Tractable Constraints in Finite Semilattices

Jakob Rehof and Torben Æ. Mogensen

DIKU, Department of Computer Science,
Universitetsparken 1, DK-2100 Copenhagen Ø, Denmark.
Electronic mail: {rehof, torbenm}@diku.dk
Fax: +45 35 32 14 01

Abstract. We introduce the notion of *definite* inequality constraints involving monotone functions in a finite meet-semilattice, generalizing the logical notion of Horn-clauses, and we give a linear time algorithm for deciding satisfiability. We characterize the expressiveness of the framework of definite constraints and show that the algorithm uniformly solves exactly the set of all meet-closed relational constraint problems, running with small linear time constant factors for any fixed problem. We give an alternative technique which reduces inequalities to satisfiability of Horn-clauses (HORNSAT) and study its efficiency. Finally, we show that the algorithm is complete for a maximal class of tractable constraints, by proving that *any* strict extension will lead to NP-hard problems in any meet-semilattice.

Keywords: Finite semilattices, constraint satisfiability, program analysis, tractability, algorithms.

1 Introduction

Many program analysis problems can be solved by generating a set of constraints over some domain and then solving these. Examples include the binding time analyses described in [11], [2] and [3], the usage count analysis in [17] and the region-size analysis described in [1]. See also Section 6 below.

In this paper we show how to solve certain classes of constraints over finite domains efficiently, and characterize classes that are not tractable. The solution methods can be used as a tool for analysis designers, and the characterization can help the designer recognize when an analysis may have bad worst-case behaviour.

Due to space limitations, details of proofs are left out. Full details can be found in [19].

2 Monotone function problems

Let P be a poset and F a finite set of monotone functions $f : P^{a_f} \rightarrow P$ with $a_f \geq 1$ the *arity* of f. We call the pair $\Phi = (P, F)$ a *monotone function problem* (MFP for short.) Given $\Phi = (P, F)$ we let T_Φ denote the set of *Φ-terms*, ranged over by τ, σ and given by $\tau ::= \alpha \mid c \mid f(\tau_1, \ldots, \tau_{a_f})$, where c ranges over constants in P, and α, β, γ range over a denumerably infinite set $\mathcal{V}$ of variables and f is a function symbol corresponding to $f \in F$. Constants and variables are collectively referred to as *atoms*, and we let A range over atoms. We assume a fixed enumeration of $\mathcal{V}$, $\mathcal{V} = v_1, v_2, \ldots, v_n, \ldots$ For each number $m > 0$ we let $\mathcal{V}_m$ denote the sequence of the m first variables in the enumeration of $\mathcal{V}$. Let $\rho \in S^m$ for some set S. We write $[\rho]_i$ for the i'th coordinate of ρ, $i \in \{1, \ldots, m\}$. Any $\rho \in S^m$ is implicitly considered a mapping $\rho : \mathcal{V}_m \to S$ by defining $\rho(v_i)$ to be $[\rho]_i$, for $i \in \{1, \ldots, m\}$. For $\rho \in S^m, 1 \leq k \leq m$ we let $\rho \downarrow k = ([\rho]_1, \ldots, [\rho]_k) \in S^k$.

A *constraint set* C over Φ is a finite set of formal inequalities of the form $\tau \leq \tau'$ with $\tau, \tau' \in T_\Phi$. The set of distinct variables occurring in a term τ is denoted $Var(\tau)$, and if C is a constraint set, then $Var(C)$ denotes the set of distinct variables occurring in C. In this paper, *we always assume that* $Var(C) = \mathcal{V}_m$ *for some* m. If $\rho \in P^m$, τ a (P, F)-term with $Var(\tau) \subseteq \mathcal{V}_m$, then $[\![\tau]\!]\rho$ is given by $[\![\alpha]\!]\rho = \rho(\alpha)$, $[\![c]\!]\rho = c$, $[\![f(\tau_1, \ldots, \tau_k)]\!]\rho = f([\![\tau_1]\!]\rho, \ldots, [\![\tau_k]\!]\rho)$. The function $[\![\tau]\!]$ is monotone for every (P, F)-term τ.

If C is a constraint set over (P, F), $Var(C) \subseteq \mathcal{V}_m$, $\rho \in P^m$, we say that ρ is a *valuation of* C *in* P; we say that ρ *satisfies* C, written $P, \rho \models C$, iff $[\![\tau]\!]\rho \leq [\![\tau']\!]\rho$ holds in P for every $\tau \leq \tau'$ in C. We say that C is *satisfiable* if and only if there exists a valuation ρ of C in P such that $P, \rho \models C$. The set of *solutions* to C, denoted $Sol(C)$, is the set $\{\rho \in P^m \mid P, \rho \models C, Var(C) = \mathcal{V}_m\}$.

If $\Phi = (P, F)$ is an MFP, then we define the decision problem Φ-SAT to be the following: *Given a constraint set* C *over* Φ, *determine whether* C *is satisfiable.* We measure the size of a term by $|\tau|$, the number of occurrences of symbols (constants, variables and function symbols) in τ; the size of a constraint set C, $|C|$, is the number of occurrences of symbols in C. We assume that $f \in F$ are given by a_f-dimensional *operation matrixes* M_f with $M_f[x_1, \ldots, x_{a_f}] = f(x_1, \ldots, x_{a_f})$. Under this representation, evaluating a function f at given arguments $x_1, \ldots, x_{a_f}$ is a constant time operation, and hence evaluating an arbitrary functional term τ is $\mathcal{O}(|\tau|)$. If P is a lattice, the component P is assumed to be given as the set of elements of P together with an additional operation matrix, $M_\sqcup$, defining the least upper bound of L, i.e., $M_\sqcup[x, y] = x \sqcup y$. From this we can recover the order relation of P, using that $x \leq y$ iff $x \sqcup y = y$. This representation will be referred to as the *matrix representation.* As we shall see, this is also an appropriate representation when P is a semilattice, since this case will be reduced to the case where P is a lattice.

There are many problems Φ for which Φ-SAT is **NP**-hard (every problem Φ-SAT is obviously in **NP**, since we can guess and verify a solution non-deterministically in polynomial time.) For instance, for any non-trivial finite lattice L, the problem $\Phi = (L, \{\sqcap, \sqcup\})$ is **NP**-complete, by reduction from CNF-SAT (propositional satisfiability, [9]), since $(P_1 \wedge \ldots \wedge P_k) \Rightarrow (Q_1 \vee \ldots \vee Q_m)$ is logically satisfiable if and only if $(P_1 \sqcap \ldots \sqcap P_k) \leq (Q_1 \sqcup \ldots \sqcup Q_m)$ is satisfiable in a two-point chain. Also, the structure of the poset P is important for complexity, see [18]. Hence we need to impose restrictions on problems to make them tractable. In the following development we shall generally assume that P is a meet-semilattice. Note, though, that the whole development transfers to join-semilattices by lattice-theoretic dualization.

