MAX-PLANCK-INSTITUT FÜR INFORMATIK

Set Constraints are the Monadic Class

Leo Bachmair
Harald Ganzinger
Uwe Waldmann

MPI–I–92–240 December 1992

INFORMATIK

Im Stadtwald
W 6600 Saarbrücken
Germany

Authors' Addresses

Leo Bachmair
Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, U.S.A, leo@cs.sunysb.edu
Harald Ganzinger, Uwe Waldmann
Max-Planck-Institut für Informatik, Im Stadtwald, D-W-6600 Saarbrücken, Germany, {hg,uwe}@mpi-sb.mpg.de

Publication Notes

This paper has been submitted for publication elsewhere and will be copyrighted if accepted.

Acknowledgements

The research described in this paper was supported in part by the German Science Foundation (Deutsche Forschungsgemeinschaft) under grant Ga 261/4-1, by the German Ministery for Research and Technology (Bundesministerium für Forschung und Technologie) under grant ITS 9102/ITS 9103 and by the ESPRIT Basic Research Working Group 6028 (Construction of Computational Logics). The responsibility for the contents of this publication lies with the authors.

Abstract

We investigate the relationship between set constraints and the monadic class of first-order formulas and show that set constraints are essentially equivalent to the monadic class. From this equivalence we can infer that the satisfiability problem for set constraints is complete for NEXPTIME. More precisely, we prove that this problem has a lower bound of NTIME($c^{n/\log n}$). The relationship between set constraints and the monadic class also gives us decidability and complexity results for certain practically useful extensions of set constraints, in particular "negative projections" and subterm equality tests.

Keywords

Constraint Solving, Set Constraints, Decidable First-Order Theories, Monadic Class, Program Analysis

1 Introduction

Set constraints describe relationships between sets of terms over some vocabulary. They arise naturally when one abstracts from concrete values of program variables in program analysis and type inference algorithms, cf. Aiken and Murphy (1991a), Aiken and Murphy (1991b), Heintze and Jaffar (1990), Heintze and Jaffar (1991), Jones and Muchnick (1979), Mishra (1984), Mishra and Reddy (1985), Reynolds (1969), Young and O'Keefe (1988), among others. Algorithms for solving certain classes of set constraints have been given in various settings, cf. section 4. Some of them are rather ad hoc or involve complicated codings.

The purpose of this paper is to show that known results about decidable fragments of first-order logic can be directly applied to set constraints. In particular, set constraints (with projections for unary functions) are, via certain natural translations, equivalent to the monadic class, for which decidability and complexity results are available. The satisfiability of set constraints can be reduced to the satisfiability of monadic formulas via length order n^2, conversely, the satisfiability of monadic formulas can be reduced to the satisfiability of set constraints via length order $n^2/\log n$. As a consequence, the satisfiability of set constraints is complete for NEXPTIME, a result that was left open in (Aiken and Wimmers 1992). More precisely, we establish NTIME($c^{n/\log n}$), for some $c > 0$, as a lower bound for the problem. The relationship between set constraints and the monadic class allows us to extend set constraints by diagonalization (i.e., *equality tests for subterms*) and by projections. By applying known results for the monadic class with equality we show that satisfiability of the extended constraints remains decidable (and, more specifically, again complete for NEXPTIME), provided projections for non-unary functions occur with *negative* polarity only. In applications to program analysis and type inference, where projections are useful, positive projections are not needed anyway, cf. (Heintze and Jaffar 1990). Unlike in the latter paper our class of constraints with projections is not restricted to definite constraints and, therefore, can also be applied to the analysis of and type inference for *disjunctive* logic programs or to other kinds of *nondeterministic* programming languages. Moreover, set constraints with equality allow one to specify some amount of sharing between variables, which results in more precise compile-time analyses of programs. The problem of set constraints with unrestricted occurrences of projections remains unsolved at this time.

In short, in this paper we demonstrate that (i) set constraints are essentially equivalent to the monadic class and (ii) decidability and complexity results for the monadic class carry over to set constraints. Moreover, our method of relating set constraints to the monadic class allows us (iii) to use standard theorem proving techniques based on ordered resolution and/or superposition as decision procedures, and (iv) to extend set constraints in several practically useful ways.

2 The Monadic Class

The *monadic class* is the class of first-order formulas without function symbols, with unary (monadic) predicates only, but with arbitrary quantification. We speak of the *monadic class with equality* if, in addition, equations (between variables) are allowed in formulas.

