Nordic Journal of Computing 3(1996), 40–61.

LEAST AND GREATEST SOLUTIONS OF EQUATIONS OVER $\mathcal{N}$

HELMUT SEIDL
FB IV – Informatik
Universität Trier
D–54286 Trier
Germany
seidl@ti.uni-trier.de

Abstract. We consider the problem of computing least and greatest solutions of a system of equations $x_i = f_i$, $i = 1, \ldots, n$, over $\mathcal{N}$, i.e., the naturals (extended by ∞), where the right hand sides f_i are expressions built up from constants and variables by various sets of operations.

We present efficient algorithms in case where the following operations occur:

(1) minimum and maximum;

(2) maximum, addition and multiplication;

(3) minimum, addition and multiplication;

(4) minimum, maximum, addition and multiplication.

We extend the methods to the cases where (one–sided) conditionals are allowed as well.

CR Classification: D.3.4, F.4.1, G.2.1

Key words: equations over integers, least or greatest solution, program analysis

1. Introduction

Assume $\mathcal{D}$ is a *complete partial order* (cpo) with least element $\bot$. Assume $X_n = \{x_1, \ldots, x_n\}$ is a set of variables, and let Ω denote a set of continuous binary operations on $\mathcal{D}$. The set of *polynomials* $\mathcal{D}_\Omega[X_n]$ with operations from Ω consists of all expressions built up from constants $d \in \mathcal{D}$, variables $x \in X_n$ by application of operations from Ω. If Ω is understood, we will also omit it in the index. Every polynomial $f \in \mathcal{D}_\Omega[X_n]$ denotes a continuous function $[f] : \mathcal{D}^n \to \mathcal{D}$. If the meaning is clear from the context, we do not distinguish (notationally) between f and $[f]$. However, we write $f \equiv g$ if we mean syntactic equality and $f = g$ if they denote the same functions, i.e., $[f] = [g]$. Let S be a system of equations $x_i = f_i, i = 1, \ldots, n$, with $f_i \in \mathcal{D}_\Omega[X_n]$. The following fact is well–known from the theory of cpo's and continuous functions.

Received December 13, 1994; revised October 25, 1995. Communicated by Neil Jones.

FACT 1. S has a unique least solution $X_1, \ldots, X_n$. X_i is given by

$$X_i = \bigsqcup_{j \geq 0} X_i^{(j)}$$

where for every i, $X_i^{(0)} = \bot$ and $X_i^{(j)} = f_i[X_1^{(j-1)}, \ldots, X_n^{(j-1)}]$ for $j > 0$. □

Many compile time analyses of programs rely on computations of least solutions of such systems of equations, cf. e.g., [2, 8, 9, 3]. The least solution of S can be computed effectively whenever $\mathcal{D}$ satisfies the *ascending chain condition* (*acc*), meaning that every ascending chain

$$d_0 \leq d_1 \leq \ldots \leq d_j \leq \ldots$$

of elements $d_j \in \mathcal{D}$, is ultimately constant. Corresponding techniques are considered in [13, 7, 11, 12]. However, there are instances of compile time program analysis which make use also of cpo's which do not satisfy the *acc*. One technique applicable in these unrestricted cases is the *widening/narrowing* approach of [2, 4]. While this approach is very general in its applicability, it may not compute the least solution itself, but only an upper bound to it. Therefore, we propose alternative methods at least for the cpo of *natural numbers* (extended by ∞)

$$\mathcal{N} = \{0 < 1 < \ldots < n < \ldots < \infty\}$$

Clearly, $\mathcal{N}$ does not satisfy the *acc*. However, meaningful analyses make use of systems of equations over $\mathcal{N}$.

In [14], an algorithm is proposed which detects whether instances of variables that exist in parallel in the `PRAM` language `FORK` are equal. It employs systems of equations over $\mathcal{N}$ with operations "$\sqcap$" (minimum) and "$\sqcup$" (maximum). In the context of register allocation, an approximative complexity analysis determines bounds to the *number of uses* of some variable.

EXAMPLE 1. Consider, e.g., the following definition of a function possibly occurring in a program (written in some strict functional language):

```
h y = letrec
        g x = if x > 0 then (d y)
              else 7 * (g (x - 1));
        d z = z * y - 9
      in
        if y < 0 then y + (g (- y))
        else d(g (y + 1))
```

The goal is to assign registers to those variables which are most often used. Calculating (some bounds to) the number of accesses to, say variable y, results in the following equations:

$$\begin{aligned} x_h &= 1 + ((2 + x_g) \sqcup (1 + x_g + x_d)) \\ x_g &= (1 + x_d) \sqcup x_g \\ x_d &= 1 \end{aligned}$$

where x_f counts the number of accesses to y during the execution of one call to function f and the conditional is abstracted with "$\sqcup$". The least solution of the system provides an upper bound. To obtain a lower bound (at least w.r.t. error–free program executions), one may compute the least solution of the system of equations obtained by replacing "$\sqcup$" with "$\sqcap$". □

Thus, we need to compute least solutions of systems of equations over $\mathcal{N}$ with sets of operations $\{\sqcap, +\}$ or $\{\sqcup, +\}$. Multiplications occur if programs may contain **for**–loops (or tail recursion where the recursion depth is determined by some variable's value). For interval analysis, one may consider an abstract domain $\bar{\mathcal{N}}$ consisting of all intervals $[n, +\infty]$, $n \in \mathcal{N}$, ordered by inclusion (cf. [4]). This domain is isomorphic to $\mathcal{N}$ where the order is *reversed*. Observe that $\bar{\mathcal{N}}$ indeed satisfies the ascending chain condition (since $\mathcal{N}$ satisfies the descending chain condition). Therefore, all monotonic functions are also continuous. Thus, least solutions over $\bar{\mathcal{N}}$ correspond to greatest solutions over $\mathcal{N}$.

Therefore, in this paper we investigate the computation of least and greatest solutions for systems of equations over $\mathcal{N}$ using various sets of continuous operations and give efficient algorithms for all of these. The operations we are interested in are "$\sqcup$" (maximum), "$\sqcap$" (minimum), "$+$" (addition, extended by $\infty + x = x + \infty = \infty$) and "$\cdot$" (multiplication, extended by $0 \cdot \infty = \infty \cdot 0 = 0$ and $\infty \cdot x = x \cdot \infty = \infty$ whenever $x \neq 0$). Note that all these operations are commutative and associative. 0 is the neutral element for "$\sqcup$" and "$+$", 1 is the neutral element for "$\cdot$" whereas ∞ is the neutral element for "$\sqcap$". Additionally to these we also consider (one–sided) conditionals "$>c;$" ($c \in \mathcal{N}$) defined by

$$x >c; y \equiv \begin{cases} y & \text{if } x > c \\ 0 & \text{otherwise} \end{cases}$$

The (theoretical) target architecture we have in mind to implement our algorithms on is a random access machine (RAM). For simplicity, we use the unit cost model to measure runtime complexities. Especially, we assume that every element of $\mathcal{N}$ can be stored in one cell of our memory, that tests $d \leq d'$ and operations from Ω always can be executed in unit time. Therefore, it makes sense to define the size $|S|$ of system S of equations $x_i = f_i, i = 1, \ldots, n$, by $|S| = \sum_{i=1}^{n}(1 + |f_i|)$ where the size $|f|$ of polynomial f is given by $|f| = 1$ if p is a constant or a variable, and $|f| = 1 + |f_1| + |f_2|$ if $f = (f_1 \Box f_2)$ for some $\Box \in \Omega$.

The unit cost assumption for every basic operation is realistic as long as the numbers involved are not too large. Consider, e.g., the system of equations

$$\begin{aligned} x_1 &= 2 \\ x_{j+1} &= x_j \cdot x_j \qquad \text{for } j = 1, \ldots, n-1 \end{aligned}$$

The least solution $X_i, i = 1, \ldots, n$, can be trivially obtained by $n - 1$ multiplications which however involve possibly exponentially large numbers. In

order to produce least solutions by a polynomial number of operations, we therefore cannot avoid to use multiplications in this case. However, we will be as restrictive with the use of multiplications as possible. So, whenever the considered systems of equations do not contain occurrences of "$\cdot$", our algorithms will not use multiplications either. Also we show that, provided the finite X_i are smaller than some h, operations on numbers of length $O(\log(h))$ suffice. Finallly, we derive efficient algorithms to determine the set of all i where $X_i < \infty$ without using multiplications.