3 Definite problems

Let $\Phi = (P, F)$ be an MFP. A constraint set C over Φ in which every inequality is of the form $\tau \leq A$, with an atom on the right hand side, is called *definite*. A definite set $C = \{\tau_i \leq A_i\}_{i \in I}$ can be written $C = C_{var} \cup C_{cnst}$ where $C_{var} = \{\tau_j \leq \beta_j\}_{j \in J}$ are the *variable expressions* in C, having a variable (β_j) on the right hand side of $\leq$, and $C_{cnst} = \{\tau_k \leq c_k\}_{k \in K}$ are the *constant expressions* in C, having a constant (c_k) on the right hand side. Note that satisfiability of *definite* inequalities over $\Phi = (\mathbf{2}, \{\sqcap\})$, with $\mathbf{2}$ the two-point boolean lattice, is exactly the HORNSAT problem (satisfiability of propositional Horn clauses [10], [7]) since Horn clauses have the form $P_1 \wedge \ldots \wedge P_n \Rightarrow Q$.

A term τ will be called *simple* if τ is a constant, a variable or has the form $\tau = f(A_1, \ldots, A_m)$ *i.e.*, there are no nested function applications; a constraint set C is called simple if all terms in it are simple. The L-normalization $\rightarrow_L$ transforms a definite set into a simple and definite set: let C be definite, $C = C' \cup \{f(\ldots g(\tau)\ldots) \leq A\}$ with $Var(C) = \mathcal{V}_m$ and τ a tuple of terms, and define $\rightarrow_L$ by

$$C \rightarrow_L C' \cup \{f(\ldots v_{m+1} \ldots) \leq A, g(\tau) \leq v_{m+1}\}$$

Lemma 1. *The reduction $\rightarrow_L$ is strongly normalizing, and if C^* is a normal form of a definite set C, then C^* is definite with $|C^*| \leq 3|C|$.*

Monotonicity guarantees that an L-normalized set is equivalent to the original set:

Lemma 2. *If $C \rightarrow_L C'$ then $Sol(C) = \{\rho' \downarrow m \mid \rho' \in Sol(C')\}$ where $Var(C) = \mathcal{V}_m$.*

Figure 1 of Appendix A gives an algorithm, called D, for solving definite constraints over an MFP $\Phi = (L, F)$ with L a finite lattice [1] Algorithm D exploits L-normalization to achieve linear time worst case complexity. Later, we shall change D slightly into an algorithm $D^{\top}$ which works for meet-semilattices. Correctness of algorithm D follows from the properties: (1) after every update of the current valuation ρ at β, $\rho(\beta)$ increases strictly in the order of L, so in particular, termination follows, since L has finite height; (2) the iteration step finds the least solution μ to the set of *variable expressions* in C, so if any solution to C exists, then μ must also satisfy the *constant expressions*, by monotonicity of all functions $[\![\tau]\!]$.

[1] Algorithm D is similar to the technique of Kildall [16] for fast fixed point computation in data-flow frameworks and also to the linear time algorithm of Dowling and Gallier [7] for solving the HORNSAT-problem. In fact, the iteration step is easily seen to be equivalent to a search for the least fixed point of a monotone operator F on a lattice, since the least fixed point of F is identical to the least post-fixed point of F. We already observed (beginning of Section 3) that definite inequalities strictly subsume Horn clauses. See Section 4.1 and Section 4.2 for more on the connection to Horn clauses.

Theorem 3. *(Correctness) If C has a solution over Φ, then it has a minimal solution, and algorithm D outputs ρ the minimal solution of C, and if C has no solution, then the algorithm fails.*

Linear time complexity for algorithm D can be shown by amortizing the number of times the test of the conditional in the **for**-loop, $L, \rho \not\models \sigma \leq \gamma$, can be executed *in total* on input C, in the worst case. We can assume (Lemma 1) that C is in L-normal form modulo an expansion by a factor 3. Let $h(L)$ denote the height of a finite lattice L, i.e., the maximal length of a chain in L, then we have

Theorem 4. *(Complexity) For fixed MFP Φ algorithm D runs in time $\mathcal{O}(|C|)$ and performs at most $3h(L) \cdot |C|$ basic computations on input C.*

Algorithm D operates uniformly in Φ, so it can be considered a decision procedure for the *uniform* problem: *Given Φ and constraint set C over Φ, is C satisfiable?.* In this case, taking $h(L) = |L|$ in the worst case, $|L|^{a_{max}}$ the size of the function matrixes with a_{max} the maximal arity of functions in Φ, we have input size $N = |L| + |L|^{a_{max}} + |C|$. With log-cost for a matrix look-up we get a maximum cost of $\mathcal{O}(a_{max} \cdot \log |L|)$ for a basic computation, resulting in $\mathcal{O}(a_{max} \cdot \log |L| \cdot |L| \cdot N)$ worst case behaviour for the uniform problem.

Algorithm D generalizes to the cases where the poset is a finite meet-semi-lattice, as follows. Let $P^\top$ denote the lattice obtained from P by adding a top element $\top$, taking $c \leq \top$ for all $c \in P$. We change algorithm D by adding the test **if** $\exists \alpha. \rho(\alpha) = \top$ **then** FAIL at the beginning of the output step. We extend the functions in Φ such that $f(x_1, \ldots, x_{a_f}) = \top$ if any $x_i = \top$. This modification of algorithm D will be referred to as algorithm $D^\top$. Algorithm $D^\top$ is obviously sound, by soundness of algorithm D, and it is a complete decision procedure for semi-lattices, since if P has no top element and the least solution to the variable expressions in C maps a variable to the top element of $P^\top$, then clearly C can have no solutions in P.

4 Relational problems

Inequality constraints are a special case of the much more general framework of relational constraints. A *relational constraint problem* is a pair $\Gamma = (P, \mathcal{S})$ with P a finite poset and $\mathcal{S}$ a finite set of finite relations $R \subseteq P^{a_R}$, with $1 \leq a_R$ (the arity of R.) Any $R \subseteq P^k$ is called a *relation over P*. A *relational constraint set* C over Γ is a finite set of *Γ-terms* of the form $R(A_1, \ldots, A_{a_R})$ where A ranges over variables in $\mathcal{V}$ and constants drawn from P. The size of a constraint term $t = R(A_1, \ldots, A_{a_R})$ is $|t| = a_R$; the size $|C|$ of a constraint set is the sum of the sizes of all terms in C. We say that a constraint set C is *satisfiable* if there exists a valuation ρ of C in P such that $([\![A_1]\!]\rho, \ldots, [\![A_{a_R}]\!]\rho) \in R$ for every term $R(A_1, \ldots, A_{a_R})$ in C. If $Var(C) = \mathcal{V}_m$, then $Sol(C)$ is the set of valuations $\rho \in P^m$ satisfying C. We define the decision problem Γ-SAT to be: *Given a constraint set C over Γ, is C satisfiable?* If $x \in P^m$ and $1 \leq i \leq m$ then $[x]_i$

denotes the i'th coordinate of x. We use the vector notation $\mathbf{x} = (x_1, \ldots, x_m)$, and if $R \subseteq P^m$ we write $R(\mathbf{x})$ as an abbreviation for $R(x_1, \ldots, x_m)$ also when this expression is considered a term.