If we *skolemize* a monadic formula in prenex form, the resulting quantifier-free formula can be characterized by the following syntactic properties: (i) all predicate symbols are unary; (ii) there exists a sequence $x_1, \ldots, x_m$ of variables such that all atoms are of the form $p(t)$, where p is a predicate and t is either a variable x_n, or a term $f(x_1, \ldots, x_n)$, for some $n \leq m$. In the following we call such formulas *flat* formulas over given vocabularies $\mathcal{F}$ and $\mathcal{P}$, respectively, of functions symbols and monadic predicate symbols. In the case of the monadic class with equality, atoms in the skolemized formulas may also be (iii) equations $s \approx t$, where s and t are terms of the form described in (ii) above. For example, the monadic formula with equality

$$\exists a \forall x \exists f \forall y \exists g \ (p(x) \wedge q(y) \rightarrow a \approx g \vee f \approx y)$$

skolemizes into the flat formula

$$p(x) \wedge q(y) \rightarrow a \approx g(x, y) \vee f(x) \approx y.$$

The monadic class has been extensively studied. Löwenheim (1915) was the first to prove the decidability of validity and satisfiability, not only for the case with equality but also for quantification over predicates. The proof by Ackermann (1954) of the same result is much simpler and, due to his syntactic method of transforming the given formula into some kind of solved form, appears to be usable in practice. In particular Ackermann employs a form of resolution with lazy unification, in which unification between terms is actually represented as an equational constraint in the resolvent.[1] Joyner Jr. (1976) and others[2] have shown that ordered resolution, which is known to be refutationally complete for arbitrary first-order theories, can be equipped with special simplification techniques so that it always terminates on flat clauses and therefore yields a decision method for *Monadic-Sat*, the satisfiability problem for the monadic class *without* equality and without second-order quantifiers. In this spirit, Bachmair, Ganzinger, and Waldmann (1992a) have shown that superposition with simplification is a decision method for the case with equality, referred to as *Monadic-E-Sat* below.[3]

Lewis (1980) has shown that for some $c > 0$, NTIME($c^{n/\log n}$) is an upper bound for the complexity of Monadic-Sat, where n is the length of the formula.

[1] Lazy unification seems to arise naturally when trying to eliminate second-order quantifiers, cf. (Bachmair, Ganzinger, and Waldmann 1992b or Gabbay and Ohlbach 1992).

[2] For an overview and a more recent treatment of the problem see (Fermüller et al. 1992).

[3] In a certain sense, this extends the results of Comon, Haberstrau, and Jouannaud (1992) about shallow equational theories to the first-order case. It is, however, not possible to extend their result in full generality. First-order clauses over flat equations, if no restrictions such as in (ii) apply, form a reduction class; any non-flat equation can then be flattened with the help of auxiliary variables.

His proof is based on the finite model property of the monadic class and checks finite structures up to cardinality 2^k, where $k = O(n/\log n)$ is the number of predicate symbols, for the model property. Although he shows that this algorithm is in some sense optimal — NTIME($c^{n/\log n}$), for some (other) $c > 0$, is at the same time a lower bound for this problem — the proof-theoretic methods of Ackermann (1954), Joyner Jr. (1976) or Fermüller et al. (1992) appear to be superior in practice. Lewis (1980) obtains this lower bound by showing that NETIME is polynomially reducible, via length order $n \log n$, to Monadic-Sat.[4] Altogether this means in particular that Monadic-Sat is complete for NEXPTIME. Looking at these proofs it is easy to see that Monadic-E-Sat is also NEXPTIME complete. For the upper bound one may use the fact that a monadic formula with equality has a model if and only if it has a model of cardinality less or equal $2^k m$, where m is the length of the quantifier prefix and k is the number of predicates (cf. e.g. Dreben and Goldfarb 1979).

3 Set Constraints

3.1 Set Constraints as Flat Formulas

A *set constraint* is a finite conjunction of subset relations $E \subseteq E'$, where E and E' are set expressions over a given finite vocabulary $\mathcal{F}$ of function symbols and $\mathcal{V}$ of set-valued variables. *Set expressions* are defined by the grammar

$$E ::= 0 \mid 1 \mid \alpha \mid E \cup E \mid E \cap E \mid \overline{E} \mid f(E_1, \ldots, E_n)$$

where α may be any variable in $\mathcal{V}$ and f any n-place function symbol in $\mathcal{F}$. Semantically, the variables in $\mathcal{V}$ are assumed to range over sets of finite ground terms over the function symbols in $\mathcal{F}$. A set constraint is said to be *satisfiable* if sets of ground terms over $\mathcal{F}$ can be assigned to the variables in such a way that the constraint evaluates to true, whereby 0 denotes the empty set and 1 the set of all ground terms; a set expression $f(E_1, \ldots, E_n)$ denotes the set of terms $\{ f(t_1, \ldots, t_n) \mid t_i \in E_i \}$; and $E \subseteq E'$, $E \cap E'$, $E \cup E'$, and $\overline{E}$ denote the subset relation, intersection, union, and complement, respectively, on sets E and E'.

Aiken and Wimmers (1992) proved that the satisfiability of set constraints is decidable by providing a specific set of transformation rules for constraints into a certain kind of "solved forms." However it turns out that this problem is a special case of the satisfiability problem for flat formulas and, hence, of Monadic-Sat.

