Więzy mnogościowe w pewnych teoriach równościowych

Witold Charatonik

Praca doktorska
napisana w
Instytucie Informatyki
Uniwersytetu Wrocławskiego

przedstawiona w
Instytucie Matematycznym
Polskiej Akademii Nauk

promotor: Leszek Pacholski

Wrocław, 1994

Set Constraints in Some Equational Theories

Witold Charatonik

Ph.D. Thesis
written in
Institute of Computer Science
University of Wrocław

submitted in
Institute of Mathematics
Polish Academy of Sciences

supervisor: Leszek Pacholski

Wrocław, 1994

Abstract.

Set constraints are relations between sets of ground terms over a given alphabet. They give a natural formalism for many problems in program analysis, type inference, order-sorted unification, constraint logic programming. In this paper we start studies of set constraints in the environment given by equational specifications. We show that in case of associativity (i.e., in free monoids) as well as in case of associativity and commutativity (i.e., in commutative monoids) the problem of consistency of systems of set constraints is undecidable; in linear non-erasing shallow theories the consistency of systems of positive set constraints is NEXPTIME-complete and in linear shallow theories the problem for positive and negative set constraints is decidable.

Contents

1 Introduction **4**
1.1 Set constraints 4
1.2 Earlier results 5
Definite and positive set constraints 5
Negative set constraints 6
Set constraints with equality 8
Set constraints with projections 8

2 Associativity, associativity and commutativity **9**
2.1 Associativity 9
2.2 Associativity and commutativity 10

3 Shallow theories **12**
3.1 Basic notions 12
3.2 Reduction to almost flat formulae 14
3.3 Positive set constraints in non-erasing linear shallow theories 29
3.4 Negative set constraints in linear shallow theories 30
3.4.1 Reduction to tree set automata 30
3.4.2 Decidability of the problem 32
3.5 An application to semantic order-sorted unification 36

1 Introduction

Set constraints give a natural formalism for many problems in program analysis, type inference, order-sorted unification, constraint logic programming. They have been studied and applied in many papers including [6], [7],[5], [4], [22], [24], [28], [29], [32], [31], [26].

Set constraints provide a tool for an analysis of (functional, logic and imperative) programs which is based on representation of variables as sets of values. In algorithms of analysis of programs sets of terms represent possible values computed by the program (or assumed by variables). Set constraints are generated by the program syntax and a solution of set constraints gives some information about the program; like types consistency or possibility of optimalization.

The motivation for the work presented here was given mainly by the paper [19] of N. Heintze, in which the ideas of set based analysis are extended to arithmetic expressions. This extension yields useful information about the arithmetic components of a program. We prove here that in presence of set intersection the problem of finding solutions of set constraints involving arithmetic is not computable.

Set constraints are also used in unification algorithms for order-sorted languages. This gives decidability of unification problems for some cases of order-sorted signatures (see [31]).

Another important application of set constraints is an approximation of logic programs. It is known that the meaning of a logic program (i.e., the set of formulae that are consequences of this program) is in general non-recursive. Set constraints give a possibility of finding a good, recursive and finitely presented approximation of this set. In [13] A. Colmerauer presents PROLOG III, an extension of PROLOG by an associative operation of list concatenation. Here we show that an attempt to extend the approximation of logic programs to PROLOG III-like languages must fail, since the problem of consistency of systems of set constraints in associative theories is undecidable.

This thesis consists of three parts. First we introduce the problem and its history. In the second part we deal with associative and associative-commutative theories and prove that in presence of set intersection proposed by N. Heintze set based analysis of programs involving arithmetic is not computable. The third part is devoted to linear shallow theories. This includes such theories as commutativity or unity, but not associativity or idempotency. Shallow theories were investigated by H. Comon, M. Haberstrau and J.-P. Jouannaud in their papers [14] and [15]. We show that the problem of consistency of systems of set constraints in such theories can be reduced to satisfiability of almost flat formulae and then solved using methods introduced in [11]. Following the ideas of [31] we were able to use this result to prove that semantic order-sorted unification is decidable for elementary sort signatures and linear shallow equational theory.

A part of this thesis ([10]) was presented during the international conference *Constraints in Computational Logics* held in Munich in September 1994.

1.1 Set constraints

A *system of set constraints* is a finite conjunction of subset relations $E \subseteq E'$ (or, in the case of negative set constraints, also $E \not\subseteq E'$), where E and E' are *set expressions* over a given finite vocabulary Σ of function symbols, and an infinite collection $\mathcal{V}$ of set-valued variables. In this thesis we consider only basic set expressions defined by the grammar

$$E ::= \bot \mid \top \mid \alpha \mid E \cup E \mid E \cap E \mid \overline{E} \mid f(E_1, ..., E_n)$$

where α is a set variable in $\mathcal{V}$ and $f \in \Sigma$. In related papers this grammar is often extended by the operators of diagonalizations $\Delta^f_{i=j}(E)$ or projections $f^i(E)$. If diagonalizations are present we speak of set constraints with equality (this notion was introduced in [9], where a partial reduction of such systems of set constraints to monadic formulae with equality was made). If projections are present we speak of set constraints with projections.

A system of set constraints is said to be *consistent* if there exists a mapping (called the *solution* of the system) assigning sets of ground terms over Σ to the variables in such a way that the conjunction of inclusions evaluates to true, where $\bot$ denotes the empty set and $\top$ the set $\mathcal{H}$ of all ground terms over Σ (the Herbrand universe); a set expression $f(E_1, \ldots, E_n)$ denotes the set of terms $\{f(t_1, \ldots, t_n) \mid t_i \in E_i\}$; $\Delta^f_{i=j}(E) = \{f(t_1, \ldots, t_n) \mid t_i = t_j\}$ and $f^i(E) = \{t_i \mid \exists t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n : f(t_1, \ldots, t_n) \in E\}$.

In the case of equational theory E we consider equivalence classes of the relation $=_E$ instead of terms — then $f(E_1, \ldots, E_n)$ denotes the set $\{q(f(t_1, \ldots, t_n)) \mid q(t_i) \in E_i\}$ where $q : \mathcal{H} \to \mathcal{H}/=_E$ is the quotient mapping.

1.2 Earlier results

Definite and positive set constraints

The first general results concerning the decidability of set constraints were obtained by Heintze and Jaffar [21], who studied the so-called definite set constraints. Definite set constraints are of the form $exp_1 \subseteq exp_2$ where exp_2 is restricted to constants, variables and function symbols, and exp_1 does not contain the complement symbol, but may contain projections.

Positive set constraints are just set constraints defined in section 1.1, without diagonalizations and projections. The decidability of systems of positive set constraints was established by A. Aiken and E.L. Wimmers [8]. Later other proofs have been obtained. R. Gilleron, S. Tison, and M. Tommasi [17] gave a proof based on automata theoretic techniques and L. Bachmair, H. Ganzinger, and U. Waldmann [9] gave a proof using the decision procedure for the first order theory of monadic predicates, providing also NEXPTIME-completeness of the problem of consistency of positive set constraints. In the paper [2] by A. Aiken, D. Kozen, M. Vardi, and E.L. Wimmers, yet another algorithm has been presented and a detailed analysis of the complexity of positive set constrains has been given. The paper [25] contains the essence of all these algorithms.

Since our approach is based on techniques introduced in [9], below we give the main idea of the proof from this paper.

Let (SC) be a system of set constraints. Let $E(SC)$ denote set of all set expressions occurring in SC. For each expression $E \in E(SC)$ we introduce a predicate symbol P_E. These predicates are defined by the following formulae:

$$
\begin{aligned}
P_\top(y_1) &\leftrightarrow true \\
P_\bot(y_1) &\leftrightarrow false \\
P_{E_1 \cup E_2}(y_1) &\leftrightarrow P_{E_1}(y_1) \vee P_{E_2}(y_1) \\
P_{E_1 \cap E_2}(y_1) &\leftrightarrow P_{E_1}(y_1) \wedge P_{E_2}(y_1) \\
P_{\overline{E}}(y_1) &\leftrightarrow \neg P_E(y_1) \\
P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n)) &\leftrightarrow P_{E_1}(y_1) \wedge \ldots \wedge P_{E_n}(y_n) \\
P_{f(E_1,\ldots,E_n)}(g(y_1, \ldots, y_m)) &\leftrightarrow false
\end{aligned}
$$

for $g \in \Sigma - \{f\}$. The universal closure of the conjunction of formulae defining predicates P_E for all $E \in E(SC)$ and the formulae $P_E(y_1) \to P_{E'}(y_1)$ for all set constraints $E \subseteq E'$ occurring in SC, expresses the satisfiability of (SC), is a flat formula (see definition 3.7), therefore is equivalent to a skolemization of some monadic formula. This reduces the problem of consistency of positive set constraints to the problem of satisfiability of monadic formulae, which is in NEXPTIME since by the result of Ackermann for satisfiable sentences without equality there exists a model of size bounded by 2^N, where N is the number of predicate symbols (see lemma 3.25).

Negative set constraints

In the case of negative set constraints we allow not only inclusions of the form $E \subseteq E'$ as in the positive case, but also negated inclusions of the form $E \not\subseteq E'$. Since we mix here inclusions with negated inclusions, these set constraints are often called *mixed*.

The problem of the decidability of such sets constraints is more difficult. Two solutions were obtained independently by Aiken, Kozen, and Wimmers [3], and Gilleron, Tison and Tommasi [18]. Both of them are quite non-trivial. The third solution was later obtained by Charatonik and Pacholski [11].

The solution by Aiken, Kozen and Wimmers uses ideas of [2] and goes through normal forms, a reduction to another problem concerning hypergraphs, a reduction to the Diophantine (nonlinear) reachability problem, and solving the last one in some sense again using graph-theoretic tools. Later an improvement of this result was obtained by K. Stefansson [30], who simplified the last part of the proof, obtaining NP-completeness of the Diophantine reachability problem, and thus establishing NEXPTIME-completeness of the original problem.

The solution by Gilleron, Tison and Tommasi extends ideas of [17] and is based on the notion of a tree set automaton. The paper presents a theory of tree set automata, describes how set constraints can be expressed using this notion and proves the decidability of the non-emptiness problems for tree set automata thus establishing the decidability of sets constraints. The first two parts of the paper are quite nice. However, the proof of decidability of the emptiness problem is long and contains several difficult technical lemmas whose proofs are rather difficult to follow.

The solution by Charatonik and Pacholski is based on the idea of Bachmair, Ganzinger, and Waldmann [9] to reduce the decidability problem for positive set constraints to the problem of consistency of first order theories of unary predicates. The translation of set constraints to monadic formula is almost the same as described for positive set constraints; the difference is that negative set constraints $E \not\subseteq E'$ are encoded in formulae of the form $\exists x P_E(x) \wedge \neg P_{E'}(x)$. The problem we encountered here was that finite models of such formulae not always can be extended to a solution of the input system of set constraints.

Example 1.1. The system $X \not\subseteq Y, a \subseteq Y, f(Y) \subseteq Y$ is not consistent over the alphabet $\Sigma = \{a, f(\cdot)\}$ because Y must contain all the terms of the form $f^n(a)$ with $n \geq 0$ (that is the whole Herbrand universe), so there is nothing to be put to X.

The following monadic formula describes this system:

$$\exists x \exists x_a \forall y \exists x_f P_X(x) \wedge \neg P_Y(x) \wedge (P_a(y) \to P_Y(y)) \wedge (P_a(x_a) \leftrightarrow true) \wedge$$

$$\wedge (P_a(x_f) \leftrightarrow false) \wedge (P_{f(Y)}(x_f) \leftrightarrow P_Y(y)) \wedge (P_{f(Y)}(x_a) \leftrightarrow false)$$

This formula has a finite model with the domain consisting of three elements e_1, e_2, e_3 and the following interpretation of predicate symbols: $P_a(e_1), P_Y(e_1), P_Y(e_2)$, $P_{f(Y)}(e_2), P_X(e_3)$. Transforming it to a Herbrand model requires introducing a new constant symbol (corresponding to e_3), which is not allowed. Note that the system is consistent over the alphabet $\Sigma = \{a, b, f(\cdot)\}$.

We avoided such situations by introducing skeletons of solutions of the system. Let SC be a consistent system of set constraints. In the definition below Φ denotes the quotient mapping from the Herbrand universe $\mathcal{H}$ to $\mathcal{H}/\equiv$ where $t \equiv s$ iff for each set expression $E \in E(SC)$ we have $t \in E \leftrightarrow s \in E$ (note that Φ determines and is determined by the solution of SC), $\prec$ is fixed linear order on terms such that terms of lower depth precede terms of greater depth, $c[i]$ denotes the i-th term with respect to $\prec$, which belongs to the set $\Phi^{-1}(c)$ and was included in the set of *milestones*. A term is called *composed* if it is not a constant symbol.

Definition 1.1. A skeleton of a solution of a system of set constraints is a labeled graph $\mathcal{S}$ such that

1. nodes of $\mathcal{S}$ are terms
2. the set of nodes contains a subset called *milestones*
3. each node t is labeled with $\Phi(t)$
4. for each $c \in \Phi(\mathcal{H})$ there exists a milestone labeled with c
5. each node being a composed term is connected by an edge with its maximal subterm
6. each edge is labeled by a rule of the form $f(c_1[j_1], \ldots, *, \ldots, c_n[j_n])$
7. each rule is consistent with the solution, i.e., if the edge from s to t is labeled by $f(c_1[j_1], \ldots, *, \ldots, c_n[j_n])$ then $\Phi(f(c_1[j_1], \ldots, s, \ldots, c_n[j_n])) = \Phi(t)$
8. if $c[j]$ is used in a label of an edge $t \to s$ then there are at least j milestones labeled by c, and each of these nodes is a term lower then t with respect to $\prec$
9. all leaves of $\mathcal{S}$ are milestones

We proved that if such skeleton exists then it can be extended to a solution, and if the system is consistent then there exists a small skeleton. Therefore the system is consistent if and only if there exists a solution with a small skeleton. Now to check if the system is consistent it is enough to check if there exists a finite model of the monadic formula which admits existence of a small skeleton.

This result can be also translated to the language of tree set automata (see also the section 3.4.1) which gives a relatively easy and elegant proof of decidability of the emptiness problem mentioned above.

Set constraints with equality

Set constraints with equality are set constraints with grammar of set expressions extended by operators of diagonalizations. They were introduced by Bachmair, Ganzinger and Waldmann in [9], where partial result of consistency for positive set constraints with negative occurrences of diagonalizations were obtained. The obvious translation to monadic class, namely $P_{\Delta^f_{i=j}(E)}(f(y_1,\ldots,y_n)) \leftrightarrow P_E(f(y_1,\ldots,y_n)) \wedge y_i = y_j$ and $P_{\Delta^f_{i=j}(E)}(g(y_1,\ldots,y_m)) \leftrightarrow false$ leads to formulae with equality and lemma 3.25 can no longer be used.

The problem with unrestricted diagonalizations in (more general than positive) negative set constraints was solved by Charatonik and Pacholski in [11].

We used there the following extension of the lemma 3.25 (the definition of the relation $\equiv$ is given just before lemma 3.25)

Lemma 1.2 (see [16]). *A monadic formula with equality, of quantifier depth at most k, is satisfiable if and only if it has a finite model (of cardinality $M \leq 2^N k$) such that in each equivalence class of the relation $\equiv$ there are at most k elements.*

Definition 1.3. Let ψ be a monadic formula with equality, of quantifier depth at most k. Two interpretations I, J of ψ are similar if for each pair c_I, c_J of corresponding equivalence classes of the relations $\equiv_I$ and $\equiv_J$ either $|c_I| = |c_J|$ or both c_I and c_J have at least k elements.

Lemma 1.4. *If I is a model of ψ and I is similar to J, then J is also a model of ψ.*

Now by applying techniques for solving negative set constraints (where we had to check if each equivalence class is nonempty, i.e., if it contains at least one term) we could check if for a given finite model there exists a similar Herbrand model (i.e., a model with the desired number of terms in each equivalence class) which gives a solution for the system of set constraints.

Set constraints with projections

There were several attempts to solve the problem with projections. The first partial solution was given by Heintze and Jaffar in their paper about definite set constraints. The second partial solution (for negative occurrences of projections in positive set constraints) was given by Bachmair, Ganzinger and Waldmann in [9]. The obvious translation to (unfortunately not flat) formulae is the following: $P_{f^i(E)}(y_i) \leftrightarrow \exists y_1,\ldots,y_{i-1},y_{i+1},\ldots,y_n : P_E(f(y_1,\ldots,y_n))$, but if $f^i(E)$ does not occur positively in the input system of set constraints, then the equivalence can be replaced by the implication from right to left, and then the formula can be normalized to a flat formula.

The third partial solution was given in the mentioned above papers [18], [3], [11] about negative set constraints, since negative set constraints are a particular case of set constraints with projections. To see it consider a set constraint $a \subseteq f^1(f(a,X))$ where a is a constant symbol, which is equivalent to $X \not\subseteq \bot$.

The full solution was given by Charatonik and Pacholski in [12]. This was done by applying the techniques developed in [11] to the idea of Bachmair, Ganzinger and Waldmann. In the same way we replaced the equivalence by the implication from right to left. Then, however, the set of terms satisfying $P_{f^i(E)}$ does not have to be equal to the f^i projection

of the set of terms satisfying P_E — it only has to contain this projection. To check if the other inclusion holds (i.e., if this set is contained in the projection) we have to check if for each term satisfying $P_{f^i(E)}$ there exists a sequence of terms such that the term being the proper application of the function symbol f to the whole sequence satisfies P_E. We do it by extending the definition of a skeleton in such a way, that each node t satisfying $P_{f^i(E)}$ gets additional label of the form $f(c_1[j_1], \ldots, *, \ldots, c_n[j_n]) : c$ with interpretation that the term $f(c_1[j_1], \ldots, t, \ldots, c_n[j_n])$ must belong to the equivalence class c which is a subset of E. This time, however, we cannot require that t is the maximal subterm of $f(c_1[j_1], \ldots, t, \ldots, c_n[j_n])$ and we have to take additional care that two such labels are not contradictory (which can happen if one such label requires that a term must belong to the equivalence class c and another requires $c' \neq c$) - in the case of negative set constraints we do not have this problem because this maximality condition gives that the *collapsing function* (see also lemma 3.36) is injective. This leads to several steps of the construction of a skeleton: first we construct a semi-skeleton, then a semi-skeleton without local conflicts, a semi-skeleton without conflicts between edges and projections, a skeleton, and a pre-germ. Each step of this construction is technical and quite non-trivial. Finally we prove that the system is consistent if and only if there exists a solution with a small pre-germ.