4.1 Representability

We are interested in the following question: How many relational constraint problems can be (efficiently) solved using algorithm D? This translates to the question: How many problems can be transformed into definite inequality problems and what is the cost of the transformation?

A relation $R \subseteq P^m$ is called *representable* with respect to a constraint problem $\Gamma = (P, \mathcal{S})$ if $R = Sol(C)$ for some constraint set C over Γ. We say that a problem Γ is a *problem with inequality* if the order relation $\leq$ on P is representable with respect to Γ. We say that a problem Γ *has minimal solutions* if $Sol(C)$ has a minimal element with respect to $\leq$ for any constraint set C over Γ with $Sol(C) \neq \emptyset$. If $\Gamma = (P, \mathcal{S})$ with P a meet-semilattice and it holds for all $R \in \mathcal{S}$ that $x, y \in R$ implies $x \sqcap y \in R$, then Γ is said to be a *meet-closed problem*. A relational problem $\Gamma = (P, \mathcal{S})$ is *representable in definite form* if P is a meet-semilattice and there exists an MFP Φ such that for all constraint sets C over Γ there exists a definite set C' over Φ with $Sol(C) = Sol(C')$.

Suppose that $R \subseteq P^m$ is a meet-closed relation, P a meet-semilattice. Then define the *partial* function $H_R : P^m \rightarrow P^m$ by $H_R(\mathbf{x}) = \bigwedge \uparrow_R (\mathbf{x})$ where $\uparrow_R (\mathbf{x}) = \{\mathbf{y} \in P^m \mid \mathbf{y} \geq \mathbf{x}, R(\mathbf{y})\}$ and with H_R *undefined* if $\uparrow_R (x) = \emptyset$. Then H_R is monotone when defined, *i.e.*, $\forall \mathbf{x}\mathbf{y} \in dom(H_R).\ \mathbf{x} \leq \mathbf{y} \Rightarrow H_R(\mathbf{x}) \leq H_R(\mathbf{y})$, and if $\uparrow_R (\mathbf{x}) \neq \emptyset$, then $\mathbf{x} \in dom(H_R)$. Moreover, for all $\mathbf{x} \in dom(H_R)$ one has $H_R(\mathbf{x}) \geq \mathbf{x}$, since every $\mathbf{y} \in \uparrow_R (\mathbf{x})$ satisfies $\mathbf{y} \geq \mathbf{x}$, and so we have $\bigwedge \uparrow_R (\mathbf{x}) \geq \mathbf{x}$.

Lemma 5. *$\mathbf{x} \in R$ if and only if $\mathbf{x} \in dom(H_R)$ and $H_R(\mathbf{x}) \leq \mathbf{x}$*

Using Lemma 5 we can characterize the class of relational problems which can be solved using algorithm D, *i.e.*, the problems which can be expressed by definite inequalities, as follows

Theorem 6. *(Representability) 1. Let $\Gamma = (P, \mathcal{S})$ with P a meet-semilattice. Then Γ is representable in definite form if and only if Γ is meet-closed. In particular, if Γ is meet-closed, then any constraint set C over Γ can be represented by a definite and simple constraint set C' with $|C'| \leq m(m+2) \cdot |C|$ where m is the maximal arity of a relation in $\mathcal{S}$.*

2. Let $\Gamma_{\leq} = (P, \mathcal{S})$ be a relational constraint problem with inequality, P arbitrary poset. Then the following conditions are equivalent: (i) $\Gamma_{\leq}$ has minimal solutions. (ii) $\Gamma_{\leq}$ is meet-closed and P is a meet-semilattice. (iii) $\Gamma_{\leq}$ is representable in definite form.

Observe that property 1 of Theorem 6 can be seen as a strict generalization of the well-known fact that a set $R \subseteq \mathbf{2}^k$ of boolean vectors is definable by a set of propositional Horn-clauses if and only if R is closed under conjunction [2] Under this view, the notion of *definite* inequalities generalizes the notion of Horn-propositions from the boolean case of $\mathbf{2}$ to an arbitrary meet-semilattice. [3]

We have seen that any meet-closed problem can be represented by definite, functional constraints. Conversely, consider *distributive constraint sets* over an MFP, *i.e.*, constraint sets C where all $\tau \leq \tau' \in C$ have the *right hand side* τ' built from distributive functions only [4] Distributive sets strictly include the definite ones, but every distributive set can be represented by a relational set over a meet-closed problem, since the functions in F can be regarded as relations (graphs), hence (Theorem 6 (1)) algorithm D can solve any Φ-SAT problem restricted to distributive constraint sets. In practice it may be convenient to translate a distributive set directly into a definite one, using the auxiliary functions H_g, defined thus: let $g : P^m \to P$ with P a meet-semilattice; then $H_g : P \to P^m$ is the *partial* function given by $H_g(x) = \bigwedge\{y \in P^m \mid x \leq g(y)\}$ with H_g *undefined* if no $y \in P^m$ satisfies $y \leq g(x)$. If P is a lattice, then H_g is a total function, and it is always monotone. We have

Lemma 7. *Let $f : P^n \to P$, $g : P^m \to P$ with g distributive. Then $f(x) \leq g(y)$ if and only if $f(x) \in dom(H_g)$ and $H_g(f(x)) \leq y$.*

The transformation $\to_R$ given by

$$C \cup \{A \leq g(\ldots h(\tau)\ldots)\} \to_R C \cup \{A \leq g(\ldots v_{m+1}\ldots), v_{m+1} \leq h(\tau)\}$$

is analogous to L-normalization and satisfies properties corresponding to Lemma 1 and Lemma 2. For MFP $\Phi = (L, F)$, L meet-semilattice, let $\Phi' = (L, F')$ with $F' = F \cup \bigcup_{g \in F_d}\{H_g^i \mid i = 1 \ldots a_g\}$ where F_d is the set of distributive functions in F and $H_g^i(x) = [H_g(x)]_i$. Using $\to_L$ and $\to_R$ one has by Lemma 7

Proposition 8. *Let $\Phi = (P, F)$ be an MFP, P a meet-semilattice. Then for any distributive constraint set C over Φ there exists a definite and simple constraint set C' over Φ' with $Sol(C) = \{\rho' \downarrow k \mid \rho \in Sol(C')\}$, where $Var(C) = V_k$, and with $|C'| \leq 3|C| + 2n(m+2)$ where n is the number of inequalities in C and m is the maximal arity of a function in F.*

[2] The definability condition for propositional Horn-clauses is a special case of a much more general model theoretic characterization of Horn-definability in first order predicate logic, by which an arbitrary first order sentence ϕ is logically equivalent to a Horn-sentence if and only if ϕ preserves reduced products of models; see [4] or [12] for in-depth treatments of this result. See also [6]

[3] Note that in definite inequalities we are allowed to use any monotone functions, whereas in the special case of Horn-implications we may use only one function, the meet operation. It is easy to see that one cannot, in general, define an arbitrary meet-closed relation using only the meet operation of the semi-lattice, since, for instance, any set C of inequalities in one variable using only the meet function must be *convex* (*i.e.*, $x, y \in Sol(C)$ and $x \leq z \leq y$ imply $z \in Sol(C)$.) So, for instance, taking L to be the three element chain $0 < 1 < \infty$, the subset $\{\infty, 0\}$ is meet-closed but not convex and hence it cannot be defined using just the meet operation.