The rest of the paper is organized as follows. The next section provides basic algorithmic facts, especially on the computation of least and greatest solutions of equations over certain finite cpo's. Sections 3, 4 and 6 present special algorithms to construct least solutions for various subsets of operations, whereas Section 7 deals with operations minimum, maximum, addition and multiplication alltogether. Section 5 proves a lower bound theorem which turns out to be crucial for the construction of least solutions of systems of equations which contain "$\sqcap$". Section 8 shows how our basic algorithms can be applied to speed up the computation of least solutions over restricted ranges of integers, and derives polynomial finiteness tests *without* multiplications even if the system contains "$\cdot$". Section 9 extends the results by additionally allowing occurrences of conditionals "$>c;$". The results of the previous sections can be applied to decide in polynomial time whether or not the costs of tree automata with cost functions of a very general form are bounded – a problem which has been left open in [16]. Finally, Section 10 considers the same questions for greatest solutions.

An extended abstract of the present paper appeared in [15]. The sections on one–sided conditionals and greatest fixpoints are new.

2. Basic Facts

Let S denote a system of equations $x_i = f_i$ over $\mathcal{N}$ with operations from Ω. It turns out that it is convenient to assume that the right hand sides f_i of S are of one of the forms ρ or $\rho_1 \Box \rho_2$ where ρ, ρ_1, ρ_2 are variables or elements of $\mathcal{N}$ and $\Box \in \Omega$. Furthermore,

- Whenever $f_i \in \mathcal{N}$ then x_i does not occur in any right hand side f_j;
- Whenever $f_i \equiv c_1 \Box c_2$ with $c_1, c_2 \in \mathcal{N}$, then $\Box \in \{\cdot, +\}$;
- Whenever $f_i \equiv \rho_1 > c; \rho_2$ then $\rho_1 \notin \mathcal{N}$;
- 1 does not occur as an operand of "$\cdot$";
- 0 does not occur as an operand at all, and ∞ may only occur as an operand of "$\cdot$" or as the second operand of some conditional.

Systems with these properties are called *normalized*. A normalized system is called *reduced* iff ∞ does not occur as an operand, and for no j and no operation $\Box$, $f_j \equiv c_1 \Box c_2$ with both $c_1 \in \mathcal{N}$ and $c_2 \in \mathcal{N}$.

EXAMPLE 2. In order to normalize the system of equations from Example 1, we break large expressions into smaller ones by the introduction of auxiliary variables y_k:

$$\begin{array}{lcllcl} x_h & = & 1 + y_1 & y_4 & = & x_g + 1 \\ y_1 & = & y_2 \sqcup y_3 & x_g & = & y_5 \sqcup x_g \\ y_2 & = & 2 + x_g & y_5 & = & 1 + 1 \\ y_3 & = & 1 + y_4 & x_d & = & 1 \end{array}$$

To obtain a reduced system, we additionally evaluate the right hand side for y_5 and replace all occurrences of y_5 in right hand sides with 2. □

Using this idea we find:

FACT 2. For every system S of equations with variables $x_1, \ldots, x_n$ a normalized system S' with variables $x_1, \ldots, x_m$ for some $m \geq n$ can be constructed (without multiplications) in time $O(|S|)$ with least solution (resp. greatest solution) $X_1, \ldots, X_m$ such that $X_1, \ldots, X_n$ is also the least solution (resp. greatest solution) of S. □

For $k \geq 2$, let $\mathbf{k}$ denote the total order $\{0 < 1 < \ldots < (k-1)\}$, and consider the set $\Omega = \{\sqcap, \sqcup, \oplus, \odot\}$ of operations on $\mathbf{k}$ where "$\sqcup$" and "$\sqcap$" are maximum and minimum as usual, and "$\oplus$" and "$\odot$" are truncated addition and multiplication, i.e., $x \oplus y = (x + y) \sqcap (k-1)$ and $x \odot y = (x \cdot y) \sqcap (k-1)$. For simplicity, we will denote "$\oplus$" and "$\odot$" by "$+$" and "$\cdot$" as well. Domain $\mathbf{k}$ together with mapping $H : \mathcal{N} \to \mathbf{k}$ defined by

$$H(y) = y \sqcap (k-1)$$

can be used to determine the behavior of system S "up to $k-2$". The reasons are the following:

- H is *faithful* on $\{0, \ldots, k-2\}$, i.e., $\{y\} = H^{-1}(H(y))$ for all $y \in \{0, \ldots, k-2\}$;
- H maps least element to least element and greatest element to greatest element;
- H commutes with operations "$\sqcap$", "$\sqcup$", "$+$", "$\cdot$" as well as with conditionals "$> c;$" whenever $0 \leq c < k-1$; moreover,
- H is continuous.

These properties allow to prove Fact 3. Let $\Omega = \{\sqcap, \sqcup, +, \cdot\}$ and Γ denote the set $\{> c; \mid 0 \leq c < k-1\}$. Consider a system S of equations $x_i = f_i, i = 1, \ldots, n$, over $\mathcal{N}$ with operations from $\Omega \cup \Gamma$. For $f \in \mathcal{N}_{\Omega\cup\Gamma}[X]$ let $(Hf) \in \mathbf{k}_{\Omega\cup\Gamma}[X]$ denote the polynomial obtained from f by replacing every coefficient d with $H(d)$ and the operations on $\mathcal{N}$ by the corresponding operations on $\mathbf{k}$. Let S_H denote the system

$$x_i = (Hf_i), \qquad\qquad i = 1, \ldots, n$$

over $\mathbf{k}$ with operations from $\Omega \cup \Gamma$. Then we have:

FACT 3. Let $X_i, i = 1, \ldots, n$, and $\bar{X}_i, i = 1, \ldots, n$, be the least and greatest solutions of S, respectively. Accordingly, let $X_{H,i}, i = 1, \ldots, n$, and $\bar{X}_{H,i}, i = 1, \ldots, n$, be the least and greatest solutions of S_H, respectively. Then

1. $X_{H,i} = H(X_i)$ for every i;
2. $\bar{X}_{H,i} = H(\bar{X}_i)$ for every i. □

Since the finite domains $\mathbf{k}$ satisfy the *acc* least solutions of systems over $\mathbf{k}$ as well as greatest solutions can be computed effectively. Using the worklist algorithm known from dataflow analysis instead of naive fixpoint iteration, we find that these solutions can even be computed efficiently.

FACT 4. Assume $k \geq 2$. Then the least (resp. greatest) solution of a normalized system S of equations over $\mathbf{k}$ with operations from $\Omega \cup \Gamma$ can be computed in time $O(k \cdot |S|)$. Provided S contains multiplications, the algorithm may use multiplications as well but only of integers of length $O(\log(k))$. □

COROLLARY 1. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\Omega \cup \Gamma$ and least solution $X_i, i = 1, \ldots, n$, (resp. greatest solution $\bar{X}_i, i = 1, \ldots, n$). For every $k \geq 0$, we can compute the sets $\{i \mid X_i = y\}$, $y = 0, \ldots, k$, (resp. $\{i \mid \bar{X}_i = y\}$, $y = 0, \ldots, k$) in time $O((k+1) \cdot |S|)$. Provided S contains multiplications, the algorithm may use multiplications as well but only of integers of length $O(\log(k+1))$.* □

By Cor. 1, we can determine in linear time the set of all i such that value X_i of the least solution for x_i equals 0 (resp. the set of all i such that value $\bar{X}_i$ of the greatest solution for x_i equals 0). Using this information, we can remove both all operands 0 and all operands ∞ (of "$\cdot$"). We conclude:

COROLLARY 2. *For every normalized system S of equations reduced systems S_1 and S_2 of equations can be constructed in time $O(|S|)$ where S_1 has the same least and S_2 has the same greatest solution as S. Multiplications are only needed if S itself contains occurrences of "$\cdot$".* □

In the following, we first consider the fast computation of least solutions for various subsets of operations. Finally in Section 10, we deal with the computation of greatest solutions as well.