This can be seen by transforming a given constraint into an equivalent set of flat formulas over $\mathcal{F}$. The set denoted by a set expression E can be represented by a monadic formula $P_E(x)$ which codes the fact "x is in E." This coding will be established by induction over the syntactic structure of constraints and set expressions. More precisely, for every set expression E which is a subexpression of the given constraint we introduce a monadic predicate P_E.

[4] We assume definitions of complexity classes as in (Johnson 1990).

These predicates are defined by the following equivalences which refer to the predicates representing the subexpressions of E.

$$\begin{array}{rcl} P_1(x) & \leftrightarrow & true \\ P_0(x) & \leftrightarrow & false \\ P_{E\cup F}(x) & \leftrightarrow & P_E(x) \vee P_F(x) \\ P_{E\cap F}(x) & \leftrightarrow & P_E(x) \wedge P_F(x) \\ P_{\overline{E}}(x) & \leftrightarrow & \neg P_E(x) \\ P_{f(E_1,\ldots,E_n)}(f(x_1,\ldots,x_n)) & \leftrightarrow & P_{E_1}(x_1) \wedge \ldots \wedge P_{E_n}(x_n) \\ P_{f(E_1,\ldots,E_n)}(g(x_1,\ldots,x_m)) & \leftrightarrow & false\,, \end{array}$$

where the x_i are pairwise distinct (first-order) variables, and where an equivalence of the last form is generated for every m-ary function symbol g different from f. It can be seen that $P_{f(E_1,\ldots,E_n)}$ is defined by as many equivalences as there are function symbols in the vocabulary. All other predicates are defined by a single equivalence. The reader may observe that these equivalences are in fact all flat.

If $K = E_1 \subseteq F_1 \wedge \ldots \wedge E_m \subseteq F_m$, then let $[\![K]\!]$ be the set of all equivalences for the subexpressions in E_i and F_i, together with the additional formulas $P_{E_i}(x) \rightarrow P_{F_i}(x)$, for $1 \leq i \leq m$, representing the subset relations.

As an example consider the constraint

$$\alpha_1 \subseteq \alpha_2 \wedge f(\alpha_2) \subseteq \overline{\alpha_2} \wedge f(\overline{\alpha_2}) \subseteq \alpha_2$$

over a vocabulary that consists of a unary function symbol f and a constant a. Here $[\![K]\!]$ is the set

$$\begin{array}{rcl} P_{\alpha_1}(x) & \rightarrow & P_{\alpha_2}(x) \\ P_{f(\alpha_2)}(x) & \rightarrow & P_{\overline{\alpha_2}}(x) \\ P_{f(\overline{\alpha_2})}(x) & \rightarrow & P_{\alpha_2}(x) \\ P_{\overline{\alpha_2}}(x) & \leftrightarrow & \neg P_{\alpha_2}(x) \\ P_{f(\alpha_2)}(f(x)) & \leftrightarrow & P_{\alpha_2}(x) \\ P_{f(\alpha_2)}(a) & \leftrightarrow & false \\ P_{f(\overline{\alpha_2})}(f(x)) & \leftrightarrow & P_{\overline{\alpha_2}}(x) \\ P_{f(\overline{\alpha_2})}(a) & \leftrightarrow & false\,. \end{array}$$

This translation of constraints into equivalences is inspired by the method of Tseitin (1970) for transforming quantifier-free first-order formulas to clausal normal form in a way that avoids the exponential growth of formulas caused by more naive methods. In the algorithm by Aiken and Wimmers (1992) an instance of the same method appears as what they call transformation of constraints into "one-level systems." Clearly $[\![K]\!]$ is a set of flat formulas, for any given K, which may be viewed as the result of skolemizing a (unique up to renaming of variables) monadic formula $[\![K]\!]_m$ in prenex form.

Theorem 1 *Let I be a Herbrand interpretation over $\mathcal{F}$. Then I is a model of $[\![K]\!]$ if and only if K is satisfied under the assignment where each set variable α in K is assigned the set of ground terms $\{\, t \in T_{\mathcal{F}} \mid P_\alpha(t) \in I \,\}$. In particular, K is satisfiable if and only if $[\![K]\!]_m$ is satisfiable.*

Corollary 1 *The problem Setc-Sat of satisfiability of set constraints is decidable.*

The proof follows from the decidability of Monadic-Sat using the preceding theorem.

A set constraint K of length n contains $\mathrm{O}(n)$ (occurrences of) subexpressions and $k = \mathrm{O}(n/\log n)$ distinct function symbols. Every occurrence of a subexpression E of K corresponds to at most one occurrence of P_E on the right hand side and at most k occurrences of P_E on the left hand side of an equivalence in $[\![K]\!]$. Hence $[\![K]\!]$ (and thus $[\![K]\!]_m$) contains $\mathrm{O}(n^2/\log n)$ atoms. As every predicate symbol or variable in the monadic formula $[\![K]\!]_m$ can be coded in $\mathrm{O}(\log n)$ space, $[\![K]\!]_m$ has length $\mathrm{O}(n^2)$. Using the upper bound of Lewis (1980), this gives us the following theorem:

Theorem 2 *Setc-Sat can be reduced to Monadic-Sat via length order n^2. Satisfiability of set constraints can hence be decided in* $\mathrm{NTIME}(c^{n^2/\log n})$*, for some constant $c > 0$.*

In comparison to what has been obtained by Aiken and Wimmers (1992), this theorem gives us a precise upper bound of the problem within NEXPTIME with a less than quadratic exponent of $n^2/\log n$.