[4] A function f is distributive if $f(x \sqcap y) = f(x) \sqcap f(y)$.

4.2 Boolean representation

We show how sets of definite inequalities over finite lattices can be translated into propositional formulae, such that there is a direct correspondence between solutions to the propositional system and solutions to the lattice inequalities.

Given a lattice L with $n+1$ elements, we represent each element of L by an element in 2^n. First we number the elements in $L \setminus \{\bot\} = \{l_1, \ldots, l_n\}$. We then represent each element x in L by a vector of boolean values $\phi(x) : L \to 2^n$ where

$$\phi(x) = (b_1, \ldots, b_n), \text{ where } b_i = 1 \text{ iff } l_i \leq x$$

We also define a mapping $\psi : 2^n \to L$:

$$\psi((b_1, \ldots, b_n)) = \bigvee \{l_i \mid b_i = 1\}$$

It is clear that $x = \psi(\phi(x))$ and $v \leq \phi(\psi(v))$. Moreover, both are monotone. Hence, ϕ and ψ form a galois connection between L and 2^n. We will translate definite inequalities over lattice terms into sets of definite inequalities over 2^n. We will assume that we have already transformed the constraints to the form $f(A_1, \ldots, A_{a_f}) \leq A_0$. We translate constraints over the L-variables $v_1, \ldots, v_k$ into sets of constraints over the boolean variables $v_{11}, \ldots, v_{1n}, \ldots, v_{k1}, \ldots, v_{kn}$. We extend ϕ to variables by setting $\phi(v_i) = (v_{i1}, \ldots, v_{in})$ and define $\phi_i(x)$ to be the i'th component of $\phi(x)$.

Even though we don't use an index for $\bot$ in our representation, we will for convenience assign it index 0 and define $\phi_0(x) = 1$ for all $x \in L$. This corresponds to extending the representation vector with an extra bit, which is always 1 (since $\bot$ is $\leq$ all elements in L).

We first generate, for each variable v_i, the set of constraints $v_{ik} \leq v_{ij}$ whenever $l_j \leq l_k$ (we actually need only do so for a set of pairs $l_j \leq l_k$ whose transitive and reflexive closure yields the ordering on L). These constraints will ensure that any solution to the constraint set will be in the image of ϕ. Note that this only works because we are in a lattice, as when we have two solutions to the constraints for a variable, the meet and join of these are also solutions. Even if we use more general Horn-clauses to model the ordering relation, it will be meet-closed, so at best we can extend the construction to meet semi-lattices.

Let the *i'th frontier of f*, $F_i(f)$ be the smallest subset of L^{a_f} such that $\phi_i(f(y_1, \ldots, y_{a_f})) = 1$ iff there exist $(x_1, \ldots, x_{a_f}) \in F_i(f)$, $(x_1, \ldots, x_{a_f}) \leq (y_1, \ldots, y_{a_f})$. This is well-defined, because the set of all $(x_1, \ldots, x_{a_f})$ such that $\phi_i(f(x_1, \ldots, x_{a_f})) = 1$ is one such, and since the intersection of two such sets is also one. While $F_i(f)$ in the worst case may be of size $O(|L^{a_f}|)$, it is often smaller. If f is distributive, $|F_i(f)| \leq 1$.

For each i between 1 and n and for each $(x_1, \ldots, x_{a_f}) \in F_i(f)$, we generate from the constraint $f(A_1, \ldots, A_{a_f}) \leq A_0$ a new constraint:

$$\phi_{j_1}(A_1) \sqcap \ldots \sqcap \phi_{j_{a_f}}(A_{a_f}) \leq \phi_i(A_0)$$

where j_k is the index of x_k in L.

The translation of $f(A_1, \ldots, A_{a_f}) \leq A_0$ is the set of all these new constraints.

Theorem 9. *The constraint $f(A_1, \ldots, A_{a_f}) \leq A_0$ is satisfiable iff all the constraints in the translation and all the ordering constraints between the components of the variables are. Moreover, any solution to the translation is in the image of ϕ and maps by ψ to a solution of $f(A_1, \ldots, A_{a_f}) \leq A_0$. Since ϕ and ψ are order-preserving, least solutions map to least solutions.*

Complexity. We have translated a set C of constraints over an $n+1$-point lattice L into a set C' of constraints over the boolean 2-point lattice. The size of the translated constraint set can be calculated as follows:

For any variable v_i in C, we introduce n variables $v_{i1}, \ldots, v_{in}$ in C'. For each v_i, C' contains at most n^2 constraints to ensure that any solution to C' will map $v_{i1}, \ldots, v_{in}$ to an image of an element in L.

For any constraint $f(A_1, \ldots, A_{a_f}) \leq A_0$, we introduce a number of constraints, each of size $a_f + 1$. The number of constraints is $\sum i = 1n|F_i(f)| \leq n \times n^{a_f}$. Since the size of the constraint $f(A_1, \ldots, A_{a_f}) \leq A_0$ is $(a_f + 1)$, the size of the translation is less than n^{1+a_f} times the size of the original constraint.

Bringing these together, we get that $|C'| \leq n^{1+a_{max}} \times |C| + n^2 \times |V|$, where a_{max} is the maximal arity of a function symbol in C and $|V|$ is the number of variables in C. For a fixed lattice and set of function symbols, this is a linear expansion. For the uniform problem, the input is given as operation matrices for the function symbols plus the constraints. The size of an operation matrix for a function with arity a is n^a, so the size of the input is greater than $n^{a_{max}} + |C|$. The size of the output is $n^{1+a_{max}} \times |C| + n^2 \times |V|$. The size of the input is hence the sum of two values and the size of the output is (approximately) the product of these. Hence, we get a quadratic worst-case expansion for the uniform problem.

The exponential dependence on the arity of the function symbols may seem bad, but it can be argued (see [19]) that any reasonable translation to boolean constraints will expand non-polynomially in the arity of the function symbols. Comparing with algorithm D (see Theorem 4) we see that D runs in time linearly dependent on a_{max} for the uniform case, hence boolean representation should, in general, only be used in case arities are known to be small.

Satisfiability of translation. Each constraint in the translation is of the form $a_1 \sqcap \ldots \sqcap a_m \leq a_0$, where the a_i are variables or constants ranging over the lattice $\{0, 1\}$. These constraints are isormorphic to Horn-clauses, and can hence be solved in time linear in the size of the constraint set using a HORNSAT-procedure [7].

