beta)$ ; by the Decomposition Lemma, this is equivalent to  $C' \models_L C''$  where  $C'$  and  $C''$  are the atomic decompositions of, respectively,  $M_C(C)$  and  $M_C(\alpha) \leq M_C(\beta)$ . By Lemma 4.1 it then follows that we have  $C \models_L \alpha \leq \beta$  if and only if  $\rho \models_L \alpha \leq \beta$  for every valuation  $\rho$  mapping every variable  $\gamma$  to a type matching  $M_C(\gamma)$  and with  $\rho \models_L C$ . It follows that the predicate  $C \models_L \alpha \leq \beta$  is completely determined by valuations  $\rho$  such that  $\rho(\alpha)[p]$  and  $\rho(\beta)[p]$  are atoms for every maximal address  $p$  for  $\alpha$  and  $\beta$  in  $C$ . From this the lemma follows easily.  $\square$

If a variable  $S\alpha$  does not get expanded by  $M_C$  (i.e.,  $\Lambda$  is maximal address for  $\alpha$  in  $C$ ), then we say that  $\alpha$  is *atomic in  $C$* . Let  $C^a$  denote the set of atomic inequalities in  $cl(C)$ .

LEMMA 4.4 *Suppose that  $\alpha$  and  $\beta$  are atomic in  $C$ , and  $C$  is structural and consistent. Then  $C \models_L \alpha \leq \beta$  if and only if  $C^a \models_L^{at} \alpha \leq \beta$ .*

PROOF For the implication from right to left, we note that, by Lemma 4.1 it is sufficient to prove that  $C^a \models_L \alpha \leq \beta$  implies  $C \models_L \alpha \leq \beta$ , which clearly holds. For the implication from left to right, let

$$\widehat{C} = \{\tau \leq \tau' \in cl(C) \mid \tau \text{ or } \tau' \text{ is atomic}\}$$

By the decomposition property for  $\models_L$  (Lemma 2.6),  $C$  is equivalent to  $\widehat{C}$ , and so it is sufficient to prove the implication

$$\widehat{C} \models_L \alpha \leq \beta \Rightarrow C_0 \models_L \alpha \leq \beta \quad (6)$$

where  $C_0 = (\widehat{C})^a$ . To prove this implication, assume

$$C_0 \not\models_L \alpha \leq \beta \quad (7)$$

Then, by Theorem 3.2 (which applies since  $C_0$  is an atomic constraint set), we have

$$\bigwedge \uparrow_{C_0}(\alpha) \not\leq_L \bigvee \downarrow_{C_0}(\beta) \quad (8)$$

Let

$$\rho = \{\alpha \mapsto \bigwedge \uparrow_{C_0}(\alpha), \beta \mapsto \bigvee \downarrow_{C_0}(\beta)\}$$

We will show that

$$\rho(C_0) \text{ is consistent} \quad (9)$$

To prove (9) we proceed as follows. We have  $C_0 \not\models_L \alpha \leq \beta$  by (7), and moreover  $\bigwedge \uparrow_{C_0}(\alpha) \notin \downarrow_{C_0}(\beta)$  (since otherwise we must have  $C_0 \models_L^{at} \bigwedge \uparrow_{C_0}(\alpha) \leq \beta$  leading to a contradiction with (8) via the satisfying valuation  $\rho_\vee$  of Lemma 2.7), and hence we have by Lemma 3.1

$$\bigvee \downarrow_{C_0}(\beta) = \bigvee \downarrow_{C_0^\dagger}(\beta) \quad (10)$$

with  $C_0^\dagger = C_0[\bigwedge \uparrow_{C_0}(\alpha)/\alpha]$ . It then follows from Lemma 2.7 that  $\rho$  can be extended to a satisfying valuation for  $C_0$ , and hence  $\rho(C_0)$  is consistent.