2 Associativity, associativity and commutativity

In this section we shall prove that in free and commutative monoids the problem *given a system of set constraints, decide whether the system is consistent*, is undecidable.

2.1 Associativity

Below we reduce a known undecidable problem *given a Turing machine M, is $L(M)$ empty?* to the problem of consistency of set constraints in free monoids.

Let M be a Turing machine (with one left-bounded tape). We say that a word w is a configuration of M if it is of the form $w_1 q w_2$, where $w_1 w_2$ is the contents of the tape of M up to the rightmost nonblank symbol or the symbol to the left of the head, (whichever is the rightmost), q is the current state of M and w_1 is the contents of the tape to the left of the head. We write $w \vdash_M v$ if w and v are configurations and v results in one move from w.

A *valid computation* of M is a (finite) string $w_1 \# w_2^R \# w_3 \# w_4^R \# \ldots$ such that each w_i is a configuration of M, w_1 is an initial configuration, w_n is a final accepting configuration, and $w_i \vdash_M w_{i+1}$ for $1 \leq i < n$. The *set of invalid computations* is the complement of the set of valid computations (with respect to the union of the work alphabet of M, the set of states of M and $\{\#\}$).

Clearly $L(M)$ is empty if and only if the set of valid computations of M is empty.

The following lemma can be found in [23]:

Lemma 2.1. *The set of valid computations of a Turing machine M is the intersection of two context free languages, L_1 and L_2, and grammars for these languages can be effectively constructed from M.*

In the lemma below the system of set constraints is written over a signature consisting of constant symbols corresponding to terminal symbols in the grammar, and one associative function symbol corresponding to concatenation. Set variables correspond to nonterminal symbols in the grammar.

Lemma 2.2. *Let G be a context free grammar and let S be the start symbol of G. Let $SC(G)$ be a system of set constraints consisting of a constraint $w \subseteq X$, for each production rule $X \to w$ in G. Then for each solution $\mathcal{S}$ of $SC(G)$ we have $L(G) \subseteq \mathcal{S}(S)$.*

Proof. Let $L(X)$ denote the set of all words that can be produced from X using production rules from G. We have to show that $L(S) \subseteq \mathcal{S}(S)$. It is easy to see (by induction on the length of the word w) that for any nonterminal symbol X and any word w, $w \in L(X)$ implies $w \in \mathcal{S}(X)$. □

Now we are ready to prove the main theorem of this section.

Theorem 2.3. *The problem of consistency of systems of set constraints in the equational theory of one associative function symbol is undecidable.*

Proof. Let M be a Turing machine and let G_1, G_2 be context free grammars such that $L(G_1) \cap L(G_2)$ is the set of valid computations of M. Let S_1 and S_2 be starting symbols of G_1 and G_2 respectively.

Now we construct a system SC of set constraints. SC consists of all the constraints in $SC(G_1)$ and $SC(G_2)$, and, additionally the constraint $S_1 \cap S_2 \subseteq \bot$.

Suppose $\mathcal{S}$ is a solution of SC. The set of valid computations is a subset of $\mathcal{S}(S_1) \cap \mathcal{S}(S_2)$. Since $\mathcal{S}(S_2) \cap \mathcal{S}(S_3) = \bot$, the set of valid computations is empty. Similar arguments show that if the set of valid computations is empty then SC is consistent. As a corollary we get that SC is consistent if and only if the set of valid computations of M is empty, i.e. if and only if $L(M)$ is empty. □

2.2 Associativity and commutativity

In this section we prove that the problem of consistency of systems of set constraints in commutative monoids (i.e. in AC1, the theory of one associative and commutative function symbol with unit element), is undecidable. The proof works also in the case of AC theory (without unit element). Using the same argumentation we show that in the presence of set intersection the problem of finding solutions of constraints involving arithmetic is not computable.

The idea is to encode a two-counter machine. A two-counter machine consists of a finite control and of two counters. Each counter holds exactly one nonnegative integer. In one move the two-counter machine checks if any of its two counters contains zero and depending on the state of its finite control changes the state and adds or subtracts 1 from one of the counters. In [23] it is shown that a two-counter machine can simulate a Turing machine, therefore the following problem is undecidable: *given two-counter machine M and one of its states q, decide whether M reaches q*. Below we prove that this problem can be reduced to the problem of consistency of set constraints in commutative monoids.

I should explain in this place that when I worked on set constraints in commutative monoids (I proved NP-hardness of the problem and fixed my attention on trying to prove NP-completeness), we had a discussion with Harald Ganzinger, who noticed that this reduction can be done. That is why the following result is due to him.

Theorem 2.4 (Ganzinger). *The problem of consistency of systems of set constraints in AC1 is undecidable.*

Proof. Let M be a two-counter machine. We represent the storage of M (two integers i, j) as the term $a^i b^j$ where a and b are two different constant symbols. For each state q of M we introduce a set variable X_q which is to encode all possible values stored by M in the counters while being in the state q. The initial configuration (both counters set to zero, initial state q_0) is represented by the set constraint $1 \subseteq X_{q_0}$ where 1 is the unit element. To test whether the counters contain zero we need the sets $A_1 = \{a^i \mid i > 0\}$, $A_2 = \{b^j \mid j > 0\}$, and $B = \{a^i b^j \mid i > 0, j > 0\}$. We get these sets by introducing new set variables A_1, A_2 and B, and constraints $A_1 = a \cup aA_1$, $A_2 = b \cup A_2$, $B = ab \cup aB \cup bB$.

Now each move is represented in the following way: if the machine is in the state q, to test whether counters are set to zero we intersect X_q with the proper set ($1, A_1, A_2$ or B), to change the state to q' we write $X_{q'}$ on the right side of the inclusion, to add 1 to a counter we multiply the expression on the left side of inclusion by a proper constant (i.e. a for the first counter and b for the second), and to subtract 1 from a counter we multiply the right side of the inclusion by proper constant. For example the instruction *if in the state q the first counter is set to zero, the second is nonzero then change the state to q' and add 1 to the first counter* is represented by $a(X_q \cap A_2) \subseteq X_{q'}$ and *if in the state q both counters are nonzero then change the state to q' and subtract 1 from the second counter* is represented by $X_q \cap B \subseteq bX_{q'}$.

It is easy to see that now if M reaches a state q with i and j stored in the counters then the system is consistent and $\mathcal{S}(X_q)$ contains $a^i b^j$ for each solution $\mathcal{S}$.

Finally, to test whether M reaches q we write $X_q \subseteq \bot$. Now the system is consistent if and only if M does not reach q. □

In [19] N. Heintze extended the ideas of the set based analysis of programs to arithmetic expressions by allowing set expressions to be of the form $E_1 op E_2$ where *op* ranges over arithmetic operators such as $+, -, *, /$. In fact set expressions presented there are generated by quite different grammar (e.g. there is no set intersection) constructed in context of ML programming language, but in the case of other languages (e.g. logic programming languages, see [20]) the grammar constructed do contain intersection. Set constraints used in program analysis are of the form $E \subseteq X$ where E is a set expression and X is a set variable. The problem therefore is not to test consistency (these constraints are trivially consistent), but to compute the minimal solution. Below we prove that in presence of intersection and arithmetic this problem is not computable. More formally, we prove the following theorem:

Theorem 2.5. *Let set expressions be generated by the grammar $E ::= X \mid i \mid E \cap E \mid i * E \mid E/i$, where i ranges over natural numbers and X over the set of set variables. The problem* given a system SC of set constraints of the form $E_i \subseteq X_i$ and a set variable X, decide whether in the minimal solution $\mathcal{I}$ of SC the set $\mathcal{I}(X)$ is empty, *is undecidable.*

Proof. The proof is almost the same as in theorem 2.4. Here the storage with integers i, j is represented by the integer $2^i 3^j$. The sets A_0, A_1 and B are defined by the constraints $2 \subseteq A_0$, $2 * A_0 \subseteq A_0$, $3 \subseteq A_1$, $3 * A_1 \subseteq A_1$, $6 \subseteq B$, $2 * B \subseteq B$, $3 * B \subseteq B$. Adding (respectively subtracting) 1 to the first (respectively the second) counter is coded by multiplying (respectively dividing) by 2 (respectively by 3). For example the instruction *if in the state q both counters are nonzero then change the state to q' and subtract 1 from the second counter* is represented by $(X_q \cap B)/3 \subseteq X_{q'}$. It is easy to see that M reaches q if and only if in the least model $\mathcal{I}$ of these constraints $\mathcal{I}(X_q)$ is nonempty. □

3 Shallow theories

In this section we will deal with linear shallow equational theories. We prove that the problem of consistency of systems of mixed (positive and negative) set constraints in these theories can be reduced to satisfiability of almost flat formulae. If the theory is also non-erasing, the obtained formula is flat and this reduction gives decidability of the problem of consistency of positive set constraints. If the theory is not non-erasing, we apply techniques from [11] and obtain the decidability result for mixed set constraints. We apply this result to prove decidability of (elementary) order-sorted unification in such theories.

Shallow theories were investigated by H. Comon, M. Haberstrau and J.-P. Jouannaud in their papers [14] and [15]. This is probably the largest class of equational theories for which all interesting problems (like word problem, unification and disunification, quantifier elimination) are decidable, and one of very few classes of theories that enjoy a uniform unification or disunification procedure.

3.1 Basic notions

In this part we describe basic properties of shallow theories and almost flat formulae.

Definition 3.1. A term t is shallow if all its variables occur at depth at most one. An equation is unordered pair of terms. An equation $l = r$ is shallow if both l and r are shallow. A shallow presentation is a finite set of shallow equations. A theory is shallow if it can be axiomatized by a shallow presentation.

Definition 3.2. A term is linear if it contains no multiple occurrences of the same variable. An equation $l = r$ is linear if both l and r are linear.

Definition 3.3. An equational theory is called

- *nontrivial* if there exist two terms which are not equal in the theory
- *linear* if it can be axiomatized by a set of linear equations
- *non-erasing* if there exists such axiomatization of the theory that for every equation $l = r$ the set of variables occurring in l is equal to the set of variables occurring in r

Example 3.1. A theory axiomatized by an equation of the form $x = f(u_1, \ldots, u_n)$ with $x \notin \{u_1, \ldots, u_n\}$ is trivial. The theory axiomatized by $K(x, y) = x$ is not non-erasing.

All theories considered in this section are nontrivial, linear and shallow; some of them are non-erasing. If a theory is not non-erasing, it is easy to encode negative set constraints, which makes the problem much more difficult (see [3], [18],[11]). For example, in the theory axiomatized by $K(x, y) = x$, the constraint $a \subseteq K(a, E)$ is equivalent to $E \not\subseteq \bot$, since if E is empty then $K(a, E)$ is empty and if E is nonempty then $K(a, E) = a$.

A position in a term is a sequence of natural numbers used to select a subterm. The empty position is denoted Λ and corresponds to the whole term. The subterm of s at position p is denoted $s[p]$ and $s[p \leftarrow u]$ denotes the term s with the subterm at position p replaced by u. If E is a set of equations we write $s \overset{p}{\underset{E}{\leftrightarrow}} t$ if there is an equation $l = r \in E$ (sometimes we also write $s \overset{p}{\underset{l=r}{\leftrightarrow}} t$) and a substitution σ such that $s[p] = l\sigma$ and $t = s[p \leftarrow r\sigma]$, or $s[p] = r\sigma$

and $t = s[p \leftarrow l\sigma]$. If t and s are terms then $t[s]$ denotes $t[p \leftarrow s]$ for some position p. $s \overset{\neq\Lambda}{\underset{E}{\leftrightarrow}} t$ denotes $s \overset{p}{\underset{E}{\leftrightarrow}} t$ for some $p \neq \Lambda$. We shall often write $s \overset{p}{\longleftrightarrow} t$ when E is clear from the context.

In the lemma below the theory E' is obtained from E by adding new constant symbols to the signature and new equations to the theory. These new constant symbols correspond to ground terms of depth at least one, occurring in the axiomatization of E. $\mathcal{H}$ and $\mathcal{H}'$ denote respectively the sets of ground terms in the original and extended theory.

Lemma 3.4. *Every shallow theory E has an extension E' such that*

- *there exists an axiomatization of E' such that for every equation $s = t$ in this axiomatization, both s and t are terms of depth at most one*
- $\forall s \in \mathcal{H}' \exists t \in \mathcal{H} \;\; t =_{E'} s$
- $\forall s, t \in \mathcal{H} \;\; t =_E s \leftrightarrow t =_{E'} s$

Proof. This is the lemma 4.2 in [15] □The depth of an equation $s = t$ is maximum of the depth of s and the depth of t.

Lemma 3.5. *The problem of consistency of systems of set constraints in a shallow theory is reducible to the problem of consistency of systems of set constraints in a shallow theory axiomatized by a finite set of equations of depth at most one.*

Proof. By lemma 3.4 $\mathcal{H}/ =_E$ is isomorphic to $\mathcal{H}'/ =_{E'}$, so a system of set constraints is consistent in E if and only if it is consistent in E'. □

Lemma 3.6. *Every shallow theory E has a syntactic shallow presentation, i.e., a shallow presentation $\hat{E}$ such that for every two terms s, t such that $s =_E t$ there is a proof in $\hat{E}$:*

$$s \overset{(\neq\Lambda)^*}{\longleftrightarrow} \overset{\Lambda}{\longleftrightarrow} \overset{(\neq\Lambda)^*}{\longleftrightarrow} t \text{ or } s \overset{(\neq\Lambda)^*}{\longleftrightarrow} t.$$

Proof. This is the lemma 3.7 in [14]. $\hat{E}$ is the closure of any axiomatization of E with respect to the following three rules:

$$\frac{g=d \;\; l=r}{d\sigma=r\sigma} \quad \text{if } l, g \notin \mathcal{V} \text{ and } \sigma = mgu(l, g)$$

$$\frac{x=d \;\; y=r}{d=r\{y:=x\}} \quad \text{if } y \in \mathcal{V} \text{ and } x \in \mathcal{C} \cup \mathcal{V}$$

$$\frac{g[a]=d \;\; a=b}{g[b]=d} \quad \text{if } a, b \in \mathcal{C}$$

where $\{y := x\}$ is the substitution assigning the term x to the variable y, $\mathcal{C}$ is the set of constant symbols in Σ, $\mathcal{V}$ is the set of variables, and Σ is the signature (i.e., the set of all function symbols in our theory). □

From now on we assume that all shallow presentations we use in this paper are syntactic, with all equations of depth at most one.

Definition 3.7. A formula ϕ is *almost flat* if the following syntactic properties are satisfied:

- ϕ is in prenex normal form with prefix $\exists x_1 \ldots \exists x_k \ldots \forall y_1 \ldots \forall y_m$

- all predicate symbols are unary
- for each atom $p(t)$ occurring in ϕ t is either a variable x_i or y_j or a term $f(y_1, \ldots, y_n)$ for some $n \leq m$

We say that ϕ is *flat* if it is almost flat with $k = 0$ (i.e., with prefix $\forall y_1 \ldots \forall y_m$)

Note that if we replace each existential variable in an almost flat formula by a new constant symbol then we get a flat formula. Flat formulae were introduced by L. Bachmair, H. Ganzinger and U. Waldmann in their paper [9], and used in the reduction of the problem of consistency of set constraints to monadic class. In this section we use similar method to show that set constraints in linear shallow theories may be reduced to monadic class.

Definition 3.8. The monadic class is the class of first order formulae without function symbols, with unary (i.e., monadic) predicates only.

If we skolemize a monadic formula in a prenex form, we get a flat formula. If we skolemize an almost flat formula, we get a formula which is a skolemized version of some monadic formula. It is well-known that a formula is satisfiable if and only if its skolemized version is satisfiable. As a corollary we get

Lemma 3.9. *An almost flat formula is satisfiable if and only if the corresponding monadic formula is satisfiable.*

3.2 Reduction to almost flat formulae

In this section we define an almost flat formula corresponding to a given system of set constraints and prove that (under some assumptions, see theorems 3.11 and 3.20) the system is consistent in EQ if and only if the formula has a model over the Herbrand universe.

The idea is the following. For a given system SC of set constraints in a theory EQ we write an almost flat formula φ. This formula contains some information about both EQ and SC. Then we look for a Herbrand model of this formula (i.e., a model with the domain being the Herbrand universe) — at this moment we ignore EQ for a while. When we find such a model, it may happen that it is not compatible with EQ (see the definition 3.10 and the example 3.3 below), but then we prove that (under some assumptions) any model of φ can be transformed to a compatible model (lemma 3.19), from which we can get (by identifying terms equal in EQ) a solution of the system SC.

Let Σ be given signature and let $\mathcal{C}$ denote the set of constant symbols in Σ. Let EQ be a (nontrivial) set of linear shallow equations. Let SC be a system $E_1 \not\subseteq E'_1, \ldots, E_k \not\subseteq E'_k, E_{k+1} \subseteq E'_{k+1}, \ldots, E_{k+k'} \subseteq E'_{k+k'}$ of mixed (positive and negative) set constraints over Σ and let $E(SC)$ denote the set of all set expressions occurring in SC, the expressions $E_i \cap \overline{E'_i}$ for $i = 1, \ldots, k$ and the expressions c for each constant symbol $c \in \mathcal{C}$ (recall that c denotes the set $\{c\}$). For each set expression $E \in E(SC)$ we introduce a predicate symbol P_E.