We also can prove that Setc-Sat is NEXPTIME-hard, more precisely we have the theorem:

Theorem 3 $\mathrm{NTIME}(c^n)$ *can be reduced to Setc-Sat via length order $n \log n$.*

To prove this theorem, we first prove a lemma which shows how to translate a certain class of flat formulas into set constraints via length order n.

Lemma 1 *Let Γ be a flat formula of length n of the form*

$$\Phi_1(x) \wedge \ldots \wedge \Phi_n(x) \wedge \Psi_1(x,y) \wedge \ldots \wedge \Psi_k(x,y)$$

where the Φ_i are flat clauses over a single variable x, constant a and unary function f, and where the Ψ_j are quantifier- and function-free formulas over the variables x and y. Then there exists a set constraint K of length $\mathrm{O}(n)$ *such that K is satisfiable if and only if $\forall x \forall y\ \Gamma$ is satisfiable. The constraint K can be constructed from Γ in polynomial time.*

Proof. In order to replace the formulas Ψ_i by equivalent constraints we introduce an auxiliary binary function symbol g. The original f-terms are distinguished by the set constraint $N = a \cup f(N)$, together with constraints $p \subseteq N$, for any predicate symbol p in Γ. The Ψ_i can now be replaced by constraints $g(N,N) \subseteq [\Psi_i]$, where $[\Psi_i]$ is defined inductively over the syntactic structure by the identities

$$\begin{array}{rcl} [\Psi \wedge \Psi'] & = & [\Psi] \cap [\Psi'] \\ [\Psi \vee \Psi'] & = & [\Psi] \cup [\Psi'] \\ [\neg \Psi] & = & \overline{[\Psi]} \cap g(N,N) \\ [p(x)] & = & g(p,N), \text{ for predicates } p \\ [p(y)] & = & g(N,p), \text{ for predicates } p. \end{array}$$

Any other boolean connectives are assumed to be defined in the usual way.

The clauses Φ_i take the form

$$L_1(x) \vee \ldots L_p(x) \vee M_1(f(x)) \vee \ldots \vee M_q(f(x)) \vee N_1(a) \vee \ldots \vee N_r(a),$$

where the L_i, M_i and N_i are predicates, possibly negated. An equivalent system of set constraints for such a formula is

$$\begin{aligned} f(\overline{L_1} \cap \ldots \cap \overline{L_p} \cap N) &\subseteq M_1 \cup \ldots \cup M_q \cup N_1^* \cup \ldots \cup N_r^* \\ N_i^* &= (N_i \cap a) \cup f(N_i^*), \text{ for } 1 \leq i \leq r. \end{aligned}$$

The conjunction of all these constraints is equivalent to Γ. The statements about the complexity are obviously satisfied with this construction. □

The proof of Theorem 3 now follows from (Lewis 1980) where the reduction of NTIME(c^n) to the monadic class via length order $n \log n$ only produces formulas Γ as required for the above lemma. In Section 3.3 we shall give a translation into set constraints for the whole class of monadic formulas in prenex form, which, however, will be of a quadratic length order and which will involve projections.

Corollary 2 *There exists a constant $c > 0$ such that Setc-Sat cannot be decided in* NTIME($c^{n/\log n}$).

Corollary 3 *Setc-Sat is NEXPTIME complete.*

3.2 Set Constraints with Equality

Set constraints have been proposed as a tool for the static analysis of programs. Extending set constraints by tests for equality of terms allows to better approximate programs in cases where the equality of two program variables is essential. Examples include non-linear heads of Prolog clauses or other kinds of equality tests on variables (for an example see Section 3.3). If we define the extension in such a way that we stay inside the monadic class with equality, the extended set constraints will enjoy basically the same properties as before.

Let Δ_Φ^f denote a family of operators on sets, called *diagonalization operators*, where f is an n-ary function symbol, with $n > 1$, and Φ a propositional formula over equations of form $i \approx j$, with $1 \leq i, j \leq n$. Then, for any set expression E, $\Delta_\Phi^f(E)$ is a set expression denoting the subset of all terms of the form $f(t_1, \ldots, t_n)$ in E, such that, in addition, the formula Φ is true when "$i \approx j$" is interpreted as "$t_i = t_j$."

