The Complexity of Set Constraints

Alexander Aiken*
Dexter Kozen**
Moshe Vardi*
Ed Wimmers*

TR 93-1352
May 1993

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120.
**Computer Science Department, Cornell University, Ithaca, NY 14853.

The Complexity of Set Constraints

Alexander Aiken*
aiken@almaden.ibm.com

Dexter Kozen†
kozen@cs.cornell.edu

Moshe Vardi*
vardi@almaden.ibm.com

Ed Wimmers*
wimmers@almaden.ibm.com

May 10, 1993

Abstract

Set constraints are relations between sets of terms. They have been used extensively in various applications in program analysis and type inference. We present several results on the computational complexity of solving systems of set constraints. The systems we study form a natural complexity hierarchy depending on the form of the constraint language.

1 Introduction

Systems of set constraints have received considerable attention as a formalism for expressing algorithms in program analysis and type inference. Many algorithms based on set constraints have been proposed and implemented, but very little is known about the computational complexity of solving systems of set constraints. In this paper we present complexity results for a natural hierarchy of decision problems involving set constraints.

Set constraints are formal inclusions between expressions representing subsets of T_Σ, the set of ground terms over a finite ranked alphabet Σ. A

*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120
†Computer Science Department, Cornell University, Ithaca, NY 14853

positive set constraint is an inclusion $E \subseteq F$, where E and F are expressions built from a set $X = \{x, y, \ldots\}$ of variables ranging over subsets of T_Σ, a constant 0 denoting the empty set, a constant 1 denoting the set T_Σ, the usual set-theoretic operators $\cup$ (set union), $\cap$ (set intersection), and $\sim$(complement in T_Σ), and an n-ary set constructor f for each n-ary symbol $f \in \Sigma$ with semantics

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

Any valuation

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

assigning a subset of T_Σ to each variable extends uniquely to a map

$$\sigma : \{\text{set expressions}\} \quad \rightarrow \quad 2^{T_\Sigma} \ .$$

A valuation σ *satisfies* the constraint $E \subseteq F$ if $\sigma(E) \subseteq \sigma(F)$. A set $\mathcal{S}$ of constraints is *satisfiable* if there is a valuation that satisfies all constraints in $\mathcal{S}$ simultaneously.

An algorithm for determining the satisfiability of general systems of positive set constraints was first presented in [4]. In this paper, we extend the results of [4] in two ways. In Section 5, we give a new characterization of the satisfiability problem that may be of independent interest. We show that deciding whether $\mathcal{S}$ is satisfiable is equivalent to deciding whether or not a certain finite hypergraph constructed from $\mathcal{S}$ has an induced subhypergraph that is *closed* (see Section 4). This characterization is simpler than the one in [4] and has the additional advantage for complexity analysis that the hypergraphs can be specified using short Boolean formulas.

In Section 6, we exploit the hypergraph characterization of the satisfiability problem to obtain a family of complexity results for the satisfiability problem. We obtain an exhaustive hierarchy of completeness results for various complexity classes depending on the number of elements of Σ of each arity. To the best of our knowledge, these are the first upper and lower bound results for set constraint problems other than the *NEXPTIME*-completeness result for the general problem, which has been obtained independently by [5]. Our complexity results are summarized in the following table.

number of elements in Σ *of*			*complexity of the*
arity 0	*arity 1*	*arity 2+*	*satisfiability problem*
0	0+	0+	trivial
1+	0	0	*NP*-complete
1+	1	0	*PSPACE*-complete
1+	2+	0	*EXPTIME*-complete
1+	0+	1+	*NEXPTIME*-complete

Intuitively, these results can be explained as follows. If Σ has only nullary symbols, then the problem is essentially Boolean satisfiability. A system with one unary symbol has the power to count in unary, which is sufficient to simulate a *PSPACE* computation. A system with at least two unary symbols can encode alternation, which gives alternating *PSPACE* or *EXPTIME*. Finally, in a system with at least one symbol of arity two or greater, there can be constraints such as $fxy \subseteq 0$, which is satisfiable iff either $x \subseteq 0$ or $y \subseteq 0$. This allows the system to encode nondeterministic choices, which raises the complexity to *NEXPTIME*.

In a companion paper [1], we consider systems with *negative* set constraints $E \not\subseteq F$. Negative constraints considerably increase the power of the constraint language. With negative constraints, the complement of the set of solutions of a system of positive constraints can be expressed as a disjunction of a set of negative constraints: σ does not satisfy the system $\mathcal{S}$ iff σ satisfies

$$\bigvee_{E \subseteq F \in \mathcal{S}} E \not\subseteq F .$$

Generalizing this observation, any Boolean combination of positive and negative constraints can be expressed as the disjunction of systems of positive and negative constraints. We show in [1] that it is decidable whether a system of positive and negative constraints is satisfiable. This result is proved by means of a reduction to a number-theoretic problem which may be of independent interest.

2 Applications and Related Work

The greatest interest in set constraints stems from the area of program analysis, where set constraints have been used for a number of years in many different settings [16, 12, 14, 15, 17, 10, 2, 3]. In these applications, set constraints are generated from the program text and then solved to obtain useful information about the program (*e.g.*, whether it is well-typed). Representing basic data structures such as lists requires binary symbols; our results show that in this general case, solving set constraints is *NEXPTIME*-complete. In practice, implementations of set constraint solvers introduce restrictions or heuristics specific to the problem domain to achieve better worst-case time complexity. Our results show that such techniques are in fact necessary to achieve more efficient implementations.

