Least Solutions of Equations over $\mathcal{N}$

Helmut Seidl

Fachbereich Informatik
Universität des Saarlandes
Postfach 151150
D–66041 Saarbrücken
Germany
seidl@cs.uni-sb.de

Abstract. We consider the problem of computing the least solution $X_i, i = 1, \ldots, n$, 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 operations taken from some set Ω. We present efficient algorithms for various subsets Ω of the operations minimum, maximum, addition and multiplication.

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 f and g 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.

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., [CC77, Ke81, MR90, CC92b]. 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 [PC87, HH91, NN92a, NN92b]. 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 [CC77, CC92a]. 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 [Se93], an algorithm is proposed which detects whether parallely existing instances of variables 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. Here, systems of equations over $\mathcal{N}$ are needed 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). Also, solving such systems of equations can used to implement interval analysis (cf. [CC92a]).

Therefore, in this paper we investigate the computation of least solutions for systems of equations over $\mathcal{N}$ using various sets of continuous operations and give efficient solutions 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 "$\sqcap$".

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, the size $|S|$ of system S of equations $x_i = f_i, i = 1, \ldots, n$, is defined 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

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

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. We derive efficient algorithms to determine the set of all i where $X_i < \infty$ without using multiplications at all. It follows that, provided the finite X_i are smaller than some h, operations on numbers of length at most $\log(h)$ suffice.

The rest of the paper is organized as follows. The next section provides basic algorithmic facts, especially on the computation of least solutions of equations over certain finite cpo's. Sections 3 through 5 present special solutions for various subsets of operations, whereas Section 6 deals with all operations together. Section 7 shows how our basic algorithms can be applied to speed up the computation of least solutions over restricted ranges of numbers, and derive polynomial finiteness tests without multiplications. This result is used 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 [Se92]. Section 8 concludes.

2 Basic Facts

Let S denote a system of equations $x_i = f_i$ over $\mathcal{N}$. It turns out that it is sometimes 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 $\Box \in \{\cdot, +\}$;
- 0 and ∞ do not occur as opperands; and
- 1 does not occur as an operand of "$\cdot$".

Systems with these properties are called *normalized*. The set of *constants* of the normalized system S is the set of all $c \in \mathcal{N}$ that occur in S as operands of "$\sqcup$" or "$\sqcap$". Observe that 0 or ∞ never can be constants of a normalized system.

A normalized system is called *reduced* iff for no j, $f_j \equiv c_1 \Box c_2$ for $c_1, c_2 \in \mathcal{N}$. We have:

Fact 2. *1. 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 multiplicatons) in time $O(|S|)$ with least solution $X_1, \ldots, X_m$ such that $X_1, \ldots, X_n$ is also the least solution of S.*

2. *For every normalized system S a reduced system S' with the same variables can be constructed in time $O(|S|)$ with the same least solution as S. Multiplications are only needed provided S itself contains multiplications.* □

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. The following fact is well–known:

Fact 3. *The least solution of a system of equations over $\mathbf{2}$ with operations from Ω can be computed in linear time.* □

Let $H : \mathcal{N} \to \mathbf{k}$ denote the mapping defined by

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

H is continuous and commutes with corresponding operations. Moreover, H is *faithful* on $\{0, \ldots, k-2\}$, i.e., $\{y\} = H^{-1}(H(y))$ for all $y \in \{0, \ldots, k-2\}$.

For $f \in \mathcal{N}[X]$ let $(Hf) \in \mathbf{k}[X]$ 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 Ω. We find:

Fact 4. *Let $X_H = X_{H,1}, \ldots, X_{H,n}$ denote the least solution of S_H. Then $X_{H,i} = H(X_i)$ for every i.* □

Call system S of equations over $\mathcal{N}$ *k–reduced* iff S is reduced and $k \leq c$ for every constant c of S.

Fact 5. *Assume S is reduced with least solution $X_i, i = 1, \ldots, n$, and $X_i \geq k$ for every i. Then a k–reduced system with the same least solution can be constructed in time $O(|S|)$.*

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 $\rho_1 \equiv c < k$ or $\rho_2 \equiv c < k$. 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 $c < k$ then define $f_i' \equiv x_j$;
- Otherwise, define $f_i' \equiv f_i$.