Now we shall define a flat formula corresponding to the system SC.

For each expression $f(E_1, \ldots, E_n) \in E(SC)$, each function symbol g occurring in SC and each equation $e \in EQ$ we define a formula $\varphi^{g(y_1,\ldots,y_{arity(g)})}_{f(E_1,\ldots,E_n)}(e)$. The idea behind this formula is the following. We want to think about terms from the Herbrand universe $\mathcal{H}$ instead of elements of the quotient set $\mathcal{H}/{=_{EQ}}$ — this is because in the case of $\mathcal{H}$ we can use almost flat formulae

and we can replace them with monadic formulae, which leads us to pure monadic logic without function symbols. So we want to define $P_{f(E_1,\ldots,E_n)}$ in such a way, that it corresponds to the set $f(E_1,\ldots,E_n)$, that is, this predicate holds exactly on terms which are equal in EQ to terms of the form $f(t_1,\ldots,t_n)$ with $q(t_i)$ belonging to E_i, where q is the quotient mapping from $\mathcal{H}$ to $\mathcal{H}/=_{EQ}$. In the empty theory this is done by the formulae $P_{f(E_1,\ldots,E_n)}(f(y_1,\ldots,y_n)) \leftrightarrow P_{E_1}(y_1) \wedge \ldots \wedge P_{E_n}(y_n)$ and $P_{f(E_1,\ldots,E_n)}(g(y_1,\ldots,y_m)) \leftrightarrow false$. In shallow theories these formulae are replaced by the formulae $\varphi^{g(y_1,\ldots,y_{arity(g)})}_{f(E_1,\ldots,E_n)}(e)$. Unfortunately we are not able to say that two terms are equal in EQ using almost flat formulae, since the order of the variables in terms is fixed and we do not have equality predicate in the language. Using these formulae we want to put to the set $q^{-1}(f(E_1,\ldots,E_n))$ as many terms that should belong there as we can. The formula $\varphi^{g(y_1,\ldots,y_{arity(g)})}_{f(E_1,\ldots,E_n)}(e)$ puts $g(y_1,\ldots,y_{arity(g)})$ in the set $q^{-1}(f(E_1,\ldots,E_n))$ if e is the distinguished (see lemma 3.6) equation in the proof of equality between $g(y_1,\ldots,y_{arity(g)})$ and $f(t_1,\ldots,t_n)$. It is defined in the following way, depending on the form of the equation e:

- $\varphi^{g(y_1,\ldots,y_m)}_{f(E_1,\ldots,E_n)}(x = g(v_1,\ldots,x,\ldots,v_m)) =$

$$(\bigwedge_{j:v_j\in\mathcal{C}} P_{v_j}(y_j) \wedge P_{f(E_1,\ldots,E_n)}(y_i))$$

where i is the position of x in $g(v_1,\ldots,x,\ldots,v_m)$, and g is any function symbol different from f, of arity m.

- $\varphi^{g(y_1,\ldots,y_m)}_{f(E_1,\ldots,E_n)}(x = f(v_1,\ldots,x,\ldots,v_n)) =$

$$(\bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(g(y_1,\ldots,y_m)) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{x\}} \exists z P_{E_j}(z))$$

where i is the position of x in $f(v_1,\ldots,x,\ldots,v_n)$.

- $\varphi^{g(y_1,\ldots,y_m)}_{f(E_1,\ldots,E_n)}(g(u_1,\ldots,u_m) = f(v_1,\ldots,v_n)) =$

$$(\bigwedge_{j:u_j\in\mathcal{C}} P_{u_j}(y_j) \wedge \bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{j_k}}(y_{i_k}) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{u_1,\ldots,u_m\}} \exists z P_{E_j}(z))$$

where $u_{i_1}\ldots u_{i_l}$ and $v_{j_1}\ldots v_{j_l}$ are all the variables occurring on both sides of the equation $f(u_1,\ldots,u_n) = f(v_1,\ldots,v_n)$, respectively, with $u_{i_k} = v_{j_k}$, and g is any function symbol different from f, of arity m.

- $\varphi^{f(y_1,\ldots,y_n)}_{f(E_1,\ldots,E_n)}(x = f(v_1,\ldots,x,\ldots,v_n)) =$

$$\begin{array}{c}(\bigwedge_{j:v_j\in\mathcal{C}} P_{v_j}(y_j) \wedge P_{f(E_1,\ldots,E_n)}(y_i)) \ \vee \\ \vee \ (\bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(f(y_1,\ldots,y_n)) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{x\}} \exists z P_{E_j}(z))\end{array}$$

where i is the position of x in $f(v_1,\ldots,x,\ldots,v_n)$.

- $\varphi^{f(y_1,\ldots,y_n)}_{f(E_1,\ldots,E_n)}(f(u_1,\ldots,u_n) = f(v_1,\ldots,v_n)) =$

$$(\bigwedge_{j:u_j\in\mathcal{C}} P_{u_j}(y_j) \wedge \bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{j_k}}(y_{i_k}) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{u_1,\ldots,u_n\}} \exists z P_{E_j}(z)) \ \vee$$

$$\vee \ (\bigwedge_{j:v_j \in \mathcal{C}} P_{v_j}(y_j) \wedge \bigwedge_{j:u_j \in \mathcal{C}} P_{E_j}(u_j) \wedge \bigwedge_{k=1}^{l} P_{E_{i_k}}(y_{j_k}) \wedge \bigwedge_{j:u_j \in \mathcal{V}-\{v_1,\ldots,v_n\}} \exists z P_{E_j}(z))$$

where $u_{i_1} \ldots u_{i_l}$ and $v_{j_1} \ldots v_{j_l}$ are all the variables occurring on both sides of the equation $f(u_1,\ldots,u_n) = f(v_1,\ldots,v_n)$, respectively, with $u_{i_k} = v_{j_k}$.

- $\varphi^{g(y_1,\ldots,y_m)}_{f(E_1,\ldots,E_n)}(e) = false$ if e is not of any of the forms above.

Example 3.2. $\varphi^{f(y_1,\ldots,y_4)}_{f(E_1,\ldots,E_4)}(f(a,x,b,y){=}f(y,d,b,x)) = (P_a(y_1){\wedge}P_b(y_3){\wedge}P_{E_2}(d){\wedge}P_{E_3}(b){\wedge}P_{E_1}(y_4){\wedge} P_{E_4}(y_2)) \quad \vee \quad (P_d(y_2) \wedge P_b(y_3) \wedge P_{E_1}(a) \wedge P_{E_3}(b) \wedge P_{E_2}(y_4) \wedge P_{E_4}(y_1))$. This formula says that $f(y_1,\ldots,y_4) \in f(E_1,\ldots,E_4)$ if either $f(y_1,\ldots,y_4)$ is of the form $f(a,y_2,b,y_4)$ and $f(y_4,d,b,y_2) \in f(E_1,\ldots,E_4)$ or $f(y_1,\ldots,y_4)$ is of the form $f(y_1,d,b,y_4)$ and $f(a,y_4,b,y_1) \in f(E_1,\ldots,E_4)$.

Let φ_{SC} be the formula $\exists x_1(P_{E_1 \cap \overline{E'_1}}(x_1) \wedge \ldots \exists x_k P_{E_k \cap \overline{E'_k}}(x_k)) \wedge (P_{E_{k+1}}(y_1) \to P_{E'_{k+1}}(y_1)) \wedge \ldots \wedge (P_{E_{k+k'}}(y_1) \to P_{E'_{k+k'}}(y_1))$. Let $\varphi_{E(SC)}$ be the formula

$$\bigwedge_{(E_1 \cup E_2) \in E(SC)} P_{E_1 \cup E_2}(y_1) \leftrightarrow (P_{E_1}(y_1) \vee P_{E_2}(y_1)) \wedge \bigwedge_{\overline{E} \in E(SC)} P_{\overline{E}}(y_1) \leftrightarrow (\neg P_E(y_1)) \wedge$$

$$\wedge \bigwedge_{(E_1 \cap E_2) \in E(SC)} P_{E_1 \cap E_2}(y_1) \leftrightarrow (P_{E_1}(y_1) \wedge P_{E_2}(y_1)) \wedge (P_\top(y) \leftrightarrow true) \wedge (P_\bot(y) \leftrightarrow false),$$

where the last two conjuncts are present only if respectively $\top$ and $\bot$ are members of $E(SC)$. Let $\varphi_{\mathcal{C}}$ be the formula $\bigwedge_{E \in E(SC),\ (a=c) \in EQ} P_E(a) \leftrightarrow P_E(c)$

Now let

$$\varphi = \forall y_1 \ldots \forall y_m \varphi_{SC} \wedge \varphi_{E(SC)} \wedge \varphi_{\mathcal{C}} \wedge \bigwedge_{f(E_1 \ldots E_n) \in E(SC), g \in \Sigma} \Bigg(P_{f(E_1 \ldots E_n)}(f(y_1,\ldots,y_n)) \leftrightarrow \left(\bigwedge_{i=1}^{n} P_{E_i}(y_i) \vee \bigvee_{e \in EQ} \varphi^{f(y_1,\ldots,y_n)}_{f(E_1 \ldots E_n)}(e) \right)$$

$$\wedge \ P_{f(E_1 \ldots E_n)}(g(y_1,\ldots,y_{arity(g)})) \leftrightarrow \left(\bigvee_{e \in EQ} \varphi^{g(y_1,\ldots,y_{arity(g)})}_{f(E_1 \ldots E_n)}(e) \right) \Bigg)$$

where m is the maximal arity of a function symbol occurring in SC.

The formula φ is equivalent to an almost flat formula. It is not almost flat itself, because it is not in prenex normal form. We prove this equivalence as follows: First, we remove $\leftrightarrow$ connectives (e.g. we can convert φ to conjunctive normal form, but treating each occurrence of $\exists z P_E(z)$ as an atom). Then, for each positive occurrence of $\exists z P_E(z)$ we introduce a fresh existential variable x_i (i is here a unique index of the occurrence) and replace this occurrence with $\exists x_i P_E(x_i)$. For each negative occurrence $\neg\exists z P_E(z)$ we introduce a fresh universal variable z_j and replace $\neg\exists z P_E(z)$ with $\forall z_j \neg P_E(z_j)$. Then, since these fresh variables occur only once in the whole formula, we can move the quantifiers anywhere in the prefix, getting e.g. prefix $\exists x_1 \ldots \exists x_k \forall y_1 \ldots \forall y_m \forall z_1 \ldots \forall z_n$. Now what we get is an almost flat formula.

Definition 3.10. We say that an interpretation $\mathcal{I}$ is compatible with EQ if terms equal in EQ are interpreted in the same way in $\mathcal{I}$; more formally, if for any two terms t, s and any predicate symbol P, $t =_{EQ} s$ implies $\mathcal{I}(P)(t) = \mathcal{I}(P)(s)$

Example 3.3. Let $\Sigma = \{a, f(\cdot), g(\cdot)\}$, $EQ = \{f(x) = g(x)\}$ and SC be the system consisting of one constraint $f(X) \subseteq \top$. Then $E(SC) = \{a, X, f(X), \top\}$, and φ is the following formula:

$$\forall y(P_{f(X)}(y) \to P_\top(y)) \wedge (P_\top(y) \leftrightarrow true) \wedge (P_a(a) \leftrightarrow true) \wedge (P_a(f(y)) \leftrightarrow false) \wedge$$

$$\wedge (P_a(g(y)) \leftrightarrow false) \wedge (P_{f(X)}(f(y)) \leftrightarrow P_X(y)) \wedge (P_{f(X)}(g(y)) \leftrightarrow P_X(y)).$$

Consider the following model of φ: $P_a(a), P_X(f(a)), P_{f(X)}(f(f(a)), P_{f(X)}(g(f(a))$ and $P_\top(t)$ for each term t. This model is not compatible with EQ because $P_X(f(a)) \not\Leftrightarrow P_X(g(a))$ despite $f(a) =_{EQ} g(a)$. Using lemma 3.19 we can transform it to the following, compatible with EQ, model: $P_a(a)$, $P_X(f(a))$, $P_X(g(a))$, $P_{f(X)}(f(f(a))$, $P_{f(X)}(f(g(a))$, $P_{f(X)}(g(f(a))$, $P_{f(X)}(g(g(a))$ and $P_\top(t)$ for each term t.

Remark. We say that a predicate P is compatible with EQ if $t =_{EQ} s$ implies $P(t) = P(s)$. We are able to prove that if predicates $P_{E_1}, \ldots, P_{E_n}$ are compatible with EQ, then so is $P_{f(E_1,\ldots,E_n)}$. If we could start the induction, we would have an inductive proof that any model of φ is compatible with EQ. In fact the formula $\varphi_{\mathcal{C}}$ gives that if E is a constant symbol then P_E is compatible with EQ. But, as the example above shows, nothing can be said if E is a set variable.

Theorem 3.11. *If SC is consistent in EQ then φ has a model over the Herbrand Universe $\mathcal{H}$ over Σ, compatible with EQ.*

Proof. Suppose SC is consistent in EQ and let $\mathcal{S}$ be its solution. Define an interpretation of predicate symbols P_E for $E \in E(SC)$ in the following way: $P_E(t) \leftrightarrow (q(t) \in \mathcal{S}(E))$ where q is the quotient mapping from $\mathcal{H}$ to $\mathcal{H}/=_{EQ}$. We will show that this interpretation (obviously compatible with EQ) is a model of φ.

Clearly $\varphi_{SC} \leftrightarrow true$ in this interpretation, since for positive constraints $E \subseteq E'$ we have

$$P_E(t) \leftrightarrow (q(t) \in \mathcal{S}(E)) \to (q(t) \in \mathcal{S}(E')) \leftrightarrow P_{E'}(t)$$

and for negative constraints $E \not\subseteq E'$ we have

$$\mathcal{S}(E) \not\subseteq \mathcal{S}(E') \leftrightarrow \exists x(x \in \mathcal{S}(E) \cap \overline{\mathcal{S}(E')}) \leftrightarrow \exists t(q(t) \in \mathcal{S}(E \cap \overline{E'})) \leftrightarrow \exists t P_{E \cap \overline{E'}}(t).$$

It is also obvious that $\varphi_{E(SC)} \leftrightarrow true$ and $\varphi_{\mathcal{C}} \leftrightarrow true$.

Now we will prove that the equivalence

$$P_{f(E_1 \ldots E_n)}(f(y_1, \ldots, y_n)) \leftrightarrow \left(\bigwedge_{i=1}^{n} P_{E_i}(y_i) \vee \bigvee_{e \in EQ} \varphi_{f(E_1 \ldots E_n)}^{f(y_1,\ldots,y_n)}(e) \right)$$

holds.

First we prove the $\to$ implication. Suppose $(P_{f(E_1 \ldots E_n)}(f(y_1, \ldots, y_n))$ holds.

From the definition of the interpretation we have $P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n))$ is equivalent to $(q(f(y_1, \ldots, y_n)) \in \mathcal{S}(f(E_1, \ldots, E_n)))$, so $P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n))$ holds if and only if there exist terms $t_1, \ldots, t_n$ such that $\bigwedge_{i=1}^{n} q(t_i) \in \mathcal{S}(E_i) \wedge f(t_1, \ldots, t_n) =_{EQ} f(y_1, \ldots, y_n)$.
Now we consider two cases:

Case 1 $f(t_1, \ldots, t_n) \overset{(\neq\Lambda)^*}{\longleftrightarrow} f(y_1, \ldots, y_n)$.

In this case $q(y_i) = q(t_i) \in \mathcal{S}(E_i)$ for $i = 1, \ldots, n$, so $\bigwedge_{i=1}^{n} P_{E_i}(y_i)$ holds.

Case 2 $\exists s, s' :\ f(t_1, \ldots, t_n) \overset{(\neq\Lambda)^*}{\longleftarrow} s \overset{\Lambda}{\longleftrightarrow} s' \overset{(\neq\Lambda)^*}{\longleftarrow} f(y_1, \ldots, y_n)$.

Since $f(t_1, \ldots, t_n) \overset{(\neq\Lambda)^*}{\longleftarrow} s$ and $s' \overset{(\neq\Lambda)^*}{\longleftarrow} f(y_1, \ldots, y_n))$, s and s' must begin with the symbol f, so $s = f(s_1, \ldots, s_n)$ and $s' = f(s'_1, \ldots, s'_n)$ with $q(s_1) = q(t_1) \in \mathcal{S}(E_1), \ldots, q(s_n) = q(t_n) \in \mathcal{S}(E_n)$ and $q(s'_1) = q(y_1), \ldots, q(s'_n) = q(y_n)$.

Now we will consider several cases, depending on the rule $(l = r) \in EQ$ used in the proof step $s \overset{\Lambda}{\longleftrightarrow} s'$:

Case 2.1 *the used rule is of the form* $f(u_1, \ldots, u_n) = f(v_1 \ldots v_n)$.

There are two possibilities:

Case 2.1.1 *there exists a substitution* σ *such that* $\sigma(f(u_1, \ldots, u_n)) = s$ *and* $\sigma(f(v_1, \ldots, v_n)) = s'$.

Since EQ is shallow, u_i and v_i for $i = 1 \ldots n$ are either constant symbols or variables. If u_i is a constant then $q(u_i) = q(\sigma(u_i)) = q(s_i) = q(t_i) \in \mathcal{S}(E_i)$, so $P_{E_i}(u_i)$ holds. If v_j is a constant then $q(v_j) = q(\sigma(v_j)) = q(s'_j) = q(y_j)$, so $P_{v_j}(y_j)$ holds.

