Set constraints with projections are in NEXPTIME[*]

Witold Charatonik[†]

University of Wrocław

wch@ii.uni.wroc.pl

Leszek Pacholski[‡]
CRIN (Nancy)
Université de Paris–Sud
Polish Academy of Sciences
pacholsk@ii.uni.wroc.pl

Abstract

Systems of set constraints describe relations between sets of ground terms. They have been successfully used in program analysis and type inference. In this paper we prove that the problem of existence of a solution of a system of set constraints with projections is in NEXPTIME, and thus that it is NEXPTIME-complete. This extends the result of A. Aiken, D. Kozen, and E.L. Wimmers [3] and R. Gilleron, S. Tison, and M. Tommasi [10] on decidability of negated set constraints and solves a problem that was open for several years.

1 Introduction

Set constraints have a form of inclusions between set expressions built over a set of set-valued variables, constants and function symbols. They have been used in program analysis and type inference algorithms for functional, imperative and logic programming languages [4], [5], [12], [13], [15], [16], [18]. Solving a system of set constraints is the main part of these algorithms, however until now the satisfiability problem for such constraints was, except of some special cases, open.

The first results concerning the satisfiability of set constraints were obtained by Heintze and Jaffar [11], who studied a narrow class of so-called definite set constraints. This class was obtained by putting several rather strong conditions on the form of set constraints, however operation of projection was allowed on one (fixed) side of the inclusion symbol. Another restricted class of set constraints was obtained by not allowing projection symbols. This class, called a class of *positive set constraints* is too restrictive to be useful for program analysis but has a relatively easy decision procedure. The decidability of systems of positive set constraints was first established by A. Aiken and E.L. Wimmers [6]. Later other proofs have been obtained. R. Gilleron, S. Tison, and M. Tommasi [9] gave a proof based on automata theoretic techniques and L. Bachmair, H. Ganzinger, and U. Waldmann [7] gave a simple and elegant proof using the decision procedure for the first order theory of monadic predicates, providing also NEXPTIME-completeness of the problem of solvability of positive set constraints. In the paper by A. Aiken, D. Kozen, M. Vardi, and E.L. Wimmers [2], yet another algorithm has been presented and a detailed analysis of the complexity of positive set constrains has been given. Finally, in a recent paper [14] D. Kozen found a mathematical theorem describing the essence of all the above mentioned proofs.

It was noticed in [7] that negated inclusion can be expressed by positive set constraints in the presence of projections, thus the class of set constraints (with projections) contains the class of set constraints without projections but with negated inclusion. The problem of the decidability of sets constraints with negated inclusion was studied by several authors as a first step towards the full problem and turned out to be much more difficult than the positive case. Two solutions were obtained independently by Aiken, Kozen, and Wimmers [3], and Gilleron, Tison and Tommasi [10]. The first solution was later improved by K. Stefansson [17], who proved NP-completeness of the non-linear Diophantine reachability problem which was used to prove the decidability of negated set constraints, and thus he established NEXPTIME-completeness of the original problem. Still another solution was found by the present authors [8], who proved NEXPTIME-

[*]This resarch has been supported by KBN grants 2 1197 91 01 and 8 S503 022 07.

[†]Institute of Computer Science, Przesmyckiego 20, 51-151 Wrocław, Poland, Partially supported by TEMPUS JEP 1941.

[‡]This research was conducted while the second author visited INRIA-CRIN (Nancy) and Université Paris–Sud on leave from the Polish Academy of Sciences. Present affiliation: Institute of Computer Science, University of Wrocław, Przesmyckiego 20, 51-151 Wrocław, Poland.

completeness of set constraints with negated inclusion and unrestricted diagonalizations.

Our proof was based on the idea of L. Bachmair, H. Ganzinger, and U. Waldmann [7] to reduce the decidability problem for positive set constraints to the problem of consistency of first order theories of unary predicates. However, our main contribution was a technical lemma which could easily be expressed and applied in the framework of tree set automata used in [10]. The framework of hypergraphs used in [3] is also closely related to the other ones, however the translation of our lemma to this framework is not immediate.