Clearly, S' is k–reduced, and S' and S have the same least solutions. □

Proposition 6. *Assume $0 < k < m$, 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 least solution $X_i, i = 1, \ldots, n$ where for all i, $X_i \geq k$. Then the set of all i with $X_i = k$ 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 Fact 5 we can also assume that S is k–reduced. Let $[k, \infty]$ denote the partial ordering

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

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

$$H_k(x) = \begin{cases} 0 \text{ if } x = k \\ 1 \text{ if } x > k \end{cases}$$

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

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

Let S_k denote the system of equations $x_i = H_k f_i, i = 1, \ldots, n$, where $H_k f_i$ is defined as follows.

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

Let $X_{k,i}, i = 1, \ldots, n$, denote the least solution of S_k. We find:

Claim. For all i, $X_{k,i} = H_k X_i$. □

This Claim together with Fact 3 implies the assertion. □

Proposition 6 can be used to speed up ordinary fixed point iteration for the finite cpo's $\mathbf{k}$.

Theorem 7. *Let $k \geq 2$. Then the least solution of a system S of n equations over $\mathbf{k}$ with operations from Ω 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 at most* $\log(k)$.

Proof. Let $X_i, i = 1, \ldots, n$ denote the least solution of S. The algorithm proceeds as follows:

(1) Construct the corresponding reduced system.
(2) For $j := 0$ to $j := k - 1$ execute steps (2.1), (2.2) and (2.3).
  (2.1) Compute the set J_j of i where $X_i = j$.

(2.2) For every $i \in J_j$ replace f_i with j, and remove all occurrences of x_i in right hand sides.
(2.3) Construct the corresponding reduced system without occurrences of constants $c \leq j$.

For $j = 0$, the algorithm of Fact 3 can be used to implement Step (2.2) whereas for $j > 0$ the implementation is given by Prop. 6. Since the j–th iteration of the **for**–loop is executed in time $O(|S|)$ (independently of j), the result follows. □

By Fact 4 and Theorem 7 we find:

Corollary 8. *Assume S is a system of equations over $\mathcal{N}$ with least solution $X_i, i = 1, \ldots, n$, and $k \geq 0$. Then we can compute the sets $\{i \mid X_i = y\}$, $y = 0, \ldots, k$, in time $O((k+1) \cdot |S|)$. Provided S contains multiplications, the algorithm may use multiplications as well but only of integers of length at most* $\log(k)$. □

3 Minimum and Maximum

Theorem 9. *The least solution of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcap, \sqcup\}$ can be computed (without multiplications) in time $O(|S| \cdot \log(|S|))$.*

Proof. Let S be the system $x_i = f_i, i = 1, \ldots, n$, with least solution $X_i, i = 1, \ldots, n$. W.l.o.g. S is reduced. Let J denote the set of all i where $f_i \notin \mathcal{N}$, and let $\mathcal{C}$ denote the set of constants of S. We observe:

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

Therefore, assume $\mathcal{C} \neq \emptyset$, and $c = \sqcup\mathcal{C}$. Clearly, $X_i \leq c$ for all $i \in J$. We show that all occurrences of c in S as an operand can be safely removed. the method is dual to the algorithm of Prop. 6 where we removed occurrences of variables corresponding to *least* constants. 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 but a smaller set of constants. 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 $c := \sqcup\mathcal{C}$.
  (3.2) Remove all occurrences of c as an operand.

(3.3) Construct again the equivalent reduced system.
(3.4) Remove c from $\mathcal{C}$, and recompute J.
(4) Finally, set $f_i \equiv 0$ 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, the maximum computations in Step (3.1) altogether consume only time $O(|S|)$. Therefore, once this representation for $\mathcal{C}$ has been computed, the remaining steps of the algorithm together take time $O(|S|)$. Therefore, if the constants $d \in \mathcal{N}$ occurring in the f_i are from a small range we may employ bucket sort instead of some general sorting algorithm. This brings the complexity down to $O(|S|)$ for this case. □

4 Maximum, Addition and Multiplication

Theorem 10. *The least solution of of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcup, +, \cdot\}$ can be computed in time $O(|S|)$. Multiplications are only needed provided S itself contains multiplications.*

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

(1) We determine the values of 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. 8, 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 the graph $G_S = (V_S, E_S)$ where $V_S = \{1, \ldots, n\}$, and $(i, j) \in E_S$ iff x_i occurs in f_j. G_S is also called *dependence graph* for 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 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 $X_j = \infty$. □

Assuming that Claims 1 and 2 together characterize all i with $X_i = \infty$, it is easy to 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 constructed the equivalent reduced system. Therefore, now assume S is a reduced system where f_j does not contain "$+$" or "$\cdot$" for every 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 $j \in Q$. Let $\mathcal{C}_Q$ denote the set of all constants c occurring in $f_i, i \in Q$. We observe:

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

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

The ideas in the proof of Theorem 10 allow to derive an important lower bound on the non–zero components X_i of the least solution.

