Decidability of Systems of Set Constraints with Negative Constraints[*]

Alexander Aiken[†]
University of California, Berkeley
aiken@cs.berkeley.edu

Dexter Kozen[‡]
Cornell University
kozen@cs.cornell.edu

Ed Wimmers[§]
IBM Almaden Research Center
wimmers@almaden.ibm.com

Abstract

Set constraints are relations between sets of terms. They have been used extensively in various applications in program analysis and type inference. Recently, several algorithms for solving general systems of positive set constraints have appeared. In this paper we consider systems of mixed positive and negative constraints, which are considerably more expressive than positive constraints alone. We show that it is decidable whether a given such system has a solution. The proof involves a reduction to a number-theoretic decision problem that may be of independent interest.

1 Introduction

Set constraints are formal inclusions or negated inclusions between expressions representing subsets of T_Σ, the set of ground terms over a finite ranked

[*] Revised and expanded version of [2].

[†] EECS Division, University of California, Berkeley, CA 94720

[‡] Computer Science Department, Cornell University, Ithaca, NY 14853

[§] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120 alphabet Σ. Formally, a *positive set constraint* is of the form $E \subseteq F$ and a *negative set constraint* is of the form $E \not\subseteq F$, where E and F are expressions built from a set $X = \{x, y, \ldots\}$ of variables ranging over subsets of T_Σ, the usual set-theoretic operators 0, 1, $\cup$, $\cap$, and $\sim$, and an n-ary set operator f for each n-ary symbol $f \in \Sigma$ with semantics

$$f(A_1, \ldots, A_n) \quad = \quad \{ft_1 \ldots t_n \mid t_i \in A_i,\ 1 \le i \le n\}\ .$$

A system $\mathcal{S}$ of constraints is *satisfiable* if there is an assignment of subsets of T_Σ to the variables satisfying all the constraints in $\mathcal{S}$.

Set constraints have numerous applications in program analysis and type inference [3, 4, 7, 14, 15, 18, 19, 20, 22]. Most of these systems deal with positive constraints only. Several algorithms for determining the satisfiability of general systems of positive constraints have appeared [1, 5, 6, 11, 13]. In [1], the satisfiability problem for a system $\mathcal{S}$ of positive constraints is shown to be equivalent to deciding whether a certain finite hypergraph constructed from $\mathcal{S}$ has an induced subhypergraph that is *closed* (see Section 4). This characterization is used to obtain an exhaustive hierarchy of complexity results depending on the number of elements of Σ of each arity.

In this paper we consider systems of mixed positive and negative constraints. Negative constraints considerably increase the power of the constraint language and have important applications in program analysis. For example, in [3, 4], opportunities for program optimization are identified by an *ad hoc* technique for checking the satisfiability of systems of negative constraints. Set constraints with only nullary symbols correspond to Boolean algebras over a finite set of atoms; in [17] general results on solving negative constraints in arbitrary Boolean algebras are given.

In this paper we give a general decision procedure for determining whether a given system of mixed positive and negative set constraints over an arbitrary signature is satisfiable. The proof reduces the satisfiability problem to a reachability problem involving Diophantine inequalities which may be of independent interest. We reduce the satisfiability problem to the Diophantine problem and then show that the Diophantine problem is decidable. The proof has a nonconstructive step involving Dickson's Lemma and does not give any complexity bounds.

The decidability result for systems of mixed positive and negative set constraints was obtained independently by Gilleron, Tison, and Tommasi [12] using automata-theoretic techniques. Stefánsson [21] has subsequently shown that the Diophantine satisfiability problem is *NP*-complete and the satisfiability problem for systems of mixed positive and negative set constraints is complete for *NEXPTIME*. Charatonik and Pacholski [8] have given an alternative proof of this result based on the approach of [6] involving monadic logic, and have extended the result to include projections [9]. Relationships between these various approaches have been drawn in [16].

2 Set Expressions and Set Constraints

There is some variation in the literature regarding the definition of set expressions and set constraints, depending on the operations allowed. The following definition is taken from [1].

Let Σ be a finite ranked alphabet consisting of symbols f, each with an associated arity $\mathbf{arity}(f) \in \mathbb{N}$. Symbols in Σ of arity 0, 1, 2, and n are called *nullary, unary, binary*, and *n-ary*, respectively. Nullary elements are often called *constants*. The set of elements of Σ of arity n is denoted Σ_n.

The set of ground terms over Σ is denoted T_Σ. This is the smallest set such that if $t_1, \ldots, t_n \in T_\Sigma$ and $f \in \Sigma_n$, then $ft_1 \ldots t_n \in T_\Sigma$. If $X = \{x, y, \ldots\}$ is a set of variables, then $T_\Sigma(X)$ denotes the set of terms over Σ and X, considering the elements of X as symbols of arity 0.

Let $\mathrm{B} = (\cup,\ \cap,\ \sim,\ 0,\ 1)$ be the usual signature of Boolean algebra. Other Boolean operators such as $\oplus$ (symmetric difference) are defined from these as usual. Let $\Sigma + \mathrm{B}$ denote the signature consisting of the disjoint union of Σ and B. A *set expression* over X is any element of $T_{\Sigma+\mathrm{B}}(X)$. The following is a typical set expression:

$$f(g(x \cup y), {\sim} g(x \cap y)) \quad \cup \quad a$$

where $f \in \Sigma_2$, $g \in \Sigma_1$, $a \in \Sigma_0$, and $x, y \in X$. We use $E, F, \ldots$ to denote set expressions. A *Boolean expression* over X is any element of $T_\mathrm{B}(X)$.

A *positive set constraint* is a formal inclusion $E \subseteq F$, where E and F are set expressions. We also allow equational constraints $E = F$, although inclusions and equations are interdefinable: $E \subseteq F$ is equivalent to $E \cup F = F$, and $E = F$ is equivalent to $E \oplus F \subseteq 0$. A *negative set constraint* is the negation of a positive set constraint: $E \not\subseteq F$ or $E \neq F$.

We interpret set expressions over the powerset 2^{T_Σ} of T_Σ. This forms an algebra of signature $\Sigma + \mathrm{B}$ where the Boolean operators have their usual set-theoretic interpretations and elements $f \in \Sigma_n$ are interpreted as functions $f : (2^{T_\Sigma})^n \to 2^{T_\Sigma}$ such that

$$f(A_1, \ldots, A_n) \quad = \quad \{ft_1 \ldots t_n \mid t_i \in A_i,\ 1 \leq i \leq n\} \ .$$

A *set assignment* is a map

$$\sigma : X \quad \to \quad 2^{T_\Sigma}$$

assigning a subset of T_Σ to each variable in X. Any set assignment σ extends uniquely to a $(\Sigma + \mathrm{B})$-homomorphism

$$\sigma : T_{\Sigma+\mathrm{B}}(X) \quad \to \quad 2^{T_\Sigma}$$

by induction on the structure of the set expression in the usual way. The set assignment σ satisfies the positive constraint $E \subseteq F$ if $\sigma(E) \subseteq \sigma(F)$, and satisfies the negative constraint $E \not\subseteq F$ if $\sigma(E) \not\subseteq \sigma(F)$. We write $\sigma \models \varphi$ if the set assignment σ satisfies the constraint φ. A system $\mathcal{S}$ of set constraints is *satisfiable* if there is a set assignment σ that satisfies all the constraints in $\mathcal{S}$; in this case we write $\sigma \models \mathcal{S}$. We write $\mathcal{S} \models \varphi$ if all set assignments that satisfy $\mathcal{S}$ also satisfy φ. The *satisfiability problem* is to determine whether a given finite system $\mathcal{S}$ of set constraints over Σ is satisfiable.

A *truth assignment* is a map $u : X \to \mathbf{2}$ where $\mathbf{2} = \{0, 1\}$ is the two-element Boolean algebra. Any truth assignment u extends uniquely to a B-homomorphism $u : T_{\mathrm{B}}(X) \to \mathbf{2}$ inductively according to the rules of Boolean algebra. If $X = \{x_1, \ldots, x_m\}$, we use the notation

$$B[x_i := a_i]$$

to denote the truth value of the Boolean formula B under the truth assignment $x_i \mapsto a_i$, $1 \leq i \leq m$.

3 Expressibility

Systems of mixed positive and negative constraints are strictly more expressive than systems of positive constraints alone. We will prove this as a corollary of a general compactness theorem for positive constraints.

Theorem 1 (Compactness) *A system $\mathcal{S}$ of positive set constraints is satisfiable if and only if all finite subsets of $\mathcal{S}$ are satisfiable.*

Proof. The implication ($\Rightarrow$) is straightforward. For the other direction, suppose $\mathcal{S}$ is finitely satisfiable. We wish to construct a satisfying set assignment for $\mathcal{S}$. By Zorn's Lemma, there exists a maximal finitely satisfiable set $\widehat{\mathcal{S}}$ of positive constraints containing $\mathcal{S}$. One can show that for all ground terms t and set expressions E, exactly one of the constraints $t \subseteq E$, $t \subseteq {\sim}E$ is in $\widehat{\mathcal{S}}$; if neither is in $\widehat{\mathcal{S}}$, then $\widehat{\mathcal{S}}$ is not maximal, and if both are, then $\widehat{\mathcal{S}}$ is not finitely satisfiable. Now define a map

$$\sigma(E) \quad = \quad \{t \mid t \subseteq E \in \widehat{\mathcal{S}}\} \ .$$

One can show by induction on the structure of set expressions that σ is a valid set assignment and satisfies $\widehat{\mathcal{S}}$. For example, to show that

$$\sigma(fE_1 \ldots E_n) \quad = \quad \{ft_1 \ldots t_n \mid t_i \in \sigma(E_i),\ 1 \leq i \leq n\} \ ,$$

note

$$t \in \sigma(fE_1 \ldots E_n) \iff t \subseteq fE_1 \ldots E_n \in \widehat{\mathcal{S}} \ . \tag{3.1}$$