Let $u_{i_1} = v_{j_1}, \ldots, u_{i_l} = v_{j_l}$ be all the variables occurring both in $\{u_1, \ldots, u_n\}$ and $\{v_1, \ldots, v_n\}$ (such a sequence exists since EQ is linear). Now $q(y_{j_k}) = q(s'_{j_k}) = q(\sigma(v_{j_k})) = q(\sigma(u_{i_k})) = q(s_{i_k}) = q(t_{i_k}) \in \mathcal{S}(E_{i_k})$, so $P_{E_{i_k}}(y_{j_k})$ holds.

Since for all i, $P_{E_i}(t_i)$ holds, we also have that $\bigwedge_{j:u_j \in \mathcal{V} - \{v_1, \ldots, v_n\}} \exists z P_{E_j}(z)$ holds.

We have just shown that
$\bigwedge_{j:v_j \in \mathcal{C}} P_{v_j}(y_j) \wedge \bigwedge_{j:u_j \in \mathcal{C}} P_{E_j}(u_j) \wedge \bigwedge_{k=1}^{l} P_{E_{i_k}}(y_{j_k}) \wedge \bigwedge_{j:u_j \in \mathcal{V} - \{v_1, \ldots, v_n\}} \exists z P_{E_j}(z)$ holds.

Case 2.1.2 *there exists a substitution* σ *such that* $\sigma(f(u_1, \ldots, u_n)) = s'$ *and* $\sigma(f(v_1, \ldots, v_n)) = s$.

This case is symmetric to the previous one; here we are able to show that
$\bigwedge_{j:u_j \in \mathcal{C}} P_{u_j}(y_j) \wedge \bigwedge_{j:v_j \in \mathcal{C}} P_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{j_k}}(y_{i_k}) \wedge \bigwedge_{j:v_j \in \mathcal{V} - \{u_1, \ldots, u_n\}} \exists z P_{E_j}(z)$.

In both cases (2.1.1 and 2.1.2) $\varphi^{f(y_1, \ldots, y_n)}_{f(E_1, \ldots, E_n)}(f(u_1, \ldots, u_n) = f(v_1 \ldots v_n))$ holds, which ends the proof of the $\rightarrow$ implication in the case 2.1.

Case 2.2 *the used rule is of the form* $x = f(v_1, \ldots, x, \ldots, v_n)$.

Let i be the position of x in $f(v_1, \ldots, x, \ldots, v_n)$. Again there are two posibilities: either there exists a substitution σ such that $\sigma(x) = s$ and $\sigma(f(v_1, \ldots, x, \ldots v_n)) = s'$ and then $\bigwedge_{v_j \in \mathcal{C}} P_{v_j}(y_j) \wedge P_{f(E_1, \ldots, E_n)}(y_i)$ holds, or there exists a substitution σ such that $\sigma(x) = s'$ and $\sigma(f(v_1, \ldots, x, \ldots v_n)) = s$, and then $\bigwedge_{j:v_j \in \mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(f(y_1, \ldots, y_n)) \wedge \bigwedge_{j:v_j \in \mathcal{V} - \{x\}} \exists z P_{E_j}(z)$ holds.

In both cases $\varphi^{f(y_1, \ldots, y_n)}_{f(E_1, \ldots, E_n)}(x = f(v_1 \ldots x, \ldots, v_n))$ holds, which gives the proof of the $\rightarrow$ implication.

Now we will prove the $\leftarrow$ implication. Suppose

$$\left((P_{E_1}(y_1) \wedge \ldots P_{E_n}(y_n)) \vee \bigvee_{e \in EQ} \varphi^{f(y_1, \ldots, y_n)}_{f(E_1 \ldots E_n)}(e) \right)$$

holds. Again we have to consider several cases, depending on which disjunct holds.

Case 1 *$P_{E_1}(y_1) \wedge \ldots P_{E_n}(y_n)$ holds.*

Then $q(y_i) \in \mathcal{S}(E_i)$ for all i, so from the definition of $f(E_1, \ldots, E_n)$ we have $q(f(y_1, \ldots, y_n)) \in \mathcal{S}(f(E_1, \ldots, E_n))$, so $P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n))$ holds.

Case 2 *some of $\varphi^{f(y_1,\ldots,y_n)}_{f(E_1,\ldots,E_n)}(t = s)$ holds.*

Case 2.1 *$t = s$ is of the form $f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)$.*

Case 2.1.1

$$\bigwedge_{j:u_j \in \mathcal{C}} P_{u_j}(y_j) \wedge \bigwedge_{j:v_j \in \mathcal{C}} P_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{j_k}}(y_{i_k}) \wedge \bigwedge_{j:v_j \in \mathcal{V}-\{u_1,\ldots,u_n\}} \exists z P_{E_j}(z)$$

holds, where $u_{i_1} \ldots u_{i_l}$ and $v_{j_1} \ldots v_{j_l}$ are all the variables occurring on both sides of the equation $f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)$, respectively, with $u_{i_k} = v_{j_k}$.

In this case, since $P_{u_j}(y_j)$ holds whenever $u_j \in \mathcal{C}$, $q(y_j) \in \mathcal{S}(u_j)$, which means that $u_j =_{EQ} y_j$. Hence $f(y_1, \ldots, y_n) =_{EQ} \sigma(f(u_1, \ldots, u_n))$, where σ is the substitution binding each variable u_k to the term y_k, and each variable $v_k \notin \{u_1, \ldots, u_n\}$ to a term t_k such that $P_{E_k}(t_k)$ holds (existence of such a term is ensured by the formula $\exists z P_{E_k}(z)$). Using the rule $f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)$ we can prove that $f(y_1, \ldots, y_n) =_{EQ} \sigma(f(v_1, \ldots, v_n))$ (note that since $u_{i_k} = v_{j_k}$, σ maps each v_{j_k} to y_{i_k}).

Now $q(\sigma(v_{j_k})) \in \mathcal{S}(E_{j_k})$, $q(\sigma(v_j)) \in \mathcal{S}(E_j)$, so $q(\sigma(f(v_1, \ldots v_n))) \in \mathcal{S}(f(E_1, \ldots E_n))$. Since $q(f(y_1, \ldots, y_n)) = q(\sigma(f(v_1, \ldots, v_n)))$, $P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n))$ holds.

Case 2.1.2

$$\bigwedge_{j:v_j \in \mathcal{C}} P_{v_j}(y_j) \wedge \bigwedge_{j:u_j \in \mathcal{C}} P_{E_j}(u_j) \wedge \bigwedge_{k=1}^{l} P_{E_{i_k}}(y_{j_k}) \wedge \bigwedge_{j:u_j \in \mathcal{V}-\{v_1,\ldots,v_n\}} \exists z P_{E_j}(z)$$

holds, where $u_{i_1} \ldots u_{i_l}$ and $v_{j_1} \ldots v_{j_l}$ are all the variables occurring on both sides of the equation $f(u_1, \ldots, u_n) = f(v_1, \ldots, v_n)$, respectively, with $u_{i_k} = v_{j_k}$.

This case is symmetric to the previous one, with almost the same proof.

Case 2.2 *$t = s$ is of the form $x = f(v_1, \ldots, x, \ldots, v_n)$.*

Case 2.2.1 *$(\bigwedge_{j:v_j \in \mathcal{C}} P_{v_j}(y_j) \wedge P_{f(E_1,\ldots,E_n)}(y_i))$ holds, where i is the position of x in $f(v_1, \ldots, x, \ldots, v_n)$.*

In this case $f(y_1, \ldots, y_n) =_{EQ} \sigma(f(v_1, \ldots, y_i, \ldots, v_n)) =_{EQ} y_i$, where σ maps v_j to y_j if v_j is a variable and does not change v_j if it is a constant, so $q(f(y_1, \ldots, y_n)) = q(y_i) \in \mathcal{S}(f(E_1, \ldots, E_n))$ and $P_{f(E_1,\ldots,E_n)}(f(y_1, \ldots, y_n))$ holds.

Case 2.2.2 *$\bigwedge_{j:v_j \in \mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(f(y_1, \ldots, y_n)) \wedge \bigwedge_{j:v_j \in \mathcal{V}-\{x\}} \exists z P_{E_j}(z)$ holds, where i is the position of x in $f(v_1, \ldots, x, \ldots, v_n)$.*

In this case $f(y_1,\ldots,y_n) =_{EQ} f(\sigma(v_1),\ldots,f(y_1,\ldots,y_n),\ldots,\sigma(v_n))$, where $\sigma(v_j) = v_j$ if $v_j \in \mathcal{C}$ and $\sigma(v_j) = t_j$ for a term t_j satisfying $P_{E_j}(t_j)$ if v_j is a variable. Since $q(\sigma(v_1)) \in \mathcal{S}(E_1),\ldots$ $q(f(y_1,\ldots y_n)) \in \mathcal{S}(E_i),\ldots q(\sigma(v_n)) \in \mathcal{S}(E_n)$, $q(f(y_1,\ldots y_n)) \in \mathcal{S}(f(E_1,\ldots E_n))$.

This ends the proof of the $\leftarrow$ implication in

$$P_{f(E_1\ldots E_n)}(f(y_1,\ldots,y_n)) \leftrightarrow \left(\bigwedge_{i=1}^{n} P_{E_i}(y_i) \vee \bigvee_{e\in EQ} \varphi^{f(y_1,\ldots,y_n)}_{f(E_1\ldots E_n)}(e)\right).$$

The proof of the equivalence

$$P_{f(E_1\ldots E_n)}(g(y_1,\ldots,y_m)) \leftrightarrow \left(\bigvee_{e\in EQ} \varphi^{g(y_1,\ldots,y_m)}_{f(E_1\ldots E_n)}(e)\right)$$

is almost the same; the difference is that we have to consider less cases since in the proof $f(\ldots) \overset{\Lambda}{\longleftrightarrow} g(\ldots)$ there may be only one possible direction of using equations from EQ. In both directions of the proof case 1 is not present (f and g are different symbols, so the proof must contain a $\overset{\Lambda}{\longleftrightarrow}$ step), case 2.1.1 corresponds to the formula $\varphi^{g(\ldots)}_{f(\ldots)}(g(\ldots) = f(\ldots))$, case 2.2.2 is not present (it would correspond to $\varphi^{f(\ldots)}_{g(\ldots)}(g(\ldots) = f(\ldots)))$, case 2.2.1 corresponds to $\varphi^{g(\ldots)}_{f(\ldots)}(x = g(\ldots))$, and case 2.2.2 to $\varphi^{g(\ldots)}_{f(\ldots)}(x = f(\ldots))$.

This ends the proof of the theorem 3.11. □

Below we will prove (under some assumptions about the existential part of φ) the converse theorem to the theorem 3.11, that is if φ has (any, not necessarily compatible with EQ) model over the Herbrand Universe $\mathcal{H}$ over Σ then SC is consistent in EQ. The idea is to change the existing model into a model compatible with EQ. This is done by choosing for each equivalence class of the relation $=_{EQ}$ a fixed term representing this class, and then interpreting every term from this class in the same way as the representative is interpreted.

Suppose φ has a model $\mathcal{I}$. We shall identify $\mathcal{I}$ with a function $\Phi : \mathcal{H} \to 2^{E(SC)}$, where $2^{E(SC)}$ is the set of functions from $E(SC)$ to $\{0,1\}$, and $\Phi(t)(E) = 1$ iff $\mathcal{I}(P_E(t)) = true$ (intuitively, if $t \in E$). Recall that q is the quotient mapping from $\mathcal{H}$ to $\mathcal{H}/=_{EQ}$. Let $r : \mathcal{H} \to \mathcal{H}$ be any fixed function giving for each term t such a representative of the set $q^{-1}(t)$, that

- if $c \in \mathcal{C}$ then $r(c) \in \mathcal{C}$
- if $r(t) \notin \mathcal{C}$, then $r(t)$ is a term of the form $f(r(t_1),\ldots,r(t_n))$.

Note that now $t =_{EQ} s$ iff $q(t) = q(s)$ iff $r(t) = r(s)$. Such function r exists, since it can be defined inductively as follows: take the minimal term t such that $r(t)$ is not defined yet; if t is a constant symbol then put $r(t) = t$, if t is a term of the form $f(t_1,\ldots,t_n)$ then put $r(t) = f(r(t_1),\ldots,r(t_n))$, and for each term s such that $s =_{EQ} t$ put $r(s) = r(t)$.

Now we shall define a new function Φ', which corresponds to another model of φ and will give us a solution of SC. Let

$$\Phi'(f(t_1,\ldots,t_n)) = \Phi(r(f(t_1,\ldots,t_n))).$$

From now on we assume that Φ is fixed and φ evaluates to $true$. We use the following convention here: P'_E denotes the value of the predicate symbol P_E under the interpretation

defined by Φ', φ' is the interpretation of φ under Φ', and P_E denotes the value of the predicate symbol P_E under the interpretation defined by Φ (thus $P'_E(t) = P_E(r(t))$).

Lemma 3.12. *For each constant symbol c and each term t, $P_c(t) \leftrightarrow true$ if and only if $t =_{EQ} c$*

Proof. (induction on the structure of t).

- $t = c$
  $P_c(c) \leftrightarrow \varphi_c^c \leftrightarrow true$
- $t = a$ for some $a \in \mathcal{C}$, $a \neq c$
  $P_c(a) \leftrightarrow \bigvee_{u=v \in EQ} \varphi_c^a(u = v) \leftrightarrow (a = c) \in EQ$
- $t = g(y_1, \ldots, y_m)$

$$P_c(t) \leftrightarrow \bigvee_{x=g(v_1,\ldots,x,\ldots,v_m)\in EQ} \varphi_c^{g(y_1,\ldots,y_m)}(x = g(v_1, \ldots, x, \ldots, v_m)) \quad \vee$$
$$\vee \quad \bigvee_{c=g(u_1,\ldots,u_m)\in EQ} \varphi_c^{g(y_1,\ldots,y_m)}(c = g(u_1, \ldots, u_m)).$$

By the definition of $\varphi_c^{g(\ldots)}$ this is equal to

$$\bigvee_{x=g(v_1,\ldots,x,\ldots,v_m)\in EQ}(\bigwedge_{v_j\in\mathcal{C}} P_{v_j}(y_j) \wedge P_c(y_i)) \quad \vee$$
$$\vee \quad \bigvee_{c=g(u_1,\ldots,u_m)\in EQ}(\bigwedge_{u_j\in\mathcal{C}} P_{u_j}(y_j)),$$

which, by the induction assumption is equivalent to

$$\bigvee_{x=g(v_1,\ldots,x,\ldots,v_m)\in EQ}(\bigwedge_{v_j\in\mathcal{C}} y_j =_{EQ} v_j \wedge y_i =_{EQ} c) \quad \vee$$
$$\vee \quad \bigvee_{c=g(u_1,\ldots,u_m)\in EQ}(\bigwedge_{u_j\in\mathcal{C}} u_j =_{EQ} (y_j)).$$

Finally, this is equivalent to $g(y_1 \ldots y_n) =_{EQ} c$.

□

Lemma 3.13. *If $f(t_1, \ldots, t_n) = r(t)$ for some term t, then for each i we have $t_i \neq_{EQ} f(t_1, \ldots, t_n)$.*

Proof. Suppose $t_i =_{EQ} f(t_1, \ldots, t_n)$. Then $r(t_i) = r(f(t_1, \ldots, t_n)) = f(r(t_1), \ldots, r(t_n))$. But $r(f(t_1, \ldots, t_n)) = f(t_1, \ldots, t_n)$, so $t_i = r(t_i) = f(t_1, \ldots, t_n)$, which is a contradiction. □

Lemma 3.14. *For each term $g(t_1, \ldots, t_m)$ either $r(g(t_1, \ldots, t_m)) = g(r(t_1), \ldots, r(t_m))$ or $r(g(t_1, \ldots, t_m)) \overset{\Lambda}{\longleftrightarrow} g(r(t_1), \ldots, r(t_m))$.*

Proof. Suppose $r(g(t_1, \ldots, t_m)) \neq g(r(t_1), \ldots, r(t_m))$.
Let $r(g(t_1, \ldots, t_m)) = h(s_1, \ldots, s_k)$ (this includes the case of $g = h$). Then there exists an equational proof of equality in EQ between $h(s_1, \ldots, s_k)$ and $g(r(t_1), \ldots, r(t_m))$ of the form

$$g(r(t_1), \ldots, r(t_m)) \overset{(\neq\Lambda)^*}{\longleftrightarrow} g(t'_1, \ldots, t'_m) \overset{\Lambda}{\longleftrightarrow} h(s'_1, \ldots, s'_k) \overset{(\neq\Lambda)^*}{\longleftrightarrow} h(s_1, \ldots, s_k).$$

Now we consider three cases:

Case 1 *The equation used in the step Λ was of the form $x = h(v_1, \ldots, x, \ldots, v_k)$.*

In this case $s_i =_{EQ} s'_i =_{EQ} g(r(t_1), \ldots, r(t_m)) =_{EQ} r(g(t_1, \ldots, t_m)) = h(s_1, \ldots, s_k)$. This contradicts lemma 3.13, which means that in the $\overset{\Lambda}{\longleftrightarrow}$ step no equation of the form $x = h(v_1, \ldots, x, \ldots, v_k)$ could be used.

Case 2 *The equation used in the step Λ was of the form $g(v_1, \ldots, x, \ldots, v_m) = x$.*

Then, since for $j \neq i$ we have $r(t_j) =_{EQ} t'_j = \sigma(v_j)$, either $r(t_j)$ is a constant symbol or v_j is a variable different from x, and there is an equation $g(v'_1, \ldots, x, \ldots, v'_m) = x$ in EQ, where $v'_j = r(t_j)$ if $v_j \in \mathcal{C}$ and $v'_j = v_j$ if $v_j \in \mathcal{V}$. Hence $g(r(t_1), \ldots, r(t_m)) \overset{\Lambda}{\longleftrightarrow} r(t_i) = r(g(r(t_1), \ldots, r(t_m))) = r(g(t_1, \ldots, t_m))$.