The final step is to show that

$$\rho(\widehat{C}) \text{ is consistent} \quad (11)$$

This can be proven from (9) by an argument similar to the one employed by Tiuryn in [22] to prove that any consistent constraint set (over a lattice of base types) is satisfiable. Since the main ideas can be found in [22] we leave out details here, but we note that the main point in the argument (in our case) is to prove that, since  $C$  is consistent and  $\alpha$  and  $\beta$  are atomic in  $C$ , we have  $b \leq b' \in cl(\rho(\widehat{C}))$  if and only if  $b \leq b' \in cl(\rho(C_0))$ ; the argument proceeds by induction in the length of a shortest chain of inequalities witnessing  $b \leq b' \in cl(\rho(\widehat{C}))$  and exploits the closure properties of  $\widehat{C}$ .

By (11) and the equivalence of consistency and satisfiability shown in [22],  $\rho$  can be extended to a satisfying

valuation  $\hat{\rho}$  for  $\hat{C}$ ; by (8) we have  $\hat{\rho}(\alpha) \not\leq_L \hat{\rho}(\beta)$  which shows  $\hat{C} \not\models_L \alpha \leq \beta$ , thereby proving (6).  $\square$

For any address  $p$ , we define special type expressions called  $p$ -templates; the idea of a  $p$ -template is that it is a most general type having  $p$  in its domain. For instance, with  $p = f s f$ , the type  $(\alpha_1 * (\alpha_2 * \alpha_3)) * \alpha_4$  is a  $p$ -template, and with  $p = f f f$ , the type  $((\alpha_1 * \alpha_2) * \alpha_3) * \alpha_4$  is a  $p$ -template. More precisely, the sets  $\Theta_p$  of  $p$ -templates are minimal sets of types satisfying

- No variable occurs twice in any type in  $\Theta_p$  for all  $p$
- $\Theta_\Lambda$  is, in each context of discussion, an infinite set of fresh variables
- $\Theta_{fp} = \{\theta_p * \alpha_s \mid \theta_p \in \Theta_p\}$
- $\Theta_{sp} = \{\alpha_f * \theta_p \mid \theta_p \in \Theta_p\}$
- $\Theta_{dp} = \{\theta_p \rightarrow \alpha_r \mid \theta_p \in \Theta_p\}$
- $\Theta_{rp} = \{\alpha_d \rightarrow \theta_p \mid \theta_p \in \Theta_p\}$

LEMMA 4.5 *Let  $\alpha \leq_0 \beta$  be the inequality  $\alpha \leq \beta$  and let  $\alpha \leq_1 \beta$  be the inequality  $\beta \leq \alpha$ . Let  $C$  be a structural constraint set, and suppose that*

- (i)  $p$  is a maximal address for  $\alpha$  and  $\beta$  in  $C$
- (ii)  $\theta_p^\alpha$  and  $\theta_p^\beta$  are two distinct  $p$ -templates which have no variables in common with each other and no variables in common with  $C$
- (iii)  $\alpha_p = \theta_p^\alpha[p]$  and  $\beta_p = \theta_p^\beta[p]$
- (iv)  $C_p = C \cup \{\alpha = \theta_p^\alpha, \beta = \theta_p^\beta\}$

Then  $C \models_L^p \alpha \leq \beta$  if and only if  $(C_p)^a \models_L^{at} \alpha_p \leq_{\pi_p} \beta_p$ .

PROOF We first show that we have

$$C \models_L^p \alpha \leq \beta \text{ iff } C_p \models_L \alpha_p \leq_{\pi_p} \beta_p \quad (12)$$