Most of the systems for program analysis cited above deal with only positive set constraints. In [2, 3], opportunities for program optimization are identified by an *ad hoc* technique for checking the satisfiability of negative set constraints; the results of [1] show that satisfiability of negative constraints is decidable in general.

Special cases of set constraints have also arisen naturally in the study of finite automata. An example is an algorithm for solving equations between regular languages with free variables [6]. In [6], no complexity analysis is given. There is a simple linear-time reduction of regular expressions to systems of set constraints over unary and nullary symbols. Thus, our results show that deciding the satisfiability of equations between regular languages with free variables is in *EXPTIME*.

Set constraints with only nullary symbols correspond to Boolean algebras over a finite set of atoms. See [13] for more general results on solving negative constraints in arbitrary Boolean algebras.

Finally, set constraints have been studied for their own sake and several algorithms for solving set constraints have been proposed [9, 4, 8]. Our results differ from these in that we are interested primarily in the complexity of the satisfiability problem for set constraints.

3 Set Expressions and Set Constraints

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, 3, 4, and n are called *nullary, unary, binary, ternary, quaternary*, and *n-ary*, respectively. Nullary elements of Σ 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 naive set theory. 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 $x, y \in X$. Set expressions are denoted $E, F, \ldots$.

A *constraint* is a formal inclusion $E \subseteq F$, where E and F are set expressions. We might also allow equational constraints $E = F$, although inclusions and equations are interdefinable.

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

$$f : (2^{T_\Sigma})^n \quad \to \quad 2^{T_\Sigma}$$

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

A *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 constraint $E \subseteq F$ if $\sigma(E) \subseteq \sigma(F)$. A family $\mathcal{S}$ of set constraints is *satisfiable* if there is a set assignment that satisfies all the constraints in $\mathcal{S}$ simultaneously. The *satisfiability problem* is to determine whether a given finite system $\mathcal{S}$ of set constraints over Σ is satisfiable.

A *Boolean expression* over X is any element of $T_{\mathrm{B}}(X)$ (*i.e.*, a set expression with no symbols from Σ). A *truth assignment* is a map

$$u : X \rightarrow \mathbf{2}$$

where $\mathbf{2} = \{0,1\}$ is the two-element Boolean algebra. Any truth assignment u extends uniquely to a B-homomorphism

$$u : T_{\mathrm{B}}(X) \rightarrow \mathbf{2}$$

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 $x_i \mapsto a_i$, $1 \leq i \leq m$.

3.1 Normal Form

We show in this section how to transform a given system $\mathcal{S}$ of set constraints into an equivalent system in a special normal form. This step simplifies the proof of correspondence between set constraints and hypergraphs, because the normal form is actually quite close to the hypergraphs defined in Section 4. The transformation to normal form does not significantly increase the size of the system.

1. For every subexpression $fE_1 \ldots E_n$ of some set expression in $\mathcal{S}$, where $f \in \Sigma_n$, let $y_0, y_1, \ldots, y_n$ be new variables, replace all occurrences of the subexpression $fE_1 \ldots E_n$ 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 constraints (*i.e.*, do not contain any occurrence of $f \in \Sigma$) or are of the form

   $$y_0 = fy_1 \cdots y_n$$

   where $f \in \Sigma_n$ and $y_0, y_1, \ldots, y_n$ are variables. Let $X = \{x_1, \ldots, x_m\}$ 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

$$Z_f = \{z^f_{ij} \mid 0 \le i \le n,\ 1 \le j \le m\}$$

and add the constraints

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

for all $1 \le i \le n$ and $1 \le j \le m$.

3. Each constraint

$$x_{j_0} = fx_{j_1}\ldots x_{j_n} \tag{1}$$

obtained in step 1 is equivalent to the constraint

$$x_{j_0} = fx_{j_1}\underbrace{1\ldots1}_{n-1} \cap f1x_{j_2}\underbrace{1\ldots1}_{n-2} \cap \cdots \cap f\underbrace{1\ldots1}_{n-1}x_{j_n}$$

which in turn is equivalent to the conjunction of constraints

$$\begin{aligned} f\underbrace{1\ldots1}_{n} \cap x_{j_0} &= fx_{j_1}\underbrace{1\ldots1}_{n-1} \cap f1x_{j_2}\underbrace{1\ldots1}_{n-2} \cap \cdots \cap f\underbrace{1\ldots1}_{n-1}x_{j_n} \\ g\underbrace{1\ldots1}_{m} \cap x_{j_0} &= 0\ ,\quad g \ne f\ ,\quad m = \mathbf{arity}(g)\ . \end{aligned}$$

Replace the constraint (1) with the constraints

$$\begin{aligned} z^f_{0j_0} &= \bigcap_{i=1}^{n} z^f_{ij_i} \\ z^g_{0j_0} &= 0\ ,\quad g \ne f\ . \end{aligned}$$

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

4. At this point we have

   - purely Boolean constraints formed in step 1 involving only the variables X

- for each $f \in \Sigma$, purely Boolean constraints formed in step 3 involving only the variables Z_f
- the mixed constraints formed in step 2.

Replace each purely Boolean constraint $E \subseteq F$ involving the variables X by the equivalent constraint

$$\sim E \cup F \quad = \quad 1 \ .$$

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 $f \in \Sigma$.

After this translation, $\mathcal{S}$ contains

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

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

for each $f \in \Sigma_n$ and each $1 \le i \le n$, $1 \le j \le m$.

This is the desired normal form.

4 Hypergraphs

For our purposes, a *hypergraph* is a structure

$$H \quad = \quad (U,\ E_i \mid i \in I)$$

consisting of a finite set U of *vertices* and an indexed family of relations E_i of various arities on U called *hyperedge relations.* An element of an n-ary hyperedge relation is called an n-ary *hyperedge*. (In our application, the index set I is the ranked alphabet Σ, and for $f \in \Sigma$, $\mathbf{arity}(E_f)$ is $\mathbf{arity}(f) + 1$.)