Then t must be of the form $ft_1 \ldots t_n$, otherwise $\widehat{\mathcal{S}}$ would not be finitely satisfiable. Now we use the fact that

$$\begin{array}{rcl} ft_1 \ldots t_n \subseteq fE_1 \ldots E_n & \models & t_i \subseteq E_i \ , \quad 1 \leq i \leq n \\ \{t_i \subseteq E_i \mid 1 \leq i \leq n\} & \models & ft_1 \ldots t_n \subseteq fE_1 \ldots E_n \end{array}$$

to argue that $t \subseteq fE_1 \ldots E_n \in \widehat{\mathcal{S}}$ iff $t_i \subseteq E_i \in \widehat{\mathcal{S}}$, $1 \leq i \leq n$, otherwise $\widehat{\mathcal{S}}$ would not be finitely satisfiable. Combining this with (3.1) and using the induction hypothesis, we get

$$t \in \sigma(fE_1 \ldots E_n) \iff t_i \in \sigma(E_i) \ , \quad 1 \leq i \leq n \ .$$

To show that σ satisfies all constraints of $\widehat{\mathcal{S}}$, let $E \subseteq F$ be any constraint in $\widehat{\mathcal{S}}$. For any term t,

$$\begin{array}{rcl} t \in \sigma(E) & \Rightarrow & t \subseteq E \in \widehat{\mathcal{S}} \\ & \Rightarrow & t \subseteq F \in \widehat{\mathcal{S}} \\ & \Rightarrow & t \in \sigma(F) \ ; \end{array} \tag{3.2}$$

the reason for the implication (3.2) is that

$$\{t \subseteq E,\ E \subseteq F\} \ \models\ t \subseteq F\ ,$$

and if $t \subseteq F$ were not in $\widehat{\mathcal{S}}$, then $t \subseteq {\sim}F$ would be, and $\widehat{\mathcal{S}}$ would not be finitely satisfiable. □

Corollary 2 *Finite systems of mixed positive and negative constraints are strictly more expressive than systems of positive constraints only.*

Proof. Consider the single negative constraint $x \neq 0$ over any ranked alphabet Σ with at least one constant and at least one symbol of higher arity. Solutions are $\sigma : \{x\} \to T_\Sigma$ with $\sigma(x)$ nonempty. Let $\mathcal{S}$ be any set, finite or infinite, of positive constraints over any set of variables X containing x. We claim that it is *not* the case that the set

$$\{\sigma(x) \mid \sigma : X \to T_\Sigma,\ \sigma \models \mathcal{S}\}$$

is exactly the set of nonempty subsets of T_Σ.

Consider the infinite set of positive constraints

$$\mathcal{S}\ \cup\ \{t \subseteq {\sim}x \mid t \in T_\Sigma\}\ .$$

Either this is satisfiable or not. If so, then there is a satisfying set assignment σ. But $t \in \sigma({\sim}x)$ for all terms t, so $\sigma(x) = \emptyset$ and $\sigma \models \mathcal{S}$, and the claim is verified. If not, then by compactness there is a finite subset $F \subseteq T_\Sigma$ such that

$$\mathcal{S}\ \cup\ \{t \subseteq {\sim}x \mid t \in F\}$$

is not satisfiable. Therefore there is no solution σ of $\mathcal{S}$ with $\sigma(x) = \{t\}$, where t is any term not in F. □

4 Set Constraints and Hypergraph Closure

In [1] it is shown how to transform a given system of positive set constraints into an equivalent system in a special normal form. The transformation is linear for fixed Σ. Applying this transformation to a system containing negative constraints, we obtain the following normal form. Let X be a set of variables, and for each $f \in \Sigma$, let

$$Z_f \quad = \quad \{z^f_{ix} \mid 0 \leq i \leq \mathbf{arity}(f),\ x \in X\}$$

be a set of variables such that the sets X and Z_f, $f \in \Sigma$ are pairwise disjoint. A system of set constraints in normal form (with respect to X and the Z_f) consists of

- a positive constraint $B = 1$, $B \in T_{\mathrm{B}}(X)$
- for each $f \in \Sigma$, a positive constraint $C_f = 1$, $C_f \in T_{\mathrm{B}}(Z_f)$
- positive constraints

  $$\begin{aligned} z^f_{0x} &= f\underbrace{1\ldots1}_{n} \cap x \\ z^f_{ix} &= f\underbrace{1\ldots1}_{i-1}x\underbrace{1\ldots1}_{n-i} \end{aligned}$$

  for each $f \in \Sigma_n$, $1 \leq i \leq n$, and $x \in X$
- a finite set of negative constraints $D \neq 0$, one for each element D of a given finite set $\mathcal{D} \subseteq T_{\mathrm{B}}(X)$.

The last component is absent with positive constraints only.

We outline here the translation of [1] along with the minor modifications necessary to handle negative constraints.

1. For every occurrence of a subexpression $fE_1 \ldots E_n$ in $\mathcal{S}$, let $y_0, y_1, \ldots, y_n$ be new variables. Replace $fE_1 \ldots E_n$ by y_0 and add new constraints $y_0 = fy_1 \ldots y_n$ and $y_i = E_i$, $1 \leq i \leq n$. Continue until all constraints are either purely Boolean or of the form $y_0 = fy_1 \ldots y_n$. Let X be the set of all variables occurring in $\mathcal{S}$ at this point.

2. For each $f \in \Sigma_n$, introduce a new set of variables

   $$Z_f \quad = \quad \{z^f_{ix} \mid 0 \leq i \leq n,\ x \in X\}$$

   and add the constraints

   $$z^f_{0x} \quad = \quad f\underbrace{1\ldots1}_{n}\cap x \qquad z^f_{ix} = f\underbrace{1\ldots1}_{i-1}x\underbrace{1\ldots1}_{n-i}$$

   for all $1 \leq i \leq n$ and $x \in X$.

3. Assume without loss of generality that there is a variable $y \in X$ and constraint $y = 1$ in $\mathcal{S}$. Each constraint $x = fx_1 \ldots x_n$ obtained in step 1 is equivalent to the constraint

$$x \quad = \quad fx_1 \underbrace{1 \ldots 1}_{n-1} \cap f 1 x_2 \underbrace{1 \ldots 1}_{n-2} \cap \cdots \cap f \underbrace{1 \ldots 1}_{n-1} x_n \cap f \underbrace{1 \ldots 1}_{n} .$$

(The last term on the right hand side is redundant except in the case $n = 0$. This was erroneously omitted in the account of [1]). This in turn is equivalent to the conjunction of constraints

$$\begin{aligned} f \underbrace{1 \ldots 1}_{n} \cap x \quad &= \quad fx_1 \underbrace{1 \ldots 1}_{n-1} \cap f 1 x_2 \underbrace{1 \ldots 1}_{n-2} \cap \cdots \cap f \underbrace{1 \ldots 1}_{n-1} x_n \cap f \underbrace{1 \ldots 1}_{n} \\ g \underbrace{1 \ldots 1}_{m} \cap x \quad &= \quad 0 \ , \ g \neq f \ , \ m = \mathbf{arity}(g) \ . \end{aligned}$$

Replace the constraint $x = fx_1 \ldots x_n$ with the constraints

$$z^f_{0x} \quad = \quad \bigcap_{i=1}^{n} z^f_{ix_i} \cap z^f_{0y} \qquad z^g_{0x} = 0 \ , \ g \neq f \ .$$

Because of the constraints introduced in step 2, the resulting system is equivalent.

4. At this point we have

   - positive and negative Boolean constraints formed in step 1 involving only the variables X
   - for each $f \in \Sigma$, positive Boolean constraints formed in step 3 involving only the variables Z_f
   - mixed constraints formed in step 2.

   Replace each positive Boolean constraint $E \subseteq F$ involving variables in X by the equivalent constraint $\sim\!E \cup F = 1$. Let B be the conjunction of all the left hand sides of such constraints, and replace all these constraints in $\mathcal{S}$ with the single constraint $B = 1$. Do the same for the purely Boolean constraints involving the variables Z_f to get a single constraint $C_f = 1$ for each $f \in \Sigma$. Finally, replace each negative Boolean constraint $E \not\subseteq F$ by the equivalent constraint $E \cap \sim F \neq 0$, and let $\mathcal{D}$ be the set of all such negative constraints.

As described in [1], a system of set constraints $\mathcal{S}$ in normal form determines a hypergraph

$$H \quad = \quad (U,\ E_f \mid f \in \Sigma)$$

as follows. The vertex set U is the set of all truth assignments $u : X \to \mathbf{2}$ satisfying B. Each such truth assignment corresponds to a conjunction of literals (also denoted u) in which each variable in X occurs exactly once, either positively or negatively, such that $u \subseteq B$ tautologically. The variable x occurs positively iff $u(x) = 1$. We often call the elements of U *atoms* because they represent atoms (minimal nonzero elements) of the free Boolean algebra on generators X modulo $B = 1$, where "minimal" is in the sense of the natural order on the Boolean algebra. It follows from elementary Boolean algebra that each Boolean expression over X is equivalent modulo $B = 1$ to a disjunction of atoms.

For each $f \in \Sigma_n$, the hyperedge relation E_f of H is defined to be the set of all $(n+1)$-tuples $(u_0, \ldots, u_n) \in U^{n+1}$ such that

$$C_f[z^f_{ix} := u_i(x)] \quad = \quad 1\ . \tag{4.3}$$

Intuitively, we think of the formula C_f as a Boolean-valued mapping on $(n+1)$-tuples of truth assignments to X. To emphasize this intuition, we abbreviate the left hand side of (4.3) by

$$C_f[u_0, \ldots, u_n]\ .$$

Thus

$$(u_0, \ldots, u_n) \in E_f \quad \text{iff} \quad C_f[u_0, \ldots, u_n] = 1\ .$$

In general, the size of H can be exponential in the size of $\mathcal{S}$.

An $(n+1)$-ary hyperedge relation E_f of the hypergraph H is said to be *closed* if for each n-tuple $u_1, \ldots, u_n \in U^n$, there exists $u_0 \in U$ such that $(u_0, u_1, \ldots, u_n) \in E_f$. In the case $n = 0$, this definition just says $E_f \cap U \neq \emptyset$. Abusing notation, we can think of E_f as a function

$$E_f : U^n \quad \to \quad 2^U$$

where

$$E_f(u_1, \ldots, u_n) \quad = \quad \{u_0 \mid (u_0, u_1, \ldots, u_n) \in E_f\}\ .$$

In this view, E_f is *closed* iff $E_f(u_1, \ldots, u_n) \neq \emptyset$ for each n-tuple $u_1, \ldots, u_n \in U^n$. The hypergraph H is said to be *closed* if all its hyperedge relations are closed.