To see the implication from left to right in (12), we note that by Lemma 4.3 we have  $C_p \models_L \alpha_p \leq_{\pi_p} \beta_p$  if and only if  $C_p \models_L^{p'} \alpha_p \leq_{\pi_p} \beta_p$  for every maximal address  $p'$  for  $\alpha_p$  and  $\beta_p$  in  $C_p$ . But, since  $p$  is maximal for  $\alpha$  and  $\beta$  in  $C$ , it is easy to see that  $\Lambda$  is the only maximal address for  $\alpha_p$  and  $\beta_p$  in  $C_p$ , and hence it is sufficient to show that  $C \models_L^p \alpha \leq \beta$  implies  $C_p \models_L^\Lambda \alpha_p \leq_{\pi_p} \beta_p$ . This implication is straight-forward and details are left for the reader. To see the implication from right to left, suppose that  $\rho \models_L C$  with  $\rho(\alpha), \rho(\beta) \in L$ ; then it is easy to see that  $\rho$  can be extended to a valuation  $\rho'$  satisfying  $C_p$  and with  $\rho'(\alpha_p) = \rho(\alpha)[p]$  and  $\rho'(\beta_p) = \rho(\beta)[p]$ . Then  $\rho'(\alpha_p) \leq_{\pi_p} \rho'(\beta_p)$  follows from the assumption of the right hand side of (12), and this proves the implication.

Since  $p$  is a maximal address for  $\alpha$  and  $\beta$  in  $C$ , we clearly have  $\alpha_p$  and  $\beta_p$  atomic in  $C_p$ , and therefore Lemma 4.4 shows that we have

$$C_p \models_L \alpha_p \leq_{\pi_p} \beta_p \text{ iff } (C_p)^a \models_L^{at} \alpha_p \leq_{\pi_p} \beta_p. \quad (13)$$

Composing (12) and (13) proves the lemma.  $\square$

As is well known ([16], [22]), testing whether a constraint set  $C$  is structural is reducible to the problem of unifying the set (regarding inequalities in  $C$  as equalities under unification.) It follows that we can test in time  $O(n)$  whether  $C$  is structural, where  $n$  is the number of symbols in  $C$ . Moreover, in case  $C$  is structural, we have  $M_C(\alpha)$  matching  $E_C(\alpha)$ , where  $E_C$  is the most general unifier for  $C$ , for any  $\alpha \in \text{Var}(C)$  (see, e.g., [16, 22]); it follows that, for any variable  $\alpha$  in  $\text{Var}(C)$ , any maximal address  $p$  for  $\alpha$  in  $C$  satisfies that  $|p|$  is  $O(n)$ , where  $|p|$  is the number of symbols in  $p$ .

We can now show:

THEOREM 4.6 *Let  $p$  be a maximal address for  $\alpha$  and  $\beta$  in  $C$ . Then the predicate  $C \models_L^p \alpha \leq \beta$  is decidable in time  $O(n^3)$  where  $n$  is the number of symbols in  $C$ .*

PROOF Recall from [22] that consistency of  $C$  is equivalent to satisfiability of  $C$ , given that  $C$  is structural. To decide  $C \models_L^p \alpha \leq \beta$ , we first decide in linear time whether  $C$  is structural. Moreover, we can decide consistency of  $C$  in time  $O(n^3)$  by computing  $cl(C)$ , which can be done by dynamic transitive closure of a graph representation of  $C$ . We can therefore assume from now on that  $C$  is structural, consistent and hence satisfiable (otherwise the entailment trivially holds.)

Now fix  $p$  to be the given maximal address for  $\alpha$  and  $\beta$  in  $C$ . By Lemma 4.5, it is sufficient to decide the predicate

$$(C_p)^a \models_L^{at} \alpha_p \leq_{\pi_p} \beta_p \quad (14)$$

with  $(C_p)^a$  and  $\alpha_p, \beta_p$  defined as in Lemma 4.5. Computing  $(C_p)^a$  can be done by computing  $cl(C_p)$  which in turn can be done in time  $O(n^3)$ , since  $|p|$  is  $O(n)$  and hence the size of each of  $\theta_p^\alpha$  and of  $\theta_p^\beta$  is  $O(n)$ . Once we have computed  $(C_p)^a$ , we know by Corollary 3.3 that the entailment (14) can be decided in time linear in the size of  $(C_p)^a$ , which is  $O(n^3)$ .  $\square$

In turn, Theorem 4.6 leads to

THEOREM 4.7 *The predicate  $C \models_L \alpha \leq \beta$  is in coNP.*

PROOF It is sufficient to show that the predicate  $C \not\models_L \alpha \leq \beta$  is in NP. By Lemma 4.3 this non-entailment holds if and only if we can find a maximal address  $p$  for  $\alpha$  and  $\beta$  in  $C$  such that  $C \not\models_L^p \alpha \leq \beta$ . The NP-algorithm for deciding non-entailment therefore starts by non-deterministically

guessing a maximal address for  $\alpha$  and  $\beta$  in  $C$ , which will serve as a succinct witness for the non-entailment.