Case 3 *The equation used in the step Λ was of the form $g(u_1, \ldots, u_m) = h(v_1, \ldots, v_k)$, with $u_{i_1} = v_{j_1}, \ldots, u_{i_l} = v_{j_l}$ being all the variables occurring on both sides of this equation.*

Let $u'_i = r(u_i)$ and $v'_j = r(v'_j)$ if u_i and v_j are constant symbols, and $u'_i = u_i$, $v'_j = v_j$ id u_i, v_j are variables. Since EQ contains all equations of the form $c = r(c)$ for every constant symbol c, it must also contain, by lemma 3.6, the equation $g(u'_1, \ldots, u'_m) = h(v'_1, \ldots, v'_k)$. Let σ be the substitution assigning t'_{i_j} to u_{i_j}. Then we have $r(t_{i_j}) =_{EQ} t'_{i_j} = \sigma(u_{i_j}) = \sigma(v_{j_j}) = s'_{j_j} =_{EQ} s_{j_j} = r(s_{j_j})$. Hence $r(t_{i_j}) = s_{j_j}$ and $g(r(t_1), \ldots, r(t_m)) \overset{\Lambda}{\longleftrightarrow} h(s_1, \ldots, s_k)$ by the equation $g(u'_1, \ldots, u'_m) = h(v'_1, \ldots, v'_k)$. □

Lemma 3.15. *For each term $g(t_1, \ldots, t_m)$ and each expression $f(E_1, \ldots, E_n) \in E(SC)$, if $P_{f(E_1,\ldots,E_n)}(r(g(t_1, \ldots, t_m)))$ holds, then there exists a term $f(s_1, \ldots, s_n)$ such that either $r(g(t_1, \ldots, t_m)) = f(s_1, \ldots, s_n)$ or $r(g(t_1, \ldots, t_m)) \overset{\Lambda}{\longleftrightarrow} f(s_1, \ldots, s_n)$, and $P_{E_j}(s_j)$ holds for every $j \in \{1, \ldots, n\}$.*

Proof. Suppose $r(g(t_1, \ldots, t_m))$ is not of the form $f(s_1, \ldots, s_n)$ with $P_{E_j}(s_j)$ for every j. Let $r(g(t_1, \ldots, t_m)) = h(s_1, \ldots, s_k)$. Then $P_{f(E_1,\ldots,E_n)}h(s_1, \ldots, s_k) \leftrightarrow \bigvee_{e \in EQ} \varphi^{h(s_1,\ldots,s_k)}_{f(E_1,\ldots,E_n)}(e)$. Suppose $\varphi^{h(s_1,\ldots,s_k)}_{f(E_1,\ldots,E_n)}(e)$ holds for some $e \in EQ$. By the definition of $\varphi^{h(\ldots)}_{f(\ldots)}$, e must be of one of the following three forms: $x = h(v_1, \ldots, x, \ldots, v_k)$, or $f(u_1, \ldots, u_n) = h(v_1, \ldots, v_k)$ or $f(u_1, \ldots, x, \ldots, u_n) = x$. By lemma 3.13 it cannot be of the form $x = h(v_1, \ldots, x, \ldots, v_k)$, so e must be either of the form $f(u_1, \ldots, u_n) = h(v_1, \ldots, v_k)$ or $f(u_1, \ldots, x, \ldots, u_n) = x$. In the first case $r(g(t_1, \ldots, t_m)) \overset{\Lambda}{\longleftrightarrow} f(\sigma(u_1), \ldots, \sigma(u_n))$ where σ maps v_j to s_j and variables u_j not in $\{v_1, \ldots, v_k\}$ to terms such that $P_{E_j}(\sigma(u_j))$ holds. In the second case $r(g(t_1, \ldots, t_m)) \overset{\Lambda}{\longleftrightarrow} f(\sigma(u_1), \ldots, r(g(t_1, \ldots, t_m)), \ldots \sigma(u_n))$ where σ maps u_j to u_j if $u_j \in \mathcal{C}$ and u_j to terms such that $P_{E_j}(\sigma(u_j))$ holds if $u_j \in \mathcal{V}$. □

Lemma 3.16. *Assume that EQ is non-erasing or, for every predicate symbol P_E, $\exists z P_E(z)$ implies $\exists z P_E(r(z))$. Then for every different function symbols f and g in Σ we have*

$$P'_{f(E_1,\ldots,E_n)}(g(t_1, \ldots, t_m)) \rightarrow \bigvee_{e \in EQ} \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e).$$

Proof. This is a technical lemma and we will have to consider many cases here. The main idea is to compose (see lemma 3.6) an equation e for which $\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e)$ holds with e' used

in the step $g(r(t_1)\ldots r(t_m)) \overset{\Lambda}{\longleftrightarrow} r(g(t_1,\ldots,t_m))$ to obtain e'' such that $\varphi^{g(r(t_1),\ldots,r(t_m))}_{f(E_1,\ldots,E_n)}(e'')$ holds.

If EQ is non-erasing then φ does not contain subformulae of the form $\exists z P_E(z)$ and we do not need the assumption that $\exists z P_E(z)$ implies $\exists z P_E(r(z))$ in the proof below — we can just skip all the fragments of the proof concerning these subformulas. So suppose EQ is not non-erasing.

By the definition of Φ' we have $P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)) \leftrightarrow P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m)))$. If $r(g(t_1,\ldots,t_m))$ equals to $g(r(t_1),\ldots,r(t_m))$ then $P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) = P_{f(E_1,\ldots,E_n)}(g(r(t_1),\ldots,r(t_m))) \rightarrow \bigvee_{e\in EQ} \varphi^{g(r(t_1),\ldots,r(t_m))}_{f(E_1,\ldots,E_n)}(e) = \bigvee_{e\in EQ} \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e)$, which ends the proof. So suppose $r(g(t_1,\ldots,t_m)) \neq g(r(t_1),\ldots,r(t_m))$.

Case 1 *$r(g(t_1,\ldots,t_m)) = h(x_1,\ldots,x_k)$ with $h \neq f$ (this includes the case of $h = g$).*

In this case

$$P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) \leftrightarrow P_{f(E_1,\ldots,E_n)}(h(x_1,\ldots,x_k)) \leftrightarrow$$
$$\leftrightarrow \bigvee_{e\in EQ} \varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(e).$$

Hence we have to prove that

$$\bigvee_{e\in EQ} \varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(e) \rightarrow \bigvee_{e\in EQ} \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e).$$

So let us suppose that $\varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(e) \leftrightarrow true$ for some equation e in EQ.

Case 1.1 *e is of the form $x = h(v_1,\ldots,x,\ldots v_k)$.*

This case is not possible since it contradicts lemma 3.13.

Case 1.2 *e is of the form $x = f(v_1,\ldots,x,\ldots,v_n)$.*

Then $\varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(x = f(v_1,\ldots x,\ldots v_n)) = \bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(h(x_1,\ldots x_k)) \wedge \bigwedge_{j:v_j\in\mathcal{V}} \exists z P_{E_j}(z)$ which (since $\varphi_{\mathcal{C}}$ evaluates to $true$, by the definition of r and by the assumption of the lemma) is equivalent to

$$\bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(r(v_j)) \wedge P_{E_i}(r(g(t_1,\ldots,t_m))) \wedge \bigwedge_{j:v_j\in\mathcal{V}} \exists z P_{E_j}(r(z)) =$$
$$= \bigwedge_{j:v_j\in\mathcal{C}} P'_{E_j}(v_j) \wedge P'_{E_i}(g(t_1,\ldots,t_m)) \wedge \bigwedge_{j:v_j\in\mathcal{V}} \exists z P'_{E_j}(z) \leftrightarrow$$
$$\leftrightarrow \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(x = f(v_1,\ldots,x,\ldots v_n)) \rightarrow \bigvee_{e\in EQ} \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e).$$

Case 1.3 *e is of the form $h(u_1,\ldots,u_k) = f(v_1,\ldots,v_n)$ with variables $u_{i_1} = v_{i'_1}, \ldots u_{i_l} = v_{i'_l}$.*

Then $\varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(h(u_1,\ldots u_k) = f(v_1,\ldots v_n)) = \bigwedge_{u_i\in\mathcal{C}} P_{u_i}(x_i) \wedge \bigwedge_{v_i\in\mathcal{C}} P_{E_i}(v_i) \wedge \bigwedge_{j=1}^{l} P_{E_{i'_j}}(x_{i_j}) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{u_1,\ldots,u_n\}} \exists z P_{E_j}(z)$.

Note that $\bigwedge_{j=1}^{n} \exists z P_{E_j}(z)$ holds. Now, by lemma 3.14, $g(r(t_1),\ldots r(t_n)) \overset{\Lambda}{\longleftrightarrow} h(x_1,\ldots,x_k)$. By lemma 3.13 no equation of the form $x = h(\ldots x,\ldots)$ could be used in this step, so there are two possibilities:

Case 1.3.1 *The equation used in the step* $g(r(t_1),\ldots,r(t_n)) \overset{\Lambda}{\longleftrightarrow} h(x_1,\ldots,x_k)$ *was of the form* $g(w_1,\ldots,x,\ldots,w_m)=x$.

Then $r(t_j)=w_j$ for $w_j\in\mathcal{C}$ and $r(t_i)=h(x_1,\ldots,x_k)$. By lemma 3.12 $P_{w_j}(r(t_j))$ holds for $w_j\in\mathcal{C}$. By the assumption of the lemma and the definition of Φ', $P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m)))$ holds, so we have

$$true \leftrightarrow \bigwedge_{j:w_j\in\mathcal{C}} P_{w_j}(r(t_j)) \wedge P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) \leftrightarrow$$

$$\leftrightarrow \bigwedge_{j:w_j\in\mathcal{C}} P'_{w_j}(t_j) \wedge P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)) = \varphi'^{g(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(g(w_1,\ldots,x,\ldots,w_m)=x)$$

$$\rightarrow \bigvee_{e\in EQ} \varphi'^{g(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(e).$$

Case 1.3.2 *The equation used in the step* $g(r(t_1),\ldots,r(t_n)) \overset{\Lambda}{\longleftrightarrow} h(x_1,\ldots,x_k)$ *was of the form* $g(w_1,\ldots,w_m)=h(u'_1,\ldots,u'_k)$.

Then, since $h(x_1,\ldots,x_k)$ is a common instance of $h(u_1,\ldots,u_k)$ and $h(u'_1,\ldots,u'_k)$, these terms are unifiable. Let $\sigma=mgu(h(u_1,\ldots,u_k),h(u'_1,\ldots,u'_k))$. Then, by lemma 3.6 the equation $g(w_1\sigma,\ldots,w_m\sigma)=f(v_1\sigma,\ldots,v_n\sigma)$ belongs to EQ. Let $w_{j_1}\sigma=v_{j'_1}\sigma,\ldots,w_{j_{l'}}\sigma=v_{j'_{l'}}\sigma$ be all the variables occurring on both sides of this equation. So now, for each $i\in\{1,\ldots,l'\}$ there exists a $j\in\{1,\ldots,l\}$ such that $w_{j_i}\sigma=v_{j'_i}\sigma=v_{i'_j}\sigma=u_{i_j}\sigma=u'_{i_j}\sigma$. Note that $j'_i=i'_j$ here.

If $v_j\sigma$ is a constant symbol, then either $v_j\sigma=v_j$ and $P_{E_j}(v_j)=P_{E_j}(v_j\sigma)\rightarrow P_{E_j}(r(v_j\sigma))\rightarrow P'_{E_j}(v_j\sigma)$, or $v_j=v_{i'_{j'}}=u_{i_{j'}}$ for some $j'\leq l$ and then $v_j\sigma=u_{i_{j'}}\sigma=x_{i_{j'}}$. Then $P_{E_{i'_{j'}}}(x_{i_{j'}})\rightarrow P_{E_{i'_{j'}}}(v_{i'_{j'}}\sigma)\rightarrow P_{E_{i'_{j'}}}(r(v_{i'_{j'}}\sigma))\rightarrow P'_{E_{i'_{j'}}}(v_{i'_{j'}}\sigma)=P'_{E_j}(v_j\sigma)$. In both cases $P'_{E_j}(v_j\sigma)\leftrightarrow true$.

If $w_j\sigma$ is a constant then $w_j\sigma=r(t_j)$ and $true\leftrightarrow P_{w_j\sigma}(r(t_j))=P'_{w_j\sigma}(t_j)$.

If $w_{j_i}\sigma$ is a variable then $w_{j_i}\sigma=u'_{i_j}\sigma$, so $r(t_{j_i})=x_{i_j}$ and $P_{E_{i'_j}}(x_{i_j})=P_{E_{j'_i}}(r(t_{j_i}))$. $P_{E_{i'_j}}(x_{i_j})$ holds by the assumption of the case 1.3, and $P_{E_{j'_i}}(r(t_{j_i}))$ is equal to $P'_{E_{j'_i}}(t_{j_i})$. Since $\bigwedge_{j=1}^n \exists z P_{E_j}(z)$ holds (remark in the case 1.3), we have $\bigwedge_{j=1}^n \exists z P'_{E_j}(z)$ holds (assumption of the lemma) which implies $\bigwedge_{j:v_j\sigma\in\mathcal{V}-\{w_1\sigma,\ldots,w_m\sigma\}}\exists z P_{E_j}(z)$ holds.

We have just shown that

$$true \leftrightarrow \bigwedge_{v_j\sigma\in\mathcal{C}} P'_{E_j}(v_j\sigma) \wedge \bigwedge_{w_j\sigma\in\mathcal{C}} P'_{w_j\sigma}(t_j) \wedge \bigwedge_{i=1}^{l'} P'_{E_{j'_i}}(t_{j_i}) \wedge \bigwedge_{j:v_j\sigma\in\mathcal{V}-\{w_1\sigma,\ldots,w_m\sigma\}} \exists z P_{E_j}(z) =$$

$$= \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(g(w_1\sigma,\ldots,w_m\sigma)=f(v_1\sigma,\ldots,v_n\sigma))$$

which implies $true\leftrightarrow\bigvee_{e\in EQ}\varphi'^{g(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(e)$.

This ends the proof of the case 1.

Case 2 $r(g(t_1,\ldots,t_m))=f(x_1,\ldots,x_n)$.

Assume $P_{f(E_1,\ldots,E_n)}(f(x_1,\ldots,x_n))$ holds. Then either $\bigwedge_{i=1}^n P_{E_i}(x_i)$ or $\bigvee_{e\in EQ}\varphi^{f(x_1,\ldots,x_n)}_{f(E_1,\ldots,E_n)}(e)$ holds. If $\bigvee_{e\in EQ}\varphi^{f(x_1,\ldots,x_n)}_{f(E_1,\ldots,E_n)}(e)$ holds, then the proof is symmetric to the proof in case 1, so suppose that $\bigwedge_{i=1}^n P_{E_i}(x_i)$ holds. By lemma 3.14 there exists $e\in EQ$ such that $g(r(t_1),\ldots,r(t_m)) \overset{\Lambda}{\underset{e}{\longleftrightarrow}} f(x_1,\ldots,x_n)$. Again we have to consider three cases:

- e cannot be of the form $x = f(\ldots x, \ldots)$, because it would contradict lemma 3.13
- if e is of the form $x = g(v_1, \ldots, x, \ldots v_m)$, then the argumentation is similar to the case 1.3.2:

  $\bigwedge_{v_i \in \mathcal{C}} P_{v_i}(v_i)$ holds. Since $\varphi_{\mathcal{C}}$ holds and by the assumption we have

  $$\bigwedge_{v_i \in \mathcal{C}} P_{v_i}(r(v_i)) \wedge P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) \leftrightarrow$$

  $$\leftrightarrow \bigwedge_{v_i \in \mathcal{C}} P'_{v_i}(v_i) \wedge P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)) = \varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e)$$

- e is of the form $g(u_1, \ldots, u_m) = f(v_1, \ldots, v_n)$

  If $u_j \in \mathcal{C}$, then $r(t_j) = u_j \in \mathcal{C}$, so $P_{u_j}(r(t_j))$ holds, which means that $P'_{u_j}(t_j)$ holds.

  If $v_j \in \mathcal{C}$ then $x_j = v_j$, and since $x_j = r(x_j)$, also $v_j = r(v_j)$. Hence $P_{E_j}(x_j)$ implies $P_{E_j}(r(v_j))$, which means that $P'_{E_j}(v_j)$ holds.

  If $v_{j_k} = u_{i_k}$ are variables then $x_{j_k} = r(t_{i_k})$ and $P_{E_{j_k}}(x_{j_k})$ implies that $P'_{E_{j_k}}(t_{i_k})$ holds.

  If v_j is a variable not occurring in $\{u_1, \ldots, u_m\}$, then $P_{E_j}(x_j)$ implies $\exists z P'_{E_j}(z)$

  We have just shown that

  $$\bigwedge_{j:u_j \in \mathcal{C}} P'_{u_j}(t_j) \wedge \bigwedge_{j:v_j \in \mathcal{C}} P'_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{j_k}}(t_{i_k}) \wedge \bigwedge_{j:v_j \in \mathcal{V} - \{u_1,\ldots,u_m\}} \exists z P'_{E_j}(z)$$

  holds, which means that $\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e)$ holds.

This ends the proof of the lemma 3.16.

Lemma 3.17. *Assume that EQ is non-erasing or, for every predicate symbol P_E, $\exists z P_E(z)$ implies $\exists z P_E(r(z))$. Then for every different function symbols f and g in Σ we have*

$$\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(e) \rightarrow P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)).$$

Proof. The assumption that EQ is non-erasing or $\exists z P_E(z)$ implies $\exists z P_E(r(z))$ is used in the same way as in lemma 3.16.

As usually, the proof depends on the form of e.