The *induced subhypergraph* of H on vertices $U' \subseteq U$ is the hypergraph

$$H' \;=\; (U',\; E'_f \mid f \in \Sigma)$$

such that $E'_f = E_f \cap (U')^{n+1}$ for $f \in \Sigma_n$.

The *hypergraph closure problem* is the problem of determining whether a given hypergraph H has a closed induced subhypergraph.

The following theorem was proved in [1].

Theorem 3 *The hypergraph H corresponding to a system $\mathcal{S}$ of positive set constraints has a closed induced subhypergraph if and only if $\mathcal{S}$ is satisfiable.*

In brief, the proof of [1] establishes a one-to-one correspondence between set assignments σ satisfying $\mathcal{S}$ and maps $\theta : T_\Sigma \rightarrow U$ such that for all $f \in \Sigma$ and for all terms $ft_1 \ldots t_n$,

$$\theta(ft_1 \ldots t_n) \;\in\; E_f(\theta(t_1), \ldots, \theta(t_n)) \; . \tag{4.4}$$

The set assignment corresponding to θ is

$$\begin{aligned} \sigma(x) &= \{t \mid \theta(t)(x) = 1\} \\ \sigma(z^f_{ix}) &= \sigma(f \underbrace{1 \ldots 1}_{i-1} x \underbrace{1 \ldots 1}_{n-i}) \\ \sigma(z^f_{0x}) &= \sigma(f \underbrace{1 \ldots 1}_{n} \cap x) \; . \end{aligned}$$

Thus deciding the satisfiability of $\mathcal{S}$ is tantamount to determining the existence of a map θ satisfying (4.4). In turn, this is equivalent to the hypergraph closure problem: if such a θ exists, then the induced subhypergraph of H on the image of θ is closed, and conversely, if there exists a closed induced subhypergraph on vertices $U' \subseteq U$, then one can inductively define $\theta(ft_1 \ldots t_n)$ to be the lexicographically first element of $U' \cap E_f(\theta(t_1), \ldots, \theta(t_n))$.

In the presence of negative constraints $D \neq 0$, $D \in \mathcal{D}$, the map θ must not only satisfy (4.4), but must also take on some value u such that $u(D) = 1$ for each $D \in \mathcal{D}$. Thus in the presence of negative constraints, the satisfiability problem becomes:

Problem 4 *Given a finite set $\mathcal{D}$ of Boolean formulas $D \in T_{\mathrm{B}}(X)$ and a hypergraph $H = (U,\ E_f \mid f \in \Sigma)$ specified by $B \in T_{\mathrm{B}}(X)$ and $C_f \in T_{\mathrm{B}}(Z_f)$, $f \in \Sigma$, determine whether there exists a map $\theta : T_\Sigma \to U$ satisfying (4.4) such that*

$$\textit{for each } D \in \mathcal{D} \textit{ there exists an atom } u \textit{ in } \theta(T_\Sigma) \textit{ satisfying } D, \tag{4.5}$$

where $\theta(T_\Sigma)$ denotes the image of T_Σ under the map θ.

5 A Reachability Problem

Our decision procedure first reduces the satisfiability problem for mixed systems of positive and negative set constraints to a certain reachability problem involving Diophantine inequalities. In this section we define the reachability problem and give the reduction.

First we describe the reachability problem on an intuitive level. Let X be a set of variables ranging over $\mathbb{N}$, the natural numbers. Suppose we are given a finite system C of formal inequalities $p \leq q$, where p and q are polynomials in the variables X with coefficients in $\mathbb{N}$, such that

- each left hand side p is a sum of variables in X
- each variable occurs in at most one left hand side.

An *assignment* is a map $u : X \to \mathbb{N}$. Each assignment u extends uniquely to an evaluation morphism $u : \mathbb{N}[X] \to \mathbb{N}$ which evaluates polynomials at u. A variable x is said to be *enabled* under an assignment u if either

- the variable x does not occur on the left hand side of any constraint in C; or
- the unique constraint in C in which x appears on the left hand side is a strict inequality under the assignment u.

Consider the following nondeterministic procedure. Starting with the zero assignment, repeatedly choose a variable that is enabled and "fire" it by incrementing it by 1. The reachability problem is to decide whether there exists a sequence of legal firings that allows a particular distinguished variable to be fired.

We give a more rigorous presentation of this problem below, then reduce the satisfiability problem to this problem. In Section 6 we show that the reachability problem is decidable.

5.1 Polynomials and Assignments

We use the term *ring* to mean commutative ring with unit and *semiring* to mean commutative semiring with unit.

Let $\mathbb{Z}$ denote the ring of integers and $\mathbb{N} \subseteq \mathbb{Z}$ the semiring of natural numbers with the usual addition and multiplication operations. For X a finite set of variables, let $\mathbb{Z}[X]$ denote the ring of polynomials in the variables X with integer coefficients and $\mathbb{N}[X] \subseteq \mathbb{Z}[X]$ the semiring of polynomials with positive coefficients. The ring $\mathbb{Z}[X]$ is the free ring on generators X and the semiring $\mathbb{N}[X]$ is the free semiring on generators X.

Any map $u : X \to R$ to a ring R extends uniquely to a ring homomorphism $u : \mathbb{Z}[X] \to R$. If S is a semiring and $S \subseteq R$, and if $u(x) \in S$ for $x \in X$, then the restriction of $u : \mathbb{Z}[X] \to R$ to domain $\mathbb{N}[X]$ is a semiring homomorphism $\mathbb{N}[X] \to S$, and is the unique semiring homomorphism extending $u : X \to S$. We will concentrate on the case $S = \mathbb{N}$ and $R = \mathbb{Z}$; we call such a map an *assignment*. However, functional composition of polynomials is effected by the same construction with $S = \mathbb{N}[X]$ and $R = \mathbb{Z}[X]$.

Intuitively, an assignment $u : X \to \mathbb{N}$ should be regarded as an assignment of values to the variables, and $u(q)$ the result of evaluating the polynomial q on those values.

The set of assignments, considered as functions of X, forms a commutative monoid $\mathcal{V}$ under pointwise addition $u + v : x \mapsto u(x) + v(x)$, $x \in X$, with identity element the zero assignment $\mathbf{0} : x \mapsto 0$, $x \in X$. The monoid $\mathcal{V}$ is isomorphic to the commutative monoid $\mathbb{N}^{|X|}$ with ordinary addition under the map $v \mapsto (v(x) \mid x \in X)$.

Care must be taken here: it is *not* the case that $(u + v)(q) = u(q) + v(q)$ for $q \in \mathbb{Z}[X]$ in general. The value of $(u + v)(q)$ is governed by the definition of the unique extension of assignments to homomorphisms. For example,

$$(u + v)(x + 1) \quad = \quad (u + v)(x) + (u + v)(1) \quad = \quad u(x) + v(x) + 1 \ ,$$

whereas

$$u(x + 1) + v(x + 1) \quad = \quad u(x) + v(x) + 2 \ .$$

However, we do have the following useful inequality:

Lemma 5 *For any $q \in \mathbb{N}[X]$,*

$$(u+v)(q) \quad \geq \quad u(q) + v(q) - \mathbf{0}(q)$$

with equality holding if q is affine (i.e., linear plus a constant term).

Proof. This can be proved by induction on the form of q. Note that $\mathbf{0}(q)$ is the constant term of q. For $x \in X$, we have $(u+v)(x) = u(x) + v(x)$, and for constants $a \in \mathbb{N}$, $(u+v)(a) = a = u(a) + v(a) - \mathbf{0}(a)$. For polynomials of the form pq where neither p nor q has a constant term,

$$\begin{aligned} (u+v)(pq) &= (u+v)(p) \cdot (u+v)(q) \\ &\geq (u(p)+v(p)) \cdot (u(q)+v(q)) \\ &\geq u(p) \cdot u(q) + v(p) \cdot v(q) \\ &= u(pq) + v(pq) \ . \end{aligned}$$

Finally, for polynomials of the form $p+q$,

$$\begin{aligned} (u+v)(p+q) &= (u+v)(p) + (u+v)(q) \\ &\geq u(p) + v(p) - \mathbf{0}(p) + u(q) + v(q) - \mathbf{0}(q) \\ &= u(p+q) + v(p+q) - \mathbf{0}(p+q) \end{aligned}$$

with equality holding if p and q are affine, by the induction hypothesis. □

In particular, $(u+v)(q) = u(q) + v(q)$ if q is linear with constant coefficient 0.

For $v : X \to \mathbb{N}$ an assignment, let $\mathbf{inc}_v : \mathbb{Z}[X] \to \mathbb{Z}[X]$ be the unique ring homomorphism such that

$$\mathbf{inc}_v(x) \quad = \quad x + v(x) \ , \quad x \in X \ .$$

Informally, $\mathbf{inc}_v(p)$ is the polynomial obtained from p by substituting $x + v(x)$ for x. Intuitively, $\mathbf{inc}_v$ says, "Automatically increase the value of $x \in X$ by $v(x)$." Restricted to domain $\mathbb{N}[X]$, $\mathbf{inc}_v$ is a semiring homomorphism $\mathbb{N}[X] \to \mathbb{N}[X]$, for which we use the same name.

The homomorphism $\mathbf{inc}_v$ is the unique map such that the diagram

$$\begin{tikzcd}
\mathbb{Z}[X] \arrow[d, "\mathbf{inc}_v"'] \arrow[dr, "u+v"] & \\
& \mathbb{Z} \\
\mathbb{Z}[X] \arrow[ur, "u"'] &
\end{tikzcd} \tag{5.6}$$

commutes, *i.e.* such that $u \circ \mathbf{inc}_v = u + v$: for $x \in X$,

$$\begin{aligned} (u+v)(x) &= u(x) + v(x) \\ &= u(x + v(x)) \\ &= u(\mathbf{inc}_v(x)) \ . \end{aligned} \tag{5.7}$$

Equation (5.7) holds since $v(x)$ is a constant. Since the homomorphisms $u+v$ and $u \circ \mathbf{inc}_v$ agree on X, they agree everywhere. The homomorphism $\mathbf{inc}_v$ is unique, since it is determined by its values on $x \in X$, and the polynomial $\mathbf{inc}_v(x)$ is determined by its set of values $u(\mathbf{inc}_v(x)) = u(x) + v(x)$.