If H is a hypergraph on vertices U and if $U' \subseteq U$, the *induced subhypergraph* of H on U' is the hypergraph H' on vertices U' whose hyperedge relations are the hyperedge relations of H restricted to U'. That is, if

$$H = (U,\ E_i \mid i \in I)\ ,$$

then

$$H' = (U',\ E'_i \mid i \in I)$$

where if E_i is n-ary then

$$E'_i = E_i \cap (U')^n\ .$$

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

$$E : U^n \rightarrow 2^U$$

where

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

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

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

Example 1 Consider the hypergraph consisting of vertices $\mathbb{Z}_p$, the field of integers modulo a prime p, and a single ternary hyperedge relation

$$E = \{(a,\ b,\ ab) \mid a,\ b \in \mathbb{Z}_p\}\ .$$

This hypergraph is not closed because there is no $a \in \mathbb{Z}_p$ such that $(a,\ 0,\ 1) \in E$. However, the induced subhypergraph on the nonzero elements of $\mathbb{Z}_p$ is closed, since for all $b,\ c \in \mathbb{Z}_p - \{0\}$ there exists an $a \in \mathbb{Z}_p - \{0\}$ such that $(a,\ b,\ c) \in E$, namely $a = c/b$. □

4.1 Succinct Specification of Hypergraphs

We work with a particular class of hypergraphs whose vertices and hyperedge relations are specified succinctly by Boolean formulas in the following way. Let

$$\begin{aligned} X &= \{x_1, \ldots, x_m\} \\ Z_f &= \{z_{ij}^f \mid 0 \le i \le \mathbf{arity}(f),\ 1 \le j \le m\}\ , \quad f \in \Sigma \end{aligned}$$

be pairwise disjoint sets of variables. Suppose we are given Boolean formulas

$$\begin{aligned} B &\in T_{\mathrm{B}}(X) \\ C_f &\in T_{\mathrm{B}}(Z_f)\ , \quad f \in \Sigma\ . \end{aligned}$$

These formulas determine a hypergraph

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

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 occasionally call the elements of U *atoms* because they represent atoms of the free Boolean algebra on generators X modulo $B = 1$. 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

$$C_f[z_{ij}^f := u_i(x_j)] = 1\ . \tag{2}$$

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 (2) 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 the hypergraph can be exponential in the size of its specification.

5 Set Constraints and Hypergraph Closure

In this section we give an efficiently computable correspondence between systems of set constraints in normal form as described in Section 3.1 and hypergraphs specified by systems of Boolean formulas as described in Section 4.1. A system of set constraints is satisfiable iff the corresponding hypergraph has a closed induced subhypergraph. Let

$$\begin{aligned} X &= \{x_1, \ldots, x_m\} \\ Z_f &= \{z^f_{ij} \mid 0 \le i \le \mathbf{arity}(f),\ 1 \le j \le m\}\ , \quad f \in \Sigma \end{aligned}$$

be pairwise disjoint sets of variables. The system $\mathcal{S}$ in normal form consisting of constraints

$$\begin{aligned} B &= 1\ , \quad B \in T_{\mathrm{B}}(X) \\ C_f &= 1\ , \quad C_f \in T_{\mathrm{B}}(Z_f)\ , \quad f \in \Sigma \end{aligned}$$

and for each $f \in \Sigma_n$, $1 \le i \le n$, and $1 \le j \le m$,

$$z^f_{0j} = f\underbrace{1\ldots1}_{n} \cap x_j \tag{3}$$

$$z^f_{ij} = f\underbrace{1\ldots1}_{i-1}x_j\underbrace{1\ldots1}_{n-i} \tag{4}$$

corresponds to the hypergraph H specified by the formulas B and C_f, $f \in \Sigma$.

Theorem 2 *The hypergraph H has a closed induced subhypergraph if and only if the system $\mathcal{S}$ of set constraints is satisfiable.*

Proof. ($\Rightarrow$) Let

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

be a closed induced subhypergraph of H. Define $\theta : T_\Sigma \to U'$ inductively such that for all $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$,

$$\theta(ft_1 \ldots t_n) \in E'_f(\theta(t_1), \ldots, \theta(t_n))\ .$$

This is possible since H' is closed. Each $\theta(t)$ is a truth assignment to X satisfying B. For each term t_0 of the form $ft_1 \ldots t_n$, extend $\theta(t_0)$ to a truth assignment to $X \cup Z_f$ as follows:

$$\theta(t_0)(z_{ij}^f) \quad = \quad \theta(t_i)(x_j) \ , \quad z_{ij}^f \in Z_f \ . \tag{5}$$

Define a set assignment σ as follows.

$$\begin{aligned} \sigma(x_j) &= \{t \mid \theta(t)(x_j) = 1\} \\ \sigma(z_{ij}^f) &= \sigma(f \underbrace{1 \ldots 1}_{i-1} x_j \underbrace{1 \ldots 1}_{n-i}) \\ \sigma(z_{0j}^f) &= \sigma(f \underbrace{1 \ldots 1}_{n} \cap x_j) \end{aligned}$$

for $f \in \Sigma_n$, $1 \le i \le n$, and $1 \le j \le m$. We show that σ satisfies $\mathcal{S}$.