5 Intractability of extensions

We have seen that algorithm D efficiently decides the *uniform* satisfiability problem (*i.e.*, uniform in both Γ and C), when instances Γ are restricted to be meet-closed. It is relevant to ask whether this can be extended to cover more relations than the meet-closed ones (perhaps by finding an algorithm entirely different from D.) The main purpose of this section is to demonstrate that, unless $\mathbf{P} = \mathbf{NP}$, *no* such extension is possible for any meet-semilattice L. This shows that algorithm D is complete for a maximal tractable class of problems, namely the meet-closed ones. If L is a meet-semilattice, we say that a problem $\Gamma = (L, \mathcal{S})$ is a *maximal* meet-closed problem if Γ is meet-closed and for any $R \subseteq L^k$ which is not meet-closed, the (particular, non-uniform) satisfiability problem $(L, \mathcal{S} \cup \{R\})$-SAT is **NP**-complete. We first show that *any distributive lattice has a maximal meet-closed problem* and deal with the general case afterwards. The proof is by composing Birkhoff's Representation Theorem for finite lattices [5] with Schaefer's Dichotomy Theorem [20] for the complexity of logical satisfiability problems.

For lattice L, let $\mathbf{Idl}(L)$ be the set of *order ideals* of L and $\mathbf{Irr}(L)$ the set of *join-irreducible* elements of L. If L is a finite, *distributive* lattice with $\mathbf{Irr}(L) = c_1, \ldots, c_n$ any fixed enumeration of $\mathbf{Irr}(L)$, then Birkhoff's Representation Theorem entails that, with $\eta : L \rightarrow \mathbf{Idl}(\mathbf{Irr}(L))$ defined by $\eta(x) = \{y \in \mathbf{Irr}(L) \mid y \leq x\}$, the map $\varphi : L \rightarrow \mathbf{2}^n$ becomes an *order-embedding* by setting $[\varphi(x)]_i = 1$, if $c_i \in \eta(x)$, and $[\varphi(x)]_i = 0$, if $c_i \notin \eta(x)$, for $i = 1, \ldots, n$. We refer to φ as the *canonical embedding* of L. For $R \subseteq L^k$ we let [6] $\varphi(R) = \{\langle \varphi(x_1), \ldots, \varphi(x_k) \rangle \mid (x_1, \ldots, x_k) \in R\}$, so $\mathbf{x} \in R \Leftrightarrow \varphi(\mathbf{x}) \in \varphi(R)$. With $\Gamma = (L, \mathcal{S})$, L distributive, define the problem $\varphi(\Gamma) = (\mathbf{2}, \varphi(\mathcal{S}))$ with $\varphi(\mathcal{S}) = \{\varphi(R) \mid R \in \mathcal{S}\}$. We use $\varphi(R)$ to denote the relation symbol corresponding to the relation $\varphi(R)$. If C is a constraint set over $\varphi(\Gamma)$, one has $\varphi(R) \subseteq \mathbf{2}^{kn}$ for all relations $\varphi(R) \in \varphi(\mathcal{S})$, where $n = |\mathbf{Irr}(L)|$ and k is the arity of R.

Now, we are interested in problems Γ such that $\varphi(\Gamma)$-SAT becomes polynomial time reducible to Γ-SAT. The problem here is that such reduction is not possible in general, because the constraint language of $\varphi(\Gamma)$ is more expressive than that of Γ. For instance, a unary relation symbol R may get translated into a symbol $\varphi(R)$ of arity $n = |\mathbf{Irr}(L)|$ with $n > 1$, so we can write (taking $n = 3$) constraints with patterns like $\{\varphi(R)(x, y, z), \varphi(R)(z, y, x)\}$ expressing that there exists $\mathbf{b} \in \mathbf{2}^3$ such that both that it and its reversal is in $\varphi(R)$; in general this cannot be expressed in the constraint language of Γ. However, if Γ has a certain kind of relations, then this becomes possible. Given distributive lattice L with canonical embedding φ and $|\mathbf{Irr}(L)| = n$ we let Π_L denote the set of "projection relations" $\pi_{ij} \subseteq L^2$, $i, j \in \{1, \ldots, n\}$, defined as follows

$$\pi_{ij} = \{(x, y) \in L^2 \mid [\varphi(x)]_i = [\varphi(x)]_j\}$$

[5] See [5]. We recall also that, for lattice L, an *order ideal* in L is a down-closed subset of L; $x \in L$ is *join-irreducible* if $x \neq \bot$ and $x = y \sqcup z$ implies $x = y$ or $x = z$ for all $y, z \in L$; L is *distributive* if $(x \sqcup y) \sqcap z = (x \sqcap z) \sqcup (y \sqcap z)$ for all $x, y, z \in L$; an *order-embedding* of lattices is an injective map preserving meet and join.

[6] We use the notation $\langle \mathbf{y}_1, \ldots, \mathbf{y}_k \rangle$, for $\mathbf{y}_i \in L^n$, to denote the "flattened" kn-vector $\mathbf{z}$ obtained by concatenating the tuples $\mathbf{y}_1, \ldots, \mathbf{y}_k$ into a single tuple, in that order; in detail, $\mathbf{z} = (z_1, \ldots, z_{kn})$ with $z_1 = [\mathbf{y}_1]_1, \ldots, z_n = [\mathbf{y}_1]_n, \ldots, z_{(k-1)n+1} = [\mathbf{y}_k]_1, \ldots, z_{kn} = [\mathbf{y}_k]_n$. For $\mathbf{x} = (x_1, \ldots, x_k) \in L^k$ we write $\varphi(\mathbf{x})$ for $\langle \varphi(x_1), \ldots, \varphi(x_k) \rangle$.

It is easy to check that π_{ij} are all meet-closed relations, since φ is an order-embedding. If L is non-trivial, there must be $a, b \in L$ with $a < b$, so that $\varphi(a) < \varphi(b)$, and hence $[\varphi(a)]_j = 0$ and $[\varphi(b)]_j = 1$ for some j; we can therefore express that $[\varphi(x)]_i = 0$ and $[\varphi(x)]_i = 1$ by $\pi_{ij}(x, a)$ and $\pi_{ij}(x, b)$, respectively. Constraints of the form $\pi_{ij}(x, y)$ will be written as $[\varphi(x)]_i = [\varphi(y)]_j$ or $[\varphi(x)]_i = b$ ($b \in \{0, 1\}$). One does not, of course, need explicit reference to φ in order to define the projection relations, so long as one can talk about the join-irreducible elements in an appropriate way:

Example 1. Let $\Phi = (L, F)$, $\mathbf{Irr}(L) = c_1, \dots, c_n$ and suppose we have a function $f_{c_i} \in F$ for each $c_i \in \mathbf{Irr}(L)$ where $f_{c_i}(x) = \top$ if $c_i \leq x$ and $f_{c_i}(x) = \bot$ otherwise. Then all f_{c_i} are distributive functions, and since the condition $[\varphi(x)]_i = [\varphi(y)]_j$ is equivalent to the condition $c_i \leq x \Leftrightarrow c_j \leq y$, the first mentioned condition can be expressed by distributive inequalities over Φ, because $c_i \leq x \Rightarrow c_j \leq y$ is equivalent to the distributive constraint $f_{c_i}(x) \leq f_{c_j}(y)$.