Case 1 *e is of the form $x = f(v_1, \ldots, x, \ldots, v_n)$*

The argumentation here is similar to that of case 1.2 of the lemma 3.16, read in the opposite direction: $\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(x = f(v_1, \ldots, x, \ldots v_n)) =$

$$= \bigwedge_{j:v_j \in \mathcal{C}} P'_{E_j}(v_j) \wedge P'_{E_i}(g(t_1,\ldots,t_m)) \wedge \bigwedge_{j:v_j \in \mathcal{V}} \exists z P'_{E_j}(z) =$$

$$= \bigwedge_{j:v_j \in \mathcal{C}} P_{E_j}(r(v_j)) \wedge P_{E_i}(r(g(t_1,\ldots,t_m))) \wedge \bigwedge_{j:v_j \in \mathcal{V}} \exists z P_{E_j}(r(z)) \leftrightarrow$$

$$\leftrightarrow \bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge P_{E_i}(r(g(t_1,\ldots,t_m))) \wedge \bigwedge_{j:v_j\in\mathcal{V}} \exists z P_{E_j}(z) =$$

$$= \varphi^{r(g(t_1,\ldots,t_m))}_{f(E_1,\ldots,E_n)}(x = f(v_1,\ldots,x,\ldots v_n)) \rightarrow P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) =$$

$$= P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m))$$

Case 2 *e is of the form* $x = g(v_1,\ldots,x,\ldots,v_n)$

$$\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(x = g(v_1,\ldots,x,\ldots v_n)) = \bigwedge_{j:v_j\in\mathcal{C}} P'_{v_j}(t_j) \wedge P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)) =$$

$$= \bigwedge_{j:v_j\in\mathcal{C}} P_{v_j}(r(t_j)) \wedge P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) \rightarrow P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) =$$

$$= P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m))$$

Case 3 *e is of the form* $g(u_1,\ldots,u_m) = f(v_1,\ldots,v_n)$ *with* $u_{i_1} = v_{i'_1},\ldots,u_{i_l} = v_{i'_l}$.

$$\varphi'^{g(t_1,\ldots,t_m)}_{f(E_1,\ldots,E_n)}(g(u_1,\ldots,u_n) = f(v_1,\ldots,v_n)) =$$

$$= \bigwedge_{j:u_j\in\mathcal{C}} P'_{u_j}(t_j) \wedge \bigwedge_{j:v_j\in\mathcal{C}} P'_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P'_{E_{i'_k}}(t_{i_k}) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{u_1,\ldots,u_m\}} \exists z P'_{E_j}(z) =$$

$$= \bigwedge_{j:u_j\in\mathcal{C}} P_{u_j}(r(t_j)) \wedge \bigwedge_{j:v_j\in\mathcal{C}} P_{E_j}(r(v_j)) \wedge \bigwedge_{k=1}^{l} P_{E_{i'_k}}(r(t_{i_k})) \wedge \bigwedge_{j:v_j\in\mathcal{V}-\{u_1,\ldots,u_m\}} \exists z P_{E_j}(r(z))$$

If $r(g(t_1,\ldots,t_m)) = g(r(t_1),\ldots,r(t_m))$ then this is equal to $\varphi^{r(g(t_1,\ldots,t_m))}_{f(E_1,\ldots,E_n)}(e)$ which means that $P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m))$ holds.

Suppose $r(g(t_1,\ldots,t_n)) = h(x_1,\ldots,x_k)$ (this includes the cases with $h = f$ and $h = g$). By lemma 3.14 there exists an equation e' such that $g(r(t_1),\ldots,r(t_m)) \overset{\Lambda}{\underset{e'}{\leftrightarrow}} h(x_1,\ldots,x_k)$. Now we will consider four possible forms of e':

Case 3.1 *e' of the form $x = h(\ldots x,\ldots)$ contradicts lemma 3.13*

Case 3.2 *e' is of the form $x = g(w_1,\ldots,x,\ldots,w_m)$.*

Since $g(r(t_1),\ldots,r(t_m))$ is a common instance of $g(u_1,\ldots,u_m)$ and $g(w_1,\ldots,w_m)$, by lemma 3.6 EQ contains an equation $\sigma(f(v_1,\ldots,v_n)) = \sigma(x)$ where σ is the most general unifier of $g(u_1,\ldots,u_m)$ and $g(w_1,\ldots,w_m)$.

Suppose that $\sigma(x)$ is a variable. Then x must be equal to one of the variables $w_{i_1},\ldots,w_{i_l}$, say w_{i_p} – otherwise $\sigma(x)$ would not be among $\sigma(w_1),\ldots,\sigma(w_m)$ and EQ would be trivial. So $r(g(t_1,\ldots,t_m)) = t_{i_p} = r(t_{i_p})$. Let $s_j = r(v_j)$ for such j that $v_j \in \mathcal{C}$, $s_{i'_q} = r(t_{i_q})$ for $q = 1,\ldots,l$, and s_j for such j that $v_j \in \mathcal{V} - \{u_1,\ldots,u_m\}$ be a term satysfying $\exists z P_{E_j}(z)$. Then we have $\bigwedge_{j=1}^{n} P_{E_j}(s_j)$. Now if $\sigma(v_j) \in \mathcal{C}$ then $\sigma(v_j) = r(t_j) = s_j$, so $P_{E_j}(\sigma(v_j))$ holds. Hence we have

$$\bigwedge_{\sigma(v_j)\in\mathcal{C}} P_{E_j}(\sigma(v_j)) \wedge P_{E_{i_p}}(s_{i_p}) \wedge \bigwedge_{\sigma(v_j)\in\mathcal{V}-\{\sigma(x)\}} \exists z P_{E_j}(z) =$$

$$= \varphi^{s_{i_p}}_{f(E_1,\ldots,E_n)}(\sigma(f(v_1,\ldots,v_n)) = \sigma(x)) \to P_{f(E_1,\ldots,E_n)}(s_{i_p}) =$$
$$= P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) = P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)).$$

If $\sigma(x)$ is not a variable, then $r(g(t_1,\ldots,t_m)) = \sigma(x)$ and

$$\bigwedge_{\sigma(v_j)\in\mathcal{C}} P_{E_j}(\sigma(v_j)) \wedge \bigwedge_{\sigma(v_j)\in\mathcal{V}} \exists z P_{E_j}(z) =$$

$$= \varphi^{\sigma(x)}_{f(E_1,\ldots,E_n)}(\sigma(f(v_1,\ldots,v_n)) = \sigma(x)) \to P_{f(E_1,\ldots,E_n)}(\sigma(x)) =$$
$$= P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) = P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)).$$

Case 3.3 *e' is of the form $g(w_1,\ldots,w_m) = h(u'_1,\ldots,u'_k)$, and $e' \neq e$.*

This case is symmetric to the case 1.3.2 of the previous lemma:

Since $g(r(t_1),\ldots,r(t_m))$ is a common instance of $g(u_1,\ldots,u_k)$ and $g(u'_1,\ldots,u'_k)$, these terms are unifiable. Let $\sigma = mgu(g(u_1,\ldots,u_k), g(u'_1,\ldots,u'_k))$. Then, by lemma 3.6 the equation $h(w_1\sigma,\ldots,w_m\sigma) = f(v_1\sigma,\ldots,v_n\sigma)$ belongs to EQ. Let $w_{j_1}\sigma = v_{j'_1}\sigma,\ldots,w_{j_{l'}}\sigma = v_{j'_{l'}}\sigma$ be all the variables occurring on both sides of this equation. So now, for each $i \in \{1,\ldots,l'\}$ there exists a $j \in \{1,\ldots,l\}$ such that $w_{j_i}\sigma = v_{j'_i}\sigma = v_{i'_j}\sigma = u_{i_j}\sigma = u'_{i_j}\sigma$. Note that $j'_i = i'_j$ here.

If $v_j\sigma$ is a constant symbol, then either $v_j\sigma = v_j$ and $P_{E_j}(v_j) = P_{E_j}(v_j\sigma) \to P_{E_j}(r(v_j\sigma)) \to P'_{E_j}(v_j\sigma)$, or $v_j = v_{i'_{j'}} = u_{i_{j'}}$ for some $j' \leq l$ and then $v_j\sigma = u_{i_{j'}}\sigma = r(t_{i_{j'}})$. Then $P_{E_{i'_{j'}}}(r(t_{i_{j'}})) \to P_{E_{i'_{j'}}}(v_{i'_{j'}}\sigma)$. In both cases $P_{E_j}(v_j\sigma)$ holds.

If $w_j\sigma$ is a constant then $w_j\sigma = x_j$ and $P_{w_j\sigma}(x_j)$ holds.

If $w_{j_i}\sigma$ is a variable then $w_{j_i}\sigma = u'_{i_j}\sigma$, so $x_{j_i} = r(t_{i_j})$ and $P_{E_{i'_j}}(r(t_{i_j})) = P_{E_{j'_i}}(x_{j_i})$. $P_{E_{i'_j}}(r(t_{i_j}))$ holds by the assumption of the case 3, so $\bigwedge_{i=1}^{l} P_{E_{j'_i}}(x_{j_i})$ holds. Since $\bigwedge_{j=1}^{n} \exists z P_{E_j}(z)$ holds, also $\bigwedge_{j:v_j\sigma\in\mathcal{V}-\{w_1\sigma,\ldots,w_m\sigma\}} \exists z P_{E_j}(z)$ holds.

We have just shown that

$$true \leftrightarrow \bigwedge_{v_j\sigma\in\mathcal{C}} P_{E_j}(v_j\sigma) \wedge \bigwedge_{w_j\sigma\in\mathcal{C}} P_{w_j\sigma}(x_j) \wedge \bigwedge_{i=1}^{l} P_{E_{j'_i}}(x_{j_i}) \wedge \bigwedge_{j:v_j\sigma\in\mathcal{V}-\{w_1\sigma,\ldots,w_m\sigma\}} \exists z P_{E_j}(z) =$$

$$= \varphi^{h(x_1,\ldots,x_k)}_{f(E_1,\ldots,E_n)}(h(w_1\sigma,\ldots,w_m\sigma) = f(v_1\sigma,\ldots,v_n\sigma)) = \varphi^{r(g(t_1,\ldots,t_n))}_{f(E_1,\ldots,E_n)}(h(\ldots) = f(\ldots)) \to$$
$$\to P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) = P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m)).$$

Case 3.4 *$e' = e$*

In this case $r(g(t_1,\ldots,t_n))$ is of the form $f(s_1,\ldots,s_n)$ and $\bigwedge_{v_j\in\mathcal{C}} P_{E_j}(v_j) \wedge \bigwedge_{k=1}^{l} P_{E_{i'_k}}(r(t_{i_k}))$ implies $\bigwedge_{j=1}^{n} P_{E_j}(s_j)$, which implies $P_{f(E_1,\ldots,E_n)}(f(s_1,\ldots,s_n)) = P_{f(E_1,\ldots,E_n)}(r(g(t_1,\ldots,t_m))) = P'_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m))$. □

Lemma 3.18. *Assume that EQ is non-erasing or, for every predicate symbol P_E, $\exists z P_E(z)$ implies $\exists z P_E(r(z))$. Then $P'_{f(E_1\ldots E_n)}(f(t_1,\ldots,t_n)) \leftrightarrow$*

$$\left((P'_{E_1}(t_1) \wedge \ldots P'_{E_n}(t_n)) \vee \bigvee_{e\in EQ} \varphi'^{f(t_1,\ldots,t_n)}_{f(E_1\ldots E_n)}(e) \right)$$

Proof. If $r(f(t_1,\ldots,t_n)) = f(r(t_1),\ldots,r(t_n))$ then the proof is straightforward:

$$P'_{f(E_1,\ldots,E_n)}(f(t_1,\ldots,t_n)) \leftrightarrow P_{f(E_1,\ldots,E_n)}(f(r(t_1),\ldots,r(t_n))) \leftrightarrow$$

$$\leftrightarrow \left(\bigwedge_{i=1}^{n} P_{E_i}(r(t_i))\right) \vee \bigvee_{e\in EQ} \varphi^{f(r(t_1),\ldots,r(t_n))}_{f(E_1,\ldots,E_n)}(e) \leftrightarrow \left(\bigwedge_{i=1}^{n} P'_{E_i}(t_i) \vee \bigvee_{e\in EQ} \varphi'^{f(t_1,\ldots,t_n)}_{f(E_1\ldots E_n)}(e)\right).$$

Assume $r(f(t_1,\ldots,t_n)) \neq f(r(t_1),\ldots,r(t_n))$. First we prove the $\leftarrow$ implication.

There are two possibilities: either $\bigwedge_{j=1}^{n} P'_{E_i}(t_i)$ or $\varphi'^{f(t_1,\ldots,t_n)}_{f(E_1\ldots E_n)}(e)$ holds for some $e \in EQ$. In the first case $\bigwedge_{j=1}^{n} P'_{E_i}(t_i) = \bigwedge_{j=1}^{n} P_{E_i}(r(t_i))$.

By lemma 3.14 there exists $e \in EQ$ such that $f(r(t_1),..r(t_n)) \overset{\Lambda}{\underset{e}{\leftrightarrow}} r(f(t_1,...t_n))$. Simple case analysis (as in case 2 of lemma 3.16, only two cases for e of the form $x = f(...)$ or g(...)=f(...), where g is the leading function symbol of $r(f(..))$, including case of $g = f$) gives that $\varphi^{r(f(t_1,..t_n))}_{f(E_1\ldots E_n)}(e)$ holds. This implies $P_{f(E_1\ldots E_n)}(r(f(t_1,\ldots,t_n)))$ which gives $P'_{f(E_1\ldots E_n)}(f(t_1,\ldots,t_n))$.

In the second case, when $\varphi'^{f(t_1,\ldots,t_n)}_{f(E_1\ldots E_n)}(e)$ holds, the argumentation is the same as in lemma 3.17.

Now we prove the $\rightarrow$ implication.

Case 1 *$r(f(t_1,\ldots,t_n)) = g(x_1,\ldots,x_m)$ with $g \neq f$.*

By lemma 3.14 there exists $e \in EQ$ such that $f(r(t_1),\ldots,r(t_n)) \overset{\Lambda}{\underset{e}{\leftrightarrow}} g(x_1,\ldots,x_m)$. By lemma 3.15 there exists $e' \in EQ$ and terms $s_1,\ldots,s_n$ such that $g(x_1,\ldots,x_m) \overset{\Lambda}{\underset{e}{\leftrightarrow}} f(s_1,\ldots,s_n)$ and $P_{E_j}(s_j)$ holds for each j. Now we will have to consider four cases, depending on e and e'.

Case 1.1 *$e = e'$ and for all $j \in \{1,\ldots,n\}$ we have $r(t_j) = s_j$*

Then $\bigwedge_{j=1}^{n} P_{E_j}(r(t_j)) \rightarrow \bigwedge_{j=1}^{n} P'_{E_j}(t_j)$

Case 1.2 *Both e and e' are either of the form $x = f(...)$ or of the form $f(...) = g(...)$, including the case with $e = e'$ but $r(t_j) \neq s_j$ for some j (which may happen if e contains erasing variables)*

By lemma 3.6 there exists a composition e'' of equations e and e' in EQ. Again simple case analysis gives that $\varphi^{f(r(t_1),\ldots,r(t_n))}_{f(E_1,\ldots,E_n)}(e'')$ holds, which (because e'' is of the form $f(...) = f(...)$) is equal to $\varphi'^{f(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(e'')$.

Case 1.3 *The equation e' is of the form $x = f(v_1,..x,\ldots,v_n)$, with x on the ith position in $f(v_1,..x,\ldots,v_n)$.*

If $v_j \in \mathcal{C}$ then $v_j = s_j$ and $P_{E_j}(v_j)$ holds. Since $s_i = r(f(t_1,\ldots,t_n))$, $P_{E_i}(r(f(t_1,\ldots,t_n)))$ holds. This gives that $\varphi'^{f(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(e')$.

Case 1.4 *The equation e is of the form $x = f(v_1,\ldots,x,\ldots,v_n)$, with x on the ith position in $f(v_1,\ldots,x,\ldots,v_n)$.*

If $v_j \in \mathcal{C}$ then $v_j = r(t_j)$ and $P_{v_j}(r(t_j))$ holds. Since $P_{f(t_1,\ldots,t_n)}(r(f(t_1,\ldots,t_n)))$ holds, $\varphi'^{f(t_1,\ldots,t_n)}_{f(E_1,\ldots,E_n)}(e')$.

Case 2 $r(f(t_1,\ldots,t_n)) = f(x_1,\ldots,x_n)$.

The proof here is symmetric to the proof of lemma 3.16, case 2. □

Lemma 3.19. *Assume that EQ is non-erasing or, for every predicate symbol P_E, $\exists z P_E(z)$ implies $\exists z P_E(r(z))$. Then the interpretation defined by Φ' is a model of φ, compatible with EQ.*

Proof. In the light of lemmas 3.16 and 3.18 it is enough to show that φ'_{SC}, $\varphi'_{E(SC)}$ and $\varphi'_{\mathcal{C}}$ hold. But this is obvious since (in the case of φ_{SC}) for positive constraints we have $P'_E(y) \leftrightarrow P_E(r(y)) \rightarrow P_{E'}(r(y)) \leftrightarrow P'_{E'}(y)$, and for negative constraints we have $\exists x P_{E\cap\overline{E'}}(x)$ implies $\exists x P_{E\cap\overline{E'}}(r(x)) \leftrightarrow \exists x P'_{E\cap\overline{E'}}(x)$. The cases of $\varphi'_{E(SC)}$ and of $\varphi'_{\mathcal{C}}$ are similar.