Proposition 11. *Consider a reduced system S 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 constants of S. Assume $X_i, i = 1, \ldots, n$ is the least solution of S where $X_i > 0$ 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.* □

Now consider a normalized system S of equations $x_i = f_i, i = 1, \ldots, n$, over $\mathcal{N}$ with operations from $\Omega = \{\sqcap, \sqcup, +, \cdot\}$ and 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 12. *Let $X_{\mu,i}, i = 1, \ldots, n$, the 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}. 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. 12 can be used to generalize the observation of Prop. 11 to systems of equations that also contain occurrences of "$\sqcap$".

Theorem 13. *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 constants of S. Assume $X_i, i = 1, \ldots, n$ is the least solution of S where $X_i > 0$ 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. 12. S_{μ_0} has the same 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 constants 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. 11, $\sqcap\mathcal{C}' \leq X_i$ for all $i \notin \mathcal{R}$, the assertion follows. □

5 Minimum, Addition and Multiplication

Theorem 14. *The least solution of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcap, +, \cdot\}$ can be computed in time $O(|S| \cdot \log(|S|))$. Multiplications are only needed provided S itself contains multiplications.*

Proof. Let S be the system $x_i = f_, i = 1, \ldots, n$, and $X_1, \ldots, X_n$ denote the least solution of S. We start by determining the set of all i where $X_i = 0$. By Cor. 8, this can be done in linear time. Therefore, let us again w.l.o.g. assume that S is reduced with $X_i > 0$ for all i. ($X_i \neq 1$ is not required by our algorithm). Let $\mathcal{C}$ denote the set of all constants of S. The following claim immediately follows from Theorem 13:

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 $X_i = c$.

Based on this claim, an algorithm may proceed 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 all right hand sides $c \sqcap x_j$ or $x_j \sqcap c$ 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., [FT87]) 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 $O(|S| \cdot \log(|S|))$. □

6 All Operations Together

Theorem 15. *The least solution of a system S of equations over $\mathcal{N}$ with operations from $\{\sqcap, \sqcup, +, \cdot\}$ can be computed in deterministic time $O(|S|^2)$. Multiplications are only needed provided S itself contains multiplications.*

Note that Prop. 12 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. Let us w.l.o.g. assume that system S of equations $x_i = f_i, i = 1, \ldots, n$, is reduced with $f_i \notin \mathcal{N}$ for all i where $X_i, i = 1, \ldots, n$, is the least solution of S and $X_i > 0$ for every i. Let $d = \sqcap\{X_i \mid i = 1, \ldots, n\}$ and $\mathcal{C}$ the set of all constants occurring in S.

If $\mathcal{C} = \emptyset$ then by Theorem 13, $X_i = \infty$ for all i. Therefore, assume $\mathcal{C} \neq \emptyset$ and let $c > 0$ denote the smallest element in $\mathcal{C}$. By Prop. 6 we know that the set of all i where $X_i = c$ can be computed in linear time.

Now assume we have replaced all f_i with the minimum c of $\mathcal{C}$ whenever $X_i \equiv c$ and afterwards reduced the resulting system. Then by Fact 5, we can remove all occurrences of c as operands of "$\sqcap$" or "$\sqcup$". This procedure can be iterated until all constants in the system are removed. Thus, we obtain the following algorithm:

(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 occurring constants.
(4) While $\mathcal{C} \neq \emptyset$ execute steps (4.1) to (4.4):
  (4.1) Compute $c := \sqcap\mathcal{C}$.
  (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 equivalent $(c+1)$–reduced system and recompute $\mathcal{C}$.
(5) Replace every remaining $f_i \notin \mathcal{N}$ with ∞.

In order to check that the proposed algorithm runs in quadratic time, we observe that the size of the system of equations under consideration after every iteration of the loop is strictly decreasing. Since for every i, the i–th iteration can be executed in time $O(|S|)$ independent of i, the result follows. □

7 Consequences

By Fact 4, Theorems 9, 10, 14 and 15 can be used to speed up Theorem 7.