3. Minimum and Maximum

THEOREM 1. *If S is a system of equations over $\mathcal{N}$ with operations from $\{\sqcap, \sqcup\}$ then the least solution of S can be computed (without multiplications) in time $O(|S| \cdot \log(|S|))$.*

Let S be the system $x_i = f_i, i = 1, \ldots, n$, with least solution $X_i, i = 1, \ldots, n$. Observe that $X_i \leq h$ for all i where $h \geq 0$ is the least upper bound

on the constants from $\mathcal{N}$ occurring in S. Therefore, Cor. 1 is applicable. However, this would result in an $O((h+1)\cdot|S|)$–algorithm.

PROOF. Because of Fact 2 we w.l.o.g. can assume that S is reduced. Let J denote the set of all i where f_i does *not* equal some constant. Furthermore, let $\mathcal{C}$ denote the set of constant operands occurring in S. We observe:

CLAIM. If $\mathcal{C} = \emptyset$ then $X_i = 0$ for all $i \in J$. □

Therefore, assume $\mathcal{C} \neq \emptyset$, and $c = \bigsqcup \mathcal{C}$. Clearly, $X_i \leq c$ for all $i \in J$. We will now show that all occurrences of c in S as an operand can be safely removed. Let S' be the system of equations $x_i = f_i', i = 1, \ldots, n$, where the f_i' are obtained as follows.

- If $f_i \equiv x_j \sqcap c$ or $f_i \equiv c \sqcap x_j$ then $f_i' \equiv x_j$.
- If $f_i \equiv x_j \sqcup c$ or $f_i \equiv c \sqcup x_j$ then $f_i' \equiv c$.
- Otherwise, $f_i' \equiv f_i$.

S' has the same least solution as S. Thus, our algorithm constructing the least solution of S works as follows.

(1) Construct the equivalent reduced system.

(2) Compute sets $\mathcal{C}$ and J.

(3) While $\mathcal{C} \neq \emptyset$ execute Steps (3.1) through (3.4):

  (3.1) Set $c := \bigsqcup \mathcal{C}$.
  (3.2) Remove all occurrences of c in S as an operand.
  (3.3) Construct again the equivalent reduced system.
  (3.4) Remove c from $\mathcal{C}$, and recompute J.

(4) Finally, set $f_i = 0$ for all remaining $i \in J$.

By a suitable data structure for S, all the removals in Step (3.2) together with the reduction in Step (3.3) can be executed in time $O(|S|)$. In order to efficiently implement the selection in Step (3.1) we can, e.g., keep $\mathcal{C}$ in a sorted list. Then, all maximum computations of Step (3.1) together consume time $O(|S|)$. It follows that, once this representation for $\mathcal{C}$ has been computed, the remaining steps of the algorithm take time $O(|S|)$. □

Note that if constants $d \in \mathcal{N}$ occurring in right hand sides are from a small range we may use bucket sort instead of some general sorting algorithm which brings complexity down to $O(|S|)$ for this case.

4. Maximum, Addition and Multiplication

THEOREM 2. *If S is a system of equations over $\mathcal{N}$ with operations from $\{\sqcup, +, \cdot\}$, then the least solution of S can be computed in time $O(|S|)$. Multiplications are only needed provided S itself contains multiplications.*

PROOF. Let S be the system $x_i = f_i, i = 1, \ldots, n$, with least solution $X_i, i = 1, \ldots, n$. To compute this least solution of S, we proceed in three steps:

(1) We determine values X_i for all i where $X_i \in \{0, 1\}$;

(2) We determine the set of all i where $X_i = \infty$;

(3) We determine the remaining values X_i.

By Cor. 1, Step (1) can be executed in linear time. Therefore w.l.o.g. assume S is reduced where $X_i > 1$ for every i. For S define graph $G_S = (V_S, E_S)$ by $V_S = \{1, \ldots, n\}$ where $(i, j) \in E_S$ iff x_i occurs in f_j. G_S is also called *dependence graph* of S.

A fast implementation of Step (2) is based on the following two claims.

CLAIM 1. If $X_i = \infty$ for some i then $X_j = \infty$ for every j reachable from i.

Claim 1 states that ∞ wherever it turns up is propagated along every edge. The reason for this is that by our assumptions, the result of an operator application cannot be less than any of its operands. □

CLAIM 2. Let Q be a strong component of G_S. If Q contains an edge (i, j) where f_j contains an occurrence of "$+$" or "$\cdot$", then $X_j = \infty$.

Claim 2 locates a source for infinity: namely, a strong component containing an edge $e = (i, j)$ corresponding to an occurrence of "$+$" or "$\cdot$" in the right hand side f_j for x_j. Thus, a cyclic path π exits which contains edge e. π represents a cyclic dependency between variables x_j and x_i. Since "$+$" or "$\cdot$" applied to finite values d_1, d_2 returns a value which is *strictly greater* than both d_1 and d_2, fixpoint iteration results in an unbounded sequence of approximations $X_j^{(t)}$ for X_j. □

Assuming that Claims 1 and 2 together characterize all i with $X_i = \infty$, we can construct a linear algorithm that implements Step (2) of our algorithm. Sufficiency of the claims however follows from Claim 3 which provides a method to compute the values X_i for the remaining i.

Assume that we determined all i where according to Claims 1 and 2, $X_i = \infty$, replaced the corresponding right hand sides f_i with ∞, and reduced the resulting system.

Therefore, now assume S is a reduced system where f_j does not contain "$+$" or "$\cdot$" for any edge (i, j) in a strong component of G_S.

Let Q be a strong component of G_S containing at least one edge such that each i in Q is reachable only from nodes in Q itself. Let $\mathcal{C}_Q$ denote the set of all constant operands of "$\sqcup$" occurring in $f_i, i \in Q$. We observe:

CLAIM 3. $X_j = \bigsqcup \mathcal{C}_Q$ for every j in Q. □

EXAMPLE 3. Considering the reduced system of equations from Example 2, we find that $\{x_g\}$ is a strong component as demanded for Claim 3. Since the right hand side for x_g equals $2 \sqcup x_g$, Claim 3 implies that $X_g = 2$. □

Based on Claim 3, Step (3) can be implemented by a suitable depth–first scan over G_S in linear time. This completes the proof of the theorem. □

5. Lower Bounds

In Sections 6 and 7 we will, in addition to arithmetical operations "$+$" and "$\cdot$" also consider the minimum operation. Moreover in Section 7, we will

exploit that, starting fixpoint iteration at some $h > 0$, may allow to simplify system S without changing the solution. Therefore, Theorem 3 does not deal with least solutions but with *h–least* solutions for $h \geq 0$. The h–least solution of S is defined as the least solution $X_i, i = 1, \ldots, n$, of S with $X_i \geq h$ for all i.

The starting point for our derivation is Theorem 2 whose generalization to h–least solutions proves the lower bound at least for systems S without occurrences of "$\sqcap$".

PROPOSITION 1. *Let $h \geq 0$, and S denote a reduced system of equations $x_i = f_i, i = 1, \ldots, n$, over $\mathcal{N}$ with operations from $\{\sqcup, +, \cdot\}$ where $f_i \notin \mathcal{N}$, and let $\mathcal{C}$ denote the set of constant operands of "$\sqcup$" occurring in of S. Assume $X_i, i = 1, \ldots, n$ is the h–least solution of S where $X_i > h$ for all i. Then $\sqcap \mathcal{C} \leq X_i$ for all i. Especially, $\mathcal{C} = \emptyset$ implies that $X_i = \infty$ for all i.*

PROOF. In case $h \geq 1$, the given lower bound on the X_i easily follows from a generalization of the characterization in Claims 1, 2 and 3 within the proof of Theorem 2. to h–least solutions. In case, $h = 0$ and $X_j = 1$ for some j, it turns out that in fact, 1 must occur as an operand of "$\sqcup$" in S. □

Now consider a system S of equations $x_i = f_i, i = 1, \ldots, n$, over $\mathcal{N}$ with operations from $\Omega = \{\sqcap, \sqcup, +, \cdot\}$ and h–least solution $X_i, i = 1, \ldots, n$. Let J denote the set of all i such that f_i contains "$\sqcap$". The J–tuple μ is called *legal choice* iff $\mu = \langle y_j \rangle_{j \in J}$ where for every $j \in J$ with $f_j \equiv \rho_{j_1} \sqcap \rho_{j_2}$, $y_j \in \{\rho_{j_1}, \rho_{j_2}\}$. Let M denote the set of all legal choices.