The diagonalization operators can be represented by flat formulas (with equality):

$$\begin{aligned} P_{\Delta_\Phi^f(E)}(f(x_1, \ldots, x_n)) &\leftrightarrow P_E(f(x_1, \ldots, x_n)) \wedge \Phi[x_i/i;\, 1 \leq i \leq n] \\ P_{\Delta_\Phi^f(E)}(g(x_1, \ldots, x_m)) &\leftrightarrow \mathit{false}, \text{ for } g \neq f. \end{aligned}$$

Theorem 4 *The satisfiability of set constraints with diagonalization is decidable. More precisely, the problem is NEXPTIME complete.*

Let us point out that tree automata with tests for equality of subterms (Bogaert and Tison 1991) are a special case of set constraints with equality. It is also known that equality tests that allow two cousins in a tree to be compared immediately lead to unsatisfiability, even in the case of definite constraints.

3.3 Set Constraints with Projections

If one is interested in satisfiability only, the class of constraints that can be decided by translation into the monadic class can be enlarged by a more refined treatment of equivalences. Let N be a set of first-order clauses. Let furthermore F denote an equivalence $p(t_1, \ldots, t_n) \leftrightarrow \Phi$, where Φ is an arbitrary formula in which p does not occur. If p occurs in N only with a single polarity (positive or negative), then the satisfiability of $N \cup \{F\}$ is equivalent to the satisfiability of $N \cup \{F'\}$ where F' represents the appropriate direction of the equivalence F. Let $F_{\rightarrow}$ and $F_{\leftarrow}$ denote the orientations $p(t_1, \ldots, t_n) \rightarrow \Phi$ and $p(t_1, \ldots, t_n) \leftarrow \Phi$, respectively, of F.

Lemma 2 *Let N and F be as before.*

(i) If p occurs only in negative literals of N, then $N \cup \{F\}$ is satisfiable if and only if $N \cup \{F_{\leftarrow}\}$ is satisfiable. In particular the minimal models of $N \cup \{F\}$ coincide with the minimal models of $N \cup \{F_{\leftarrow}\}$.

(ii) If p occurs only in positive literals of N, then $N \cup \{F\}$ is satisfiable if and only if $N \cup \{F_{\rightarrow}\}$ is satisfiable. In particular the maximal models of $N \cup \{F\}$ coincide with the maximal models of $N \cup \{F_{\rightarrow}\}$.

Proof. We prove case (i), the other case being dual. If I is a model of $N \cup \{F\}$, it is clearly also a model of $N \cup \{F_{\leftarrow}\}$. If I is a model of $N \cup \{F_{\leftarrow}\}$, define I' from I by only changing the interpretation for p such that $p(t_1, \ldots, t_n)\sigma$ is true in I' if and only if $\Phi\sigma$ is true in I, where σ ranges over all ground substitutions for the free variables in F. As p does not occur positively in N and does not occur in Φ, I' is a model of $N \cup \{F\}$.

If I' is not a minimal model of $N \cup \{F\}$, that is, there exists a smaller model I'' of $N \cup \{F\}$, then I' and I'' cannot differ in the interpretation of p alone. In this case I'' is also a model of $N \cup \{F_{\leftarrow}\}$ which is smaller than I. □

Polarity of a subexpression E in a set constraint K corresponds to parity or disparity, respectively, of the nesting of complement operators around E. More precisely, let n be the number of complement operators enclosing a particular occurrence of E in K. Then this occurrence is called *positive*, if either n is even and the occurrence is in the right side of a subset relation, or else n is odd and the occurrence is in the left side of a subset relation. Otherwise the occurrence is called *negative*.

If E is a positive [negative] subexpression of K, we need to consider only those clauses that correspond to the $\rightarrow$-direction [$\leftarrow$-direction] of the equivalences defining P_E. (If an expression occurs in both polarities, both directions have to be taken.) By $[\![K]\!]_{\pm}$ we denote the resulting polarity-based clausal form of K.

For example, in $\alpha_1 \subseteq \alpha_2 \wedge f(\alpha_2) \subseteq \overline{\alpha_2} \wedge f(\overline{\alpha_2}) \subseteq \alpha_2$, both α_2 and $\overline{\alpha_2}$ occur positively and negatively, while $f(\alpha_2)$ and $f(\overline{\alpha_2})$ occur only negatively. Therefore $[\![K]\!]_\pm$ consists of the clauses

$$\begin{array}{rcl} P_{\alpha_1}(x) & \rightarrow & P_{\alpha_2}(x) \\ P_{f(\alpha_2)}(x) & \rightarrow & P_{\overline{\alpha_2}}(x) \\ P_{f(\overline{\alpha_2})}(x) & \rightarrow & P_{\alpha_2}(x) \\ P_{\overline{\alpha_2}}(x), P_{\alpha_2}(x) & \rightarrow & \\ & \rightarrow & P_{\overline{\alpha_2}}(x), P_{\alpha_2}(x) \\ P_{\alpha_2}(x) & \rightarrow & P_{f(\alpha_2)}(f(x)) \\ P_{\overline{\alpha_2}}(x) & \rightarrow & P_{f(\overline{\alpha_2})}(f(x))\,. \end{array}$$

Repeated application of Lemma 2 yields the following lemma:

Lemma 3 *A constraint K is satisfiable if and only if $[\![K]\!]_\pm$ is satisfiable.*

The optimized translation allows us to admit *projection functions* to a certain extent. If f is an n-ary function symbol, we call f^i, for $1 \leq i \leq n$, the i-th projection of f, and admit set expressions of the form $f^i(E)$, which are defined to denote the sets of all terms x_i for which there exist ground terms $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, such that $f(x_1, \ldots, x_n) \in E$. The denotation of a set expression $f^i(E)$ can be defined by the equivalence

$$P_{f^i(E)}(x_i) \quad \leftrightarrow \quad \exists\, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\ P_E(f(x_1, \ldots, x_n)).$$

If $n > 1$, the $\rightarrow$-direction of this equivalence is second-order as the existential quantifiers range over the ground terms of the given vocabulary. However, if $n = 1$ or $f^i(E)$ does not occur positively in K, then $[\![K]\!]_\pm$ is a set of (flat) clauses (without existential quantifiers). Let *SetcP-Sat* [*SetcEP-Sat*] denote the satisfiability problem for set constraints without [with] equality and without positive occurrences of projections for non-monadic function symbols.

Theorem 5 *SetcEP-Sat is decidable. More precisely, it is a NEXPTIME complete problem.*

Set constraints with projections of monadic function symbols allow the translation of arbitrary monadic formulas in prenex form.

Theorem 6 *Monadic-Sat can be reduced to SetcP-Sat via length order $n^2/\log n$.*

Proof. Let Φ be a monadic formula in prenex form, and let Φ' be its skolemized form. Assume that Φ' is a formula over the set of predicate symbols $\mathcal{P}$, the set of function symbols $\mathcal{F}$, and the variables $x_1, \ldots, x_k$. We will show how to compute a set constraint K_Φ over a signature $\hat{\mathcal{F}}$ that is solvable if and only if Φ' (and thus Φ) is solvable.

The new signature $\hat{\mathcal{F}}$ contains a constant e and a binary operator ":" (written in infix form), that are used as list constructors, and a unary operator $\hat{g}$

for every g in $\mathcal{F}$. We generally omit parentheses and abbreviate $A : (B : C)$ as $A : B : C$ to simplify notation. Furthermore we somewhat sloppily write $A^i : B$ for $A : A : \cdots : A : B$, and similarly $A^0 : B$ for B. The set of $\mathcal{F}$-terms is mapped into the set of $\hat{\mathcal{F}}$-terms via a function τ that is defined inductively by $\tau(g(t_1, \ldots, t_i)) = \hat{g}(\tau(t_i) : \cdots : \tau(t_1) : e)$. The function τ is injective, its range, denoted by T, is the only solution of the set equation

$$T = \bigcup_{g \in \mathcal{F}} \hat{g}(T^{|g|} : e) \ ,$$

where $|g|$ denotes the arity of g.

We define a function ξ from flat boolean expressions over $\mathcal{P}$, $\mathcal{F}$, and $x_1, \ldots, x_k$ to set expressions as follows:

$$\begin{aligned} \xi(p(x_i)) &= T^{k-i} : \alpha_p : T^{i-1} : e \\ \xi(p(\hat{g}(x_1, \ldots, x_i))) &= T^{k-i} : g^{-1}(\alpha_p) \\ \xi(\Psi_1 \wedge \Psi_2) &= \xi(\Psi_1) \cap \xi(\Psi_2) \\ \xi(\Psi_1 \vee \Psi_2) &= \xi(\Psi_1) \cup \xi(\Psi_2) \\ \xi(\neg\Psi) &= T^k : e \cap \overline{\xi(\Psi)} \ . \end{aligned}$$

The set constraint K_Φ is now defined as the conjunction of the following subset relations: (i) the set equation defining T that was given above, (ii) the subset relation $\alpha_p \subseteq T$ for every predicate symbol $p \in \mathcal{P}$, and (iii) the subset relation $T^k : e \subseteq \xi(\Phi)$. If n is the size of the monadic formula Φ, then K_Φ has size $\mathrm{O}(n^2/\log n)$.

Suppose that I is a Herbrand model for Φ'. For every $p \in \mathcal{P}$, we assign to α_p the set of all $\tau(t)$ such that $p(t) \in I$. An easy induction proof shows that this assignment constitutes a solution of K_Φ. Conversely, assume that we have a solution of K_Φ, then the set of all $p(t)$ such that α_p contains $\tau(t)$ is a Herbrand model for Φ'. In other words, K_Φ is satisfiable if and only if Φ is satisfiable. $\Box$

Theorems 2 and 6 indicate that set constraints without positive occurrences of projections for non-monadic functions and the monadic class are two essentially equivalent logics, with respect to expressiveness as well as complexity.

The problem of set constraints with unrestricted positive projections is still open. Positive projections can be used to formulate negations of subset relationships. In fact, $a \subseteq f^1(f(a, \alpha))$, where a is a constant, is equivalent to $\alpha \not\subseteq \emptyset$. Therefore, being able to solve constraints with negated subset relationships would be a major step towards admitting projections in full and an interesting case of its own. We have learned that this latter problem has recently been solved (Sophie Tison, personal communication), but we have not yet seen the solution.

