Set Constraints are the Monadic Class

Leo Bachmair* Harald Ganzinger† Uwe Waldmann†

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.

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, 1991b), Heintze and Jaffar (1990b, 1991), Jones and Muchnick (1979), Mishra (1984), Mishra and Reddy (1985), Reynolds (1969), Young and O'Keefe (1988), among others. Set constraints are also related to term declarations in order-sorted languages and have been used to describe unification algorithms in such a framework (Uribe 1992). An overview over known algorithms for solving various classes of set constraints can be found in Section 4. These methods are often 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 and diagonalizations occur with *negative* polarity only. In applications to program analysis and type inference, where projections and equality tests are useful, positive projections are not needed anyway, cf. (Heintze and Jaffar 1990b). 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 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 or diagonalizations remains unsolved at this time.

In short, in this paper we demonstrate that (i) set constraints are essentially equivalent to the monadic class, that (ii) decidability and complexity results for the monadic class carry over to set constraints, and that (iii) these results are also valid for several useful extensions of set constraints. Besides these theoretical consequences, our method of relating set constraints to the monadic class opens up a practical way to decide these extensions of set constraints using standard theorem proving methods based on ordered resolution and/or superposition that are known for the monadic

*Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794, U.S.A, leo@cs.sunysb.edu

†Max-Planck-Institut für Informatik, Postfach 1150, D-66041 Saarbrücken, Germany, {hg,uwe}@mpi-sb.mpg.de

1043-6871/93 $03.00 © 1993 IEEE

class (Joyner 1976, Fermüller et al. 1992, Bachmair, Ganzinger and Waldmann 1993).

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 (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 (1993) 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 $c > 0$, $\mathrm{NTIME}(c^{n/\log n})$ is an upper bound for the complexity of Monadic-Sat, where n is the length of the formula. His proof is based on the finite model property of the monadic class and checks finite structures up to cardinality 2^k, where $k = \mathrm{O}(n/\log n)$ is the number of predicate symbols, for the model property. Although he shows that this algorithm is in some sense optimal — $\mathrm{NTIME}(c^{n/\log n})$, for some (other) $c > 0$, is at the same time a lower bound for this problem — the proof-theoretic methods of Ackermann (1954), Joyner (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. 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

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

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

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

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

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

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

$$\beta \subseteq \alpha \wedge f(\alpha) \subseteq \overline{\alpha} \wedge f(\overline{\alpha}) \subseteq \alpha$$

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

$$\begin{aligned}
P_\beta(x) &\rightarrow P_\alpha(x) \\
P_{f(\alpha)}(x) &\rightarrow P_{\overline{\alpha}}(x) \\
P_{f(\overline{\alpha})}(x) &\rightarrow P_\alpha(x) \\
P_{\overline{\alpha}}(x) &\leftrightarrow \neg P_\alpha(x) \\
P_{f(\alpha)}(f(x)) &\leftrightarrow P_\alpha(x) \\
P_{f(\alpha)}(a) &\leftrightarrow \mathit{false} \\
P_{f(\overline{\alpha})}(f(x)) &\leftrightarrow P_{\overline{\alpha}}(x) \\
P_{f(\overline{\alpha})}(a) &\leftrightarrow \mathit{false}\,.
\end{aligned}$$

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 *If I is a Herbrand model of $[\![K]\!]$, then K is satisfied under the assignment where each set variable α in K is assigned the set of ground terms $\{ t \in T_{\mathcal{F}} \mid P_\alpha(t) \in I \}$. Conversely, if the assignment ϕ satisfies K, then there exists a Herbrand model I of $[\![K]\!]$ such that $\phi(\alpha) = \{ t \in T_{\mathcal{F}} \mid P_\alpha(t) \in I \}$ for each set variable α in K. 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 $O(n)$ (occurrences of) subexpressions and $k = O(n/\log n)$ 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 $O(n^2/\log n)$ atoms. As every predicate symbol or variable in the monadic formula $[\![K]\!]_m$ can be coded in

$O(\log n)$ space, $[\![K]\!]_m$ has length $O(n^2)$. 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 formula of length n of the form*

$$\forall x \forall y\ \Phi_1(x) \wedge \ldots \wedge \Phi_n(x) \wedge \Psi(x, y),$$

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

Proof. In order to replace Ψ by an equivalent constraint 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 formula Ψ can now be replaced by the constraint $g(N, N) \subseteq [\Psi]$, where $[\Psi]$ is defined inductively over the syntactic structure by the identities

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

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

The clauses Φ_i take the form

$$\begin{aligned}
&L_1(x) \vee \ldots \vee L_p(x) \\
&\quad \vee M_1(f(x)) \vee \ldots \vee M_q(f(x)) \\
&\quad \vee N_1(a) \vee \ldots \vee N_r(a),
\end{aligned}$$

where the L_j, M_j and N_j are predicates, possibly negated. An equivalent system of set constraints for such a formula is

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

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 $\mathrm{NTIME}(c^n)$ to the monadic class via length order $n \log n$ only produces formulas Γ as required for the above lemma. In Section 3.2 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* $\mathrm{NTIME}(c^{n/\log n})$.