It is immediate from the definition of σ that the constraints (3) and (4) are satisfied. We now show that σ satisfies $B = 1$. For $t \in T_\Sigma$, let $\chi_t : 2^{T_\Sigma} \to \mathbf{2}$ be the characteristic function

$$\chi_t(A) \quad = \quad \begin{cases} 1 \ , & \text{if } t \in A, \\ 0 \ , & \text{if } t \notin A. \end{cases}$$

By definition of σ,

$$\begin{aligned} \chi_t(\sigma(x_j)) = 1 \quad &\Leftrightarrow \quad t \in \sigma(x_j) \\ &\Leftrightarrow \quad \theta(t)(x_j) = 1 \end{aligned} \tag{6}$$

so $\chi_t \circ \sigma$ and $\theta(t)$ agree on X. Since both are B-homomorphisms (*i.e.*, preserve Boolean operations), they agree on all elements of $T_{\mathrm{B}}(X)$; in particular, they agree on B. Since $\theta(t)(B) = 1$ for all t by definition of U, we have that $\chi_t(\sigma(B)) = 1$ for all t. Thus $\sigma(B) = T_\Sigma = \sigma(1)$.

Finally, we show that σ satisfies $C_f = 1$. Recall that C_f is a conjunction of constraints of the form $\sim E \cup F$, where E and F are either 0 or a conjunction of elements of Z_f. Since σ satisfies (3) and (4), it also satisfies $E \subseteq f \underbrace{1 \ldots 1}_{n}$.

By Boolean reasoning it follows that σ also satisfies

$$\sim f \underbrace{1 \ldots 1}_{n} \quad \subseteq \quad \sim E \quad \subseteq \quad \sim E \cup F \ ,$$

therefore

$$\sim f \underbrace{1 \ldots 1}_{n} \quad \subseteq \quad C_f \ .$$

Thus it remains to show that σ satisfies

$$f \underbrace{1 \ldots 1}_{n} \quad \subseteq \quad C_f \ ,$$

or in other words, for all terms t_0 of the form $ft_1 \ldots t_n$,

$$t_0 \quad \in \quad \sigma(C_f) \ .$$

As above, we argue that $\chi_{t_0} \circ \sigma$ and $\theta(t_0)$ agree on Z_f:

$$\begin{array}{rcll} \theta(t_0)(z_{ij}^f) & = & \theta(t_i)(x_j) & \text{by (5)} \\ & = & t_i \in \sigma(x_j) & \text{by (6)} \\ & = & t_0 \in \sigma(f \underbrace{1 \ldots 1}_{i-1} x_j \underbrace{1 \ldots 1}_{n-i}) & \text{by definition of } \sigma \\ & = & \chi_{t_0}(\sigma(z_{ij}^f)) & \text{since } \sigma \text{ satisfies (4)} \end{array}$$

for $0 \leq i \leq n$ and $1 \leq j \leq m$. Since both $\chi_{t_0} \circ \sigma$ and $\theta(t_0)$ are B-homomorphisms, they agree on all elements of $T_{\mathrm{B}}(Z_f)$; in particular, they agree on C_f. Since

$$(\theta(t_0), \theta(t_1), \ldots, \theta(t_n)) \in E'_f \ ,$$

we have

$$\begin{array}{rcl} C_f[\theta(t_0), \ldots, \theta(t_n)] = 1 & \Rightarrow & C_f[z_{ij}^f := \theta(t_i)(x_j)] = 1 \\ & \Rightarrow & C_f[z_{ij}^f := \theta(t_0)(z_{ij}^f)] = 1 \\ & \Rightarrow & \theta(t_0)(C_f) = 1 \\ & \Rightarrow & \chi_{t_0}(\sigma(C_f)) = 1 \\ & \Rightarrow & t_0 \in \sigma(C_f) \ . \end{array}$$

($\Leftarrow$) Suppose σ satisfies $\mathcal{S}$. Regarding $u \in U$ as a conjunction of literals, let

$$U' \quad = \quad \{u \in U \mid \sigma(u) \neq \emptyset\} \ .$$

We claim that the induced subhypergraph

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

of H is closed. For $f \in \Sigma_n$ and for all $u_1, \ldots, u_n \in U'$, let $t_i \in \sigma(u_i)$, $1 \le i \le n$. The t_i exist by definition of U'. There is a unique atom u_0 such that $t_0 = ft_1 \ldots t_n \in \sigma(u_0)$, and $u_0 \in U'$.

Extend u_0 to domain $X \cup Z_f$ by defining $u_0(z^f_{ij}) = u_i(x_j)$, $0 \le i \le n$, $1 \le j \le m$. Then

$$\begin{aligned}
u_0(z^f_{ij}) &= u_i(x_j) \\
&= \begin{cases} 1\,, & \text{if } t_i \in \sigma(x_j) \\ 0\,, & \text{otherwise} \end{cases} \\
&= \begin{cases} 1, & \text{if } i = 0 \text{ and } t_0 \in \sigma(x_j) \\ 1, & \text{if } i \ge 1 \text{ and } t_0 \in \sigma(f\underbrace{1\ldots1}_{i-1}x_j\underbrace{1\ldots1}_{n-i}) \\ 0, & \text{otherwise} \end{cases} \\
&= \begin{cases} 1, & \text{if } i = 0 \text{ and } t_0 \in \sigma(z^f_{0j}) \\ 1, & \text{if } i \ge 1 \text{ and } t_0 \in \sigma(z^f_{ij}) \\ 0, & \text{otherwise} \end{cases} \\
&= \chi_{t_0}(\sigma(z^f_{ij}))\ .
\end{aligned}$$