By composing two copies of (5.6), one observes that the set

$$\mathcal{I} \quad = \quad \{\mathbf{inc}_v \mid v \text{ an assignment}\}$$

forms a monoid under functional composition $\circ$ with identity $\mathbf{inc}_{\mathbf{0}}$. Moreover, $\mathcal{I}$ is isomorphic to the monoid of assignments $\mathcal{V}$ under the map $v \mapsto \mathbf{inc}_v$; *i.e.*,

$$\mathbf{inc}_{u+v} \quad = \quad \mathbf{inc}_u \circ \mathbf{inc}_v \ .$$

The map $v \mapsto \mathbf{inc}_v$ is bijective, since v can be recovered uniquely from $\mathbf{inc}_v$ by taking $u = \mathbf{0}$ in (5.7).

It follows immediately that $\mathbf{inc}_u$ and $\mathbf{inc}_v$ commute under composition, *i.e.* $\mathbf{inc}_u \circ \mathbf{inc}_v = \mathbf{inc}_v \circ \mathbf{inc}_u$.

One application of particular importance will be incrementing the value of a variable x under an assignment u by 1. The new assignment is $u + \delta_x$, where $\delta_x(x) = 1$ and $\delta_x(y) = 0$ for $y \neq x$. The effect of applying $\mathbf{inc}_{\delta_x}$ to a polynomial q is the same as substituting $x + 1$ for x in q.

Let X^* denote the monoid of finite-length strings over X. This is the free monoid on generators X. Elements of X^* will be denoted $\sigma, \tau, \rho, \ldots$

There is a unique monoid homomorphism $[\![\ ]\!] : X^* \rightarrow \mathcal{V}$ extending the map $x \rightarrow \delta_x$, $x \in X$. The image of $\sigma = x_1 \cdots x_n$ under this map is $[\![\sigma]\!] =$

$\sum_{i=1}^{n} \delta_{x_i}$. Applied to x, the function $[\![\sigma]\!]$ gives the number of occurrences of x in the string σ. This is known in formal language theory as the *Parikh map*. By a slight abuse of notation, we omit the braces $[\![\ ]\!]$ when using $[\![\sigma]\!]$ as a function; thus $\sigma(x)$ denotes the number of occurrences of x in σ, and $\sigma(q)$ is the value of the polynomial q under the assignment $[\![\sigma]\!]$.

5.2 Systems of Diophantine Inequalities

We consider finite systems C of Diophantine inequalities of the form $p \leq q$ where $p, q \in \mathbb{N}[X]$ such that

- each left hand side p is a sum of distinct variables; and
- each variable in X occurs in at most one left hand side.

There is no restriction on the form of the right hand sides q except that they be in $\mathbb{N}[X]$. The inequalities in C are called *(Diophantine) constraints*. A variable $x \in X$ is said to be *constrained* in C if x occurs on the left hand side of some constraint in C. In this case we denote the unique such constraint by $\mathbf{con}\,(x, C)$. If x does not occur on the left hand side of any constraint in C, then x is said to be *unconstrained* in C, and we write $\mathbf{con}\,(x, C) = *$.

We say that the assignment u *satisfies* the constraint $p \leq q$ if $u(p) \leq u(q)$. We say that u *satisfies* C if u satisfies all the constraints in C. We say that $\sigma \in X^*$ *satisfies* a constraint or set of constraints if $[\![\sigma]\!]$ does.

5.3 The Nonlinear Reachability Problem

Let C be a system of Diophantine constraints as described above and let $x_0 \in X$ be a fixed distinguished variable.

Definition 6 Let $\sigma \in X^*$. The constraint $p \leq q \in C$ is said to be σ-*enabled* if $\sigma(p) < \sigma(q)$; *i.e.*, the inequality is strict under the assignment $[\![\sigma]\!]$. The variable $x \in X$ is said to be (σ, C)-*enabled* if either

- x is unconstrained in C, or
- x is constrained in C and $\mathbf{con}\,(x, C)$ is σ-enabled.

□

A *tree*, for our purposes, is a nonempty prefix-closed subset T of X^*. The *root* of T is ϵ. The *parent* of $\sigma \neq \epsilon$ is the longest proper prefix of σ. A *leaf* of T is an element of T that is not a parent. A *path* of T is a maximal subset of T linearly ordered by the prefix relation.

The system C gives rise to a tree

$$T_C \;=\; \{\sigma \in X^* \mid \text{for all prefixes } \tau x \text{ of } \sigma,\ x \text{ is } (\tau, C)\text{-enabled.}\}$$

The tree T_C describes the possible legal sequences of firings that can take place according to the informal description of the nonlinear reachability problem given in Section 5.

Definition 7 The *Nonlinear Reachability Problem (NRP)* is to determine, given C, whether T_C contains an element σ such that $\sigma(x_0) > 0$. Such a σ is called a *solution* of the given instance C of the NRP. □

In other words, determine whether there exists a legal firing sequence such that the distinguished variable x_0 is fired.

Note that ϵ satisfies C since $[\![\epsilon]\!] = \mathbf{0}$, and if σ satisfies C and x is (σ, C)-enabled, then σx satisfies C, since $[\![\sigma x]\!] = [\![\sigma]\!] + \delta_x$. It follows by induction that σ satisfies C for every $\sigma \in T_C$. In other words, if σ satisfies C and x is (σ, C)-enabled, then we can fire x and the resulting assignment still satisfies C. The converse is false in general; *i.e.*, it is possible that both σ and σx satisfy C but x is not (σ, C)-enabled: consider the constraint $x \leq x$.

5.4 Reduction of Set Constraint Satisfiability to Nonlinear Reachability

Theorem 8 *The satisfiability problem for systems of mixed positive and negative set constraints reduces effectively to a finite disjunction of instances of the Nonlinear Reachability Problem.*

Proof. As argued in Section 4, the satisfiability problem for systems of mixed positive and negative constraints is equivalent to Problem 4. Using the notation of Problem 4, let $\mathcal{U}$ be the set of all subsets $V \subseteq U$ such that for all $D \in \mathcal{D}$ there exists a $v \in V$ with $v(D) = 1$. Consider a modified version of Problem 4 in which condition (4.5) is replaced by the condition

$$V \subseteq \theta(T_\Sigma)\ . \tag{5.8}$$

Then Problem 4 is equivalent to the disjunction over all $V \in \mathcal{U}$ of instances of the modified version. Furthermore, we will only need to construct a finite partial approximation θ' to θ satisfying (4.4) and (5.8), provided

- the domain of θ' is closed downward under the subterm relation
- there is a closed induced subhypergraph of H containing the image of θ'.

The second property will allow θ' to be completed to a total function θ, as described below.

Thus the problem now becomes:

Problem 9 *Given a hypergraph $H = (U,\ E_f \mid f \in \Sigma)$ specified by B and C_f, $f \in \Sigma$, and a subset $V \subseteq U$, determine whether there exist $U' \subseteq U$ and a partial map $\theta : T_\Sigma \to U'$ with finite domain such that*

- *the induced subhypergraph on U' is closed*
- *the domain of θ is closed downward under the subterm relation*
- *θ satisfies (4.4) on all terms in its domain*
- *$V \subseteq \theta(T_\Sigma) \subseteq U'$.*

Consider the following nondeterministic procedure for constructing θ. We first guess the subset U' containing the target set V and check that it is closed. We start with θ totally undefined. At any point, say we have a partial θ with finite domain closed downward under the subterm relation. We nondeterministically pick some term $ft_1 \ldots t_n$ such that the $\theta(t_i)$ are defined but $\theta(ft_1 \ldots t_n)$ is not yet defined, nondeterministically choose some u in $E_f(\theta(t_1), \ldots, \theta(t_n)) \cap U'$, and assign $\theta(ft_1 \ldots t_n) := u$. We are always able to continue, since U' is closed. We halt successfully when and if all elements of V have been chosen as $\theta(t)$ for some t.

During this process, we use an integer variable $x_{u,f,u_1,\ldots,u_n}$, $n = \mathbf{arity}(f)$, to count the number of terms of the form $ft_1 \ldots t_n$ such that

- $\theta(t_i)$ exists and equals u_i, $1 \leq i \leq n$, and
- $\theta(ft_1 \ldots t_n)$ exists and equals u.

There is one such variable for each choice of f in Σ, $u_1, \ldots, u_n \in U'$ where $n = \mathbf{arity}(f)$, and $u \in U' \cap E_f(u_1, \ldots, u_n)$.

Now for each $f \in \Sigma_n$ and $u_1, \ldots, u_n \in U'$, consider the formal inequality

$$\sum_{u \in U' \cap E_f(u_1,\ldots,u_n)} x_{u,f,u_1,\ldots,u_n} \quad \leq \quad \prod_{i=1}^{n} \sum_{m=0}^{M} \sum_{\substack{v_1,\ldots,v_m \in U' \\ g \in \Sigma_m}} x_{u_i,g,v_1,\ldots,v_m} \quad (5.9)$$

where M is the maximum arity of symbols in Σ. This inequality has the following significance. Given a partial map θ, let

$$\begin{aligned} B_u &= \{t \mid \theta(t) \text{ exists and equals } u\} \\ A_{f,u_1,\ldots,u_n} &= \{ft_1 \ldots t_n \mid t_i \in B_{u_i},\ 1 \leq i \leq n\}\ . \end{aligned}$$

The value of the right hand side of (5.9) is the size of $A_{f,u_1,\ldots,u_n}$, which is the the size of the direct product $B_{u_1} \times \cdots \times B_{u_n}$. The value of the left hand side of (5.9) is the size of the subset of $A_{f,u_1,\ldots,u_n}$ consisting of all elements t for which $\theta(t)$ is defined. The inequality expresses the fact that θ is defined on the subterms of t before being defined on t.

Consider the collection C of all such inequalities (5.9). To say that a variable $x_{u,f,u_1,\ldots,u_n}$ is enabled says that there exists a term t with head symbol f such that θ is defined on the n immediate subterms and takes values $u_1, \ldots, u_n$ on those subterms respectively, but $\theta(t)$ is not yet defined. To fire $x_{u,f,u_1,\ldots,u_n}$ says that we choose one such t and define $\theta(t) := u$.

The process of defining θ from the bottom up as described above corresponds to a sequence of legal firings. Conversely, any legal sequence of firings gives a corresponding sequence of definitions of θ starting with the totally undefined map.