In this paper we give solution to the full problem, i.e we prove that the problem of existence of a solution of set constraints with projections is NEXPTIME-complete. Since in the presence of projections negative set constraints can be expressed using positive set constraints, our result extends the earlier results. Moreover, the technique of this paper can be combined with the technique of our earlier paper [8] on set constraints with diagonalization to obtain the decidability of the full system of set constraints with projections and diagonalizations. We are planning to include this result in the full paper.

Our proof here is an extension of the proof in [8]. As in [8] we prove that if there is a solution, then there is a certificate (a finite description of a solution) of this fact. Moreover, there exists a certificate of the size exponential in the size of set constraints and the fact that it provides a solution can be checked in time polynomial in the size of the certificate. In the case of negative set constraints a certificate was provided by a (small) finite part of a solution (called a *history* in [8]) having an element in each non-empty atom of the Boolean algebra of sets of terms generated by the sets of terms defined by set expressions that appear in the system of set constraints. In this paper the certificate of consistency provides in addition a set of generators of *witnesses*, which in the case when a term t is in the projection of a set E (say $t \in f^1(E)$), provide an element s such that $f(s,t) \in E$

To make the proof easier we first prove that the problem of existence of a solution for set constraints reduces (in polynomial time) to the case, where all function symbols have the arity at most two.

2 Basic definitions

Let $\Sigma = \{a_1, \ldots, a_{i_0}, f_{1,1}, \ldots, f_{1,i_1}, \ldots, f_{m,1}, \ldots, f_{m,i_m}\}$, where $a_1, \ldots, a_{i_0}$ are constant symbols, $f_{1,1}, \ldots, f_{1,i_1}, \ldots, f_{m,1}, \ldots, f_{m,i_m}$ are function symbols and $f_{i,j}$ stands for j-th symbol of arity i in Σ, and let $\mathcal{V}$ be a set of second order variables. Set expressions are defined by the grammar

$$\begin{aligned} E ::= 0 \mid 1 \mid \alpha \mid E \cup E \mid E \cap E \mid \bar{E} \mid \\ f_{k,i}(E_1, ..., E_k) \mid (f_{k,i})^j(E) \end{aligned}$$

where α is a set variable in $\mathcal{V}$, $f_{k,i} \in \Sigma$, and $j \leq k$. The last operation denoted by $(f_{k,i})^j(E)$ is called a *projection*. A *positive* set constraint is a relation of the form $E \subseteq E'$, and a *negative* set constraint has the form $E \not\subseteq E'$. In the presence of projections negative set constraints can be expressed using the positive ones, so here it suffices to consider only positive set constraints.

Consider a system of set constraints

$$(SC) \qquad E_1 \subseteq E'_1 \wedge \ldots \wedge E_k \subseteq E'_k.$$

Let $T(\Sigma)$ denote the set of ground (constant) terms over Σ and for a set X, let $\mathcal{P}(X)$ denote the set of all subsets of X. Let $E(SC)$ denote the set of all subexpressions of $E_1, E'_1, \ldots, E_k, E'_k$. Each function $\mathcal{S} : \mathcal{X} \to \mathcal{P}(T(\Sigma))$, where $\mathcal{X}$ is the set of variables occurring in (SC), is extended in a unique way to a function $\mathcal{S}$ from $E(SC)$ to $\mathcal{P}(T(\Sigma))$, putting $\mathcal{S}(0) = \emptyset$, $\mathcal{S}(1) = T(\Sigma)$, $\mathcal{S}(E_1 \cup E_2) = \mathcal{S}(E_1) \cup \mathcal{S}(E_2)$, $\mathcal{S}(E_1 \cap E_2) = \mathcal{S}(E_1) \cup \mathcal{S}(E_2)$, $\mathcal{S}(\neg E) = T(\Sigma) \setminus \mathcal{S}(E)$, $\mathcal{S}(f_{k,i}(E_1, ..., E_k)) = f_{k,i}(\mathcal{S}(E_1), ..., \mathcal{S}(E_k)) = \{f(t_1, \ldots, t_k) : t_1 \in \mathcal{S}(E_1), \ldots, t_k \in \mathcal{S}(E_k)\}$, and finally

$$\begin{aligned} \mathcal{S}((f_{k,i})^j(E)) = \{t_j \in T(\Sigma) : \\ (\exists t_1 \ldots \exists t_{j-1} \exists t_{j+1} \ldots \exists t_k) f_{k,i}(t_1, \ldots, t_k) \in \mathcal{S}\}. \end{aligned}$$