Since the length of any maximal address for  $\alpha$  and  $\beta$  in  $C$  is linearly bounded by the size of  $C$ , guessing  $p$  can be done non-deterministically in polynomial time. Checking that  $p$  is indeed maximal for  $\alpha$  and  $\beta$  in  $C$  is reducible to a deterministic linear time unification problem. Checking that  $p$  is a witness for the non-entailment comes down to checking  $C \not\models_L^p \alpha \leq \beta$ , which can be done deterministically in polynomial time, by Theorem 4.6.  $\square$

A few remarks on Theorem 4.6 and Theorem 4.7 are in place. Theorem 4.6 is used to prove Theorem 4.7, but the former theorem has independent interest (which is why it is not just a lemma.) The theorem says, when composed with Lemma 4.3, that if we want to decide whether  $C \models_L \alpha \leq \beta$ , then we can do so by considering entailments at all the maximal addresses for  $\alpha$  and  $\beta$  *only*, and each such entailment can be decided in cubic time. Now, there may be exponentially many addresses to consider in the worst case, but the interesting fact stated in the theorem is that we need *only* consider addresses for  $\alpha$  and  $\beta$ . This is in sharp contrast to a naive approach where the entire constraint set is expanded under matching and decomposed to atomic constraints. In contrast, the theorem shows the possibility of deciding entailment with only a “demand-driven” form of expansion, since we only need to consider addresses in the expansion of those variables which we are actually interested in. This, in turn, could be useful to engineer, *e.g.*, simplification of constraint sets without expanding the entire set.

In case  $L$  is finite, we have a shorter proof of Theorem 4.7 (without relying on Theorem 4.6); in this case we could guess a path  $p$  and then test, for each pair  $(b, b') \in L^2$  such that  $b \not\leq_L b'$ , whether or not the set  $C \cup \{\alpha = \tau_b, \beta = \tau_{b'}\}$  is consistent, where  $\tau_b$  is  $\theta_p^\alpha$  with  $b$  substituted for  $\theta_p^\alpha[p]$ , and  $\tau_{b'}$  is  $\theta_p^\beta$  with  $b'$  substituted for  $\theta_p^\beta[p]$ . This is just an implementation of the fact that we can express negation of a constraint by explicit enumeration of all facts contradicting it, when the universe of facts is finite.

## 4.2. coNP-hardness

In this section we prove that deciding the predicate  $C \models_L \alpha \leq \beta$  is coNP-hard; this holds for any non-trivial lattice  $L$ , and in the presence of just a co-variant type constructor. This strongly suggests that there is no way we can bypass the exponential expansion incurred by matching a structural set, in the worst case, in that we must guess maximal addresses to decide non-entailment.

A basic idea in the proof presented below is that, by exploiting the characterizations of entailment developed in Section 3 Section 4.1, we can use maximal addresses to encode truth valuations.

**THEOREM 4.8** *For any non-trivial lattice  $L$ , the predicate  $C \models_L \alpha \leq \beta$  is hard for coNP under log-space reductions in the presence of a single binary, co-variant type constructor.*

**PROOF** In the following construction, we assume only the type constructor  $*$ . Moreover, since  $L$  is assumed to be a non-trivial lattice, we can assume that there are two elements  $\perp, \top \in L$  such that  $\perp \leq_L \top$ .

Fix two distinct variables  $\alpha$  and  $\beta$ . Let NENT be the problem:

- Given  $C$ , decide whether  $C \not\models_L \alpha \leq \beta$

We reduce SAT (propositional satisfiability, [10]) to NENT. This shows that NENT is NP-hard, which in turn shows that the problem of deciding  $C \models_L \alpha \leq \beta$  is coNP-hard.