We have thus reduced the satisfiability problem for systems of mixed positive and negative set constraints to a disjunction of instances of the problem of determining, given C and V, whether there is a finite sequence of legal firings after which for all $v \in V$ there are f and $u_1, \ldots, u_n$ such that the value of $x_{v,f,u_1,\ldots,u_n}$ is nonzero.

We reduce this problem to a finite disjunction of instances of the NRP as follows. For each $v \in V$, choose f and $u_1, \ldots, u_n$ and let $y_v = x_{v,f,u_1,\ldots,u_n}$. Add the constraint

$$x_0 \quad \leq \quad \prod_{v \in V} y_v$$

where x_0 is a new variable, and make x_0 the distinguished variable of the NRP so obtained. The variable x_0 can be fired only after all the y_v have been fired. The problem above is equivalent to the disjunction of all such instances of the NRP over all possible choices of the y_v. □

6 Decidability of the Nonlinear Reachability Problem

In this section we prove the decidability of the NRP. We will start by defining several technical concepts on which our proof is based and deriving their basic properties. The most important of these concepts are the notions of *exposed* and *inhibited* variables and *admissible* strings. Intuitively, a variable is *exposed* in a polynomial q if incrementing it causes the value of q to increase. The intuition behind the definition *inhibited variable* is that it does no good to increment such a variable under the current state of affairs. A string σ is *admissible* if it never increments any inhibited variable. We show that if there exists a solution, then there exists an admissible one. The final argument shows that if we construct the tree T_C breadth-first, ignoring nonadmissible strings, then along any path we will eventually encounter either a solution, a leaf with no admissible extensions, or a configuration that allows us to reduce the size of the system.

6.1 Reset

We first describe a useful technical device called a *reset*. Intuitively, after executing a firing sequence σ that is legal with respect to a set of constraints C, we can construct a new instance of the problem $\mathbf{inc}_\sigma(C)$ (defined below) which allows us to proceed as if we were starting afresh.

Definition 10 Let C be a system of Diophantine constraints as defined in Section 5.2. If $\sigma \in T_C$, we define T_C^σ to be the subtree of T_C rooted at σ:

$$T_C^\sigma \quad = \quad \{\tau \in X^* \mid \sigma\tau \in T_C\} \ .$$

This set is nonempty and prefix-closed, therefore a tree. □

Note that $[\![\sigma]\!]$ alone determines whether a variable is σ-enabled. It follows inductively that if $\sigma, \tau \in T_C$ and $[\![\sigma]\!] = [\![\tau]\!]$, then $T_C^\sigma = T_C^\tau$.

Let v be any assignment satisfying C, and let $\mathbf{inc}_v$ be as in Section 5.1. Let $\mathbf{inc}_v(C)$ denote the system of constraints

$$\mathbf{inc}_v(C) \quad = \quad \{p \leq \mathbf{inc}_v(q) - v(p) \mid p \leq q \in C\} \ .$$

The right hand sides $\mathbf{inc}_v(q) - v(p)$ are in $\mathbb{N}[X]$, since the constant coefficient of $\mathbf{inc}_v(q)$ is at least $v(p)$. This is a consequence of the fact that v satisfies C:

$$v(p) \quad \leq \quad v(q) \quad = \quad \mathbf{0}(\mathbf{inc}_v(q)) \ .$$

Moreover, x is constrained in C iff it is constrained in $\mathbf{inc}_v(C)$, since all the left hand sides are the same.

Note also that the constraint $p \leq \mathbf{inc}_v(q) - v(p) \in \mathbf{inc}_v(C)$ is equivalent to $\mathbf{inc}_v(p) \leq \mathbf{inc}_v(q)$, since $\mathbf{inc}_v(p) = p + v(p)$ for p a sum of variables.

Lemma 11 *Let C be a set of constraints and $\sigma \in T_C$. Then*

$$T_C^\sigma \quad = \quad T_{\mathbf{inc}_\sigma(C)} \ .$$

Proof. Certainly ϵ is a member of both trees. Moreover, for any constraint $p \leq q \in C$, we have from (5.6) that

$$\tau(\mathbf{inc}_\sigma(q-p)) \quad = \quad \sigma\tau(q-p) \ ,$$

and $\mathbf{con}\,(x, C) = \mathbf{con}\,(x, \mathbf{inc}_\sigma(C))$, thus x is $(\tau, \mathbf{inc}_\sigma(C))$-enabled iff x is $(\sigma\tau, C)$-enabled. Thus the trees are identical. □

6.2 Order

Our algorithm will construct part of the tree T_C. During this construction, we will want to keep track of the values of $q - p$ for $p \leq q \in C$, since this information will help us determine when we have reached a situation in which progress has been made. We define the order $\leq_C$ for this purpose. We also define the order $\leq_X$, which is just the natural order on the set of assignments.

Definition 12 For C a system of constraints and $\sigma, \tau \in X^*$, define

- $\sigma \leq_X \tau$ if $\sigma(x) \leq \tau(x)$ for all $x \in X$
- $\sigma \leq_C \tau$ if $\sigma(q-p) \leq \tau(q-p)$ for all $p \leq q \in C$
- $\sigma \leq_{X,C} \tau$ if both $\sigma \leq_X \tau$ and $\sigma \leq_C \tau$
- $\sigma \equiv_C \tau$ if both $\sigma \leq_C \tau$ and $\tau \leq_C \sigma$.

□

It follows from Lemma 5 and the observation that $\mathbf{0}(q)$ is the constant coefficient of q that for $q \in \mathbb{N}[X]$, if $\sigma \leq_X \tau$ then $\sigma(q) \leq \tau(q)$.

Note that the relations $\leq_X$ and $\leq_C$ depend only on the assignments $[\![\sigma]\!]$ and not on the strings σ themselves. Note also that if $\sigma\tau \in T_C$ then $\sigma \leq_X \sigma\tau$. The same statement is not true in general for $\leq_C$; for example, take $\sigma = \epsilon$, $\tau = x$, and $C = \{x \leq y+1\}$.

Lemma 13 *Let $x \in X$, $\sigma, \tau \in X^*$ such that $\sigma \leq_X \tau$, and $p \leq q \in C$. Then*

$$\sigma x(q-p) - \sigma(q-p) \quad \leq \quad \tau x(q-p) - \tau(q-p) \ .$$

Proof. Using Lemma 5 and the facts that $[\![\sigma x]\!] = [\![\sigma]\!] + \delta_x$ and p is linear, it follows that the inequality in the statement of the Lemma is equivalent to

$$\sigma x(q) - \sigma(q) \quad \leq \quad \tau x(q) - \tau(q) \ .$$

By (5.6), this is equivalent to

$$\sigma(\mathbf{inc}_x(q) - q) \quad \leq \quad \tau(\mathbf{inc}_x(q) - q) \ .$$

But this inequality follows from the assumptions of the Lemma, since $\sigma \leq_X \tau$ and $\mathbf{inc}_x(q) - q \in \mathbb{N}[X]$. □

Lemma 14 *Let $\sigma, \tau \in T_C$ and $x \in X$.*

(i) If x is (σ, C)-enabled and $\sigma \leq_C \tau$, then x is (τ, C)-enabled.

(ii) If $\sigma \leq_X \tau$ then $\sigma x \leq_X \tau x$.

(iii) If $\sigma \leq_{X,C} \tau$, then $\sigma x \leq_{X,C} \tau x$.

Proof. The assertions (i) and (ii) are straightforward consequences of the definitions. The assertion (iii) follows from (ii) and Lemma 13. □

6.3 Well Partial Orders and Dickson's Lemma

A *well partial order* is a partially ordered set in which every infinite sequence has an infinite monotone nondecreasing subsequence. That is, for every infinite sequence $d_0, d_1, \ldots$, there exist indices $i_0 < i_1 < \cdots$ such that $d_{i_0} \leq d_{i_1} \leq \cdots$.

Lemma 15 (Dickson's Lemma) *The set $\mathbb{N}^k$ of k-tuples of natural numbers under the componentwise order is a well partial order.*

For a proof of Dickson's Lemma, see [10].

We will use Dickson's Lemma in the argument below to conclude that along any infinite path in T_C, we must eventually have $\sigma \leq_C \sigma\tau$. Here we are taking $k = |C|$ and comparing the k-tuples $(\sigma(q - p) \mid p \leq q \in C)$.

6.4 Exposed Variables

Intuitively, a variable x is *σ-exposed* in a polynomial q iff, after executing σ, firing x would cause the value of q to increase strictly. The following definition and lemma make this intuition precise.

Definition 16 Let $x \in X$ and $\sigma \in T_C$. We say that x is *σ-exposed* in a monomial qx^i, where x does not appear in q, if $i \geq 1$ and $\sigma(q) \neq 0$. For $q \in \mathbb{N}[X]$, we say that x is *σ-exposed* in q if x is σ-exposed in some monomial of q. We say that x is *(σ, C)-exposed* if x is σ-exposed in q for some $p \leq q \in C$. □

Lemma 17 *Let $x \in X$, $q \in \mathbb{N}[X]$, and $\sigma \in T_C$. Then x is σ-exposed in q iff $\sigma(q) < \sigma x(q)$.*

Proof. Since σ and σx are homomorphisms and all values are nonnegative, it suffices to show the result for monomials ax^i, $a \in \mathbb{N}[X - \{x\}]$. Since $\sigma x(a) = \sigma(a)$,

$$\sigma x(ax^i) - \sigma(ax^i) \quad = \quad \sigma(a)((\sigma(x) + 1)^i - \sigma(x)^i) \quad \geq \quad 0 \ ,$$

with equality holding iff $i = 0$ or $\sigma(a) = 0$. □

The following lemma establishes some basic properties of the notion of exposure and its relation to enabling and the relation $\leq_C$.

Lemma 18 *Let $x \in X$, $p \leq q \in C$, and $\sigma, \tau \in T_C$.*

(i) If x is σ-exposed in q and $\sigma \leq_X \tau$, then x is τ-exposed in q (once exposed, always exposed).

(ii) If x is σ-exposed in q, then $\sigma x(q-p) \geq \sigma(q-p)$; moreover, if x does not occur in p, then the inequality is strict.

(iii) If x is not (σ, C)-exposed, then $\sigma x \leq_C \sigma$.

(iv) The property of exposure in the right hand side of a constraint $p \leq q \in C$ is preserved under a reset. Formally, x is $\sigma\tau$-exposed in q iff x is τ-exposed in $\mathbf{inc}_\sigma(q) - \sigma(p)$.