Since u_0 and $\chi_{t_0} \circ \sigma$ agree on Z_f and both are B-homomorphisms, they agree on C_f. Thus

$$u_0(C_f) \;=\; \chi_{t_0}(\sigma(C_f)) \;=\; 1\ ,$$

since σ satisfies $C_f = 1$. Then

$$\begin{aligned}
u_0(C_f) = 1 \;&\Rightarrow\; C_f[z^f_{ij} := u_0(z^f_{ij})] = 1 \\
&\Rightarrow\; C_f[z^f_{ij} := u_i(x_j)] = 1 \\
&\Rightarrow\; C_f[u_0, u_1, \ldots, u_n] = 1 \\
&\Rightarrow\; (u_0, u_1, \ldots, u_n) \in E_f\ .
\end{aligned}$$

Since the u_i are in U', we also have

$$(u_0, u_1, \ldots, u_n) \in E'_f\ .$$

Thus H' is closed. □

Corollary 3 *The following two decision problems are efficiently interreducible:*

(i) Given a system $\mathcal{S}$ of set constraints, is it satisfiable?

(ii) Given a hypergraph H specified by Boolean formulas, does it have a closed induced subhypergraph?

6 Complexity Bounds

In this section we give complexity bounds for a hierarchy of satisfiability problems for systems of set constraints based on the arities of the elements of Σ. By Theorem 2, we are free to work with either the constraints $\mathcal{S}$ directly or the hypergraph H. It is usually easier to deal with H because it is a finite object, whereas in general $\mathcal{S}$ involves infinitely many terms and can have infinitely many solutions.

The results of this section are summarized in the following table.

number of elements in Σ of			*complexity of the*
arity 0	*arity 1*	*arity 2+*	*satisfiability problem*
0	0+	0+	trivial
1+	0	0	*NP*-complete
1+	1	0	*PSPACE*-complete
1+	2+	0	*EXPTIME*-complete
1+	0+	1+	*NEXPTIME*-complete

The first line of the above table is really a triviality, because with no constants in Σ, the set of ground terms T_Σ is empty. We handle each of the other cases separately.

6.1 Nullary Symbols

With at least one constant in Σ but no symbol of higher arity, we have $T_\Sigma = \Sigma = \Sigma_0$. In this case the satisfiability problem is *NP*-complete. The hypergraph H has only unary hyperedge relations, and the closure problem amounts to determining whether for each $c \in \Sigma$ there exists a truth assignment u satisfying both B and C_f (in the sense that $u(B) = 1$ and $C_c[u] = 1$). This is essentially Boolean satisfiability.

6.2 One Unary Symbol

6.2.1 *PSPACE* Upper Bound

Suppose Σ consists of one unary symbol f, one or more constants, and no other symbols. Then the hypergraph H is simply a conventional directed graph with binary edge relation E_f and a distinguised subset E_c for each constant c. The vertices U are the truth assignments satisfying a Boolean formula B, the edge relation E_f is the set of pairs $(u,v) \in U^2$ such that $C_f[u,v] = 1$, and the distinguished subset E_c of U is the set of u such that $C_c[u] = 1$. In this case, the closure problem is to determine whether there is a subset U' of U such that

- each E_c intersects U'
- for any $v \in U'$ there is a $u \in U'$ such that $(u,v) \in E_f$.

This can be determined nondeterministically in linear space as follows. For each constant c in turn, guess $u \in E_c$ and verify that $E_c[u] = u(B) = 1$. Starting from u, guess an E_f-path of length $2^m + 1$, where $m = |X|$. At each step, verify the new vertex v by evaluating $v(B)$ and the new edge (v,w) by evaluating $C_f[v,w]$. If such a path is found for all c, accept.

Since $|U| \leq 2^m$, any E_f-path of length $2^m + 1$ must repeat a vertex. Thus the procedure accepts iff for every nullary $c \in \Sigma$ there is an E_f-path $u_0, u_1, \ldots, u_k$ such that $u_0 \in E_c$ and $u_k = u_j$ for some $j < k$. The induced hypergraph on the set of all such u_i for all c is closed. Conversely, any closed induced subhypergraph must contain an induced subhypergraph of this form.

This procedure requires only linear space to record the current vertex, m bits for a counter, and enough space to evaluate the formulas. Thus the algorithm runs in nondeterministic *PSPACE*, which is the same as deterministic *PSPACE* by Savitch's Theorem [11].

6.2.2 *PSPACE* Lower Bound

We show that the problem of deciding whether a system with one unary symbol f and one constant c is satisfiable is *PSPACE*-hard by a reduction from the halting problem for linear-bounded automata (LBA), a well known *PSPACE*-complete problem. Consider an LBA M and input string w of length n. A *configuration* of M on input w is an instantaneous description of M's current tape contents, state, and head position. Each legal configuration is represented as a string of symbols of length n over a finite alphabet. Assume without loss of generality that there is a unique accept configuration and a unique reject configuration on inputs of length n, that all computation paths of M halt and either accept or reject (equip M with a binary exponential-time counter if necessary), that the accept configuration enters a trivial loop, and that the reject configuration has no successor.

The configurations of M on w can be encoded as truth assignments to Boolean variables

$$\begin{aligned} A_j^a &= \text{“symbol } a \text{ is written on tape cell } j\text{”}, \quad 1 \leq j \leq n, \ a \in \Gamma \\ Q_j^q &= \text{“}M \text{ is in state } q \text{ scanning tape cell } j\text{”}, \quad 1 \leq j \leq n, \ q \in Q \ . \end{aligned}$$

These variables comprise the set X. One can write down short formulas describing the action of M on input w:

- a formula B describing all legal configurations (exactly one symbol occupying each tape cell, exactly one current state, exactly one tape cell currently being scanned);
- a formula C_c describing the start configuration of M on input w (the symbol occupying tape cell j is the j^{th} symbol of w and M is in state s scanning the leftmost tape cell);
- a formula $C_f[u, v]$ describing legal pairs of configurations such that u follows from v in one step according to the transition rules of M.