Note that if EQ is non-erasing and SC does not contain negative constraints then we do not need the assumption that $\exists z P_E(z)$ implies $\exists z P_E(r(z))$. □

Theorem 3.20. *Assume that φ has a model over the Herbrand Universe $\mathcal{H}$ over Σ. If either EQ is non-erasing and SC does not contain negative constraints or $\exists z P_E(z)$ implies $\exists z P_E(r(z))$ for every predicate symbol P_E then SC is consistent in EQ.*

Proof. We will prove that Φ' defines a solution of SC. Let $\mathcal{S}(E) = \{q(t) \mid P_E(t)\}$ for $E \in E(SC)$, where $q : \mathcal{H} \rightarrow \mathcal{H}/=_{EQ}$ is the quotient mapping, and P_E is the interpretation of predicate symbol P_E under Φ'. Note that by the lemma 3.19, $q^{-1}(\mathcal{S}(E))$ is precisely the set $\{t \mid P_E(t)\}$. We have to show that this definition of $\mathcal{S}$ agrees with the natural extension of $\mathcal{S}$ restricted to set variables and constants to $E(SC)$, i.e., that $\mathcal{S}(f(E_1,\ldots,E_n)) = f(\mathcal{S}(E_1),\ldots,\mathcal{S}(E_n))$, $\mathcal{S}(E_1 \cup E_2) = \mathcal{S}(E_1) \cup \mathcal{S}(E_2)$ and so on. This can be easily done by induction on the structure of E. The formula $\varphi_{E(SC)}$ assures us that everything is correct if E is a boolean expression (that is a union, intersection or complement of other expressions). The equivalence defining $P_{f(E_1,\ldots,E_n)}(f(t_1,\ldots,t_n))$ gives that $P_{f(E_1,\ldots,E_n)}$ is $true$ on all the terms from $q^{-1}(\mathcal{S}(f(E_1,\ldots,E_n)))$, that is $\mathcal{S}(f(E_1,\ldots,E_n)) \subseteq f(\mathcal{S}(E_1),\ldots,\mathcal{S}(E_n))$. The equivalence defining $P_{f(E_1,\ldots,E_n)}(g(t_1,\ldots,t_m))$ gives that if $P_{f(E_1,\ldots,E_n)}$ is $true$ on some term, it must be a term from $q^{-1}(\mathcal{S}(f(E_1,\ldots,E_n)))$, that is $\mathcal{S}(f(E_1,\ldots,E_n)) \supseteq f(\mathcal{S}(E_1),\ldots,\mathcal{S}(E_n))$.

Finally, φ_{SC} assures that the constraints are satisfied. □

3.3 Positive set constraints in non-erasing linear shallow theories

The following corollary is a particular case of the theorem 3.20:

Corollary 3.21. *If φ has a model over the Herbrand Universe $\mathcal{H}$ over Σ and EQ is non-erasing and SC is a system of positive set constraints then SC is consistent in EQ.*

In the light of theorem 3.11, corollary 3.21 and lemma 3.9 we get the following

Theorem 3.22. *A system SC of set constraints in a linear nonerasing shallow equational theory is consistent if and only if the corresponding monadic formula is satisfiable.*

and since satisfiability of monadic formulae is in NEXPTIME (see [27]) and consistency of systems of positive set constraints in the empty theory is NEXPTIME-complete (see [9]), we get

Theorem 3.23. *Consistency of systems of set constraints in a linear nonerasing shallow equational theory is decidable; more precisely it is a NEXPTIME-complete problem.*

3.4 Negative set constraints in linear shallow theories

3.4.1 Reduction to tree set automata

In the definitions below we keep notations from the previous section: Σ is a fixed set of function symbols, EQ is a fixed nontrivial linear shallow theory, $\mathcal{H}$ is the Herbrand universe over Σ, φ is an almost flat formula corresponding to the sytem SC of mixed set constraints, and $r : \mathcal{H} \to \mathcal{H}$ is a fixed function giving for each term t its reprezentant $r(t)$ in the theory EQ such that $t =_{EQ} r(t)$, if $c \in \mathcal{C}$ then $r(c) \in \mathcal{C}$, if $r(t) \notin \mathcal{C}$ then it is a term of the form $f(r(t_1), \ldots, r(t_n))$.

Definition 3.24. A *tree set automaton* is a triple $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{S})$. Σ is a finite ranked alphabet, $\mathcal{Q}$ is a finite set of states and $\mathcal{S}$ is a finite set of *rules* of the form $f(q_1, \ldots, q_n) \to q$ where $f \in \Sigma$, $q, q_1, \ldots, q_n \in \mathcal{Q}$, and n is the arity of f.

The automaton is *complete* if for each number n, each $f \in \Sigma$ of arity n and each sequence $q_1, \ldots, q_n$ of states there must exist a state q such that $f(q_1, \ldots, q_n) \to q$ belongs to $\mathcal{S}$.

An $\mathcal{A}$-*run* of $\mathcal{A}$ is a mapping $\Phi : \mathcal{H} \to \mathcal{Q}$ such that if $\Phi(f(t_1, \ldots, t_n)) = q$ then $\mathcal{S}$ contains the rule $f(\Phi(t_1), \ldots, \Phi(t_n)) \to q$.

An $\mathcal{A}$-run Φ is EQ-*succesful* if there exists a function r as above such that $\Phi(r(\mathcal{H})) = \mathcal{Q}$.

Now we will define a tree set automaton corresponding to the system SC (or, equivalently, to the formula φ). Recall that each almost flat formula has a corresponding monadic formula. Let $monadic(\varphi)$ be the monadic formula corresponding to φ. To define the automaton, we need the following definitions and lemma:

Let N be the number of predicate symbols in $\mathcal{P} = \{P_E \mid E \in E(SC)\}$. If I is a model of $monadic(\varphi)$ then we consider the equivalence relation $\equiv$ in I such that $x \equiv y$ if and only if for all $P \in \mathcal{P}$ we have $P^I(x) \leftrightarrow P^I(y)$, where P^I is the interpretation of P in I. Let M be the number of different nonempty equivalence classes of the relation $\equiv$.

Lemma 3.25 (see [1], p.34). *A monadic formula (without equality) is satisfiable if and only if it has a finite model (of cardinality $M \leq 2^N$) such that in each equivalence class of the relation $\equiv$ there is at most one element.*

Recall that φ is equivalent to an almost flat formula ψ, which is a partial skolemization of $monadic(\varphi)$ — $monadic(\varphi)$ can be obtained from ψ by replacing each term $f(y_1, \ldots, y_n)$ by an existential variable x_f and adding the quantifier $\exists x_f$ between the quantifiers $\forall y_n$ and $\forall y_{n+1}$ in the prefix of ψ.

Definition 3.26. If $\mathcal{M}$ is a finite model of $monadic(\varphi)$ then we define a tree set automaton $\mathcal{A}(\mathcal{M})$ in the following way:

- Σ is the set of function symbols occurring in φ
- $\mathcal{Q}$ is the domain of $\mathcal{M}$
- $f(q_1, \ldots, q_n) \to q$ belongs to $\mathcal{S}$ if the formula $monadic(\varphi)$ with y_i instantiated to q_i and x_f instantiated to q is satisfied.

Note that Σ here is exactly the initial set of function symbols with which we write set constraints, and it is a bit less than the set of Skolem function symbols for the formula $monadic(\varphi)$ — the difference is the set of skolem constants corresponding to the existential part of the almost flat formula ψ (skolemization of ψ introduces new constant symbols and gives a formula which is also a skolemization of $monadic(\varphi)$).

Note that $\mathcal{A}(\mathcal{M})$ may be nondeterministic, i.e., $\mathcal{S}$ may contain two (or more) rules with the same lefthand side and with different righthand sides, but must be complete. Hence each partial run (a run defined on some closed subset of $\mathcal{H}$, where *closed* means that whenever the set contains a term, it must also contain all its subterms) can be extended to a run.

Lemma 3.27. *If SC is consistent in EQ then there exists a finite model $\mathcal{M}$ of $monadic(\varphi)$ and an EQ-succesful run of $\mathcal{A}(\mathcal{M})$.*

Proof. Assume that SC is consistent in EQ. By theorem 3.11 φ has a model over $\mathcal{H}$, compatible with EQ. Consider this model and the equivalence relation $\equiv$ defined on $\mathcal{H}$ by $t \equiv s$ iff $P_E(t) \leftrightarrow P_E(s)$ for all $E \in E(SC)$. Let $[t]$ denotes the equivalence class of the term t in $\equiv$. Now we build the model $\mathcal{M}$. Its domain is the set $\mathcal{H}/\equiv$ of (nonempty) equivalence classes of $\equiv$. Note that its size is bounded by 2^N where $N = |E(SC)|$. The interpretation of predicate symbols is the following: for each element $[t]$ of the domain of $\mathcal{M}$ and each predicate symbol P we put $P^{\mathcal{M}}([t]) \leftrightarrow P^{\mathcal{H}}(t)$ where $P^{\mathcal{M}}$ and $P^{\mathcal{H}}$ are the respective interpretations of the symbol P. It is easy to check that $\mathcal{M}$ is a model of $monadic(\varphi)$.

To define an EQ-succesful run of $\mathcal{A}(\mathcal{M})$ put $\Phi(t) = [t]$. This really defines a run because if $t = f(t_1, \ldots, t_n)$ then (since φ is universally quantified over y_i) φ with y_i instantiated to t_i for $i = 1, \ldots, n$ is satisfied, which means that $monadic(\varphi)$ (interpreted in $\mathcal{H}$) with y_i instantiated to t_i and x_f instantiated to $f(t_1, \ldots, t_n)$ is satisfied, so $monadic(\varphi)$ (interpreted in $\mathcal{M}$) with y_i instantiated to $[t_i]$ and x_f instantiated to $[f(t_1, \ldots, t_n)]$ is satisfied, and $f([t_1], \ldots, [t_n]) \rightarrow [f(t_1, \ldots, t_n)]$ belongs to the set of rules.

The run is EQ-succesful because of the two following facts. First is that the domain of $\mathcal{M}$ consists only of nonempty equivalence classes of $\equiv$. The second is that considered model of φ is compatible with EQ, so for all terms t and any function r giving for each term its representant in EQ we have $[t] = [r(t)]$, so each nonempty equivalence class contains an element of $r(\mathcal{H})$. □

Lemma 3.28. *If there exists a model $\mathcal{M}$ of $monadic(\varphi)$ and an EQ-succesful run of $\mathcal{A}(\mathcal{M})$ then SC is consistent in EQ.*

Proof. Let $\mathcal{M}$ be a model of $monadic(\varphi)$ and $\Phi : \mathcal{H} \rightarrow \mathcal{Q}$ an EQ-succesful run of $\mathcal{A}(\mathcal{M})$. Let r be the function from the definition of EQ-succesful run. Define an interpretation over $\mathcal{H}$ of predicate symbols occurring in φ as follows: $P^{\mathcal{H}}(t) \leftrightarrow P^{\mathcal{M}}(\Phi(t))$.

It is easy to see that φ is satisfied in this interpretation. Since Φ is an EQ-succesful run of $\mathcal{A}(\mathcal{M})$, we have that $\exists z P_E(z)$ implies $\exists z P_E(r(z))$ for every predicate symbol P_E interpreted over $\mathcal{H}$. By theorem 3.20 SC is consistent in EQ. □

As a corollary to lemmas 3.27 and 3.28 we get the following

Theorem 3.29. *SC is consistent in EQ if and only if there exists a finite model $\mathcal{M}$ of $monadic(\varphi)$ and an EQ-succesful run of $\mathcal{A}(\mathcal{M})$.*

3.4.2 Decidability of the problem

In this section we will prove that the problem *given a complete tree set automaton, does there exist an EQ-succesful run of* $\mathcal{A}$, is decidable. Together with theorem 3.29 it will give the decidability result for systems of mixed set constraints in linear shallow theories.

Before we present the proof we want to give some intuition about why this problem is difficult. The first idea of solving it is to use pumping lemma techniques. The (false) pumping lemma here would look as follows: if there exists an EQ-succesful run Φ such that $\Phi(s) = q$ where s is a big term then there is a big subterm t_1 of s and a small term t_2 such that $\Phi(t_1) = \Phi(t_2)$ and Φ' defined by $\Phi'(u) = \Phi(u[t_2/t_1])$ is another EQ-succesful run. Here $u[t_2/t_1]$ denotes a term obtained by replacing (each or some occurrence of) t_2 by t_1. Using such lemma we would be able to prove that if there exists an EQ-succesful run Φ then there exists another EQ-succesful run Φ' and a small set T of small terms such that $\Phi'(T) = \mathcal{Q}$. The following example shows that this approach does not work.

Example 3.4. Let $\Sigma = \{a, f(\cdot), g(\cdot,\cdot)\}$, $\mathcal{Q} = \{q_0, q_1, q\}$ and $\mathcal{S} = \{a \to q_0,\ f(q_0) \to q_0,\ f(q_0) \to q_1, g(q_0,q_0) \to q_0,\ g(q_0,q_0) \to q_1,\ g(q_0,q_1) \to q, \ldots\}$. Let $\Phi(a) = q_0, \Phi(f^{700}(a)) = q_0, \Phi(g(a,a)) = q_0, \Phi(g(a, f^{700}(a))) = q_1, \Phi(g(g(a,a), g(a, f^{700}(a)))) = q$, where $f^{700}(a)$ denotes function symbol f applied 700 times to the constant a. Now the intention is to replace $f^{700}(a)$ by a to get $\Phi'(g(g(a,a), g(a,a))) = q$. But then we would have to have $\Phi'(g(a,a)) = q_0$ and $\Phi'(g(a,a)) = q_1$ which is impossible because Φ' must be a function.

The method used here is taken from [11], slightly modified to deal with equational theories, and translated to the language of tree automata. Similiar results (for empty theory) were obtained in [18], but the proof presented there is much more complicated.

We fix a linear ordering $\prec$ on terms such that if t is a term of depth lower then the depth of s, then $t \prec s$. The maximal subterm of a term $f(s_1, \ldots, s_k)$ is a term s_j among $s_1, \ldots, s_k$ such that for all $i \leq k$, $s_i \prec s_j$ (if there are two maximal subterms $s_j = s_l$ with $j < l$ then the first occurrence, i.e., s_j is the maximal subterm).

In the text below, when we speak of equivalence classes, we mean the equivalence relation $\equiv$ defined by $t \equiv s$ iff $\Phi(t) = \Phi(s)$. Note that if the automaton is of the form $\mathcal{A}(\mathcal{M})$ then this is exactly the relation $\equiv$ used in the proof of lemma 3.27.

We will show that each EQ-succesful run can be approximated by some finite set. We fix some EQ-succesful run Φ with function r. We treat the set $\mathcal{H} - r(\mathcal{H})$ as some inconvinient balast, so when we speak of terms, we mean rather terms from $r(\mathcal{H})$ than terms from $\mathcal{H}$. Whenever we choose a term which is used in the proof below, it must be an element of $r(\mathcal{H})$.

Definition 3.30. If Φ is an EQ-succesful run of $\mathcal{A}$ then *the initial part of* Φ is Φ restricted to the minimal finite closed subset of $r(\mathcal{H})$ containing the minimal (according to the order $\prec$) terms (from $r(\mathcal{H})$) in each equivalence class.

Note that initial part of a run is a partial run and can be extended to a run.

We will show that if this set is too big then we are able to construct a new EQ-succesful run with a smaller initial part. The key observation here is that if $f(s_1, \ldots, t, \ldots, s_n)$ belongs to the domain of the initial part, and this set contains no other term containing t as a subterm, then if we can replace $f(s_1, \ldots, t, \ldots, s_n)$ by $f(s'_1, \ldots, t, \ldots, s'_n)$ where $s_i \equiv s'_i$ and s'_i are minimal in their equivalence classes, we obtain a run with smaller initial part. Of course the terms s'_i must be taken from $r(\mathcal{H})$.

In the text below $c[i]$ denotes the i-th term (according to the ordering $\prec$) in the set $\Phi^{-1}(c) \cap r(\mathcal{H})$, and the set of *milestones* is a subset of the domain of the initial part (defined in the background of the definitions below). We call a term *composed* if it is not a constant symbol.

Definition 3.31. The semi-skeleton of the initial part of an EQ-succesful run Φ is a labeled graph $\mathcal{K}$ constructed in the following way:

- initialize the set of *milestones* as the set containing all the minimal terms in each equivalence class,
- initialize the set of nodes of $\mathcal{K}$ as the minimal set containing the set of milestones and closed under the operation of taking the maximal subterm,
- label each node t in $\mathcal{K}$ with $\Phi(t)$,
- connect each composed term in $\mathcal{K}$ with its maximal subterm by an edge directed from a term to its superterm,
- label each edge $t \to f(s_1, \ldots, t, \ldots, s_n)$ with the rule $f(\Phi(s_1)[1], \ldots, *, \ldots, \Phi(s_n)[1])$.

Before we say how to extend the semi-skeleton to a skeleton, we want to give some intuition of how to interpret this graph. A label of an edge can be interpreted as information how to build a new EQ-succesful run. The rule $f(c_1[1], \ldots, *, \ldots, c_n[1])$ says: take the predecessor t of the edge and first ([1] indicates the first term) terms $t_1, \ldots, t_n$ from the sets $\Phi^{-1}(c_1), \ldots, \Phi^{-1}(c_n)$ and define $\Phi(f(t_1, \ldots, t, \ldots, t_n))$ as the label of the successor of the edge. Of course we have to decide what to do if a conflict arises, i.e., if we are to define Φ on one term in several different ways.

Definition 3.32. A fork of degree $k \geq 2$ is a term t such that in the semi-skeleton there are k edges connecting t with k its superterms, labeled with the same rule.