Negated subset relations amount to deciding fragments of the inductive theory for the class of flat formulas that correspond to set constraints. Without loss of generality one may restrict attention to constraints K of form

$$E_1 \subseteq \emptyset \wedge \ldots \wedge E_m \subseteq \emptyset \wedge \alpha \not\subseteq \emptyset$$

where α is a set variable. Such a constraint is satisfiable if and only if

$$[\![E_1 \subseteq \emptyset \wedge \ldots \wedge E_m \subseteq \emptyset]\!]$$

is satisfiable and

$$[\![E_1 \subseteq \emptyset \wedge \ldots \wedge E_m \subseteq \emptyset]\!] \not\models \forall x\ \neg P_\alpha(x)$$

where the universal quantification ranges over the set of ground terms of the given signature.

We believe that by relating set constraints to the monadic class it should be possible to decide universally quantified consequences of set constraints by exploiting the finite model property of the monadic class. More precisely, if Φ is a monadic formula over predicate symbols in $\mathcal{P}$ and I is a $\mathcal{P}$-structure, then one can consider the equivalence relation $\equiv$ in I defined by $x \equiv y$ if and only if for all p in $\mathcal{P}$ we have $I \models p(x)$ if and only if $I \models p(y)$. Clearly, I is a model of Φ if and only if $I/\!\equiv$ (with the canonical interpretation of the predicate symbols) is a model of Φ. Moreover, $I/\!\equiv$ is finite so that the validity of formulas in I can be effectively decided. If we now consider a set constraint K and its equivalent monadic formula $[\![K]\!]_m$, the only remaining problem is to characterize those finite models of $[\![K]\!]_m$ which result from (Herbrand) models I of $[\![K]\!]$ through factorization by $\equiv$. This problem we have not yet been able to solve.

4 Related Work

The present paper was motivated by the work of Aiken and Wimmers (1992) who considered the basic form of set constraints defined in Section 3.1. Their constraint solving algorithm employs a step similar to our translation of constraints into flat formulas and then employs similar techniques as ordered resolution to saturate constraints. This is not surprising as ordered resolution can be turned into a decision procedure for the whole monadic class. Their algorithm was shown to be in NEXPTIME. We have made this result more precise by establishing a less than quadratic exponent, i.e., an upper bound of NTIME($c^{n^2/\log n}$).

In an earlier approach Heintze and Jaffar (1990) used ad-hoc formalisms for solving the satisfiability problem of the class of *definite* set constraints with projections. Definite constraint are of the form $X \subseteq Y$, where Y is a variable and X does not contain the complement operator (but may contain projections). Their approximative consequence operator for Prolog programs can be defined in terms of these constraints. For example, the reverse program

$$\begin{array}{rcl} rev(nil, L, L) & . & \\ rev(cons(X, L), R, S) & :- & rev(L, cons(X, R), S). \end{array}$$

is approximated by the constraints

$$\begin{array}{rcl} c(nil, 1, 1) & \subseteq & rev \\ c(cons(cons^1(c^2(rev)), c^1(rev)), cons^2(c^2(rev)), c^3(rev)) & \subseteq & rev, \end{array}$$

where c is a ternary constructor which combines the arguments of rev. The least solution of the constraints is the least solution of the approximate consequence operator. Our present result for constraints with projections is more general in that we also admit positive occurrences of unions and negative occurrences of complements. Therefore we can also handle analysis and type inference problems for disjunctive logic programs and nondeterministic programming languages. Moreover, we have exhibited a decidable class of set constraints with equality. For the above example

$$\begin{array}{rcl} \Delta^c_{2=3}(c(nil, 1, 1)) & \subseteq & rev \\ c(cons(cons^1(c^2(rev)), c^1(rev)), cons^2(c^2(rev)), c^3(rev)) & \subseteq & rev, \end{array}$$

is a better approximation of the reverse program which captures the nonlinearity (i.e., the multiple occurrence of the variable L) in the first clause.

Frühwirth et al. (1991) studied the same class of constraints as Heintze and Jaffar (1990) and presented a proof method similar to ours in flavor. They reduce the problem to the decidability of a certain class of Horn clauses and show that the complexity of determining membership in their minimal models is EXPTIME-complete. Here we have seen that adding disjunctions to this class of formulas makes the complexity jump to NEXPTIME completeness.

Tree automata with tests for equality of subterms (Bogaert and Tison 1991) are a special case of set constraints with equality.

More recently, Gilleron, Tison, and Tommasi (1992) have introduced a new class of tree automata which exactly accept solutions of set constraints. This gives rise to another constraint solving algorithm. They also prove the existence of regular, as well as minimal and maximal regular solutions in case solutions do exist. The problem of projections is left open in this paper. Although the new class of automata may be of interest in its own, the proposed treatment of set constraints in terms of these automata appears to be technically involved. The reader will perhaps appreciate the simplicity of the treatment in the present paper.