The encoding technique is similar to that used in the proof of Cook's Theorem; see [11] for details. Then M accepts input w if and only if the hypergraph specified by B, C_c, and C_f has a closed induced subhypergraph consisting of the configurations in the accepting computation path.

6.3 Two or More Unary Symbols

6.3.1 *EXPTIME* Upper Bound

With two or more unary symbols, one or more constant symbols, and no other symbols, consider the following deterministic exponential-time algorithm for determining whether the specified hypergraph has a closed induced subhypergraph. Write down all truth assignments to X and delete those not satisfying B. For each remaining u, check whether it has an E_f-successor for all unary f and delete it if not (inductively, such a u cannot be contained in any closed subhypergraph). Repeat until no more vertices are deleted. The resulting subhypergraph contains all closed subhypergraphs, and is closed itself if for each nullary c there is a $u \in E_c$. There are at most 2^m truth assignments, and the tests can be done efficiently by evaluating $C_f[u,v]$ and $C_c[u]$.

6.3.2 *EXPTIME* Lower Bound

The exponential time lower bound for two unary symbols is obtained by generalizing the lower bound construction for one unary symbol. Instead of a deterministic LBA, we encode an alternating LBA M on input w [7]. We assume without loss of generality that M has no negating transitions, that there is a unique accept and a unique reject configuration for inputs of length n, that M alternates strictly between universal and existential branches, that all branches are at most binary, that the start, accept, and reject configurations are universal branches, that all computation paths either accept or reject, that the unique accept configuration enters a trivial loop, and that the unique reject configuration has no successor. We will construct a hypergraph that has a closed induced subhypergraph iff M accepts w.

Let α be a universal configuration with successors $\alpha 0$ and $\alpha 1$ in lexicographical order. By assumption, $\alpha 0$ and $\alpha 1$ are existential configurations. Let $\alpha 00$ and $\alpha 01$ be the two successors of $\alpha 0$ and let $\alpha 10$ and $\alpha 11$ be the two successors of $\alpha 1$ in lexicographical order. Then $\alpha 00$, $\alpha 01$, $\alpha 10$, and $\alpha 11$ are universal configurations.

As in Section 6.2.2, we let B and C_c be Boolean formulas describing the set of all legal configurations and the start configuration, respectively. In addition, we let C_f describe the relation consisting of all pairs $(\alpha, \alpha 00)$ and $(\alpha, \alpha 01)$, and we let C_g describe the relation consisting of all pairs $(\alpha, \alpha 10)$ and $(\alpha, \alpha 11)$. The idea is that in the semantics of alternating Turing machines, α leads to acceptance iff both $\alpha 0$ and $\alpha 1$ lead to acceptance, which occurs iff at least one of $\alpha 00$ or $\alpha 01$ leads to acceptance and at least one of $\alpha 10$ or $\alpha 11$ lead to acceptance. Thus M accepts the input w iff there is a closed induced subhypergraph consisting of an accepting computation tree of M.

6.4 One Binary Symbol

6.4.1 *NEXPTIME* Upper Bound

With any number of symbols of any arity, we can determine in nondeterministic exponential time whether there exists a closed induced subhypergraph by guessing a subset $U' \subseteq U$ and verifying that the induced subhypergraph on U' is closed; *i.e.*, for all $u \in U'$, $u(B) = 1$ and for all $f \in \Sigma_n$ and $u_1, \ldots, u_n \in U'$, there exists a $u_0 \in E_f(u_1, \ldots, u_n) \cap U'$. The set U has at most 2^m elements, where m is the number of variables, and the predicates $C_f[u_0, \ldots, u_n]$ require polynomial time to evaluate. The entire algorithm runs in nondeterministic exponential time.

6.4.2 *NEXPTIME* Lower Bound

We show first that with a constant c and one ternary symbol f, we can encode computations of a nondeterministic exponential-time Turing machine. In Section 6.5 we show later how to reduce f to binary. Let M be such a machine with time and space bound $N = 2^{O(n)}$. Without loss of generality, assume that M starts in its start state scanning the left endmarker $\vdash$, that it has unique accept and reject configurations on inputs of length n and that all computation paths lead to acceptance or rejection, that all nondeterministic branches are at most binary, that once M accepts or rejects it enters a trivial loop in which it remains in the same state.

Computation histories of M on inputs of length n can be represented as $N \times N$ matrices. Each row i of the matrix encodes a possible configuration of M at time i. The ij^{th} entry of the matrix records the symbol occupying the j^{th} tape cell at time i and whether that cell is being scanned by the machine at time i. If so, the current state of the finite control is also recorded. An accepting computation history of M on input w is represented by a matrix whose first row encodes the start configuration of M on input w, whose $i+1^{\text{st}}$ row follows from the i^{th} by the transition rules of M, and whose final row $N-1$ encodes the accept configuration.

Given M and input $w = w_1 \ldots w_n$, we construct Boolean formulas B, C_c and C_f specifying a hypergraph

$$H = (U,\ E_c,\ E_f)$$

where c is nullary and f is ternary (thus E_c is a unary and E_f is quaternary).

Each entry of the matrix is represented by a vertex of the hypergraph, which is a truth assignment satisfying B. The hyperedge relation E_f enforces constraints between adjacent entries. The hypergraph has a closed induced subhypergraph if and only if there exists a matrix representing an accepting computation history of M.