If $\Gamma = (L, \mathcal{S})$ and C is a constraint set over $\varphi(\Gamma)$, we describe a translation of C, called $\varphi^{-1}(C)$, to a constraint set over $\Gamma' = (L, \mathcal{S} \cup \Pi_L)$, as follows. Let $t_1, \dots, t_m$ be an enumeration of the constraint terms in C. Each t_s can be written as a term of the form $t_s = \varphi(R)(\langle \mathbf{A}_1, \dots, \mathbf{A}_k \rangle)$, where each $\mathbf{A}_i$ is a vector of n atomic terms, $n = |\mathbf{Irr}(L)|$, k the arity of R. We let $t_s \Downarrow (p, i) = [\mathbf{A}_p]_i$ ($1 \leq p \leq k, 1 \leq i \leq n$.) For each term t_s we let $\alpha_1^s, \dots, \alpha_k^s$ be k unique, fresh variables to be used in the translation of term t_s. The relational term $t_s = \varphi(R)(\langle \mathbf{A}_1, \dots, \mathbf{A}_k \rangle)$ then gets translated into the constraint set $C_s = C_{s,1} \cup C_{s,2} \cup C_{s,3}$, where

$$C_{s,1} = \{R(\alpha_1^s, \dots, \alpha_k^s)\}$$
$$C_{s,2} = \{[\varphi(\alpha_j^s)]_i = b \mid [\mathbf{A}_j]_i = b \text{ with } b \in \{0, 1\}\}$$
$$C_{s,3} = \left\{[\varphi(\alpha_p^s)]_i = [\varphi(\alpha_q^u)]_j \,\middle|\, t_s \Downarrow (p, i) \text{ is variable } x \wedge \exists t_u \in C.\, t_u \Downarrow (q, j) = x \right\}$$

For $C = t_1, \dots, t_m$ we define $\varphi^{-1}(C) = \bigcup_{s=1}^m C_s$. The constraints in $\varphi^{-1}(C)$ using π_{ij} can simulate all patterns in $\varphi(C)$, so we can show

Lemma 10. *Let $\Gamma = (L, \mathcal{S})$ with L non-trivial distributive lattice, and let C be a constraint set over $\varphi(\Gamma)$. Then C is satisfiable over $\varphi(\Gamma)$ if and only if $\varphi^{-1}(C)$ is satisfiable over $\Gamma' = (L, \mathcal{S} \cup \Pi_L)$.*

Since each set $C_{s,3}$ in $\varphi^{-1}(C)$ contributes at most $|C|^2$ new constraints, because each new constraint is determined by a distinct pair of occurrences of variables in C, it is straight-forward to show

Lemma 11. *For every constraint set C over $\varphi(\Gamma)$ one has $|\varphi^{-1}(C)| \leq |C|^2 \cdot (2|C| + 3)$.*

We now recall the contents of Schaefer's Dichotomy Theorem [20]. It yields the following very powerful classification (see also [14]):

Theorem 12 Schaefer [20]. *Let $\Gamma = (\mathbf{2}, \mathcal{S})$ be any boolean problem. Then the satisfiability problem Γ-SAT is polynomial time decidable, if one of the following four conditions are satisfied: either* (1) *every relation in $\mathcal{S}$ is closed under* *disjunction, or* (2) *every relation in $\mathcal{S}$ is closed under conjunction, or* (3) *every relation in $\mathcal{S}$ is* bijunctive, *i.e., it satisfies the closure condition $\forall x, y, z \in \mathbf{2}^k . x, y, z \in R \Rightarrow (x \wedge y) \vee (y \wedge z) \vee (z \wedge x) \in R$, or* (4) *every relation in $\mathcal{S}$ is* affine, *i.e., it satisfies the closure condition $\forall x, y, z \in \mathbf{2}^k . \, x, y, z \in R \Rightarrow x \oplus y \oplus z \in R$, where $\oplus$ is the exclusive disjunction.*

*Otherwise, if none of the above conditions are satisfied, the problem Γ-*SAT *is* **NP***-complete under log-space reductions.*

If L is a meet-semilattice which is not a lattice, we let $L^{\top}$ denote the extension of L to a lattice under addition of a top element, as earlier. If L is already a lattice, we let $L^{\top} = L$, by definition. We say that L is a ***distributive meet-semilattice*** if L is a meet-semilattice such that $L^{\top}$ is a distributive lattice. If L is a distributive meet-semilattice we know by previous remarks that $L^{\top}$ has a canonical embedding into $\mathbf{2}^n$, and this map φ will be referred to as the canonical embedding of L also. For any meet-semilattice L we let $M_L = \{(x, y, z) \in L^3 \mid z = x \sqcap y\}$, the meet-relation of L. The following results are proved in detail in [19].

Lemma 13. *Let L be a non-trivial distributive meet-semilattice, and let φ be its canonical embedding. Then the relation $\varphi(M_L)$ satisfies: $\varphi(M_L)$ is closed under conjunction, $\varphi(M_L)$ is not closed under disjunction, $\varphi(M_L)$ is not bijunctive, and $\varphi(M_L)$ is not affine.*

Let L be distributive and $\mathcal{S} = \Pi_L \cup \{M_L\}$; then the problem $\widetilde{\Gamma} = (L, \mathcal{S})$ is meet-closed. We can show that, for *any* relation R over L which is not meet-closed, the problem Γ^{+}-SAT with $\Gamma^{+} = (L, \mathcal{S} \cup \{R\})$ is **NP**-hard, by reduction from $\varphi(\Gamma)$-SAT, which, in turn, can be shown to be **NP**-hard by Lemma 10, Lemma 11, Lemma 13 together with Schaefer's Dichotomy Theorem. We therefore have

Theorem 14. (Intractability of extensions, distributive case) *For any non-trivial, distributive meet-semilattice L the problem $\widetilde{\Gamma} = (L, \Pi_L \cup \{M_L\})$ is maximal meet-closed.*

By Birkhoff's Theorem we know that L embeds into $\mathbf{2}^n$ if *and only* if L is distributive, hence the method used to prove Theorem 14 will not work for arbitrary finite lattices. However, if L is an arbitrary meet-semilattice we have the following weaker result by direct reduction from CNF-SAT:

Theorem 15. (Intractability of extensions, general case) *Let L be any non-trivial meet-semilattice and let $R \subseteq L^k$ be any relation over L which is not meet-closed. Then there exists a meet-closed problem $\Gamma = (L, \mathcal{S})$ such that the problem $\widetilde{\Gamma}$-*SAT *is* **NP***-complete with $\widetilde{\Gamma} = (L, \mathcal{S} \cup \{R\})$.*

Theorem 15 entails that the uniform satisfiability problem restricted to meet-closed relations becomes **NP**-hard no matter how it is extended, since the theorem says that there will always be a particular (non-uniform) problem over such an extension which is **NP**-hard; here the hard problem depends on the given extension. In contrast, Theorem 14 asserts the existence of a particular (non-uniform) problem which becomes hard no matter how it is extended. The results given here extend, in the case of finite domains, the results of [13], which considers totally ordered domains. See also [15].