A solution of (SC) is a function $\mathcal{S} : \mathcal{X} \to \mathcal{P}(T(\Sigma))$, where $\mathcal{X}$ is the set of variables occurring in (SC), such that $\mathcal{S}(E_i) \subseteq \mathcal{S}(E'_i)$, for each $i \leq k$.

2.1 Reduction to the Monadic Class

Following [7] for each expression $E \in E(SC)$ we introduce a predicate symbol P_E. The predicates P_E are defined inductively by the following formulas: $P_1(y_1) \leftrightarrow true$, $P_0(y_1) \leftrightarrow false$, $P_{E_1 \cup E_2}(y_1) \leftrightarrow P_{E_1}(y_1) \vee P_{E_2}(y_1)$, $P_{E_1 \cap E_2}(y_1) \leftrightarrow P_{E_1}(y_1) \wedge P_{E_2}(y_1)$, $P_{\bar{E}_1}(y_1) \leftrightarrow \neg P_{E_1}(y_1)$,

$$\begin{aligned} P_{f_{k,i}(E_1, \ldots, E_k)}(f_{k,i}(y_1, \ldots, y_k)) \leftrightarrow \\ P_{E_1}(y_1) \wedge \ldots \wedge P_{E_k}(y_k), \qquad (1) \\ P_{f_{k,i}(E_1, \ldots, E_k)}(g(y_1, \ldots, y_{k'})) \leftrightarrow false \\ \text{for} \quad g \in \Sigma \setminus \{f_{k,i}\}, \qquad (2) \end{aligned}$$

and finally

$$P_{(f_{k,i})^j(E)}(y_k) \leftrightarrow \exists y_1 \dots \exists y_{j-1} \exists y_{j+1} \dots \exists y_k P_E(f_{k,i})(y_1, \dots, y_k).$$

Let φ' be the universal closure of the conjunction of the formulas defining predicates P_E, for all $E \in E(SC)$. For the system (SC) the formula φ

$$(\forall y_1)((P_{E_1}(y_1) \to P_{E_1'}(y_1)) \wedge \dots \dots \wedge (P_{E_k}(y_1) \to P_{E_k'}(y_1))) \wedge \varphi'$$

expresses the satisfiability of (SC) in $T(\Sigma)$.

Clearly, φ is not a pure monadic formula since it contains function symbols (in φ'). It can, however, be considered as a (partial) skolemization of a pure monadic formula ψ. More exactly ψ is obtained from φ by replacing each conjunct of the form (1) or (2) respectively by $P_{f_{k,i}(E_1,\dots,E_k)}(z) \leftrightarrow (P_{E_1}(y_1) \wedge \dots \wedge P_{E_k}(y_k))$, and $P_{f_{k,i}(E_1,\dots,E_k)}(z') \leftrightarrow \mathit{false}$, replacing $f(y_1, \dots, y_k)$ everywhere in φ by z and prefixing the matrix of the formula so obtained with $\forall y_1 \dots \forall y_k \exists z \exists z'$. It is worth noticing (it is useful when checking whether ψ holds), that ψ is equivalent to a formula with the quantifier prefix of the form $\forall^* \exists^*$.

Let N be the number of predicate symbols in $\mathcal{P} = \{P_E : E \in E(SC)\}$. If I is a model of ψ, 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 $\equiv$. We identify the interpretation of each unary predicate symbol P^I with the subset $\{x : P(x)\}$ of the domain of I. It has been known for some time (see e.g. [1], p.34) that 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.