(v) If $\sigma(x) > 0$, x is not σ-exposed in q, and x is σy-exposed in q, then y is σ-exposed in q.

Proof. Except for (iv) and (v), all statements are direct consequences of Definition 16 and Lemma 17.

To prove (iv), we use (5.6) and Lemma 17:

$$\begin{aligned} \sigma\tau x(q) - \sigma\tau(q) &= \tau x(\mathbf{inc}_\sigma(q)) - \tau(\mathbf{inc}_\sigma(q)) \\ &= \tau x(\mathbf{inc}_\sigma(q) - \sigma(p)) - \tau(\mathbf{inc}_\sigma(q) - \sigma(p)) \ , \end{aligned}$$

since $\sigma(p)$ is a constant.

For (v), there must be a monomial ax^i of q, $i \geq 1$, $a \in \mathbb{N}[X - \{x\}]$, such that $\sigma(a) = 0$ and $\sigma y(a) > 0$. Since $\sigma y(x) \geq \sigma(x) > 0$, we have $\sigma y(ax^i) > 0 = \sigma(ax^i)$, thus $\sigma y(q) > \sigma(q)$. By Lemma 17, y is σ-exposed in q. □

6.5 Inhibited Variables and Admissible Strings

The technical notion of an *inhibited variable* captures the idea that, under the current state of affairs, firing the variable makes no progress toward a solution. Intuitively, firing a variable makes progress only if the variable is exposed, so that firing it might enable another variable, or has value 0, so that firing it might contribute to the exposure of another variable.

We will formalize and prove a result that says intuitively that any string σ can be simulated by another string τ in which no inhibited variable is ever fired. Such a string τ is called *admissible*.

Definition 19 Let C be a system of Diophantine constraints and $\sigma \in T_C$. We say $x \in X$ is (σ, C)-*inhibited* if

- x is unconstrained in C,
- x is not (σ, C)-exposed, and
- $\sigma(x) > 0$.

We say that $\sigma \in X^*$ is C-*admissible* if $\sigma \in T_C$, and for all prefixes τy of σ, y is not (τ, C)-inhibited. □

Lemma 20 *(i) If y is (σ, C)-inhibited, then $\sigma(p) = \sigma y(p)$ and $\sigma(q) = \sigma y(q)$ for all constraints $p \leq q \in C$. In particular, $\sigma y \equiv_C \sigma$.*

(ii) If y, z are (σ, C)-inhibited, then z is $(\sigma y, C)$-inhibited. (This also applies to the case $y = z$.)

Proof.

(i) Since y is unconstrained, it does not appear in p, therefore $\sigma(p) = \sigma y(p)$. Since y is not σ-exposed in q, we have $\sigma(q) = \sigma y(q)$ by Lemma 17.

(ii) Surely $\sigma y(z) \geq \sigma(z) > 0$ and z is still unconstrained in C. Since y and z are not (σ, C)-exposed, they are not exposed in q for any $p \leq q \in C$. Since $\sigma(z) > 0$, it follows from Lemma 18(v) that z is not σy-exposed in q. Thus z is not $(\sigma y, C)$-exposed.

□

The following two lemmas imply that we can restrict our attention to admissible strings when looking for solutions.

Lemma 21 *For every $\sigma \in T_C$, there exists a C-admissible string $\tau \in T_C$ such that $\sigma \leq_C \tau$.*

Proof. Let us call a prefix $\sigma_1 y$ of σ *bad* if y is (σ_1, C)-inhibited. The proof is by lexicographical induction on the length of σ; among strings of the same length, the number of bad prefixes; and among strings of the same length and same number of bad prefixes, the length of the longest bad prefix ("longer" is "smaller" in the induction). If σ is null or has no bad prefix, there is nothing to prove. If the longest bad prefix $\sigma_1 y$ is σ itself, then since y is not (σ_1, C)-exposed, we have by Lemma 18(iii) that $\sigma_1 y \leq_C \sigma_1$, and we are done by the induction hypothesis. Otherwise, there exists a z and σ_2 such that $\sigma = \sigma_1 y z \sigma_2$. Now z is not $(\sigma_1 y, C)$-inhibited, by the maximality of $\sigma_1 y$. Neither is it (σ_1, C)-inhibited, by Lemma 20(ii). Moreover, z is (σ_1, C)-enabled, by Lemma 20(i) and the fact that it is $(\sigma_1 y, C)$-enabled, and y is $(\sigma_1 z, C)$-enabled since it is unconstrained. Therefore $\sigma_1 z y \sigma_2 \in T_C$ is of the same length as σ, but with either strictly fewer bad prefixes (if $\sigma_1 z y$ is not a bad prefix) or the same number of bad prefixes and a strictly longer maximal one (if it is). The result follows from the induction hypothesis. □

Lemma 22 *If a given instance of the NRP with constraints C has a solution, then it has an admissible solution.*

Proof. Let σ be a solution of minimal length. Then σ is of the form τx_0 and $\tau(x_0) = 0$. By Lemma 21, there exists an admissible ρ such that $\tau \leq_C \rho$. If $\rho(x_0) > 0$, then ρ is the desired admissible solution. Otherwise, x_0 is (ρ, C)-enabled (since $\tau \leq_C \rho$ and x_0 is (τ, C)-enabled) and not (ρ, C)-inhibited (since $\rho(x_0) = 0$), therefore ρx_0 is the desired admissible solution. □

6.6 The Graphs $H(\sigma, C)$

We now describe a family of graphs $H(\sigma, C)$ defined in terms of a given system C of constraints and $\sigma \in T_C$. The purpose of these graphs is to keep track of the exposed variables and how firing them can enable other constraints, so that we can monitor the progress of a firing sequence.

Formally, $H(\sigma, C)$ is a finite labeled directed graph with vertices $C \cup \{*\}$. For each $p \leq q \in C$ and $x \in X$ such that x is σ-exposed in q, there is an edge labeled x from $\mathbf{con}(x, C)$ to $p \leq q$. (Recall that $\mathbf{con}(x, C)$ is $*$ if x is unconstrained in C, otherwise $\mathbf{con}(x, C)$ is some constraint $p \leq q \in C$.) Self-loops are allowed in this definition: if x is constrained in C by the constraint

$p \leq q$ and x is σ-exposed in q, $H(\sigma, C)$ has a self-loop labeled x on the vertex $p \leq q$.

It follows from Lemma 18(i) that if $\sigma \leq_X \tau$ then $H(\sigma, C)$ is a subgraph of $H(\tau, C)$. In particular, $H(\sigma, C)$ is a subgraph of $H(\sigma x, C)$. Moreover, it follows from Lemma 18(ii) that if $\sigma \in T_C$, x is (σ, C)-enabled, and $H(\sigma, C)$ contains an edge labeled x into $p \leq q$, then $p \leq q$ is σx-enabled.

We can think of $H(\sigma, C)$ as a net in which tokens are passed around as variables are fired. Firing a variable x causes at least one token to be passed from $\mathbf{con}\,(x, C)$ along all edges labeled x to other constraints in which x is exposed, enabling those constraints. The number of tokens that are passed depends on the values of $\sigma(q - p)$ for $p \leq q \in C$, but by Lemma 18(ii), it is always at least one.

Lemma 23 *Let $\sigma\tau \in T_C$ such that $\sigma \leq_C \sigma\tau$. Assume further that τ contains at least one variable constrained in C. Then $H(\sigma\tau, C)$ contains either a cycle all of whose labels are in τ or an edge out of $*$ whose label is in τ.*

Proof. Let x be constrained in C by the constraint $p \leq q$, and suppose that x occurs in τ at least once. Then $\sigma(p) < \sigma\tau(p)$. Also, $\sigma(q-p) \leq \sigma\tau(q-p)$, since $\sigma \leq_C \sigma\tau$. Combining these inequalities, we obtain $\sigma(q) < \sigma\tau(q)$. By Lemma 17, there must be a $y \in X$ and a prefix ρy of τ such that y is $\sigma\rho$-exposed in q. Then $H(\sigma\rho, C)$ contains an edge labeled y from $\mathbf{con}\,(y, C)$ to $\mathbf{con}\,(x, C)$. Since $H(\sigma\rho, C)$ is a subgraph of $H(\sigma\tau, C)$, this edge also exists in $H(\sigma\tau, C)$.

Now either y is unconstrained in C, in which case $\mathbf{con}\,(y, C) = *$ and we are done, or we can continue in the same fashion with y. Following these edges backwards, we must eventually either arrive at $*$ or cycle. □

6.7 Equivalence of Problem Instances

In our decidability proof, we will show that as a computation σ unfolds, the graph $H(\sigma, C)$ develops in certain ways that occasionally allow us to simplify C, for instance by discarding a constraint or a variable. In such cases we will construct a new system D that is structurally simpler than C but equivalent in the sense that D has a solution iff C does. The following definition gives the formal notion of equivalence of systems that we have in mind.

Definition 24 Let C, D be systems of constraints. We write $C \leq D$ if for every $\sigma \in T_C$ there is a $\tau \in T_D$ such that $\sigma \leq_X \tau$. We write $C \equiv D$ and say that C and D are *equivalent* if both $C \leq D$ and $D \leq C$. □

It follows immediately from this definition that if $C \equiv D$, then C has a solution if and only if D does.

6.8 Proof of Decidability

Let C be a system of Diophantine constraints. The following three lemmas, Lemmas 25, 26, and 27, identify three situations that will allow a structural simplification of the system C. We suggest that the reader skip the proofs of these lemmas on first reading and go directly to Theorem 28.

Lemma 25 *Let $p \leq q \in C$. If C has an unconstrained variable $\mathbf{0}$-exposed in q, then*

$$C \quad \equiv \quad C - \{p \leq q\}\ .$$

Proof. Let $C' = C - \{p \leq q\}$. The easier direction is $C \leq C'$. If y is (σ, C)-enabled then y is also (σ, C')-enabled, since y is either constrained by the same constraint in C and C' or unconstrained in C'. It follows that $T_C \subseteq T_{C'}$.