6 Applications to program analysis

An application of the constraint solving technology shown in this paper is program analysis by annotated type systems. In these systems, programs are given types annotated with elements from a finite lattice. The type rules impose constraints between the annotations of the types of a term and its sub-terms. A typing of a program will generate a set of constraints, which must then be solved. Often, the constraints will have a form suitable for solution by the methods presented here.

One such example is the usage count analysis from [17] which has annotations in the lattice $\{0, 1, \infty\}$ with $0 \le 1 \le \infty$. The constraints use the following binary operators:

$k_1 + k_2$

$k_1 \backslash k_2$	0	1	∞
0	0	1	∞
1	1	∞	∞
∞	∞	∞	∞

$k_1 \cdot k_2$

$k_1 \backslash k_2$	0	1	∞
0	0	0	0
1	0	1	∞
∞	0	∞	∞

$k_1 \rhd k_2$

$k_1 \backslash k_2$	0	1	∞
0	0	0	0
1	0	1	∞
∞	0	1	∞

$+$ and $\cdot$ are addition and multiplication over counts. $k_1 \rhd k_2$ is 0 if $k_1 = 0$, otherwise $k_1 \rhd k_2 = k_2$.

The constraints are of one of the forms $k_1 = k_2$, $1 \le k$, $k_1 \le k_2$, $k_1 + k_2 \le k_3$, $\infty \cdot k_1 \le k_2$, $k_1 \cdot k_2 \le k_3$ or $k_1 \rhd k_2 \le k_3$. Hence (noting that $k_1 = k_2$ is equivalent to $k_1 \le k_2, k_2 \le k_1$), they are of the kind that can be solved by the methods shown, either by the direct method or by translation into boolean constraints. The translation uses the mapping shown below for lattice elements.
Hence, we replace each constraint variable k_i by two variables l_i and r_i over the binary domain, with the constraint $l_i \le r_i$.

The constraints are translated using the translation shown in section 4.2. The table below shows the result after reduction has been made for the constraints that involve constants.

count	translation
0	00
1	01
∞	11

constraint	translation
$k_1 = k_2$	$l_1 = l_2$, $r_1 = r_2$
$1 \le k$	$r = 1$
$k_1 \le k_2$	$l_1 \le l_2$, $r_1 \le r_2$
$k_1 + k_2 \le k_3$	$l_1 \le l_3$, $r_1 \le r_3$, $l_2 \le l_3$, $r_2 \le r_3$, $r_1 \wedge r_2 \le l_3$
$\infty \cdot k_1 \le k_2$	$r_1 \le l_2$
$k_1 \cdot k_2 \le k_3$	$r_1 \wedge r_2 \le r_3$, $l_1 \wedge r_2 \le l_3$, $l_2 \wedge r_1 \le l_3$
$k_1 \rhd k_2 \le k_3$	$r_1 \wedge l_2 \le l_3$, $r_1 \wedge r_2 \le r_3$

We end this section by giving some further examples of how our results can be used to reason about problems of interest in program analysis.

Example 2. The problem $\Phi = (2, \{\wedge\})$ is maximal meet-closed. To see this, represent Φ as relational problem $\Gamma = (2, \{M_2, \leq\})$. It then follows from Theorem 14 that Γ is maximal, because all relations in the set Π_2 can already be defined in terms of M_2, by Horn-definability (cf. comment after Theorem 6) together with the fact that all relations in Π_2 are meet-closed.

Example 3. Let **3** denote the three-point chain $0 < 1 < \infty$. The distributive lattice **3** occurs in many practical contexts, such as, *e.g.*, program analyses involving usage counting [17], [21],[1].

Let *pred* denote the predecessor function on **3**, with $pred(0) = 0, pred(1) = 0, pred(\infty) = 1$. Then *pred* is a distributive function. Consider the problem $\Phi = (\mathbf{3}, \{\sqcap, pred\})$, it is clearly meet-closed (viewed relationally), and, moreover, it is maximal. To see this, first note that the join-irreducible elements are $\{1, \infty\}$. Now define the function f_1 by setting $f_1(x) = 1 \sqcap x$ and define the function f_∞ by setting $f_\infty(x) = pred(x)$. Then $f_1(x) = 1$ if $1 \leq x$ and $f_1(x) = 0$ otherwise; moreover, $f_\infty(x) = 1$ if $\infty \leq x$ and $f_\infty(x) = 0$ otherwise. Recalling Example 1 we see that the functions f_1 and f_∞ are sufficient to represent the projection relations in Π_3, since we can express the condition $c_i \leq x \Rightarrow c_j \leq y$, for $c_i, c_j \in \mathbf{Irr}(\mathbf{3})$, by the distributive constraint $f_{c_i}(x) \leq f_{c_j}(y)$. It then follows from Theorem 14 that Φ is a maximal problem.

Example 4. Consider the *uniform* function problem restricted to *distributive* inequalities. By Theorem 6 this problem contains all meet-closed problems. Therefore, it follows from Theorem 15 that any uniform extension of the problem becomes **NP**-hard. In other words, if monotone function symbols can occur on the left side of inequalities and *any single* non-distributive function symbol can occur arbitrarily on the right side, then the uniform problem is **NP**-hard.

Example 5. In many applications in program analysis the semilattice L will be thought of as a domain of abstract program properties with lower elements representing more information than higher elements [7] If an analysis can be implemented as a constraint problem with minimal solutions it will have the desirable property that it is guaranteed to yield a uniquely determined piece of information which is optimal relative to the abstraction of the analysis. Theorem 6 says that *all and only* inequality constraint problems with this natural property can be represented as definite inequalities and hence can be solved in linear time using algorithm D.

7 Conclusion

We have studied efficient solution methods for the following classes of constraint problems over finite meet-semilattices:

[7] Alternatively, elements with more information may sit higher in the semilattice. Our results still apply, since the whole development in this paper can of course be dualized in the lattice-theoretic sense to encompass join-semilattices and join-closed problems over such.

- Definite inequalities involving monotone functions.
- Distributive inequalities.
- Boolean inequalities (Horn clauses).
- Meet-closed relational constraints.

We have shown that these classes are equivalent modulo linear time transformations for any fixed problem, and we have estimated the constant factors involved. For any fixed lattice and any fixed set of function/relation symbols the methods are linear time in the size of the constraint set, but, in the case of boolean representation, the time depends non-linearly on the maximal arity of relation symbols used. Furthermore, we have shown that these classes can be solved uniformly in polynomial time and that they are are maximal in the sense that any extension leads to **NP**-hard uniform problems.

Acknowledgements We wish to thank Fritz Henglein, Neil Jones, Helmuth Seidl, Mads Tofte and our referees for helpful discussions and comments.