For legal choice $\mu = \langle y_j \rangle_{j \in J}$, let S_μ denote the system of equations $x_i = g_i, i = 1, \ldots, n$, where $g_i \equiv y_i$ whenever $i \in J$ and $g_i \equiv f_i$ otherwise.

PROPOSITION 2. *Let $X_{\mu,i}, i = 1, \ldots, n$, denote the h–least solution of S_μ. Then*

1. $\forall \mu \in M : \forall i \in [1, n] : X_i \leq X_{\mu,i}$;
2. $\exists \mu_0 \in M : \forall i \in [1, n] : X_i = X_{\mu_0,i}$.

PROOF. Assertion 1 follows by usual fixed point induction. Therefore, consider Assertion 2. For every $f_i \equiv \rho_1 \sqcap \rho_2$, we have $X_i = \rho_1[X_1, \ldots, X_n] \sqcap \rho_2[X_1, \ldots, X_n]$. Consequently, $X_i = \rho_{\nu_i}[X_1, \ldots, X_n]$ for some $\nu_i \in \{1, 2\}$. Hence, we define $\mu_0 = \langle \rho_{\nu_j} \rangle_{j \in J}$. By construction, $X_i, i = 1, \cdots, n$, is a solution of S_{μ_0} with $X_i \geq h$ for all i. Therefore, $X_{\mu_0,i} \leq X_i$ for all i. Since by Assertion 1, also $X_i \leq X_{\mu_0,i}$ for all i, Assertion 2 follows. □

Prop. 2 can be used to derive the desired generalization of Prop. 1.

THEOREM 3. *Let $h \geq 0$, and consider a reduced system S of equations over $\mathcal{N}$ with operations from Ω where $f_i \notin \mathcal{N}$ for all i, and let $\mathcal{C}$ denote the set of constant operands of "$\sqcap$" or "$\sqcup$" occurring in S. Assume $X_i, i = 1, \ldots, n$ is the h–least solution of S. where $X_i > h$ for all i. Then $\sqcap \mathcal{C} \leq X_i$ for all i. Especially, $\mathcal{C} = \emptyset$ implies that $X_i = \infty$ for all i.*

PROOF. Consider the system of equations S_{μ_0} as constructed in the proof of Prop. 2. S_{μ_0} has the same h–least solution as S but does not contain occurrences of "$\sqcap$". Let S' denote the reduced system $x_i = f'_i, i = 1, \ldots, n$, corresponding to S_{μ_0}. Let $\mathcal{C}'$ denote the set of constant operands of "$\sqcap$" or "$\sqcup$" occurring in S'.

Moreover, let $\mathcal{R}$ denote the set of i where $f'_i \in \mathcal{N}$. We observe:

1. $\sqcap \mathcal{C} \leq X_i$ for all $i \in \mathcal{R}$;
2. $\sqcap \mathcal{C} \leq \sqcap \mathcal{C}'$.

Since by Prop. 1, $\sqcap \mathcal{C}' \leq X_i$ for all $i \notin \mathcal{R}$, the assertion follows. □

6. Minimum, Addition and Multiplication

THEOREM 4. *If S is a system of equations over $\mathcal{N}$ with operations from $\{\sqcap, +, \cdot\}$, then the least solution of S can be computed in time $O(|S| \cdot \log(|S|))$. Multiplications are only needed provided S itself contains multiplications.*

PROOF. Let S denote the system $x_i = f_i, i = 1, \ldots, n$, with least solution $X_i, i = 1, \ldots, n$. We start by determining the values of the X_i for all i where $X_i = 0$. By Cor. 1, this can be done in linear time. Therefore, let us again assume w.l.o.g. that S is reduced with $f_i \notin \mathcal{N}$ and $X_i > 0$ for all i. ($X_i \neq 1$ is not required by our algorithm). Let $\mathcal{C}$ denote the set of all constant operands of "$\sqcap$" occurring in S. The following claim immediately follows from Theorem 3:

CLAIM.

1. If $\mathcal{C} = \emptyset$ then $X_i = \infty$ for all i.
2. Assume $\mathcal{C} \neq \emptyset$ and $c = \sqcap \mathcal{C}$. If $f_i \equiv x_j \sqcap c$ or $f_i \equiv c \sqcap x_j$ then $X_i = c$.

□

EXAMPLE 4. Consider the following system of equations:

$$\begin{array}{lcllcl} x_1 & = & x_3 + 1 & x_5 & = & 7 \sqcap x_7 \\ x_2 & = & x_3 \sqcap x_6 & x_6 & = & 4 + x_1 \\ x_3 & = & 3 \sqcap x_1 & x_7 & = & x_1 \cdot x_2 \\ x_4 & = & x_5 \sqcap x_6 & & & \end{array}$$

Then, $X_i > 0$ for all i. The set $\mathcal{C}$ of constant operands of "$\sqcap$" equals $\{3, 7\}$ where 3 occurs in the third equation. Hence, $X_3 = 3$. □

Based on our claim, we construct the algorithm as follows.

(1) Compute the sets of all i where $X_i = 0$, and replace the corresponding right hand sides with 0.

(2) Construct the equivalent reduced system.

(3) Compute the set $\mathcal{C}$.

(4) While $\mathcal{C} \neq \emptyset$, execute steps (4.1) to (4.4).

(4.1) Compute $c := \sqcap \mathcal{C}$.
(4.2) Replace every right hand side $c \sqcap x_j$ and every right hand side $x_j \sqcap c$ with c.
(4.3) Construct the corresponding reduced system.
(4.4) Recompute $\mathcal{C}$.

(5) For all remaining i, replace f_i with ∞.

By appropriate data structures for S, all the updates in (4.2) and (4.3) together can be executed in time $O(|S|)$. The set $\mathcal{C}$ is kept in a priority queue. Using an efficient implementation for a priority queue (see, e.g., [5]) we find, that all minimum extractions of (4.1) together with all insertions and deletions of (4.4) can be executed in time $O(|S| \cdot \log(|S|))$. Thus, the overall complexity of the algorithm is $O(|S| \cdot \log(|S|))$. □

7. All Operations Together

THEOREM 5. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\{\sqcap, \sqcup, +, \cdot\}$, and $X_i, i = 1, \ldots, n$, is the least solution of S. Then the following holds:*

1. *Let $h > 1$ and $d \leq h$ for every constant d occurring in any of the f_i. Then*

$$X_i < \infty \quad \textit{iff} \quad X_i \leq \begin{cases} h \cdot 2^{|S|} & \textit{if } f_i \in \mathcal{N}_{\{\sqcup,\sqcap,+\}}[X_n] \textit{ for all } i \\ h^{2^{|S|}} & \textit{otherwise} \end{cases}$$

2. *The least solution of S can be computed in deterministic time $O(|S|^2)$. Multiplications are only needed provided S itself contains multiplications.*

Note that by Assertion (1) of the Theorem and Cor. 1, the least solution of S can be effectively determined. However, the resulting algorithm may have double exponential running time. Alternatively, Prop. 2 could be used to determine whether for given Y and i, $X_i < Y$. However, this algorithm would only run in *nondeterministic* polynomial time.

PROOF OF STATEMENT (1). First observe that we w.l.o.g. may assume that

1. S is normalized, and
2. $1 < X_i < \infty$ for all i.

Secondly, we observe that the given bounds easily can be seen to hold provided S only contains operations from $\{+, \cdot\}$. Therefore, our proof is based on the following claim:

CLAIM 1. A system S' of equations $x_i = f_i', i = 1, \ldots, n$, can be constructed having the same least solution as S but which uses operations from $\{+, \cdot\}$ alone.

Because of Prop. 2 we w.l.o.g. may now assume that S is a system of equations with operations from $\{\sqcup, +, \cdot\}$.

Let Q denote some strong component of the dependence graph G_S of S and some node i from Q.