For the other direction, suppose $\sigma \in T_{C'}$. Let x be a C-unconstrained variable $\mathbf{0}$-exposed in q. Let $n = |\sigma|$ and let

$$\tau \quad = \quad \underbrace{xx\cdots x}_{n}\sigma \quad = \quad x^n\sigma\ .$$

Then $\sigma \leq_X \tau$. We show that $\tau \in T_C$. Certainly $x^n \in T_C$, since x is unconstrained. It remains to show that $\sigma \in T_C^{x^n}$. Resetting and using Lemma 11, it suffices to show $\sigma \in T_{\mathbf{inc}_{x^n}(C)}$. Thus we need to show that for any prefix ρy of σ, y is $(\rho, \mathbf{inc}_{x^n}(C))$-enabled. This follows from the fact that y is (ρ, C')-enabled: for any $f \leq g \in C'$,

$$\begin{aligned} \rho(\mathbf{inc}_{x^n}(g - f)) \quad &= \quad \rho x^n(g) - \rho(f) && \text{since } x \text{ does not occur in } f \\ &\geq \quad \rho(g - f)\ , \end{aligned}$$

and for the constraint $p \leq q$,

$$\begin{array}{rcll} \rho(\mathbf{inc}_{x^n}(q-p)) & = & \rho x^n(q) - \rho(p) & \text{since } x \text{ does not occur in } p \\ & \geq & \rho(q) + n - \rho(p) & \text{by Lemmas 17 and 18(i)} \\ & \geq & \rho(q) + n - |\rho| & \text{since } p \text{ is linear} \\ & > & 0 & \text{since } |\rho| < n. \end{array}$$

□

Lemmas 26 and 27 deal with two different kinds of cycles that can arise in $H(\sigma, C)$. The first is used when the cycle is a self-loop on a single vertex, and the latter is used when the cycle has at least two vertices.

Lemma 26 *If $H(\epsilon, C)$ has a self-loop labeled x on vertex $p \leq q$, and if x is $(\mathbf{0}, C)$-enabled, let*

$$C' = \begin{cases} (C - \{p \leq q\}) \cup \{p - x \leq q - x\} , & \textit{if } q - x \in \mathbb{N}[X] \\ C - \{p \leq q\} , & \textit{otherwise.} \end{cases}$$

Then $C \equiv C'$.

Proof. Since x is $\mathbf{0}$-exposed in q, by Definition 16 that q has a term of the form ax^k where $a, k \in \mathbb{N}$ and $a, k \geq 1$; *i.e.*, q can be written $q' + x^k$ with $q' \in \mathbb{N}[X]$. If the first alternative in the definition of C' holds, *i.e.* if q has a linear term ax, then we can take $k = 1$. If the second alternative holds, we can take $k > 1$. Let us call these two cases (i) and (ii), respectively. Either way, since **con** (x, C) is $p \leq q$, x also occurs in p, and since p is linear, $p = p' + x$ for some $p' \in \mathbb{N}[X]$.

First we show $C \leq C'$. This is immediate for case (ii) as in Lemma 25. For case (i), note that $q - p = q' - p'$. Thus for any $\sigma \in X^*$, any variable $y \in X - \{x\}$ is (σ, C)-enabled iff it is (σ, C')-enabled, and since x is unconstrained in C', x is always (σ, C')-enabled. It follows that $T_C \subseteq T_{C'}$, thus $C \leq C'$.

Now we show $C' \leq C$ for both cases. Let $\sigma \in T_{C'}$, and let $n = \max\{2, |\sigma|\}$. Let σ' be obtained by deleting all occurrences of x from σ, and let $\tau = x^n\sigma'$. Then $\sigma \leq_X \tau$. We claim that $\tau \in T_C$. Since x is $\mathbf{0}$-exposed in q and $(\mathbf{0}, C)$-enabled, by Lemmas 17 and 18(i), $x^n \in T_C$, so we need only prove that

$\sigma' \in T_C^{x^n}$. Resetting by Lemma 11, it suffices to prove that $\sigma' \in T_{\mathbf{inc}_{x^n}(C)}$. We need to show that for any prefix $\rho' y$ of σ', y is $(\rho', \mathbf{inc}_{x^n}(C))$-enabled. This will follow from the fact that y is (ρ, C')-enabled, where ρy is the unique prefix of σ such that $\rho' y$ is ρy with all occurrences of x removed (note $y \neq x$, since it occurs in σ').

Suppose ρ has m occurrences of x. For any $f \leq g \in C - \{p \leq q\}$,

$$\begin{aligned} \rho'(\mathbf{inc}_{x^n}(g-f)) &= \rho' x^n(g) - \rho(f) && \text{since } x \text{ does not occur in } f \\ &= \rho x^{n-m}(g) - \rho(f) \\ &\geq \rho(g) - \rho(f) \\ &= \rho(g-f)\ . \end{aligned}$$

For the argument involving constraint $p \leq q$, we split on cases. In case (i),

$$\begin{aligned} \rho'(\mathbf{inc}_{x^n}(q-p)) &= \rho' x^n(q'-p') \\ &= \rho' x^n(q') - \rho(p') && \text{since } x \text{ does not occur in } p' \\ &= \rho x^{n-m}(q') - \rho(p') \\ &\geq \rho(q') - \rho(p') \\ &= \rho(q'-p')\ . \end{aligned}$$

In case (ii),

$$\begin{aligned} \rho'(\mathbf{inc}_{x^n}(q-p)) &= \rho' x^n(q-p) \\ &= \rho' x^n(q') + \rho' x^n(x^k) - \rho' x^n(p') - \rho' x^n(x) \\ &\geq \delta_x^n(x^k) - \rho'(p') - \delta_x^n(x) \\ &\geq n^k - (n-1) - n \\ &\geq (n-1)^2 \\ &> 0\ . \end{aligned}$$

□

Lemma 27 *If there is a cycle in $H(\epsilon, C)$ on vertices*

$$D \;=\; \{p_0 \leq q_0, \ldots, p_{n-1} \leq q_{n-1}\}\ ,$$

then $C \equiv C'$, where

$$\begin{aligned} p' &= \sum_{i=0}^{n-1} p_i \\ q' &= \sum_{i=0}^{n-1} q_i \\ C' &= (C - D) \cup \{p' \leq q'\} \ . \end{aligned}$$

Proof. First we show $C \leq C'$. As above, it suffices to show that for any assignment $\sigma \in T_C$ and variable y, if y is (σ, C)-enabled then y is (σ, C')-enabled. If $\mathbf{con}\,(y, C) \notin D$, then $\mathbf{con}\,(y, C') = \mathbf{con}\,(y, C)$, thus y is (σ, C)-enabled iff it is (σ, C')-enabled. Otherwise, if $\mathbf{con}\,(y, C) \in D$, say $p_k \leq q_k$ for some $0 \leq k \leq n-1$, then $\mathbf{con}\,(y, C')$ is $p' \leq q'$. Since $\sigma \in T_C$, we have $\sigma(p_i) \leq \sigma(q_i)$, $0 \leq i \leq n-1$. Moreover, since y is (σ, C)-enabled, we have $\sigma(p_k) < \sigma(q_k)$. Thus $\sigma(p') < \sigma(q')$, so y is (σ, C')-enabled.

Now we show $C' \leq C$. Assume without loss of generality that the vertices in D occur on the cycle of $H(\epsilon, C)$ in the order $p_0 \leq q_0$, …, $p_{n-1} \leq q_{n-1}$ and that y_i is the label on the edge from $p_i \leq q_i$ to $p_{i+1} \leq q_{i+1}$, $0 \leq i \leq n-1$ (arithmetic on subscripts is modulo n).

The intuitive idea behind the following argument is that if some y_i is enabled, then firing y_i enables y_{i+1}, and so on; thus we can imagine a token being passed around the cycle D, enabling whichever $p_j \leq q_j \in D$ is needed.

Let $\sigma \in T_{C'}$. We construct by induction on the length of σ a string $\sigma' \in T_C$ such that $\sigma \leq_{X,C'} \sigma'$. Define $\epsilon' = \epsilon$. Now suppose $\sigma y \in T_{C'}$ and σ' has been defined. By the induction hypothesis,

(i) $\sigma \leq_{X,C'} \sigma'$

(ii) $\sigma' \in T_C$.

Since y is (σ, C')-enabled, by (i) we have that y is (σ', C')-enabled.

If $\mathbf{con}\,(y, C)$ is in $C - D$ or $\mathbf{con}\,(y, C) = *$, let $(\sigma y)' = \sigma' y$. Then $\sigma y \leq_X (\sigma y)'$, and since $\mathbf{con}\,(y, C) = \mathbf{con}\,(y, C')$, y is (σ', C)-enabled. Moreover, $\sigma y \leq_{C'} \sigma' y$ by Lemma 14(iii).

If $\mathbf{con}\,(y, C)$ is in D, say $p_k \leq q_k$, then $\mathbf{con}\,(y, C')$ is $p' \leq q'$. By (i) and (ii),

$$\begin{aligned} \sigma'(p') &< \sigma'(q') \ , \\ \sigma'(p_i) &\leq \sigma'(q_i) \ , \quad 0 \leq i \leq n-1 \ . \end{aligned}$$

It follows that there must exist an i, $0 \leq i \leq n-1$, such that

$$\sigma'(p_i) \quad < \quad \sigma'(q_i) \; . \tag{6.10}$$

Define

$$(\sigma y)' \quad = \quad \sigma' y_i y_{i+1} y_{i+2} \cdots y_{k-1} y$$

(the sequence $i, i+1, \ldots, k-1$ wraps modulo n if necessary). Then $\sigma y \leq_X (\sigma y)'$. By (6.10), y_i is (σ', C)-enabled. Since each y_j is $\mathbf{0}$-exposed in q_{j+1}, $0 \leq j \leq n-1$, it follows inductively that each y_j is $(\sigma' y_i y_{i+1} \cdots y_{j-1}, C)$-enabled, and y is $(\sigma' y_i y_{i+1} \cdots y_{k-1}, C)$-enabled. Thus $\sigma' y_i y_{i+1} \cdots y_{k-1} y \in T_C$.