5 Conclusions and Future Work

In this paper we have demonstrated that set constraints can be viewed as a logic that is essentially equivalent to the monadic class of first-order formulas. The translations establishing the equivalence are natural and have allowed us to directly apply decidability and complexity results for the monadic class to set constraints and settle open problems posed by Aiken and Wimmers (1992). Moreover we have shown that extensions of set constraints by equality and by negative projections do not lead beyond the monadic class (with equality). At present, however, projections for non-monadic functions are restricted to negative subexpressions in constraints, and the problem of admitting unrestricted projections remains open.

References

W. Ackermann, 1954. *Solvable Cases of the Decision Problem.* North-Holland, Amsterdam.

A. Aiken and B. Murphy, 1991. Implementing regular tree expressions. In *ACM Conference on Functional Programming Languages and Computer Architecture*, pp. 427–447.

A. Aiken and B. Murphy, 1991. Static type inference in a dynamically typed language. In *Eighteenth Annual ACM Symposium on Principles of Programming Languages*, pp. 279–290.

A. Aiken and E.L. Wimmers, 1992. Solving Systems of Set Constraints (Extended Abstract). In *Proc. IEEE Symposium on Logic in Computer Science*, pp. 329–340.

L. Bachmair, H. Ganzinger and U. Waldmann, 1992. Superposition with Simplification as a Decision Procedure. In preparation.

L. Bachmair, H. Ganzinger and U. Waldmann, 1992. Theorem proving for hierarchic first-order theories. In H. Kirchner, G. Levi, editors, *Algebraic and Logic Programming*, Lecture Notes in Computer Science, vol. 632, pp. 420–445, Berlin, Springer-Verlag. (Invited paper for special AAECC issue on ALP'92.).

B. Bogaert and S. Tison, 1991. Automata with equality tests. Technical Report IT 207, Laboratoire d'Informatique Fondamentale de Lille.

H. Comon, M. Haberstrau and J.-P. Jouannaud, 1992. Decidable Problems in Equational Theories. In *Proc. IEEE Symposium on Logic in Computer Science*, pp. 255–265.

B. Dreben and W.D. Goldfarb, 1979. *The Decision Problem. Solvable Classes of Quantificational Formulas.* Addison-Wesley Publishing Company, Inc.

C. Fermüller, A. Leitsch, T. Tammet and N. Zamov, 1992. *Resolution Methods for the Decision Problem.* To appear.

T. Frühwirth, E. Shapiro, M. Vardi and E. Yardeni, 1991. Logic programs as types for logic programs. In *Proc. IEEE Symposium on Logic in Computer Science*, pp. 300–309.

D. M. Gabbay and H. J. Ohlbach, 1992. Quantifier elimination in second–order predicate logic. *South African Computer Journal*, Vol. 7, pp. 35–43.

R. Gilleron, S. Tison and M. Tommasi, 1992. Solving Systems of Set Constraints using Tree Automata. Technical Report, Laboratoire d'Informatique Fondamentale de Lille. Paper accepted for STACS'93.

N. Heintze and J. Jaffar, 1990. A finite presentation theorem for approximating logic programs. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pp. 197–209.

N. Heintze and J. Jaffar, 1991. Set-based program analysis. Draft manuscript.

D.S. Johnson, 1990. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pp. 67–161. Elsevier.

N.D. Jones and S.S. Muchnick, 1979. Flow analysis and optimization of LISP-like structures. In *Sixth Annual ACM Symposium on Principles of Programming Languages*, pp. 244–256.

W.H. Joyner Jr., 1976. Resolution Strategies as Decision Procedures. *Journal of the Association for Computing Machinery*, Vol. 23, No. 3, pp. 398–417.

H.R. Lewis, 1980. Complexity Results for Classes of Quantificational Formulas. *Journal of Computer and System Sciences*, Vol. 21, pp. 317–353.

L. Löwenheim, 1915. Über Möglichkeiten im Relativkalkül. *Math. Annalen*, Vol. 76, pp. 228–251.

P. Mishra, 1984. Towards a theory of types in PROLOG. In *Proc. IEEE Symposium on Logic in Computer Science*, pp. 289–298.

P. Mishra and U. Reddy, 1985. Declaration-free type checking. In *Twelfth Annual ACM Symposium on the Principles of Programming Languages*, pp. 7–21.

J.C. Reynolds, 1969. Automatic Computation of Data Set Definitions. *Information Processing*, Vol. 68, pp. 456–461.

G.S. Tseitin, 1970. On the complexity of derivation in propositional calculus. *Seminars in Mathematics, V.A. Steklov Math. Institute, Leningrad, Consultants Bureau, New York-London*, Vol. 8, pp. 115–125. Reprinted in J. Siekmann and G. Wrightson (eds): Automation of Reasoning, Vol. 2, 1983, Springer-Verlag, pp. 466–486.

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