Let $\mathcal{S}$ be a solution of (SC). In the following we write simply f instead $f_{k,i}$ hoping that the arity of f will be either clear from the context or irrelevant. If $E(SC) = \{E_1, \dots, E_N\}$, then let $\mathcal{D} = \{\mathcal{S}(\varepsilon(E_1)) \cap \dots \cap \mathcal{S}(\varepsilon(E_N)) : \varepsilon(E_i) \in \{E_i, \overline{E_i}\}\}$ be the set of atoms in the Boolean algebra of subsets of $T(\Sigma)$ generated by the sets $\mathcal{S}(E)$, for $E \in E(SC)$. We can identify $\mathcal{D}$ with the set $T(\Sigma)/\equiv$ of equivalence classes of the relation $\equiv$, so $\mathcal{D}$ contains at most 2^N elements, which are disjoint subsets of $T(\Sigma)$. Let Φ be the quotient mapping from $T(\Sigma)$ into $\mathcal{D}$, i.e. $\Phi(t) = d \in \mathcal{D}$ if $t \in d$. Clearly Φ can be also considered a Boolean homomorphism from $\mathcal{P}(T(\Sigma))$ onto $\mathcal{D}$. Let $E^{\mathcal{D}} = \Phi(E) = \{\Phi(t) : t \in E\}$. It is easy to check, that if we put $(P_E)^{\mathcal{D}} = E^{\mathcal{D}}$, then $(\mathcal{D}; (P_E)^{\mathcal{D}} : E \in E(SC))$ becomes a model of ψ. Of course $\mathcal{S}$ can reconstructed from Φ by putting $\mathcal{S}(X) = \Phi^{-1}(\bigcup\{\varepsilon(E_1^{\mathcal{D}}) \cap \dots \cap \varepsilon(E_N^{\mathcal{D}}) : \varepsilon(X) = X\})$, for all $X \in \mathcal{X}$ (notice, that X is one of E's). So, we shall often call Φ a solution of (SC).

Above we described how to get a model of ψ from a solution of (SC). Also it is easy to check that if Φ is a function from $T(\Sigma)$ to a finite model $\mathcal{D}$ of ψ, then by putting $E = \Phi^{-1}(P_E^{\mathcal{D}})$ we obtain a solution of the *positive part* of (SC). Now, using the fact that z, and the quantifiers $\forall y_1 \dots \forall y_k \exists z$ ($\forall y_1 \dots \forall y_k' \exists z'$) mimic the functions f and g in (1) and (2), given any model $\mathcal{D}$ of (SC), a homomorphism Φ from $T(\Sigma)$ to $\mathcal{D}$ can be defined by induction on the depth of terms. In particular in the case of positive set constraints (SC) without projections each (finite) model of ψ immediately gives a solution to (SC). (See [7] for more details and a more detailed discussion of the notion of *positive*.)

Since the positive part of (SC) is satisfied in any inverse image by Φ of a finite model of ψ, whose existence can be checked by the above mentioned result in [1], it suffices to take care of the negative part. To do this in the case of negative set constraints, it suffices to make sure that the subsets of $T(\Sigma)$ corresponding to non-empty equivalence classes are nonempty, i.e that Φ is onto. In the case of set constraints with projections, it is necessary to make sure that, whenever $F \subseteq f^j(E)$, and $t_j \in F$, then there exist $t_1, \dots, t_{j-1}, t_{j+1}, \dots t_k$ such that $f(t_1, \dots, t_k) \in E$.

2.2 Tree Set Automata

We have studied set constraints using the approach described above. However, we have been told that this approach was difficult to follow, so we shall now say how, what we have done, can be translated into the language of *tree set automata*. In fact, in this version of the paper we tried to switch to the language of tree set automata. However, since it was done in the haste, the switch was perhaps not complete. First we cite some definitions from [10].