Corollary 3 *Setc-Sat is* NEXPTIME*-complete.*

3.2 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, and where all free variables of Φ appear in $p(t_1, \ldots, t_n)$. 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 $N \cup \{F_{\leftarrow}\}$, we define a new interpretation I' that differs from I only in the denotation of the predicate p. For every ground substitution σ for the free variables in F, $p(t_1, \ldots, t_n)\sigma$ is true in I' if and only if $\Phi\sigma$ is true in I. As p does not occur positively in N and does not occur in Φ, $I' \subseteq I$ is a model of both $N \cup \{F\}$ and $N \cup \{F_{\leftarrow}\}$.

If I is a minimal model of $N \cup \{F_{\leftarrow}\}$, then $I' = I$ is a minimal model of $N \cup \{F\}$. Conversely, if a minimal model J of $N \cup \{F\}$ is not a minimal model of $N \cup \{F_{\leftarrow}\}$, then there exists a minimal model $I \subset J$ of $N \cup \{F_{\leftarrow}\}$. In this case $I' \subseteq I \subset J$, which is a contradiction. □

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 $\beta \subseteq \alpha \wedge f(\alpha) \subseteq \overline{\alpha} \wedge f(\overline{\alpha}) \subseteq \alpha$, both α and $\overline{\alpha}$ occur positively and negatively, while $f(\alpha)$ and $f(\overline{\alpha})$ occur only negatively. Omitting trivial formulas of the form *false* $\rightarrow \Gamma$, the set $[\![K]\!]_{\pm}$ consists therefore of the clauses

$$\begin{aligned}
P_\beta(x) &\rightarrow P_\alpha(x) \\
P_{f(\alpha)}(x) &\rightarrow P_{\overline{\alpha}}(x) \\
P_{f(\overline{\alpha})}(x) &\rightarrow P_\alpha(x) \\
P_{\overline{\alpha}}(x), P_\alpha(x) &\rightarrow \\
&\rightarrow P_{\overline{\alpha}}(x), P_\alpha(x) \\
P_\alpha(x) &\rightarrow P_{f(\alpha)}(f(x)) \\
P_{\overline{\alpha}}(x) &\rightarrow P_{f(\overline{\alpha})}(f(x)) .
\end{aligned}$$

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. The set expression $f^i(E)$ denotes the set of all terms t_i for which there exist ground terms $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n$, such that $f(t_1, \ldots, t_n) \in E$. 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 if $f^i(E)$ does not occur positively in K, then $[\![K]\!]_{\pm}$ is a set of (flat) clauses (without existential quantifiers). Let *SetcP-Sat* denote the satisfiability problem for set constraints without positive occurrences of projections for non-monadic function symbols.

Theorem 4 *SetcP-Sat is decidable. More precisely, it is an* NEXPTIME*-complete problem.*

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

Theorem 5 *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 $x_1, \ldots, x_k$. Without loss of generality we assume that $\mathcal{F}$ contains at least one constant symbol (otherwise we add one to $\mathcal{F}$). 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 write $A^i : B$ for $A : (A : \cdots : (A : B)\ldots)$, 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 formulas over $\mathcal{P}$, $\mathcal{F}$, and $x_1, \ldots, x_k$ to set expressions over $\hat{\mathcal{F}}$ as follows:

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

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 $O(n^2/\log n)$.

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. □

Theorems 2 and 5 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 (Marc Tommasi, 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.

3.3 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 nonlinear heads of Prolog clauses or other kinds of equality tests on variables (for an example see Section 4).

Let $\Delta^f_{i=j}$ denote a family of operators on sets, called *diagonalization operators*, where f is an n-ary function symbol with $n > 1$ and $1 \leq i, j \leq n$. Then, for any set expression E, $\Delta^f_{i=j}(E)$ 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 $t_i = t_j$.

We will translate set constraints involving diagonalization operators into monadic formulas with equality.

$$\begin{aligned}
P_{\Delta^f_{i=j}(E)}(f(x_1, \ldots, x_n)) \quad &\leftrightarrow \\
&P_E(f(x_1, \ldots, x_n)) \wedge x_i = x_j \\
P_{\Delta^f_{i=j}(E)}(g(x_1, \ldots, x_m)) \quad &\leftrightarrow \quad \mathit{false}, \text{ for } g \neq f\,.
\end{aligned}$$

However, a model of the monadic formula corresponds to a solution of the set constraint only if in this model the equality symbol is interpreted by syntactic equality. For this reason, as in the case of projections, we have to restrict to negative occurrences of $\Delta^f_{i=j}$, since then the $\rightarrow$-direction of the equivalences above can be ignored.