CLAIM 2. If f_i contains "$\sqcup$", then some j in Q exists such that f_j contains an operand ρ_Q which is not some variable from Q such that $X_i = \rho_Q[X_1, \ldots, X_n]$. □

With the help of Claim 2, S' is obtained from S as follows.

If f_i does not contain "$\sqcup$", then $f_i' \equiv f_i$. Otherwise, we put $f_i' \equiv \rho_Q$ where Q is the strong component in which i is contained.

In order to prove the correctness of the construction, we simply execute the algorithm from the proof of Theorem 2 on both systems. It turns out that it indeed computes the same least solution on input S as on input S'. □

The proof of statement (2) is based on Theorem 3. Additionally, we need the following observations on h–least solutions of system S.

FACT 5. If $h \geq h' \geq 0$ and $X_i \geq h$ for all i then the h–least solution of S equals the h'–least solution of S. □

For $h \geq 0$, system S is called *h–reduced* if S is reduced and $h < c$ for every constant operand of "$\sqcap$" or "$\sqcup$" occurring in S.

FACT 6. Assume $h \geq 0$, and S is reduced where $X_i, i = 1, \ldots, n$, is the h-least solution of S. Assume furthermore, that $X_i > h$ for every i. Then a h–reduced system S' can be constructed in time $O(|S|)$ such that the h–least solution of S' equals $X_i, i = 1, \ldots, n$.

PROOF. For a proof observe that S cannot contain any equation $x_i = f_i$ where $f_i \equiv \rho_1 \sqcap \rho_2$ with $\rho_1 \equiv c \leq h$ or $\rho_2 \equiv c \leq h$. Therefore, consider system S' of equations $x_i = f_i'$ which is obtained from S as follows.

- If $f_i \equiv c \sqcup x_j$ or $f_i \equiv x_j \sqcup c$ where $c \in \mathcal{N}$ and $c \leq h$ then define $f_i' \equiv x_j$;
- Otherwise, define $f_i' \equiv f_i$.

Clearly, S' is h–reduced, and the h–least solution of S' equals the h–least solution of S. □

Observe that the h–reduced system S' constructed in the proof of Fact 6 has not necessarily the same least solution as S. This can be seen from the following simple system S of equations:

$$x_1 = x_1 + x_2 \qquad x_2 = x_3 \sqcup 1 \qquad x_3 = x_1 \sqcap 2$$

S has least solution $X_1 = \infty$, $X_2 = X_3 = 2$. Especially, $X_i > 1$ for all i. By Fact 5, this is also the 1–least solution of S. Therefore, we can construct the corresponding 1–reduced system S'. It is given by:

$$x_1 = x_1 + x_2 \qquad x_2 = x_3 \qquad x_3 = x_1 \sqcap 2$$

The 1–least solution of S' is indeed equal to the 1–least solution of S. The *least* solution of S', however, is different. It is given by $X_1' = X_2' = X_3' = 0$.

FACT 7. Assume $0 < h < m$, and S is a reduced system of equations $x_i = f_i$, $i = 1, \ldots, n$, over $\mathbf{m}$ (or $\mathcal{N}$ if $m = \infty$) with operations from Ω and h–least solution $X_i, i = 1, \ldots, n$ where for all i, $X_i \geq h$. Then the set of all i with $X_i = h$ can be computed in time $O(|S|)$.

PROOF. W.l.o.g. we only consider the case where $m = \infty$, i.e., S is a system of equations over $\mathcal{N}$. By Facts 5 and 6 we can also assume that S is $(h-1)$–reduced. Let $[h, \infty]$ denote the partial ordering

$$h < (h+1) < (h+2) < \ldots < \infty$$

We introduce the map $H_h : [h, \infty] \to \mathbf{2}$ defined by

$$H_h(x) = \begin{cases} 0 & \text{if } x = h \\ 1 & \text{if } x > h \end{cases}$$

We have for all $x_1, x_2 \in [h, \infty]$:

1. $H_h(x_1 + d) = 1$ for all $d > 0$.
2. If $h = 1$ then $H_h(x_1 \cdot x_2) = H_h(x_1) \sqcup H_h(x_2)$.
   If $h > 1$ then $H_h(x_1 \cdot d) = 1$ for all $d > 1$.
3. $H_h(x_1 \sqcup x_2) = H_h(x_1) \sqcup H_h(x_2)$; and finally,
4. $H_h(x_1 \sqcap x_2) = H_h(x_1) \sqcap H_h(x_2)$.

Let J denote the set of all i such that *not* $f_i \equiv c < h$, and S_h denote the system of equations $x_i = H_h f_i, i \in J$, where $H_h f_i$ is defined as follows.

- If $f_i \equiv c \geq h$ then $H_h f_i \equiv H_h c$.
- If f_i contains "+" then $H_h f_i \equiv 1$.
- If $f_i \equiv \rho_1 \cdot \rho_2$ we have to distinguish between the cases $h = 1$ and $h > 1$. If $h = 1$ then $H_h f_i \equiv H_h(\rho_1) \sqcup H_h(\rho_2)$ (where $H_h(x_j) \equiv x_j$). If $h > 1$ then $H_h f_i \equiv 1$.
- Otherwise, $H_h f_i \equiv f_i$.

Let $X_{h,i}, i = 1, \ldots, n$, denote the least solution of S_h. We find:
CLAIM. For all $i \in J$, $X_{h,i} = H_h X_i$. □
This Claim together with Fact 4 implies the assertion. □

EXAMPLE 5. Consider system S of equations

$$\begin{array}{lcllcl} x_1 & = & x_2 \sqcap x_3 & x_4 & = & x_1 \sqcup x_5 \\ x_2 & = & x_1 + x_4 & x_5 & = & x_3 \sqcap 9 \\ x_3 & = & x_4 \sqcup 7 & x_6 & = & x_2 + 4 \end{array}$$

Then S is 6–reduced with $X_i \geq 7$ for all i. To determine the set of all i with $X_i = 7$, we construct the following system of equations:

$$\begin{array}{lcllcl} x_1 & = & x_2 \sqcap x_3 & x_4 & = & x_1 \sqcup x_5 \\ x_2 & = & 1 & x_5 & = & x_3 \sqcap 1 \\ x_3 & = & x_4 & x_6 & = & 1 \end{array}$$

The occurrences of 7 and 9 are replaced with 0 and 1 respectively; moreover, the right hand sides for x_2 and x_6 have been replaced with 1. Solving this system we find that $\{i \mid X_i = 7\} = \{1, 3, 4, 5\}$. □

Proof of Statement (2). Assume system S is reduced where $f_i \notin \mathcal{N}$ and $X_i > 0$ for all i. Let $\mathcal{C}$ the set of all constant operands of "$\sqcap$" or "$\sqcup$" occurring in S.

If $\mathcal{C} = \emptyset$ then by Theorem 3, $X_i = \infty$ for all i. Therefore, assume $\mathcal{C} \neq \emptyset$ and let $c > 0$ denote the smallest element in $\mathcal{C}$. By Fact 7 we know that the set of all i where $X_i = c$ can be computed in linear time.

Assume we have replaced all f_i with the minimum c of $\mathcal{C}$ whenever $X_i \equiv c$ and afterwards constructed the corresponding reduced system. Then by Fact 6, we can remove all occurrences of c as operands of "$\sqcap$" or "$\sqcup$". This procedure can be iterated until all constant operands of "$\sqcap$" or "$\sqcup$" in the system are removed. Thus, we obtain the following algorithm to compute the least solution of a system of equations S:

(1) Compute the set of all i where $X_i = 0$, and replace the corresponding right hand sides with 0.

(2) Construct the equivalent reduced system.

(3) Compute the set $\mathcal{C}$ of all constant operands of "$\sqcap$" or "$\sqcup$" occurring in S.

(4) While $\mathcal{C} \neq \emptyset$ execute steps (4.1) to (4.4):

  (4.1) Compute $c := \bigsqcap \mathcal{C}$.

  (4.2) Determine the set J of all i where $X_i = c$.

  (4.3) For every $j \in J$ replace every right hand side f_j with c.

  (4.4) Construct the corresponding c–reduced system and recompute $\mathcal{C}$.