Definition 2.1 A tree set automaton (abbreviated by TSA) is a 5-tuple $\mathcal{A} = (\Sigma, \mathcal{Q}, \mathcal{F}, \mathcal{S}, \Omega)$. Σ is a finite *ranked alphabet*, $\mathcal{Q}$ is a finite set of *states*, $\mathcal{F}$ a tuple of sets of *finite states*, $\Omega \subseteq 2^{\mathcal{Q}}$ a set of *accepting states* and $\mathcal{S}$ a finite set of *rules* of the form $f(c_1, ..., c_p) \to c$, with $f \in \Sigma_p$ and $c, c_1, ..., c_p \in \mathcal{Q}$.

If $\mathcal{F} = (F_1, \dots, F_n)$, with $F_i \subseteq \mathcal{Q}$, then n is called the *rank* of $\mathcal{A}$.

Definition 2.2 An $\mathcal{A}$-run r in a tree automaton $\mathcal{A}$ is a function $r : T(\Sigma) \to \mathcal{Q}$ such that whenever $r(t_1) = c_1, \ldots, r(t_p) = c_p$ and $r(f(t_1, \ldots, t_p)) = c$, then $(f(c_1, \ldots, c_p) \to c) \in \mathcal{S}$. A run r is successful if $r(T(\Sigma)) \in \Omega$.

The set of all $\mathcal{A}$-runs of is denoted by $\mathcal{R}$ and the set of all successful $\mathcal{A}$-runs is denoted by $\mathcal{SR}$.

Definition 2.3 Let $\mathcal{A}$ be a TSA of rank n. Then the set $\mathcal{L}(\mathcal{A})$ of n-tuples of tree languages *recognized* by $\mathcal{A}$ is defined as follows:

$$\mathcal{L}(\mathcal{A}) = \{\mathcal{L}(\mathcal{A}, r) : r \in \mathcal{SR}\}$$
$$\mathcal{L}(\mathcal{A}, r) = (L_1(\mathcal{A}, r), \ldots, L_n(\mathcal{A}, r))\}$$
$$\forall i \in [1, n], L_i(\mathcal{A}, r) = \{t \in T(\Sigma) : r(t) \in F_i\}$$

A TSA is complete, if for each $f \in \Sigma_p$, and for each $c_1, \ldots, c_p \in \mathcal{Q}$, there exists $c \in \mathcal{Q}$, such that $(f(c_1, \ldots, c_p) \to c) \in \mathcal{S}$.

The translation of the approach using the monadic class to the tree set automata is easy. The states of $\mathcal{A}$ correspond roughly to the elements of the finite model I. The above mentioned result on the monadic class from [1], gives, for a consistent system (SC) of set constraints, a complete TSA $\mathcal{A}$, such that any run of $\mathcal{A}$ provides a solution of the positive part of (SC) and the notion of a successful run correspond to the fact that the function Φ is onto. A rule $f(c_1, \ldots, c_i) \to c$ of a TSA is the homomorphic image of the function $f : (T(\Sigma))^i \to T(\Sigma)$ in $\mathcal{D}$. To save space we omit the details.

The notion of a successful run describes a solution of a system of negative set constraints. It is not difficult to define a (perhaps not very automata-theoretic) notion of a TSA which captures set constraints with projections. To do so, it suffices to replace the definition of Ω by:

Ω is a finite set of quadruples of the form (Q_1, Q_2, f, i), where $Q_1, Q_2 \subseteq \mathcal{Q}$, $f \in \Sigma_k$, $i \leq k$

and to say that

a $\mathcal{A}$-run r is very successful if, whenever $r(t) \in Q_1$, for $t \in T(\Sigma)$, then there exist $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots t_k$, such that $r(f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots t_k)) \in Q_2$.

2.3 Reduction to the binary case

Lemma 2.4 *The problem whether a systems of set constraints has a solution can be reduced (in polynomial time) to such a problem over vocabularies containing no function symbols of arity greater then two.*

Proof. For each n-ary function symbol f we introduce $n - 1$ binary function symbols $f_1, \ldots f_{n-1}$. Then, we replace each occurrence of an expression $f(E_1, \ldots, E_n)$ by $f_1(E_1, f_2(E_2, \ldots f_{n-1}(E_{n-1}, E_n) \ldots)$ and each occurrence of a projection $f^i(E)$ by an expression $f_1^2(f_2^2(\ldots f_{i-1}^2(f_i^1(E \cap f_1(1, f_2(1, \ldots f_{n-1}(1, 1) \ldots)$.