We first define the set of Boolean variables X. There is a variable A_a for each symbol a of the tape alphabet, a variable Q_q for each machine state q, $m = \lceil \log_2 N \rceil$ variables $t_0, \ldots, t_{m-1}$ encoding the time (row number of the matrix) in binary, m variables $s_0, \ldots, s_{m-1}$ encoding the position of the tape cell (column of the matrix) in binary, and a variable *choice* determining the nondeterministic choice.

The vertices of the hypergraph are the truth assignments to X such that at most one state and exactly one tape symbol have Boolean value 1. This is specified by the formula

$$\begin{aligned} B \;&=\; (\bigcup_a A_a) \;\cap\; \bigcap_{a \neq b} ({\sim}A_a \cup {\sim}A_b) \\ &\quad\;\cap\; \bigcap_{p \neq q} ({\sim}Q_p \cup {\sim}Q_q) \end{aligned}$$

Each truth assignment u to X corresponds to an index ij into the array, $0 \leq i, j \leq N-1$, where i and j are the numbers encoded in binary by the truth values $u(t_k)$ and $u(s_k)$, $0 \leq k \leq m-1$. We denote i and j by $\mathit{time}(u)$ and $\mathit{space}(u)$, respectively. Also, each u satisfying B has exactly one symbol a with $u(A_a) = 1$, which we denote by $\mathit{sym}(u)$. Finally, each u satisfying B has at most one state q with $u(Q_q) = 1$, which we denote by $\mathit{state}(u)$. If $u(Q_q) = 0$ for all states q, we write $\mathit{state}(u) = \perp$.

We will need to perform modular addition and comparisons on numbers in the range $0 \leq i \leq N-1$ in terms of their binary representations. For example, we need formulas expressing conditions such as $\mathit{time}(u) = \mathit{time}(v) + 1$ and $\mathit{time}(u) > \mathit{time}(v)$. These constructions are quite easy and well known, so we omit the details.

The unary predicate C_c specifies the 00^{th} entry (upper left corner) of the matrix. It specifies that the machine is in its start state s scanning the leftmost tape cell, which contains the left endmarker $\vdash$:

$$C_c[u] \;=\; (\mathit{time}(u) = 0) \;\cap\; (\mathit{space}(u) = 0) \;\cap\; Q_s \;\cap\; A_{\vdash}\;.$$

There are exactly two truth assignments satisfying B and C_c, one for each value of *choice*, and one of these must be contained in any closed subhypergraph.

The quarternary predicate $C_f[u, v, w, x]$ serves several purposes. It is defined as the conjunction of several formulas describing the format of configurations, the initial configuration (first row of the matrix), the final configuration (last row of the matrix), and the legal transitions.

First we specify that there is at most one truth assignment for every ij:

$$(time(w) = time(x) \ \cap \ space(w) = space(x)) \ \Rightarrow \ w = x \ . \tag{7}$$

Inclusion of this formula as a conjunct of $C_f[u, v, w, x]$ guarantees that there can be no closed induced subhypergraph containing two distinct vertices w and x such that $time(w) = time(x)$ and $space(w) = space(x)$.

We also wish to specify that for every i, the value of the variable *choice* at all tape cells in row i of the matrix is the same:

$$time(w) = time(x) \ \Rightarrow \ (w(choice) = x(choice)) \ . \tag{8}$$

To specify the initial configuration, we must ensure that the first n tape cells after the left endmarker contain the input string $w = w_1 \cdots w_n$, that all remaining cells to their right except the last contain the blank symbol $\natural$, the last tape cell contains the right endmarker $\dashv$, and no other cell besides the leftmost contains a state:

$$\begin{aligned}
&(v = w = x \ \cap \ time(v) = 0 \ \cap \ space(v) < N - 1) \\
&\quad \Rightarrow \ (time(u) = 0 \ \cap \ space(u) = space(v) + 1 \ \cap \ state(u) = \bot \\
&\qquad \cap \ \bigcap_{i=1}^{n} (space(u) = i \ \Rightarrow \ sym(u) = w_i) \\
&\qquad \cap \ n < space(u) < N - 1 \ \Rightarrow \ sym(u) = \natural \\
&\qquad \cap \ space(u) = N - 1 \ \Rightarrow \ sym(u) = \dashv) \ .
\end{aligned} \tag{9}$$

If the premise of (9) is true of v, w and x, then there is exactly one choice of u that satisfies the conclusion. The two truth assignments satisfying C_c satisfy the premise of (9), and it can be shown inductively that any closed subhypergraph must contain the entire first row of the matrix representing the initial configuration.

To capture valid transitions of the machine, we write

$$\begin{aligned}
&(time(v) = time(w) = time(x) < N - 1 \\
&\cap\ space(x) = space(w) + 1 = space(v) + 2) \\
&\quad \Rightarrow\ \ (time(u) = time(w) + 1\ \cap\ space(u) = space(w) \\
&\qquad\quad \cap\ (sym(u), state(u)) = \\
&\qquad\quad next(choice(v), sym(v), state(v), sym(w), state(w), sym(x), state(x)))
\end{aligned} \tag{10}$$

where the function *next* encodes the transition relation of M. The nondeterministic choice is determined by the value of the variable *choice*. Addition in this expression is modulo N. We are using the fact that the state and symbol at time $i+1$ and position j depends only on the state and symbol at time i and positions $j-1$, j, and $j+1$. The function *next* also encodes the fact that if the machine is not scanning tape cell j at time i, then the symbol on tape cell j is unchanged at time $i+1$.