Definition 3.33. A skeleton of the initial part of a run is a graph with the following properties:

1. it is a forest with directed edges leading from roots to leaves
2. it contains the semi-skeleton as a subgraph
3. labels of edges outgoing from a single node are different
4. labels of edges are compatible with transition rules of the automaton, i.e., if $f(c_1[j_1], \ldots, *, \ldots, c_k[j_k])$ is a label of an edge $t \to s$, then $\mathcal{S}$ contains the rule $f(c_1, \ldots, \Phi(t), \ldots, c_k) \to \Phi(s)$
5. if $c[j]$ is used in a label of an edge $t \to s$ then there are at least j milestones labeled by c,
6. if $c[j]$ is used in a label of an edge $t \to s$ then the node $c[j]$ (according to the linear ordering $\prec$) is a term lower than t
7. all leaves are milestones

Lemma 3.34. *If Φ is an EQ-succesful run then there exists a skeleton $\mathcal{K}$ of the initial part of Φ.*

Proof. We will show here how to extend a semi-skeleton to a skeleton.

If t is a fork of degree k in a semi-skeleton, then the initial part of Φ contains k different terms $f(s_1^1, \ldots, t, \ldots, s_n^1), \ldots f(s_1^k, \ldots, t, \ldots, s_n^k)$, with $s_j^i \prec t$. Without loss of generality we can assume that the terms s_1^1 and s_1^2 are different. Now if the terms $s_1^1, \ldots, s_1^k$ are different, then we add $k-1$ minimal terms from $\Phi^{-1}(\Phi(s_1^1))$ to the set of milestones (remember that $\Phi(s_1^i) = \Phi(s_1^j)$, so we have at least k terms in this set) and replace the labels of edges with k different rules $f(\Phi(s_1^1)[1], \ldots, *, \ldots, \Phi(s_n^1)[1]), \ldots, f(\Phi(s_1^1)[k], \ldots, *, \ldots, \Phi(s_n^1)[1])$. If there are less than k terms among $s_1^1, \ldots, s_1^k$, then it is even better, because we need less new milestones. Consider the following examples

Example 3.5. t is a fork of degree 4 and we have the following four terms as nodes in the semi-skeleton: $f(t, x_1, y), f(t, x_2, y), f(t, x_3, y), f(t, x_4, y)$ with $\Phi(x_1) = \Phi(x_2) = \Phi(x_3) = \Phi(x_4) = x$. Then we label edges $t \rightarrow f(t, x_j, y)$ with the rules $f(*, x[j], \Phi(y)[1])$ which (while reconstructing a run from the skeleton) we interpret as follows: take the term t (predecessor of these edges), the four minimal terms $t_1, \ldots, t_4$ from the set $\Phi^{-1}(x)$ (these are also four minimal terms from $\Phi'^{-1}(x)$), the minimal term s in $\Phi^{-1}(\Phi(y))$ and define $\Phi'(f(t, t_j, s)) = \Phi(f(t, x_j, y))$

Example 3.6. t is a fork of degree 4 and we have the following four terms as nodes in the semi-skeleton: $f(t, x_1, y_1), f(t, x_2, y_1), f(t, x_1, y_2), f(t, x_2, y_2)$ with $\Phi(x_1) = \Phi(x_2) = x$ and $\Phi(y_1) = \Phi(y_2) = y$. Then we have to create only two new milestones - the second minimal terms from the sets $\Phi^{-1}(x)$ and $\Phi^{-1}(y)$ and label the four edges with the rules $f(*, x[1], y[1]), f(*, x[2], y[1]), f(*, x[1], y[2]), f(*, x[2], y[2])$

Note that if we have a fork of degree k then we introduce at most $k-1$ new milestones, which are among the first $k-1$ terms in some equivalence classes. Moreover, these milestones are of depth less than or equal to the depth of the fork. □

Below we shall prove that an EQ-succesful run can be reconstructed from a skeleton of its initial part. In fact a solution can be reconstructed from any labeled graph satisfying all the conditions described in Definition 3.33 except 2, and with the condition 6 replaced by the thesis of lemma 3.36.

Definition 3.35. Given a skeleton $\mathcal{K}$ we say that a term t' is a collapse of a node $t \in \mathcal{K}$ if either t is a constant symbol and $t' = t$ or $t' = f(t_1, \ldots, s' \ldots, t_k)$ and there is an edge $s \rightarrow t$ labeled with $f(c_1[j_1], \ldots, *, \ldots, c_k[j_k])$, the term s' is the collapse of the node s and $t_1, \ldots, t_k$ are collapses of respective milestones. The collapsing function is a function mapping each node of $\mathcal{K}$ to its collapse.

Lemma 3.36. *The collapsing function is 1 to 1.*

Proof. (induction on depth of terms) It is obvious that a constant symbol cannot be a collapse of two different nodes. Let us suppose that all collapsed terms of depth less than or equal to n are collapses of at most one node and there is a term u of depth $n+1$ which is a collapse of two different nodes. Without loss of generality we can assume $u = f(t', s')$ where t' and s' are collapses of nodes t and s respectively, with $s \prec t$. Clearly $f(t', s')$ has been obtained using two rules of the form $f(c_1[k], *)$ and $f(*, c_2[l])$, so t' is the collapse of $c_1[k]$, and $t \neq c_1[k]$,

since $c_1[k] \prec s \prec t$ (note that if $t = s$ then $c_1[k] \neq s$ — otherwise $c_1[k]$ would be choosen as the maximal subterm in $f(c_1[k], s)$). Hence t' is a collapse of two different nodes which is impossible. □

Lemma 3.37. *If Φ is an EQ-succesful run then the collapse of the skeleton of the initial part of Φ is an initial part of another EQ-succesful run Φ' (with the same function r).*

Proof. By lemma 3.36 the set of pairs consisting of collapsed nodes and their labels is a function (mapping collapsed terms to states of the automaton). By the condition 4 of the definition of the skeleton, this function is a partial run. By lemma 3.34 this partial run is EQ-succesful. □

Lemma 3.38. *There are at most $|\mathcal{Q}|^2$ milestones.*

Proof. We start building the skeleton (from a semi-skeleton) with $|\mathcal{Q}|$ milestones. This gives for each number d at most $|\mathcal{Q}|$ different terms of depth d in the semi-skeleton. Then we add new milestones only if a fork is encountered. Let t be a fork of degree k and let d denotes its depth. We have in the skeleton at least k superterms of t of depth $d+1$ and t is the only one their subterm of depth d. Then we add at most $k-1$ new terms (milestones) of depth less than or equal to d. Hence still we have at most $|\mathcal{Q}|$ terms of depth d in the skeleton, which means that degrees of forks are bounded by $|\mathcal{Q}|$ and the set of milestones has at most $|\mathcal{Q}|$ minimal terms in each equivalence class. So the cardinality of this set is bounded by $|\mathcal{Q}|^2$. □

Lemma 3.39. *There are at most $|\mathcal{Q}|^2$ nodes of outdegree greater than one.*

Proof. In a forest the number of nodes of outdegree greater than one is less than the number of leaves. Each leaf of the skeleton is a milestone. □

Lemma 3.40. *If there exists an EQ-succesful run of the automaton, then there exists one with the domain of the initial part consisting of at most $2|\mathcal{Q}|^3$ terms.*

Proof. Assume that Φ is an EQ-succesful run. By lemma 3.37 we can assume that all terms in the domain of the initial part of Φ belong to the skeleton of this initial part. Suppose that there are more than $2|\mathcal{Q}|^3$ nodes. By lemmas 3.38 and 3.39 the number of milestones and the nodes of outdegree greater than one is bounded by $2|\mathcal{Q}|^2$, so there exists a path $u_1 \to \ldots \to u_m$ with $m \geq |\mathcal{Q}|$ such that $u_1, \ldots, u_m$ are terms of outdegree one and none of them is a milestone. So, there are two terms u_i and u_j with $1 \leq i < j \leq m$ such that $\Phi(u_i) = \Phi(u_j)$. Now, we can cut off the path $u_i \to \ldots \to u_{j-1}$ thus obtaining a smaller graph satisfying the conditions in the remark above the definition 3.35. We have only to show that the collapsing function on this graph is 1 to 1. This is quite easy since if a term s is a collapse two nodes s_1, s_2 then $s[u_i/u_j]$ is a collapse of two nodes $s_1[u_i/u_j]$ and $s_2[u_i/u_j]$ where $s[u_i/u_j]$ denotes a term with all occurrences of u_i replaced by u_j, and this contradicts lemma 3.36. Now, to build a new EQ-succesful run Φ' with a lower number of terms in the domain of the initial part we can proceed as in the proof of lemma 3.37.

Repeating the procedure described above we can find an EQ-succesful run with at most $2|\mathcal{Q}|^3$ different terms in the domain of the initial part. □

Theorem 3.41. *It is decidable whether a complete tree set automaton admits an EQ-succesful run.*

Proof. To decide whether a complete tree set automaton admits an EQ-succesful run it is enough to check if there exists a skeleton with at most $2|\mathcal{Q}|^3$ nodes such that for every two nodes t, s we have $t \neq_{EQ} s$.

Lemma 3.40 proves that this method is complete. To prove that it is sound we will show that if there exists a skeleton such that for every two nodes t, s we have $t \neq_{EQ} s$ then there exists an EQ-succesful run. But this is quite simple since the mapping assigning to each node its label is a partial run. To prove that this run is EQ-succesful we have to find a proper function r. For every term t such that there exists a node t' with $t =_{EQ} t'$ we put $r(t) = t'$. Then inductively we do the following: we take the minimal (according to $\prec$) term s for which $r(s)$ is not defined; if s is a constant symbol then we put $r(s) = s$; if s is a composed term of the form $f(s_1, \ldots, s_n)$ then we put $r(s) = f(r(s_1), \ldots, r(s_n))$; for all terms $s' =_{EQ} s$ we put $r(s') = r(s)$. Now it is obvious that our run is EQ-succesful. □

As a corollary to theorems 3.41, 3.29 and lemma 3.25 we get the final result:

Theorem 3.42. *Consistency of systems of set constraints in a linear shallow equational theory is decidable.*

3.5 An application to semantic order-sorted unification

In this section, following the ideas of [31], we show how to use set constraints for solving unification problems in order-sorted theories. It is known that general order-sorted unification is undecidable, and that unification in elementary order-sorted (syntactic) unification is decidable. T. Uribe in [31] extended this result to so-called semi-linear ordered signatures. Very little is known about semantic order-sorted unification. Below we prove decidability of (semantic) elementary order-sorted unification in linear shallow theories.

Definition 3.43. An *elementary sorted signature* is a 4-tuple $(SO, \leq, SD, OD)$ where SO is a finite set of sorts, $\leq$ is a partial order on SO, SD is set of sort declarations of the form $S \leq S'$ with $S, S' \in SO$, and OD is set of operation declarations of the form $f : S_1, \ldots, S_n \to S$.

An $\mathcal{S}$-term t is of sort S if either it is a variable (e.g. x) of sort S' (written $x : S'$) with $S' \leq S$ or it has the form $f(t_1, \ldots, t_n)$ where every term t_i is of sort S'_i and OD contains declaration $f : S'_1, \ldots, S'_n \to S'$ with $S' \leq S$.

An $\mathcal{S}$-substitution σ is a finite set $\{x_1 := s_1, \ldots, x_l := s_l\}$ such that it is a substitution and for each variable x of sort S, $\sigma(x)$ is a term of sort S.

An elementary sorted signature $\mathcal{S}$ is equivalent to a system $SC(\mathcal{S})$ of set constraints of the form $S \subseteq S'$ for each subsort declaration $S \leq S'$, and $f(S_1, \ldots, S_n) \subseteq S$ for each operation declaration $f : S_1, \ldots, S_n \to S$.

Definition 3.44. A *unification problem* in signature $\mathcal{S}$ is to determine for given pair t, s of $\mathcal{S}$-terms if there exists an $\mathcal{S}$-substitution σ such that $t\sigma = s\sigma$.

Theorem 3.45. *Let $(\mathcal{S}, E)$ be an order-sorted equational theory such that*

- *$\mathcal{S}$ is elementary*
- *E is linear and shallow*

Then the unification problem for $(\mathcal{S}, E)$ is decidable.

Proof. Consider the following algorithm:

input: equation $t =?s$

output: **yes** if t and s are unifiable, **no** otherwise

1. Solve the unification problem $t' =?s'$ in E, where t', s' are obtained by erasing sort information from t, s. If these terms are not unifiable in E then output **no**, otherwise let $U = \{\theta_1, \ldots, \theta_n\}$ be the set of most general idempotent unifiers of this equation (such set always exist, see [14])

2. for each $\theta = \{x_1 := s_1, \ldots, x_l := s_l\}$ in U we will construct a system $SC(\theta)$ of set constraints. Let $\mathcal{V}$ be the set of individual variables occurring in $t, s, s_1, \ldots, s_l$. Let η be a mapping assigning to each variable $x \in \mathcal{V}$ a set variable V_x. $SC(\theta)$ consists of all constraints from $SC(\mathcal{S})$, constraints $\{V_{x_i} \subseteq s_i\eta \mid i = 1, \ldots, l\}$ and $\{V_x \subseteq S(x) \mid x : S(x) \in \mathcal{V}\}$, and negative constraints $\{V_x \not\subseteq \bot \mid x \in \mathcal{V}\}$

3. If some of the systems $SC(\theta)$ is consistent then output **yes**, otherwise output **no**

Below we prove that the terms t and s are unifiable if and only if the algorithm outputs **yes**.

Proof of $\rightarrow$. Let γ be a (sorted) unifier of t and s. Then there exists an (unsorted) most general unifier θ and a substitution ξ such that $\gamma = \theta\xi$. Let $\theta = \{x_1 := s_1, \ldots, x_l := s_l\}$. Let σ be a substitution mapping all individual variables to some ground terms. Then $V_x = \{x\gamma\sigma\}$ (in particular $V_{x_i} = \{s_i\xi\sigma\}$) gives the solution of the system.

Proof of $\leftarrow$. Assume the system is consistent. Then each set V_x is nonempty. Let u_x be any term from V_x. The substitution $\{x := u_x \mid x \in \mathcal{V}\}$ is a unifier of t and s. □

Acknowledgements

I would like to thank Harald Ganzinger and Leszek Pacholski for their suggestions and discussions. I am also grateful to Jerzy Marcinkowski for reading a draft of this thesis and giving me comments.

References

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

[2] A. Aiken, D. Kozen, M. Vardi, and E. L. Wimmers. The complexity of set constraints. In *1993 Conference on Computer Science Logic*, pages 1–17, September 1993.

[3] A. Aiken, D. Kozen, and E. L. Wimmers. Decidability of systems of set constraints with negative constraints. Technical Report 93-1362, Computer Science Department, Cornell University, June 1993.

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

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

[6] A. Aiken and E. Wimmers. Type inclusion constraints and type inference. In *Proceedings of the 1993 Conference on Functional Programming Languages and Computer Architecture*, pages 31–41, Copenhagen, Denmark, June 1993.

[7] A. Aiken, E. Wimmers, and T.K. Lakshman. Soft typing with conditional types. In *Twenty-First Annual ACM Symposium on Principles of Programming Languages*, Portland, Oregon, January 1994.

[8] A. Aiken and E. L. Wimmers. Solving systems of set constraints (extended abstract). In *Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 329–340, 1992.

[9] L. Bachmair, H. Ganzinger, and U. Waldmann. Set constraints are the monadic class. In *Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 75–83, 1993.

[10] W. Charatonik. Set constraints in some equational theories. In *1st International Conference Constraints in Computational Logics*, LNCS 845, pages 304–319. Springer-Verlag, 1994.

[11] W. Charatonik and L. Pacholski. Negative set constraints with equality. In *Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 128–136, 1994.

[12] W. Charatonik and L. Pacholski. Set constraints with projections are in NEXPTIME. In *Proceedings of the* 35^{th} *Symp. on Foundations of Computer Science*, pages 642–653, 1994.

[13] A. Colmerauer. An introduction to PROLOG III. *Communications of the ACM*, 33:69–90, 1990.

[14] H. Comon, M. Haberstrau, and J.-P. Jouannaud. Decidable problems in shallow equational theories (extended abstract). In *Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 255–265, 1992.

[15] H. Comon, M. Haberstrau, and J.-P. Jouannaud. Syntacticness, cycle-syntacticness, and shallow theories. *Information and Computation*, 111:154–191, May 1994.

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

[17] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints using tree automata. In *10th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 665, pages 505–514. Springer-Verlag, 1993.

[18] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints with negated subset relationships. In *Proceedings of the* 34^{th} *Symp. on Foundations of Computer Science*, pages 372–380, 1993. A full version *Technical report IT 247, Laboratoire d'Informatique Fondamentale de Lille.*

[19] N. Heintze. Set based analysis and arithmetic. Technical Report CMU–CS–93–193, School of Computer Science, Carnegie Mellon University, July 1993.

[20] N. Heintze. Set constraints in program analysis. Draft manuscript, July 1993.

[21] N. Heintze and J. Jaffar. A decision procedure for a class of set constraints (extended abstract). In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 42–51, 1990.

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

[23] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley Publishing Company, Inc., 1979.

[24] N. D. Jones and S. S. Muchnick. Flow analysis and optimization of lisp-like structures. In *Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 244–256, January 1979.

[25] D. Kozen. Logical aspects of set constraints. In *1993 Conference on Computer Science Logic*, pages 175–188, September 1993.

[26] D. Kozen. Set constraints and logic programming (abstract). In *1st International Conference Constraints in Computational Logics*, LNCS 845. Springer-Verlag, 1994. Also to appear in *Information and Computation.*

[27] H. R. Lewis. Complexity results for classes of quantificational formulas. *Journal of Computer and System Sciences*, 21:317–353, 1980.

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

[29] J. C. Reynolds. Automatic computation of data set definitions. *Information Processing*, 68:456–461, 1969.

[30] K. Stefansson. Systems of set constraints with negative constraints are *NEXPTIME*-complete. In *Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 137–141, 1994.

[31] T. E. Uribe. Sorted unification using set constraints. In *11th International Conference on Automated Deduction*, LNAI 607, pages 163–177. Springer-Verlag, 1992.

[32] 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.