It is easy to see that the system so obtained has a solution if and only if the original one has. □

2.4 A sketch of the proof

A germ of a solution is a fragment of a solution (perhaps with some additional structure) sufficient to reconstruct a (regular) solution. In the case of positive set constraints the empty solution is a germ (assuming a complete automaton is given). Given an automaton arising from a solution of the monadic translation each run of this automaton will generate a solution. In the case of negated constraints it is not sufficient. It may happen that a run omits a state i.e. a set, corresponding to an expression in (SC), which should be non-empty, is empty. Now, to give the reader some idea of how our proof works, we shall discuss in more details the solution for the case on negative set constraints. It is simpler, but already contains some ideas and notions necessary for the full solution.

A set of terms is *closed*, if with every term it contains all its subterm. A germ G is a *partial run*, defined on a closed set of terms, such that each state, which is reached in the solution, is already assumed by a term in G. We claim that if a solution exists, then there is a solution generated by a germ whose size is exponential in the size of the set constraints, and in fact polynomial in the number of states of the automaton (More exactly $2M^3$, where M is the number of states).

To do this we extend the structure a a germ to a structure of a *skeleton*. The idea is as follows. We imagine that a germ is generated from the empty set in a *sequential* way, i.e we assume that there is a fixed linear order (order of construction) on G. Moreover, we assume that this linear order is an extension of the subterm relation, i.e. that if in some step we introduce and label a new term, then its two immediate subterms are already in the germ and are already labeled. The label of the newly added term is such that the labels so far defined describe a partial run of the automaton. If the process so described could be considered linear, then some kind of *pumping lemma* argument could be used to prove that a large germ can be replaced by a small *germ like* structure. However the construction is not linear, rather it has an (order) structure of an acyclic directed graph. In each step of the construction two subterm are used. Moreover, as it has been noticed by several persons, the *pumping lemma* type of argument can not be applied to the paths in the terms. We are going to make the process *almost everywhere* linear by adding some extra structure - in fact the process will have the structure of a forest (a collection of trees) with a lot of straight branches.

In each term we distinguish the *main subterm* - which is the one constructed later, we call it *the mother*, and we treat the other one, *the father*, as a parameter of the construction step. Notice, that in each construction step we want a term which can be labeled with a given state (of the proper color) - this depends only on the colors the the parents, and we want the new term to be really *new*, i.e. we want it to be different from all terms constructed so far. But if the *mother* was *new*, i.e. was not yet used with the given function symbol and with any *father* of a given color, then instead of the actual father we can take arbitrary *father* of the proper color and we still get a good, i.e. *new* and properly colored term. So, we can always try to use the first father of the proper color (i.e. the one that has been first constructed). When it works, the construction becomes linear, and we can cut off the segments of the construction joining the terms of the same color. However, the construction is not so simple, since there are *forks*, i.e. mothers which have (inside the germ) several children with fathers of the same color and using the same function symbol. Then several different fathers have to be used, but again we choose them as small as possible.

We have almost defined a *skeleton*, which is a germ, with a structure of a directed graph, directed edges going from each mother to her children, each edge being labeled with the function symbol, the color of the father, its number (in the numbering of the terms of the same color), and the function symbol (we write $f(*, c[i])$ or $f(c[i], *)$ to denote that f is the function symbol, c is the color of the father, i is his number and $*$ denotes whether the mother is the left or the right subterm).

So, in a germ we have three types of terms: milestones - the first k elements of each color, that will serve as fathers (or are our goals), forks, and the generic elements (neither forks nor milestones). Now, an easy combinatorial argument can show, that if there exists a germ, then we can always find a skeleton with at most M milestones in each class, and so, with at most M^2 forks, each of degree at most M. Therefore in such a skeleton there is at most $2M^2$ *straight lines* where a pumping lemma arguments work, so the size of each *straight line* can be bounded by M giving the germ of size $2M^3$.