By (9), any closed subhypergraph contains the start configuration of the computation. Inductively, assume any closed subhypergraph contains the first i configurations. By (8), entries in configuration i must agree on the value of the variable *choice*. Furthermore, given any v, w, and x satisfying the premise of (10), there are exactly two u satisfying the conclusion of (10) with different values of *choice* but otherwise identical. One of these must be in any closed subhypergraph.

Finally, we need to check that the accept state occurs someplace in the last row of the matrix. Since the machine has either accepted or rejected by time $N-1$, and since we have already insured that the matrix accurately encodes a computation history, we need only check that the reject state r does not occur in the last row. We use the formula

$$time(x) = N - 1 \ \Rightarrow\ state(x) \neq r\ . \tag{11}$$

Finally, we define $C_f[u, v, w, x]$ to be the conjunction of (7), (8), (9), (10), and (11).

We have argued that the problem of deciding whether a given system of constraints with one ternary and one nullary symbol is satisfiable is *NEXPTIME*-complete. It will follow from the result of the next section that the problem with one nullary and one binary symbol is also *NEXPTIME*-complete.

6.5 Symbols of Greater Arity

In this section we show that any system with symbols of arbitrary arity can be reduced to a system with a single binary symbol and a single constant. By the results of Section 6.4, it suffices to prove this result for a signature with one ternary symbol and one constant. This will establish the *NEXPTIME*-completeness of the satisfiability problem for systems with at least one constant and at least one symbol of arity two or greater.

Let $\Gamma = \{g, b\}$ and let $\Sigma = \{f, a\}$, where a and b are constants, f is binary, and g is ternary. Let B, C_b, and C_g be formulas describing a hypergraph

$$H = (U,\ E_b,\ E_g)\ ,$$

where E_b is unary and E_g is quaternary. Define a new hypergraph

$$\widehat{H} = (\widehat{U},\ E_a,\ E_f)$$

specified by formulas $\widehat{B}$, C_f, and C_a, as follows. The vertices of $\widehat{H}$ are

$$\widehat{U} = U \cup (U \times U)\ .$$

Elements of $U \times U$ are denoted $\langle u, v\rangle$. If X is the set of variables used in the definition of H such that elements of U are truth assignments to X, then we can take $\widehat{U}$ to be a set of truth assignments to $X \cup X' \cup \{z\}$, where X' is a disjoint copy of X and z is a new variable whose sole purpose is to distinguish between U and $U \times U$. We define $\widehat{B}$, C_f, and C_a such that

$$\begin{aligned}
\widehat{B}[u] &= B[u] \\
\widehat{B}[\langle u, v]\rangle &= B[u] \cap B[v] \\
C_a[u] &= C_b[u] \\
C_a[\langle u, v\rangle] &= 0 \\
C_f[\langle v, w\rangle, v, w] &= 1 \\
C_f[u, v, w] &= 0 \\
C_f[u, \langle v, w\rangle, x] &= C_g[u, v, w, x] \\
C_f[\langle t, u\rangle, \langle v, w\rangle, x] &= 0
\end{aligned}$$

and for $q, r \in \widehat{U}$ not covered by any of the preceding cases,

$$\begin{aligned}
C_f[q, q, r] &= 1 \\
C_f[p, q, r] &= 0\ ,\quad p \neq q\ .
\end{aligned}$$

These formulas are easily derived from B, C_b, and C_g. Then

$$\begin{aligned} E_f &= \{(\langle u,v\rangle, u, v) \mid u,v \in U\} \\ &\cup \{(u, \langle v,w\rangle, x) \mid (u,v,w,x) \in E_g\} \\ &\cup \{(q,q,r) \mid (q,r) \text{ not of the form } (u,v) \text{ or } (\langle v,w\rangle, x)\}\ . \end{aligned}$$

It is easily shown that the induced subhypergraph of H on vertices U' is closed if and only if the induced subhypergraph of $\widehat{H}$ on vertices $U' \cup (U' \times U')$ is closed.

7 Future Work

We would like to extend these techniques to *projection functions.* For every symbol $f \in \Sigma_n$, one can define a family of projection functions $f^{-1}, \ldots, f^{-n}$ with semantics

$$\sigma(f^{-i}(E)) \quad = \quad \{t_i \mid ft_1 \ldots t_n \in \sigma(E)\}$$

Algorithms for solving special cases are known [16, 12, 9], but substantial restrictions must be placed on the other set operations. Projection functions subsume negative constraints because the constraint system

$$\mathcal{S} \cup \{b \subseteq f^{-1}(fby)\}$$

is satisfiable only if $\mathcal{S}$ is satisfiable with $y \neq 0$. The result of [1] on negative constraints is presumably a step towards solving systems with projections.

Acknowledgements

Dexter Kozen was supported by the Danish Research Academy, the National Science Foundation, the John Simon Guggenheim Foundation, 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. Decidability of systems of set constraints with negative constraints. manuscript, January 1993.

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

[3] A. Aiken and B. Murphy. Static type inference in a dynamically typed language. In *Proc. 18th Symp. Principles of Programming Languages*, pages 279–290. ACM, January 1991.

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

[5] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Set constraints are the monadic class. In *Proc. 8th Symp. Logic in Computer Science*, page to appear. IEEE, June 1993.

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

[7] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *J. Assoc. Comput. Mach.*, 28(1):114–133, 1981.

[8] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints using tree automata. manuscript, July 1992.

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

[10] N. Heintze and J. Jaffar. A finite presentation theorem for approximating logic programs. In *Proc. 17th Symp. Principles of Programming Languages*, pages 197–209. ACM, January 1990.

[11] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

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

[13] K. Marriott and M. Odersky. Systems of negative boolean constraints. Technical Report YALEU/DCS/RR-900, Computer Science Department, Yale University, April 1992.

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

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

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

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