The basic idea is that an address  $p$  defines a truth assignment of an instance of SAT. Given  $n$  variables  $x_1, \dots, x_n$  containing all the propositional variables occurring in an instance of SAT, we say that an address  $p = l_1 \dots l_n \in \{f, s\}^n$  of length  $n$  defines a truth assignment  $T_p$  as follows:  $T_p(x_i) = \text{true}$  if  $l_i = f$  and  $T_p(x_i) = \text{false}$  if  $l_i = s$  (note that  $f$  means *true* and not false in this encoding.) Henceforth we shall think of addresses both as such and as the truth assignments they induce in this fashion.

Let us assume now that we are given an instance  $I$  of SAT, a set of clauses  $\{C_1, \dots, C_m\}$  over  $x_1, \dots, x_n$ . Each clause is a finite disjunction of *atoms*  $A$ . An atom is a propositional variable or its negation. Each clause defines a set of truth valuations which *falsify* the clause; we call this set the *exclusion set* of the clause. Then, a truth valuation  $T \in \{f, s\}^n$  satisfies  $I$ , if and only if  $T$  is not in the exclusion set of any clause in  $I$ .

Given  $I$  we construct an instance  $C(I)$  of NENT (that is, a constraint set) with the following intuition. For every clause  $C_i = A_1 \vee \dots \vee A_k$  in  $I$  we build a set of constraints that *excludes* exactly every address that (when read as a truth assignment) *falsifies* the clause. By excluding we mean that every such address  $p$  satisfies

$$C(I) \models_L^p \alpha \leq \beta$$

so that, by Lemma 4.3,  $p$  is *not* a witness that the NENT-problem has a positive answer. This is done by making sure that  $p$  becomes a maximal address for  $\alpha$  and  $\beta$  in  $C(I)$  such that  $C(I)_p$  contains the inequalities  $\alpha_p \leq \perp$  and  $\top \leq \beta_p$ ; here  $C(I)_p$  and  $\alpha_p, \beta_p$  are as in Lemma 4.5. Furthermore, our construction is such that  $\alpha_p \leq \beta_p$  is not deducible from  $C(I)$ . By our characterizations of entailment, this means that existence or nonexistence in  $C(I)$  of an address  $p$  such that  $\alpha_p \leq \perp$  and  $\top \leq \beta_p$  determines whether  $C(I) \not\models_L \alpha \leq \beta$  or  $C(I) \models_L \alpha \leq \beta$ .

Let us now describe in detail the construction of the set  $C(I)$ . The set of addresses that falsify clause  $C_i$  can be described by a unique *address pattern*  $P_i \in \{f, s, \#\}^n$ .

An address pattern defines the set of addresses that arise by replacing # arbitrarily by either  $f$  or  $s$ . For example,  $s\#\#fs\#\#\#$  is the address pattern that falsifies clause  $x_1 \vee \neg x_4 \vee x_5$  for  $n = 8$ .

Given an address pattern  $P$  we define  $\mathcal{C}(P, \gamma)$  as follows:

$$\begin{aligned}\mathcal{C}(FP', \gamma) &= \{\delta * \delta' \leq \gamma\} \cup \mathcal{C}(P', \delta) \quad (\delta, \delta' \text{ new}) \\ \mathcal{C}(SP', \gamma) &= \{\delta * \delta' \leq \gamma\} \cup \mathcal{C}(P', \delta') \quad (\delta, \delta' \text{ new}) \\ \mathcal{C}(\#P', \gamma) &= \{\delta * \delta \leq \gamma\} \cup \mathcal{C}(P', \delta) \quad (\delta \text{ new}) \\ \mathcal{C}(\Lambda, \gamma) &= \{\top \leq \gamma\}\end{aligned}$$

Similarly, we define  $\mathcal{C}^-(P, \gamma)$ : This is done by *reversing* the inequalities in the first three clauses for  $\mathcal{C}$  above and replacing the last clause by  $\mathcal{C}^-(\Lambda, \gamma) = \{\gamma \leq \perp\}$ .