It remains to show that $\sigma y \leq_{C'} (\sigma y)'$. For $p \leq q$ in $C - D$,

$$\begin{aligned}
(\sigma y)'(q-p) &= (\sigma y)'(q) - (\sigma y)'(p) \\
&\geq \sigma y(q) - (\sigma y)'(p) \\
&= \sigma y(q) - \sigma y(p) \qquad\qquad \text{since the } y_i \text{ do not appear in } p \\
&= \sigma y(q-p) \; .
\end{aligned}$$

For $p' \leq q'$, since each y_j is $\mathbf{0}$-exposed in q_{j+1} and hence also in q', by Lemma 18(ii) we have

$$\begin{aligned}
(\sigma y)'(q'-p') &= \sigma' y_i y_{i+1} y_{i+2} \cdots y_{k-1} y(q'-p') \\
&\geq \sigma' y_{i+1} y_{i+2} \cdots y_{k-1} y(q'-p') \\
&\geq \sigma' y_{i+2} \cdots y_{k-1} y(q'-p') \\
&\geq \cdots \\
&\geq \sigma' y(q'-p')
\end{aligned} \tag{6.11}$$

By Lemma 14(iii) and the induction hypothesis, (6.11) is bounded below by $\sigma y(q'-p')$. □

Theorem 28 *It is decidable whether a given instance C of the NRP has a solution.*

Proof. We proceed by induction on the complexity of C. If $C = \emptyset$, then all variables are unconstrained and therefore enabled, thus we can increment x_0 immediately. Otherwise assume C is nonempty.

We identify a number of cases below, each of which allows us to reduce the size of C in some respect (either fewer constraints or fewer constrained variables). In each case, the induction hypothesis gives a procedure for deciding whether the smaller system has a solution, and this will determine whether C has a solution.

Case 1 C contains an unconstrained $(\mathbf{0}, C)$-exposed variable. By Lemma 25, C is equivalent to a system with fewer constraints.

Case 2 $H(\epsilon, C)$ has a self-loop labeled x, and x is $(\mathbf{0}, C)$-enabled. By Lemma 26, C is equivalent to a system with either fewer constrained variables or fewer constraints.

Case 3 $H(\epsilon, C)$ has a cycle on a set of at least two vertices. By Lemma 27, C is equivalent to a system with fewer constraints.

Case 4 None of Cases 1, 2, or 3 apply. In this case, consider the set T_C^{adm} consisting of all admissible strings in T_C. The set T_C^{adm} contains the empty string ϵ and is closed under the prefix relation, so it is a tree. For any $\sigma \in T_C^{\mathrm{adm}}$, $\sigma x \in T_C^{\mathrm{adm}}$ iff x is (σ, C)-enabled but not (σ, C)-inhibited. By Lemma 22, C has a solution if and only if it has one in T_C^{adm}.

Now let T'_C be the subtree of T_C^{adm} obtained by deleting all strings containing a proper prefix of the form $\sigma\tau$, where $|\tau| > |X|$ and $\sigma \leq_C \sigma\tau$. The tree T'_C has no infinite paths, since Dickson's Lemma (Lemma 15) says that any infinite path must contain $v_0, v_1, v_2, \ldots$ such that each v_i is a proper prefix of v_{i+1} and each $v_i \leq_C v_{i+1}$; thus $v_0 \leq_C v_{|X|+1}$ and the difference in their lengths is at least $|X| + 1$, so this infinite path would be pruned in the construction of T'_C. By König's Lemma, T'_C is finite, since it is finitely branching. The tree T'_C can be constructed effectively since the conditions for extending a branch and for pruning are effective.

Since any extension in T_C of a solution is a solution, C has a solution iff it has a solution of the form $\sigma\tau \in T_C^{\mathrm{adm}}$ for some leaf σ of T'_C. The leaves σ are of two types, not necessarily mutually exclusive:

(i) All (σ, C)-enabled variables are (σ, C)-inhibited. Leaves of this form are leaves of T_C^{adm}, since they have no C-admissible extensions.

(ii) The leaf σ is of the form $\tau\rho$, where $\tau \leq_C \tau\rho$ and $|\rho| > |X|$. Leaves of this form are not necessarily leaves of T_C^{adm}, but are obtained by pruning T_C^{adm} in the construction of T'_C.

If $\sigma(x_0) > 0$ or x_0 is (σ, C)-enabled for some leaf σ, we are done: in the former case, σ is a solution, and in the latter, σx_0 is a solution. Otherwise, there is no admissible solution extending a leaf of the form (i). Thus we are left with leaves of the form (ii). For each such leaf $\tau\rho$, where $\tau \leq_C \tau\rho$ and $|\rho| > |X|$, since $\tau\rho$ is C-admissible, for every prefix vx of ρ, either

- x is constrained in C,
- x is $(\tau v, C)$-exposed, or
- $\tau v(x) = 0$.

Suppose ρ contains a variable constrained in C. By Lemma 23, $H(\tau\rho, C)$ contains either an edge out of $*$ or a cycle whose labels are in ρ. If the former, we revert to Case 1 after resetting. If the latter and the cycle is of length at least two, we revert to Case 3 after resetting. Otherwise there is a self-loop in $H(\tau\rho, C)$ with label x, where vx is a prefix of ρ. If that self-loop already exists in $H(\tau v, C)$, then since x is τv-enabled, we revert to Case 2 after resetting. Otherwise, let vy be the shortest prefix of ρ such that $H(\tau vy, C)$ contains that self-loop. By Lemma 18(v), x is τvy-enabled, and we revert to Case 2 after resetting.

If all variables occurring in ρ are unconstrained in C and at least one is $(\tau v, C)$-exposed for some prefix v of ρ, then $H(\tau v, C)$ has an edge out of $*$, and we revert to Case 1 after resetting.

Finally, if all variables occurring in ρ are unconstrained in C and not $(\tau\rho, C)$-exposed, we must have $\tau v(x) = 0$ for every prefix vx of ρ, otherwise the string would not be admissible. But since $|\rho| > |X|$, at least one variable must be fired twice, so this situation cannot occur. □

Acknowledgements

We are indebted to Moshe Vardi for many valuable ideas and the anonymous referees for a thorough reading and excellent suggestions that substantially improved the presentation.

We gratefully acknowledge the support of the National Science Foundation under grant CCR-9317320, BRICS (Basic Research in Computer Science), a Centre of the Danish National Research Foundation, the John Simon Guggenheim Foundation, and the U.S. Army Research Office through the ACSyAM branch of the Mathematical Sciences Institute of Cornell University under contract DAAL03-91-C-0027.

References

[1] A. Aiken, D. Kozen, M. Vardi, and E. Wimmers, *The complexity of set constraints*, in Proc. 1993 Conf. Computer Science Logic (CSL'93), E. Börger, Y. Gurevich, and K. Meinke, eds., vol. 832 of Lect. Notes in Comput. Sci., Eur. Assoc. Comput. Sci. Logic, Springer, September 1993, pp. 1–17.

[2] A. Aiken, D. Kozen, and E. Wimmers, *Decidability of systems of set constraints with negative constraints*, Tech. Rep. 93-1362, Computer Science Department, Cornell University, June 1993.

[3] A. Aiken and B. Murphy, *Implementing regular tree expressions*, in Proc. 1991 Conf. Functional Programming Languages and Computer Architecture, August 1991, pp. 427–447.

[4] ——, *Static type inference in a dynamically typed language*, in Proc. 18th Symp. Principles of Programming Languages, ACM, January 1991, pp. 279–290.

[5] A. Aiken and E. Wimmers, *Solving systems of set constraints*, in Proc. 7th Symp. Logic in Computer Science, IEEE, June 1992, pp. 329–340.

[6] L. Bachmair, H. Ganzinger, and U. Waldmann, *Set constraints are the monadic class*, in Proc. 8th Symp. Logic in Computer Science, IEEE, June 1993, pp. 75–83.

[7] J. A. Brzozowski and E. Leiss, *On equations for regular languages, finite automata, and sequential networks*, Theor. Comput. Sci., 10 (1980), pp. 19–35.

[8] W. Charatonik and L. Pacholski, *Negative set constraints with equality*, in Proc. 9th Symp. Logic in Computer Science, IEEE, July 1994, pp. 128–136.

[9] ——, *Set constraints with projections are in NEXPTIME*, in Proc. 35th Symp. Foundations of Computer Science, IEEE, November 1994, pp. 642–653.

[10] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms*, Springer-Verlag, 1992.

[11] R. Gilleron, S. Tison, and M. Tommasi, *Solving systems of set constraints using tree automata*, in Proc. Symp. Theor. Aspects of Comput. Sci., vol. 665, Springer-Verlag Lect. Notes in Comput. Sci., February 1993, pp. 505–514.

[12] ——, *Solving systems of set constraints with negated subset relationships*, in Proc. 34th Symp. Foundations of Comput. Sci., IEEE, November 1993, pp. 372–380.

[13] N. Heintze and J. Jaffar, *A decision procedure for a class of set constraints*, in Proc. 5th Symp. Logic in Computer Science, IEEE, June 1990, pp. 42–51.

[14] ——, *A finite presentation theorem for approximating logic programs*, in Proc. 17th Symp. Principles of Programming Languages, ACM, January 1990, pp. 197–209.

[15] N. D. Jones and S. S. Muchnick, *Flow analysis and optimization of LISP-like structures*, in Proc. 6th Symp. Principles of Programming Languages, ACM, January 1979, pp. 244–256.

[16] D. Kozen, *Logical aspects of set constraints*, in Proc. 1993 Conf. Computer Science Logic (CSL'93), E. Börger, Y. Gurevich, and K. Meinke, eds., vol. 832 of Lect. Notes in Comput. Sci., Eur. Assoc. Comput. Sci. Logic, Springer, September 1993, pp. 175–188.

[17] K. Marriott and M. Odersky, *Systems of negative Boolean constraints*, Tech. Rep. YALEU/DCS/RR-900, Computer Science Department, Yale University, April 1992.

[18] P. Mishra, *Towards a theory of types in PROLOG*, in Proc. 1st Symp. Logic Programming, IEEE, 1984, pp. 289–298.

[19] P. Mishra and U. Reddy, *Declaration-free type checking*, in Proc. 12th Symp. Principles of Programming Languages, ACM, 1985, pp. 7–21.

[20] J. C. Reynolds, *Automatic computation of data set definitions*, in Information Processing 68, North-Holland, 1969, pp. 456–461.

[21] K. Stefánsson, *Systems of set constraints with negative constraints are NEXPTIME-complete*, in Proc. 9th Symp. Logic in Computer Science, IEEE, June 1994, pp. 137–141.

[22] J. Young and P. O'Keefe, *Experience with a type evaluator*, in Partial Evaluation and Mixed Computation, D. Bjørner, A. P. Ershov, and N. D. Jones, eds., North-Holland, 1988, pp. 573–581.