This finishes the sketch of the proof of the NEXPTIME upper bound on the complexity of negative set constraints.

We extend this idea to the case with projections as follows. In addition to milestones and forks we have another kind of non-generic elements: *universal witnesses.* Let c be a color of a term t such that $t \in f^1(E)$. Then the universal witness for c, i.e. for (almost all) terms t of the color c, and for $f^1(E)$, is an element s such that $f(s,t) \in E$ and s is not a milestone. A relatively simple argument shows that it is possible to saturate a skeleton with *universal witnesses* in such a way, that all generic elements (and all forks) will have all necessary witnesses (universal or natural) in the skeleton. It remains to add some *special witnesses* for some troublesome elements (milestones and universal witnesses). This is quite technical and requires consideration of many cases. In fact, the technical proof below can be slightly simplified (some cases can be removed), but we did not risk changing the technical proof at the last moment.

3 The Result

In this section we prove that if a system of set constraints has a solution, then there exists a finite description of some solution. We call this finite description a *germ* of the solution. The germ is obtained by collapsing a *pre-germ* and a pre-germ is obtained by a "pumping lemma surgery" from a *skeleton.* To define the notion of a skeleton and to prove that it exists and is small we need some definitions and lemmas.

We fix a linear ordering $\prec$ extending the relation of being a subterm (i.e. if s is a subterm of t, then $s \prec t$). We assume, that we are given a solution, and that Φ is a quotient mapping of $T(\Sigma)$ onto a finite model I. We can also think, that we are given an automaton $\mathcal{A}$ corresponding to (SC) whose states can be identified with the elements of I and that Φ is a (very successful) run of $\mathcal{A}$.

In the definition below $c[i]$ 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 3.1 A semi-skeleton of a solution Φ 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 $c \in \Phi(t(\Sigma))$ 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 one of the forms: $f(*, c[i])$, $f(c[i], *)$, $f(*)$

7. each rule is consistent with the rules of $\mathcal{A}$, i.e. if the edge from s to t is labeled by $f(*, c[i])$, $f(c[i], *)$, or $f(*)$, then, respectively $f(\Phi(s), c) \rightarrow \Phi(t)$, $f(c, \Phi(s)) \rightarrow \Phi(t)$, or $f(\Phi(s)) \rightarrow \Phi(t)$, is a rule of $\mathcal{A}$.

8. if $c[j]$ is used in a label of an edge $t \rightarrow s$ 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

Convention In all the proofs below we tacitly consider only one of the two symmetric cases: $f(*, c[i])$ and $f(c[i], *)$. The same applies to other notions that are introduced later and are symmetric.

Lemma 3.2 *If a system of set constraints has a solution Φ then there exists a semi-skeleton of Φ with at most M^2 milestones, where M is the cardinality of $\Phi(T(\Sigma))$.*

Proof. This lemma has been implicitly proved in [8] and so, has been omitted. We shall include the proof in the a journal version. □

Lemma 3.3 *For any set of terms of cardinality T, there exists its closure to a semi-skeleton with at most MT milestones.* □

Definition 3.4 Given a semi-skeleton $\mathcal{S}$ we say that a term t' is a collapse of a node $t \in \mathcal{S}$ if either t is a constant symbol and $t' = t$, or $t' = f(t_1, s')$ (respectively $t' = f(s', t_1)$ or $t' = f(s')$) and there is an edge $s \rightarrow t$ labeled by $f(c[j], *)$ (respectively $f(*, c[j])$ or $f(*)$), the term s' is the collapse of the node s and t_1 is the collapse of $c[j]$. The collapsing function is a function mapping each node of the semi-skeleton to its collapse.

Lemma 3.5 *The collapsing function is 1 to 1.*

Proof. A version of this lemma has been given in [8] and so, its proof has been omitted.

By the closure of a set of terms (or a graph) to a semi-skeleton we mean a minimal semi-skeleton containing this set of terms (or this graph).