A Algorithm D

Algorithm D is shown in Figure 1. We outline the data-structures and operations assumed for a linear time implementation (more details are in [19].) A list *Ilist* of records representing the inequalities in C. Each record also holds a boolean variable, called *inserted*. An array $Clist[\beta]$ indexed by variables in $Var(C)$. Each entry $Clist[\beta]$ holds an array of pointers to inequalities in *Ilist*, one entry for each inequality in C in which β occurs. Thus, to each item $Clist[\beta][k]$ there corresponds a unique occurrence of β in C. Hence, the number of distinct items $Clist[\beta][k]$ is bounded by $|C|$. The structure NS is a doubly linked list of of pointers to inequalities in *Ilist* which are also in C_{var}, and the item in *Ilist* has a back-pointer set to the representing item in NS. There is a pointer to the head element of NS. The idea is that NS holds holds pointers to those inequalities in C_{var} which are *not satisfied* under the current interpretation ρ (see below.) A finite map ρ mapping each distinct variable of C to an element in L. The map ρ holds the current "guess" at a satisfying valuation for C_{var}. The map $\perp_{Var(C)\to L}$ is the map which sends every variable of C to the bottom element of L. The map ρ is implemented as an array of lattice elements indexed by the variables. We write $\rho(\beta)$ for $\rho[\beta]$. Evaluating $\rho(\beta)$ is a constant time operation, and so is the operation of updating ρ at β.

Operation POP removes the head element of NS and returns it after setting the corresponding *inserted* field to false. INSERT(i) inserts an inequality pointer i at the front of NS, updating the *inserted* field. If the *inserted* field is true, then INSERT does nothing. DROP(i), removes i, a pointer to an element in NS, from NS, using the *inserted* filed in analogy with INSERT. All these operations can be implemented as constant time operations, using the data-structures above. Operation L-NORMALIZE transforms a definite set into an equivalent L-normal set and runs in time $O(|C|)$ (using Lemma 2 and Lemma 1.)

```
1. Input
   A finite set $C = \{\tau_i \leq A_i\}_{i \in I}$ of definite inequalities over $\Phi = (L, F)$, $F = \{f : L^{a_f} \to L\}$ with each $f$ monotone, $L$ finite lattice, $Var(C) = \mathcal{V}_m$ for some $m$.
2. Initially
    $C := \text{L-NORMALIZE}(C)$
    $\rho := \bot_{\mathcal{V}_m \to L}$
    $C_{var} := \{\tau \leq A \in C \mid A \text{ is a variable}\}$
    Initialize the lists $Clist[\alpha]$, for every distinct variable $\alpha$ in $C$.
    Initialize $Ilist$ to hold the inequalities in $C$.
    $NS := \{\tau \leq \beta \in C_{var} \mid L, \rho \not\models \tau \leq \beta\}$
3. Iteration
    while $NS \neq \emptyset$ do
                $\tau \leq \beta := \text{POP}(NS)$;

                $\rho(\beta) := [\![\tau]\!]\rho \sqcup \rho(\beta)$;

                for $\sigma \leq \gamma \in Clist[\beta]$ do
                        if $\rho, L \not\models \sigma \leq \gamma$
                        then INSERT($\sigma \leq \gamma$)
                        else DROP($\sigma \leq \gamma$)
                end; (* for *)

       end;(* while *)
4. Output
   If $L, \rho \models \tau \leq c$ for all $\tau \leq c \in C_{cnst}$ then output $\rho$ else FAIL.
```

Figure 1: Algorithm D for satisfiability of definite constraints over Φ

In the algorithm we have followed the convention of writing the pattern of an inequality pointed to by an inequality pointer instead of the pointer itself.

References

1. L. Birkedal, M. Tofte, and M. Vejlstrup. From region inference to von Neumann machines via region representation inference. In *Proc. 23rd Annual ACM Symposium on Principles of Programming Languages (POPL)*, pages 171–183. ACM Press, January 1996.
2. L. Birkedal and M. Welinder. Binding-time analysis for Standard ML. *Lisp and Symbolic Computation*, 8(3):191–208, September 1995.
3. A. Bondorf and J. Jørgensen. Efficient analyses for realistic off-line partial evaluation. *Journal of Functional Programming*, 3(3):315–346, July 1993.
4. C.C. Chang and H.J. Keisler. *Model Theory*. Studies in Logic and the Foundation of Mathematics, Vol. 73, 3rd ed., North Holland, 1990.
5. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge Mathematical Textbooks, Cambridge University Press, 1990.
6. R. Dechter and J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 58:237–270, 1992.

7. William F. Dowling and Jean H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 3:267–284, 1984.
8. Dirk Dussart, Fritz Henglein, and Christian Mossin. Polymorphic recursion and subtype qualifications: Polymorphic binding-time analysis in polynomial time. In *Proc. 2nd Int'l Static Analysis Symposium (SAS), Glasgow, Scotland*, Lecture Notes in Computer Science. Springer-Verlag, September 1995.
9. M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman, 1979.
10. R. Greenlaw, H.J. Hoover, and W. Ruzzo. *Limits to Parallel Computation. P-Completeness Theory*. Oxford University Press, 1995.
11. F. Henglein. Efficient type inference for higher-order binding-time analysis. In J. Hughes, editor, *Functional Programming Languages and Computer Architecture, Cambridge, Massachusetts, August 1991 (Lecture Notes in Computer Science, vol. 523)*, pages 448–472. ACM, Berlin: Springer-Verlag, 1991.
12. W. Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications, Vol. 42, Cambridge University Press, 1993.
13. P. Jeavons and M. Cooper. Tractable constraints on ordered domains. *Artificial Intelligence*, 79:327–339, 1995.
14. Peter Jeavons and David Cohen. An algebraic characterization of tractable constraints. In *First Annual Conference on Computing and Combinatorics (COCOON)*, pages 633–642. Springer Verlag, LNCS 959, 1995.
15. Peter Jeavons, David Cohen, and Marc Gyssens. A unifying framework for tractable constraints. In *First International Conference on Principles and Practice of Constraint Programming*, pages 276–291. Springer Verlag, LNCS 976, 1995.
16. G. Kildall. A unified approach to global program optimization. *Proc. ACM Symp. on Principles of Programming Languages (POPL)*, 1973.
17. T. Æ. Mogensen. Types for 0, 1 or many uses. Technical report, In preparation, 1996.
18. Vaughan Pratt and Jerzy Tiuryn. Satisfiability of inequalities in a poset. *Studia Logica, Helene Rasiowa memorial issue (to appear)*, 1996.
19. Jakob Rehof and Torben Mogensen. Report on tractable constraints in finite semilattices. Technical report, DIKU, Dept. of Computer Science, University of Copenhagen, Denmark. Available at http://www.diku.dk/research-groups/topps/personal/ rehof/publications.html, 1996.
20. Thomas J. Schaefer. The complexity of satisfiability problems. In *Tenth Annual Symposium on the Theory of Computing (STOC)*, pages 216–226. ACM, 1978.
21. Mads Tofte and Jean-Pierre Talpin. Implementation of the typed call-by-value λ-calculus using a stack of regions. In *Proc. 21st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), Portland, Oregon*. ACM, ACM Press, January 1994.