Lemma 4 *Let N be a set of first-order clauses with equality such that equality literals occur only negatively in N. Then N has a model if and only if it has a model where the equality symbol is interpreted by the syntactical equality relation (i.e., in which any two different ground terms are interpreted differently).*

Proof. The "if" part is trivial. To prove the "only if" part assume that I is an arbitrary model of N. We define a new interpretation I' in which the denotation of the equality predicate is the syntactical equality relation and in which every non-equality ground atom $p(t_1, \ldots, t_n)$ has the same truth value as in I. As equality literals occur only negatively in N, I' is a model of N. □

Theorem 6 *The satisfiability of extended set constraints without positive diagonalizations or projections for non-monadic function symbols is decidable. More precisely, the problem is* NEXPTIME*-complete.*

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 contains 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 $\mathrm{NTIME}(c^{n^2/\log n})$.

In an earlier approach Heintze and Jaffar (1990a) 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

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

is approximated by the constraints

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

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,

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

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

Frühwirth, Shapiro, Vardi, et al. (1991) studied the same class of constraints as Heintze and Jaffar (1990a) 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. Our result is more general in the sense that it is not restricted to Horn clauses. On the other hand it is weaker in that we cannot deal with tests for disequality. It is 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.

More recently, Gilleron, Tison and Tommasi (1993) 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. Complexity results are not stated, and the problem of projections is also left open in this paper.

Checking whether the intersection of two or more sorts is empty is a central task in unification procedures for an order-sorted logic with term declarations. For semi-linear term declarations, as defined by Uribe (1992) and Socher-Ambrosius (1993), this problem can be reduced to the satisfiability problem for set constraints with negative diagonalization. Uribe considers constraints of the form $\alpha_1 = E_1, \ldots, \alpha_n = E_n$,

where all α_i are different and the E_i may contain inter-variable dependencies. Translated to our framework, this yields constraints where diagonalizations occur both positively and negatively. However, the E_i do not contain negations, hence the set constraint has always a smallest solution. To test whether two sorts α_j and α_k are disjoint in this smallest model, it is sufficient to check whether the somewhat weaker constraint $E_1 \subseteq \alpha_1, \ldots, E_n \subseteq \alpha_n, \alpha_j \cap \alpha_k \subseteq 0$ is satisfiable: if $\alpha_j \cap \alpha_k$ is non-empty in the smallest model, then it is non-empty in every model. This weaker constraint contains only negative diagonalizations, and hence is amenable to our decision method.

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 negative diagonalizations and projections do not lead beyond the monadic class (with equality). These extended constraints can thus still be solved using decision procedures for the monadic class that are based on standard theorem proving methods. At present, however, projections for non-monadic functions and diagonalizations are restricted to negative subexpressions in constraints, and to weaken these restrictions remains an open problem.

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 Ministry for Research and Technology (Bundesministerium für Forschung und Technologie) under grants ITS 9102/ITS 9103 and by the ESPRIT Basic Research Working Group 6028 (Construction of Computational Logics).

References

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

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

A. Aiken and B. Murphy, 1991b. 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 *Seventh Annual IEEE Symposium on Logic in Computer Science*, pp. 329–340.

L. Bachmair, H. Ganzinger and U. Waldmann, 1992. Theorem Proving for Hierarchic First-Order Theories. In *Algebraic and Logic Programming, Third International Conference*, LNCS 632, pp. 420–434. Springer-Verlag.

L. Bachmair, H. Ganzinger and U. Waldmann, 1993. Superposition with Simplification as a Decision Procedure for the Monadic Class with Equality. Technical Report MPI-I-93-204, Max-Planck-Institut für Informatik, Saarbrücken. To appear in *3rd Kurt Gödel Colloquium: Computational Logic and Proof Theory.*

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 Shallow Equational Theories (Extended Abstract). In *Seventh Annual 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 *Sixth Annual 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. Also appeared in *3rd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 425–435, 1992.

R. Gilleron, S. Tison and M. Tommasi, 1993. Solving Systems of Set Constraints using Tree Automata. In *10th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 665, pp. 505–514. Springer-Verlag.

N. Heintze and J. Jaffar, 1990a. A Decision Procedure for a Class of Set Constraints (Extended Abstract). In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pp. 42–51.

N. Heintze and J. Jaffar, 1990b. 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 optimisation 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 *IEEE International Symposium on Logic Programming*, 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.

R. Socher-Ambrosius, 1993. Unification in Order-Sorted Logic With Term Declarations. To appear in *Conference on Logic Programming and Automated Reasoning*, 1993.

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, editors, *Automation of Reasoning*, volume 2, pp. 466–486, 1983. Springer-Verlag.

T. E. Uribe, 1992. Sorted Unification Using Set Constraints. In *11th International Conference on Automated Deduction*, LNAI 607, pp. 163–177. Springer-Verlag.

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.