(5) Replace every remaining $f_i \notin \mathcal{N}$ with ∞.

To determine the complexity of the algorithm, first observe that the size of the system of equations under consideration after every iteration of the loop is strictly decreasing. Since every iteration can be executed in time $O(|S|)$ the result follows. □

8. Consequences

Depending on Ω, let t_Ω denote the function with

$$t_\Omega(x) = \begin{cases} x & \text{if } \Omega = \{\sqcup, +, \cdot\} \\ x \cdot \log(x) & \text{if } \Omega = \{\sqcup, \sqcap\} \text{ or } \Omega = \{\sqcap, +, \cdot\} \\ x^2 & \text{if } \Omega = \{\sqcup, \sqcap, +, \cdot\} \end{cases}$$

By Fact 3, Theorems 1, 2, 4 and 5 can be used to speed up Cor. 1.

Corollary 3. *For $k \geq 2$ let S be a system of equations over* $\mathbf{k}$ *with operations from $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$. Using multiplications of integers of length $O(\log(k))$, the least solution of S can be computed in time $O(t_\Omega(|S|))$.* □

Note that the given complexity bounds are independent of k.

Instead of determining the precise values of the least solution only up to some given bound k, we are sometimes also interested in computing the set of all i where $X_i = \infty$. It turns out that in this case, we can eliminate all multiplications from our algorithm. The key tool for this is provided by the following proposition.

PROPOSITION 3. *Assume S is a normalized system of equations over $\mathcal{N}$ with operations from $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$ and least solution $X_i, i = 1, \ldots, n$. Assume $X_i > 1$ for all i. Then a system $\bar{S}$ can be constructed (without multiplications) in time $O(|S|)$ with least solution $\bar{X}_i, i = 1, \ldots, n$, such that*

1. *$\bar{S}$ contains operations only from $\Omega \backslash \{\cdot\}$;*
2. *For every i, $X_i = \infty$ iff $\bar{X}_i = \infty$.*

PROOF. Define $\bar{S}$ as the system of equations obtained from S by replacing every occurrence of every $c \in \mathcal{N}$ with 1 and every occurrence of "$\cdot$" with "$+$".

CLAIM 1. Some $h > 1$ exists such that for all i, $\bar{X}_i \leq X_i \leq h^{\bar{X}_i}$.

The assertion of the proposition obviously follows from Claim 1. To prove Claim 1, we successively consider Claims 2, 3, 4 and 5.

Let S_1 be the system of equations obtained from S where every occurrence of "$\cdot$" is replaced by "$+$", and let $X_{1,i}, i = 1, \ldots, n$, denote the least solution of S_1.

CLAIM 2. For all $i = 1, \ldots, n$,

(1) $X_{1,i} \leq X_i$;

(2) $2 \leq X_{1,i}$.

Since $2 \leq X_i$ for every i, some j_0 exists such that for every i, $2 \leq X_i^{(j_0)}$. Since $y_1 + y_2 \leq y_1 \cdot y_2$ whenever $2 \leq y_1$ and $2 \leq y_2$, we conclude that for all i and j, $X_{1,i}^{(j)} \leq X_i^{(j+j_0)}$. Thus, Assertion (1) of Claim 2 follows.

To prove Assertion (2) of Claim 2 just observe that $y_1 \cdot y_2 \leq y_1 + y_2$ whenever $y_1 \leq 2$ and $y_2 \leq 2$. □

Choose some $h > 1$ such that $c \leq h$ for every $c \in \mathcal{N}$ occurring in S.

CLAIM 3. For all i,

(1) $\bar{X}_i \leq X_{1,i} \leq h \cdot \bar{X}_i$;

(2) $1 \leq \bar{X}_i$.

Assertion (1) follows by usual fixed point induction, whereas Assertion (2) is implied by Assertion (1). □

The first Assertions of Claims 2 and 3 imply the left inequality of Claim 1. Also for later use, observe that by Assertion (2) of Claim 3, $\bar{X}_i, i = 1, \ldots, n$, is also the 1–least solution of $\bar{S}$.

Next, we approximate S from above. Define S_2 as the system of equations obtained from S by replacing every occurrence of every $c \in \mathcal{N}$ with h. Let $X_{2,i}, i = 1, \ldots, n$, denote the least solution of S_2.

CLAIM 4. For all i,

(1) $X_i \leq X_{2,i}$;

(2) $h \leq X_{2,i}$.

Assertion (1) of Claim 4 again follows by fixed point induction, whereas Assertion (2) follows from Theorem 3. Observe that by (2), $X_{2,i}, i = 1, \ldots, n$, is also the h–least solution of S_2. □

Let S_3 be the system of equations obtained from S_2 by replacing every occurrence of "+" with "$\cdot$". Let $X_{3,i}, i = 1, \ldots, n$, denote the h–least solution of S_3.

CLAIM 5. For all i,

(1) $X_{2,i} \leq X_{3,i}$;

(2) $X_{3,i} = h^{\bar{X}_i}$

Both assertions follow by fixed point induction where for (1), we need that $h > 1$. For (2) recall that $\bar{X}_i = \bigsqcup_{j \geq 0} \bar{X}_i^{(j)}$ where $\bar{X}_i^{(0)} = 1$, and $X_{3,i} = \bigsqcup_{j \geq 0} X_{3,i}^{(j)}$ where $X_{3,i}^{(0)} = h$. Since $E_h : [1, \infty] \to [h, \infty]$ defined by $E_h(y) = h^y$ commutes with "$\sqcup$" and "$\sqcap$" and maps "+" to "$\cdot$", Assertion (2) follows by induction on j. □

Claims 4 and 5 together imply the right inequality of Claim 1. □

The following theorem collects our results on finiteness for subsets $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$ considered so far.

THEOREM 6. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$ and least solution $X_i, i = 1, \ldots, n$. Then the set of all i with $X_i = \infty$ can be determined in time $O(t_\Omega(|S|))$* without *multiplications.*

PROOF. The case where $\Omega = \{\sqcap, \sqcup\}$ is trivially implied by Theorem 1. Otherwise, we can by Cor. 1 and Fact 2 w.l.o.g. assume that the assumptions of Prop. 3 are satisfied. Then the results follow from Theorems 2, 4 resp. 5. □

9. Conditionals

In this section we extend the methods of the last sections to the case of systems S which additionally contain conditionals. Observe that in presence of conditionals, removal of variables x_i with, e.g., $X_i = 0$ or $X_i = 1$ is more involved, since computing the least solution of (the image of) S over **3** may not be meaningful if S contains occurrences of conditionals "$> c;$" with $c \geq 2$.

For the following assume $k \geq 2$, $\Gamma = \{> c; \mid 0 \leq c < k - 1\}$, and $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$.

THEOREM 7. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\Omega \cup \Gamma$ containing m occurrences of conditionals. Then the least solution $X_i, i = 1, \ldots, n$, of S can be computed in time $O((m + 1) \cdot t_\Omega(|S|))$. Multiplications are only needed provided S itself contained occurrences of "$\cdot$".*

One immediate method to construct the least solution of S would be to perform ordinary fixpoint iteration in the finite domain $\mathbf{k}+\mathbf{2}$ to determine all the sets $\{i \mid X_i = r\}$, $r = 0, \ldots, k$. This information can then be used in a second step to remove all conditionals which allows to apply our algorithms for systems of equations over Ω. The first step however, consumes time $O((k+1) \cdot |S|)$ which depends on the *size* of the numbers occurring in conditions whereas the complexity of the new algorithm only depends on the *number of occurrences* of conditionals. Therefore, the trivial solution favorably competes with our algorithm only provided k is sufficiently small.

PROOF. W.l.o.g. assume S is reduced, and assume that only for $i = 1, \ldots, m$, $f_i \equiv x_{j_i} > c_i; \rho_i$, and f_i does not contain a conditional for $i > m$. Let $\mathcal{D} = \mathbf{2}^m$. Then S defines a mapping $\mu_S : \mathcal{D} \to \mathcal{D}$ as follows.