Let  $P_i$  be the address pattern that falsifies clause  $C_i$ . The constraints generated for  $C_i$  are  $\mathcal{C}(P_i, \beta) \cup \mathcal{C}^-(P_i, \alpha)$ . The constraints  $C(I)$  generated for  $I$  is the union of the constraints generated for the individual clauses in  $I$ .

Now, to see that  $C(I)$  is a reduction of SAT to NENT we have to check:

$$I \text{ is satisfiable if and only if } C(I) \not\models_L \alpha \leq \beta \quad (15)$$

However, by Lemma 4.3 together with Lemma 4.5, we have (15) equivalent to

$$p \text{ satisfies } I \text{ if and only if } C_p^I \not\models_L^{\text{at}} \alpha_p \leq \beta_p \quad (16)$$

for every truth valuation  $p$  on the  $n$  variables of  $I$ ; in (16),  $C_p^I$  is the atomic set  $(C(I)_p)^a$ , defined as in Lemma 4.5, and  $\alpha_p, \beta_p$  likewise.

Inspection the construction of  $C(I)$  shows that there are no chains of inequalities from any variable in any of the constraints in  $\mathcal{C}^-(P_i, \alpha)$  to any variable in any of the constraints in  $\mathcal{C}(P_j, \beta)$ , for all  $i, j$ . Hence,

$$C_p^I \not\models_L \alpha_p \leq \beta_p \quad (17)$$

To prove (16), assume first that  $p$  satisfies  $I$ . Since  $p$  satisfies  $I$ , there can be no pattern  $P_i$  defining the exclusion set of  $C_i$  such that  $p$  matches  $P_i$ . It then follows by construction of  $C(I)$  that

$$\uparrow_{C_p^I}(\alpha_p) = \emptyset = \downarrow_{C_p^I}(\beta)$$

and hence

$$\bigwedge \uparrow_{C_p^I}(\alpha_p) = \top \not\leq_L \perp = \bigvee \downarrow_{C_p^I}(\beta_p) \quad (18)$$

which proves  $C_p^I \not\models_L^{\text{at}} \alpha_p \leq \beta_p$  by Theorem 3.2 together with (17).

Finally, consider the implication from right to left in (16). If  $C_p^I \not\models_L^{\text{at}} \alpha_p \leq \beta_p$ , then we must have

$$\bigwedge \uparrow_{C_p^I}(\alpha_p) \not\leq_L \bigvee \downarrow_{C_p^I}(\beta_p) \quad (19)$$

by Theorem 3.2 and (17). Let  $P_i$  be any pattern which defines an exclusion set for a clause in  $I$ . If  $p$  matches  $P_i$ , then, by construction of  $C(I)$ , we see that  $\alpha_p \leq \perp \in C_p^I$  and  $\top \leq \beta_p \in C_p^I$ , contradicting (19). Hence,  $p$  cannot be in the exclusion set of any clause in  $I$ , and hence  $p$  satisfies  $I$ .  $\square$

By Theorem 4.8 and Theorem 4.7 we have

**COROLLARY 4.9** *The entailment problem for simple types over any non-trivial lattice is coNP-complete under log-space reductions; the problem remains coNP-complete in the presence of a single co-variant type constructor.*