Corollary 16. *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 only of length $\log(k)$, the least solution of S can be computed in polynomial time independent of k.*

Depending on Ω, we can achieve the following complexity bounds:

$$\begin{array}{lll} \Omega = \{\sqcup, \sqcap\} & : & O(|S| \cdot \log(|S|)) \\ \Omega = \{\sqcup, +, \cdot\} & : & O(|S|) \\ \Omega = \{\sqcap, +, \cdot\} & : & O(|S| \cdot \log(|S|)) \\ \Omega = \{\sqcup, \sqcap, +, \cdot\} & : & O(|S|^2) \end{array}$$

Instead of determining the precise values of the least solution only up to some given bound k, we are 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 17. *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 occurence of every $c \in \mathcal{N}$ with 1 and every occurrence of "$\cdot$" with "+". Then the assertion is implied by the following observation:

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

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

Theorem 18. *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 polynomial time* without *multiplications.*

Depending on Ω, we can achieve the following complexity bounds:

$$\begin{array}{lll} \Omega = \{\sqcup, +, \cdot\} & : & O(|S|) \\ \Omega = \{\sqcap, +, \cdot\} & : & O(|S| \cdot \log(|S|)) \\ \Omega = \{\sqcup, \sqcap, +, \cdot\} & : & O(|S|^2) \end{array}$$

Proof. By Cor. 8 and Fact 2 we can w.l.o.g. assume that the assumptions of Prop. 17 are satisfied. Then the results follow from Theorems 10, 14 resp. 15. □

The strength of Theorem 18 can be examplified by an application in the field of finite tree automata (see, e.g., [CM93, HVK91, Se92] 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 can be computed from the costs for the subcomputations. In [Se92] 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 18 allows both an improvement in the complexity and a generalization to much more complicated cost functions. We obtain:

Theorem 19. *For every finite tree automaton A and every cost function for A over $\mathcal{N}$ using operations from $\Omega = \{\sqcap, \sqcup, +, \cdot\}$, it can be decided in deterministic polynomial time (on a RAM with uniform cost measure but without multiplications) whether the least upper bound on the costs of all accepting computations is finite.* □

8 Conclusion

We presented efficient algorithms which compute the least solution $X_i, i = 1, \ldots, n$, of a system S of equations over $\mathcal{N}$ with various sets Ω of operations. The algorithms used multiplications only provided S itself contained multiplications. In order to compute the set of all i where $X_i = \infty$, we derived polynomial time algorithms without multiplications at all.

References

[CM93] B. Courcelle, M. Mosbah: Monadic Second–Order Evaluations on Tree–Decomposable Graphs. Theor. Comp. Sci. 109 (1993), 49–82

[CC77] P. Cousot, R. Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. 4th Symp. on Principles of Programming Languages, 238–252, Los Angeles, California, 1977

[CC92a] P. Cousot, R. Cousot: Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. Tech. Report LIENS – 92 – 16, Paris, 1992

[CC92b] P. Cousot, R. Cousot: Abstract Interpretation and Application to Logic Programs. Tech. Report LIX/RR/92/08, Palaiseau, 1992

[FT87] M.L. Fredman, R.E. Tarjan: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. JACM (34), 597–615, 1987

[HKV91] A. Habel, H.–J. Kreowski, W. Vogler: Decidable Boundedness Problems for Sets of Graphs Generated by Hyperedge-Replacement. Theor. Comp. Sci. 89 (1991), 33–62

[HH91] C. Hankin, S. Hunt: Fixed Points and Frontiers: A New Perspective. J. of Functional Programming (1), 91–120, 1991

[Ke81] K. Kennedy: A Survey of Data Flow Analysis Techniques. In: S.S. Muchnick, N.D. Jones (eds.): *Program Flow Analysis. Theory and Applications*. Prentice–Hall, 1981

[MR90] T.J. Marlowe, B.G. Ryder: Properties of Data Flow Frameworks. Acta Informatica (28), 121–163, 1990

[NN92a] F. Nielson, H.R. Nielson: Bounded Fixed Point Iteration. J. Logic Computat. (2), 441–464, 1992

[NN92b] F. Nielson, H.R. Nielson: Finiteness Conditions for Fixed Point Iteration. Prog. of LISP and Functional Programming 1992, 96–108

[PC87] S. Peyton–Jones, C. Clack: Finding Fixpoints in Abstract Interpretations. In: S. Abramsky, C. Hankin (Eds.): *Abstract Interpretation of Declarative Languages,* 246–265 Ellis Horwood Ltd. and John Wiley, 1987

[Se92] H. Seidl: Tree Automata with Cost Functions. Proc. CAAP'92, LNCS 581, 279–299, 1992; long version to appear in TCS, special issue on CAAP'92

[Se93] H. Seidl: Equality of Instances of Variables in `FORK`. Tech. Rep. 6/93, SFB 124–C1, Saarbrücken, 1993

This article was processed using the LaTeX macro package with LLNCS style