For $\alpha \equiv (\alpha_1, \ldots, \alpha_m) \in \mathcal{D}$ define S_α as the system of equations $x_i = f_i', i = 1, \ldots, n$, where for $1 \leq i \leq m$,

$$f_i' \equiv \begin{cases} \rho_i & \text{if } \alpha_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

and $f_i' \equiv f_i$ for the remaining i. Let $X_{\alpha,i}, i = 1, \ldots, n$, denote the least solution of S_α. Then define $\mu_S(\alpha) = (\beta_1, \ldots, \beta_m)$ where $\beta_i = 1$ iff the i–th condition is satisfied, i.e.,

$$\beta_i = \begin{cases} 1 & \text{if } X_{\alpha,j_i} > c_i \\ 0 & \text{otherwise} \end{cases}$$

CLAIM 1. μ_S is monotonic. □

Since $\mathcal{D}$ is finite, μ_S is even continuous. Define $\alpha = \bigsqcup_{j \geq 0} \alpha^{(j)}$ where $\alpha^{(0)} = (0, \ldots, 0)$ and $\alpha^{(j)} = \mu_S(\alpha^{(j-1)})$ for $j > 0$. Then α is the least fixpoint of μ_S, and we have:

CLAIM 2. For all $i = 1, \ldots, n$,

(1) $X_{\alpha^{(j)},i} \leq X_i$ for all $j \geq 0$;

(2) $X_{\alpha,i} = X_i$.

Assertion (1) of Claim 2 follows by usual induction on j whereas for a proof of assertion (2) it suffices to show that $X_{\alpha,i}, i = 1, \ldots, n$, is indeed a solution of S. □

Observe that all the systems of equations $S_{\alpha^{(j)}}$ contain operations only from Ω. Therefore, Claims 1 and 2 give rise to the following algorithm:

(1) Initialize $\alpha := (0, \ldots, 0)$;

(2) For $j := 1$ to m execute Steps (2.1), (2.2) and (2.3):

  (2.1) Compute S_α;
  (2.2) Set $X_1, \ldots, X_n$, to the least solution of S_α;
  (2.3) Set $\alpha := \mu_S(\alpha)$;

(3) Return $X_1, \ldots, X_n$.

Since the length of the longest chain in $\mathcal{D}$ is $m+1$, the least fixpoint of μ_S equals $\alpha^{(m)}$. Therefore, m iterations suffice. Since Step (2.2) can be implemented by appropriate algorithms of the preceding sections the result follows. □

Using Fact 3, the idea for the algorithm of Theorem 7 can also be applied to compute least solutions over some finite domain $\mathbf{k}$.

COROLLARY 4. *S be a system of equations over $\mathbf{k}$ with operations from $\Omega \cup \Gamma$ containing m occurrences of conditionals. Using multiplications of integers of length $O(\log(k))$, the least solution of S can be computed in time $O((m+1) \cdot t_\Omega(|S|))$.* □

Similar to the last section, we would like to determine the set $\{i \mid X_i = \infty\}$ efficiently without using unrestricted multiplications. This is indeed possible. W.l.o.g. let us assume that our system S of equations is already normalized. Let $k-2$ be the least upper bound on the values occurring in conditions, and consider the map $H : \mathcal{N} \to \mathbf{k}$ defined by $H(x) = x \sqcap (k-1)$. Applying the algorithm of Cor. 4 to S_H we determine the sets $\{i \mid X_i = r\}$, $r = 1, \ldots, k-2$, in time $O((m+1) \cdot t_\Omega(|S|))$ using multiplications of integers of length $O(\log(k))$. Using this information, we remove in a second step all occurrences of conditions from S arriving at a system of equations S' without conditionals but with the same least solution. To S' we finally apply the algorithm from Theorem 6 to compute $\{i \mid X_i = \infty\}$. Thus, we have proved:

THEOREM 8. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\Omega \cup \Gamma$ and least solution $X_i, i = 1, \ldots, n$. If S contains m occurrences of conditionals, the set of all i with $X_i = \infty$ can be determined in time $O((m+1) \cdot t_\Omega(|S|))$ using multiplications of integers of length $O(\log(k))$.* □

As a final remark on the strength of Theorem 8 we consider an application in the field of finite tree automata (see, e.g., [1, 6, 16] for motivation and precise definitions).

A *finite tree automaton* A is a finite state device operating on (finite ordered labeled) trees; a *cost function* $\mathbf{c}$ for A over semiring $\mathcal{R}$ maps every transition of A to some polynomial over $\mathcal{R}$ which determines how the cost of the whole computation is computed from the costs for the subcomputations. In [16] cost functions over $\mathcal{N}$ are considered with operations "$\sqcap$" and "$+$", and it is proven that it can be decided whether or not the least upper bound on the costs of all accepting computations is finite. This was done by proving an explicit upper bound on the costs of accepting computations provided the least upper bound is finite. The decision procedure implied by this upper bound could be implemented in polynomial space.

It turns out that the given problem can be described by a system of equations over $\mathcal{N}$ with operations from $\{\sqcap, \sqcup, +\}$. Therefore, Theorem 8 allows both an improvement in the complexity and a generalization to much more complicated cost functions. We obtain:

THEOREM 9. *For every finite tree automaton A and every cost function for A over $\mathcal{N}$ using operations from $\Omega = \{\sqcap, \sqcup, +, \cdot\} \cup \{>c;\ |\ c \in \mathcal{N}\}$, it can be decided in deterministic polynomial time whether the least upper bound on the costs of all accepting computations is finite.* □

10. Greatest Solutions

In this section, we consider the computation of *greatest* solutions. Let S denote the system of equations $x_i = f_i, i = 1, \ldots, n$ where f_i are polynomials over $\mathcal{N}$ with operations from $\mathcal{N}$. Since the f_i are monotone, and $\mathcal{N}$ satisfies the *descending chain condition* (*dcc*), the f_i are also continuous w.r.t. "$\geq$". Hence, the greatest solution $\bar{X}_i, i = 1, \ldots, n$, of S is be obtained by

$$\bar{X}_i = \bigsqcap_{j \geq 0} \bar{X}_i^{(j)}$$

where $\bar{X}_i^{(0)} = \infty$ and for $j > 0$,

$$\bar{X}_i^{(j)} = f_i[\bar{X}_1^{(j-1)}, \ldots, \bar{X}_n^{(j-1)}]$$

Note that (because of *dcc*), fixpoint iteration always terminates. However, it need not be efficient since descending chains may have arbitrary length. Therefore, it makes sense to ask whether efficient algorithms exist analogous to those for computing least solutions. It turns out that this is indeed the case. Let us again start by considering systems S of equations without occurrences of conditionals. Recall that by Cor. 2, we w.l.o.g. may assume that S is reduced.

THEOREM 10. *The greatest solution of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcup, +, \cdot\}$ can be computed in time $O(|S|)$.*

PROOF. Let S be the system $x_i = f_i, i = 1, \ldots, n$, and assume w.l.o.g. that S is reduced. Let $\bar{X}_i, i = 1, \ldots, n$, denote the greatest solution of S. Our key observation is:

CLAIM. Let Q be a strong component of G_S containing at least one edge. Then $\bar{X}_i = \infty$ for all $i \in Q$. □

Since moreover, $\bar{X}_j = \infty$ whenever $\bar{X}_i = \infty$, and j is reachable from i, we obtain the following simple algorithm:

(1) Reduce S;

(2) Replace every right hand side $f_i \notin \mathcal{N}$ with ∞;

This algorithm trivially runs in linear time. □

Next, we consider the general case where system S contains occurrences of "$\sqcap$" as well. The main result of this section is:

THEOREM 11. *The greatest solution of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcup, \sqcap, +, \cdot\}$ can be computed in time $O(|S| \cdot \log(|S|))$. Multiplications are only needed if S itself contains occurrences of "$\cdot$".*

PROOF. Let S be the system $x_i = f_i, i = 1, \ldots, n$, and assume w.l.o.g. that S is reduced and $f_i \notin \mathcal{N}$ for all i. Let $\bar{X}_i, i = 1, \ldots, n$, denote the greatest solution of S and $\mathcal{C}$ the set of constant operands of "$\sqcap$" or "$\sqcup$" occurring in S. Analogous to Theorem 3, we would like to establish a lower bound on the components $\bar{X}_i$.