## References

- [1] A. Aiken, E. Wimmers, and J. Palsberg. Optimal representations of polymorphic types with subtyping. Technical Report UCB/CSD-96-909, University of California, Berkeley, July 1996.
- [2] M. Benke. Efficient type reconstruction in the presence of inheritance. In *Mathematical Foundations of Computer Science (MFCS)*, pages 272–280. Springer Verlag, LNCS 711, 1993.
- [3] M. Benke. Some complexity bounds for subtype inequalities. Technical Report TR 95-20 (220), Warsaw University, Institute of Informatics, Warsaw University, Poland, December 1995.
- [4] W. Charatonik and A. Podelski. The independence property of a class of set constraints. In *Conference on Principles and Practice of Constraint Programming*, pages 76–90. Springer-Verlag, 1996. Lecture Notes in Computer Science, Vol. 1118.
- [5] P. Curtis. Constrained quantification in polymorphic type analysis. Technical Report CSL-90-1, Xerox Parc, February 1990.
- [6] J. Eifrig, S. Smith, and V. Trifonov. Sound polymorphic type inference for objects. In *Proceedings OOPSLA '95*, 1995.
- [7] M. Fahndrich and A. Aiken. Making set-constraint program analyses scale. In *Workshop on Set Constraints, Cambridge MA*, 1996.
- [8] C. Flanagan and M. Felleisen. Modular and polymorphic set-based analysis: Theory and practice. Technical Report Rice COMP TR96-266, Rice University, November 1996.
- [9] Y. Fuh and P. Mishra. Polymorphic subtype inference: Closing the theory-practice gap. In *Proc. Int'l J't Conf. on Theory and Practice of Software Development*, pages 167–183, Barcelona, Spain, March 1989. Springer-Verlag.
- [10] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, 1979.
- [11] M. Hoang and J. Mitchell. Lower bounds on type inference with subtypes. In *Proc. 22nd Annual ACM Symposium on Principles of Programming Languages (POPL)*, pages 176–185. ACM Press, 1995.
- [12] S. Kaes. Type inference in the presence of overloading, subtyping and recursive types. In *Proc. ACM Conf. on LISP and Functional Programming (LFP), San Francisco, California*, pages 193–204. ACM Press, June 1992. also in *LISP Pointers*, Vol. V, Number 1, January-March 1992.

- [13] D. Kozen, J. Palsberg, and M. Schwartzbach. Efficient recursive subtyping. In *Proc. 20th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, pages 419–428. ACM, ACM Press, January 1993.
- [14] P. Lincoln and J. Mitchell. Algorithmic aspects of type inference with subtypes. In *Proc. 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), Albuquerque, New Mexico*, pages 293–304. ACM Press, January 1992.
- [15] J. Mitchell. Coercion and type inference (summary). In *Proc. 11th ACM Symp. on Principles of Programming Languages (POPL)*, pages 175–185, 1984.
- [16] J. Mitchell. Type inference with simple subtypes. *Journal of Functional Programming*, 1(3):245–285, July 1991.
- [17] J. Palsberg and P. O’Keefe. A type system equivalent to flow analysis. *ACM Transactions on Programming Languages and Systems*, 17(4):576–599, July 1995.
- [18] F. Pottier. Simplifying subtyping constraints. In *Proceedings ICFP ’96, International Conference on Functional Programming*, pages 122–133. ACM Press, May 1996.
- [19] V. Pratt and J. Tiuryn. Satisfiability of inequalities in a poset. *Studia Logica (to appear)*.
- [20] J. Rehof. Minimal typings in atomic subtyping. To appear in *proceedings POPL ’97, 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France*, January 1997.
- [21] G. S. Smith. Principal type schemes for functional programs with overloading and subtyping. *Science of Computer Programming*, 23:197–226, 1994.
- [22] J. Tiuryn. Subtype inequalities. In *Proc. 7th Annual IEEE Symp. on Logic in Computer Science (LICS), Santa Cruz, California*, pages 308–315. IEEE Computer Society Press, June 1992.
- [23] J. Tiuryn and M. Wand. Type reconstruction with recursive types and atomic subtyping. In M.-C. Gaudel and J.-P. Jouan-  
naud, editors, *Proc. Theory and Practice of Software Development (TAPSOFT), Orsay, France*, volume 668 of *Lecture Notes in Computer Science*, pages 686–701. Springer-Verlag, April 1993.
- [24] V. Trifonov and S. Smith. Subtyping constrained types. In *Proceedings SAS ’96, Static Analysis Symposium, Aachen, Germany*, pages 349–365. Springer, 1996. *Lecture Notes in Computer Science*, vol.1145.