CLAIM 1. $\sqcap \mathcal{C} \leq \bar{X}_i$ for all i. Especially, $\mathcal{C} = \emptyset$ implies that $\bar{X}_i = \infty$ for all i.

The proof of this claim follows directly by fixpoint induction. □

Next, we show how occurrences of the minimal operand in $\mathcal{C}$ safely can be removed. For $c = \sqcap \mathcal{C}$, define the system S' of equations $x_i = f'_i, i = 1, \ldots, n$, where f'_i is defined as follows.

- If $f_i \equiv \rho \sqcup c$ or $f_i \equiv c \sqcup \rho$ then $f'_i \equiv \rho$.
- If $f_i \equiv \rho \sqcap c$ or $f_i \equiv c \sqcap \rho$ then $f'_i \equiv c$.
- Otherwise, $f'_i \equiv f_i$.

Let $\bar{X}'_i, i = 1, \ldots, n$, denote the greatest solution of S'. Then the following holds:

CLAIM 2. $\bar{X}_i = \bar{X}'_i$ for all i. □

Observe that $|S'| < |S|$. Therefore, Claims 1 and 2 allow to construct the following algorithm:

(1) Construct the corresponding reduced system;

(2) Compute the set $\mathcal{C}$;

(3) While $\mathcal{C} \neq \emptyset$ execute steps (3.1), (3.2) and (3.3):

  (3.1) Compute $c := \sqcap \mathcal{C}$;
  (3.2) Remove all occurrences of c as an operand of "$\sqcup$" or "$\sqcap$";
  (3.3) Construct the corresponding reduced system, and recompute $\mathcal{C}$;

(4) Replace the remaining right hand sides $f_i \notin \mathcal{N}$ with ∞.

Since every iteration of the loop strictly decreases the size of the system of equations, we have at most $O(|S|)$ iterations. By keeping $\mathcal{C}$ in an appropriate data structure, all minimum extractions together with all new insertions take time $O(|S| \cdot \log(|S|))$. Since all remaining actions only consume time $O(|S|)$ the theorem follows. □

A simple special case of Theorem 11 is the problem of determining for every vertex v in a graph G the minimal cost of paths (w.r.t. to a positive cost measure on the edges of G) from a distinguished source v_0 to v. Here, our algorithm specializes to (a variant of) Dijkstra's algorithm (see, e.g., [10]).

Similar to the case of least solutions, we would like to determine for some $k > 1$, the sets $\{i \mid \bar{X}_i = y\}$ with $y = 0, \ldots, k-2$, without using multiplications of arbitrary numbers. The simplest idea is to execute the ordinary fixpoint iteration for system S but doing all computations within finite domain $\mathbf{k}$. This results in an algorithm whose runtime depends on k. Instead however, we may use the algorithm from Theorem 11 resulting in an algorithm with a runtime independent of k. We obtain:

COROLLARY 5. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\{\sqcup, \sqcap, +, \cdot\}$ with greatest solution $\bar{X}_i, i = 1, \ldots, n$. For $k > 1$, the sets $\{i \mid \bar{X}_i = y\}$ with $y = 0, \ldots, k-2$, can be computed in time $O(|S| \cdot \log(|S|))$ by using multiplications only of numbers up to length $O(\log(k))$.* □

We also would like to compute the set $\{i \mid \bar{X}_i = \infty\}$ efficiently without using multiplications.

THEOREM 12. *Assume S is a system of equations over $\mathcal{N}$ with operations from $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$ and greatest solution $\bar{X}_i, i = 1, \ldots, n$. Then the set of all i with $\bar{X}_i = \infty$ can be determined in polynomial time.*

Depending on Ω, we achieve the following complexity bounds:

$$\begin{array}{lll} \Omega = \{\sqcup, +, \cdot\} & : & O(|S|) \\ \Omega = \{\sqcup, \sqcap, +, \cdot\} & : & O(|S| \cdot \log(|S|)) \end{array}$$

The proof is based on the next proposition which is the analogue to Prop. 3 in case of least solutions. The proof of this proposition is based on a similar construction.

PROPOSITION 4. *Assume S is a normalized system of equations over $\mathcal{N}$ with operations from $\Omega \subseteq \{\sqcup, \sqcap, +, \cdot\}$ and greatest solution $\bar{X}_i, i = 1, \ldots, n$. Assume $\bar{X}_i > 0$ for all i. Then a system S' can be constructed (without multiplications) in time $O(|S|)$ with greatest solution $X'_i, i = 1, \ldots, n$, such that*

1. *S' contains operations only from $\Omega \backslash \{\cdot\}$;*
2. *For every i, $\bar{X}_i = \infty$ iff $X'_i = \infty$.* □

An extension of the presented greatest fixpoint methods to systems of equations containing one–sided conditionals is possible as well. Since we compute greatest solutions we start the iteration with the assumption that all conditions hold true. The rest of the computation proceeds completely along the lines of Section 9.

11. Conclusion

We presented efficient algorithms which compute the least and greatest solutions of a system S of equations over $\mathcal{N}$ for various sets of operations. The algorithms use multiplications only provided S itself contains multiplications. In order to compute the set of all components where the result is ∞, we derived polynomial time algorithms without multiplications. We extended our results by allowing occurrences of conditionals as well.

References

[1] Courcelle, B. and Mosbah, M. 1993. Monadic Second–Order Evaluations on Tree–Decomposable Graphs. *Theoretical Computer Science 109*, 49–82.

[2] Cousot, P. and Cousot, R. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *4th Symposium on Principles of Programming Languages*. Los Angeles, California, 238–252.

[3] Cousot, P. and Cousot, R. 1992. Abstract Interpretation and Application to Logic Programs. *Journal of Logic Programming 13*, 2 and 3 (July), 103–179.

[4] Cousot, P. and Cousot, R. 1992. Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. Report LIENS-92-16, Paris.

[5] Fredman, M. L. and Tarjan, R. E. 1987. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM 34*, 597–615.

[6] Habel, A., Kreowski, H.-J., and Vogler, W. 1991. Decidable Boundedness Problems for Sets of Graphs Generated by Hyperedge-Replacement. *Theoretical Computer Science 89*, 33–62.

[7] Hankin, C. and Hunt, S. 1991. Fixed Points and Frontiers: A New Perspective. *Journal of Functional Programming 1*, 91–120.

[8] Kennedy, K. 1981. A Survey of Data Flow Analysis Techniques. In S. S. Muchnick and N. D. Jones, editors, *Program Flow Analysis. Theory and Applications*. Englewood Cliffs, New Jersey, Prentice–Hall, 5–54.

[9] Marlowe, T. J. and Ryder, B. G. 1990. Properties of Data Flow Frameworks. *Acta Informatica 28*, 121–163.

[10] Mehlhorn, K. 1984. *Data Structures and Algorithms, Vol. 2: Graph Algorithms and NP–Completeness*. EATCS Monographs on Theoretical Computer Science. Springer, New York, Heidelberg.

[11] Nielson, F. and Nielson, H. R. 1992. Bounded Fixed Point Iteration. *Journal of Logic and Computation 2*, 441–464.

[12] Nielson, F. and Nielson, H. R. 1992. Finiteness Conditions for Fixed Point Iteration. In *Proceedings of the 1992 ACM Conference on LISP and Functional Programming*, 96–108.

[13] Peyton-Jones, S. and Clack, C. 1987. Finding Fixpoints in Abstract Interpretations. In S. Abramsky and C. Hankin, editors, *Abstract Interpretation of Declarative Languages*. Ellis Horwood Ltd. and John Wiley, 246–265.

[14] Seidl, H. 1993. Equality of Instances of Variables in `FORK`. Report 6/93, SFB 124–C1, Saarbrücken.

[15] Seidl, H. 1994. Least Solution of Equations over $\mathcal{N}$. In E. Shamir S. ASbiteboul, editor, *Proceedings of the 21st International Colloquium on Automata, Languages and Programming (ICALP)*, Volume 820 of *Lecture Notes in Computer Science*. Springer Verlag, 400–411.

[16] Seidl, H. 1994. Tree Automata with Cost Functions. *Theoretical Computer Science 126*, 113–142. Special Issue on CAAP'92.