Definition 3.6 A semi-skeleton (of a solution Φ of a system (SC) of set constraints) without local conflicts is a semi-skeleton with additional labels of nodes such that

- if for some node t we have $t \in f^1(E)$ (respectively $t \in f^2(E)$) for some binary function symbol f and $f^1(E)$ (respectively $f^2(E)$) occurs in (SC) then t is labeled with additional label of the form $f(*, c_1[i]) : c_2$ (respectively $f(c_1[i], *) : c_2$) where $c_2 \in \Phi(E)$
- if $c[j]$ is used in a label of any node then there are at least j milestones labeled by c
- if a node t is labeled by $f(*, c_1[i]) : c_2$ (respectively $f(c_1[i], *) : c_2$) and there is an edge $t \rightarrow s$ labeled by $f(*, c_1[i])$ (respectively $f(c_1[i], *)$) then s is labeled by c_2.

Below we shall prove that if a system of set constraints has a solution, than there exists a semi-skeleton without local conflicts and with bounded number of milestones. We need more definitions and lemmas.

Definition 3.7 A milestone $c[i]$ is called an *universal witness (u-witness)* if $c[i]$ is not used in the label of any edge.

Definition 3.8 A term s is a witness of a projection $f^1(E)$ (respectively $f^2(E)$) for a term t, if $f(t, s) \in E$ (respectively $f(s, t) \in E$).

Definition 3.9 We say that there is an unsolved local conflict at a node t in a semi-skeleton, if $f^1(E)$, (or $f^2(E)$) appears in (SC), and for each witness s of this projection the following holds: for some $k \geq 1$, there are k milestones $\Phi(s)[1], \ldots, \Phi(s)[k]$ in $\Phi(s)$, for some $k' \geq 0$, there are edges outgoing from t labeled by $f(*, \Phi(s)[1]), \ldots, f(*, \Phi(s)[k'])$, t is labeled by $f(*, \Phi(s)[k'+1]) : c_{k'+1}, \ldots, f(*, \Phi(s)[k]) : c_k$, and none of $\Phi(f(*, \Phi(s)[1])), \ldots, \Phi(f(*, \Phi(s)[k']))$, $c_{k'+1}, \ldots, c_k$ belongs to $\Phi(E)$.

Lemma 3.10 *Let a semi-skeleton $\mathcal{S}$ of a solution Φ of (SC) be a closure to a semi-skeleton of at most*

$M+n$ milestones, including n u-witnesses. Suppose, that there is a node t in $\mathcal{S}$ with an unsolved local conflict. Then, there exists a semi-skeleton $\mathcal{S}'$ containing $\mathcal{S}$, with solved conflict at t, and without new unsolved local conflicts. Moreover, $\mathcal{S}'$ is a closure to a semi-skeleton of at most $M+n+1$ terms, and has at most $n+1$ u-witnesses.

Proof. Suppose, there is an unsolved local conflict at a node t and let s be a witness of the projection $f^1(E)$ (arguments for $f^2(E)$ are symmetric). We have $k+1$ terms $s, s_1, \ldots, s_k$ with $\Phi(s) = \Phi(s_1) = \ldots = \Phi(s_k)$, where $f(t, s_1), \ldots f(t, s_{k'})$ are successors of edges outgoing from t and $s_{k'+1}, \ldots s_k$ are witness of respective projections.

So, there are at least $k+1$ terms in the set $\Phi^{-1}(\Phi(s))$ and we can take $k+1$-th (according to the ordering $\prec$) term $\Phi(s)[k+1]$ in this set, put it into the set of milestones, close the graph so obtained to a semi-skeleton $\mathcal{S}'$ and label t by $f(*, \Phi(s)[k+1]) : \Phi(f(t,s))$. This certainly solves the conflict at t, but some new local conflicts can be created, since degrees of forks in $\mathcal{S}$, which were bounded by $M+n$, can increase to $M+n+1$, so some milestones can stop being u-witnesses.

Suppose, $c[j]$ is an u-witness in $\mathcal{S}$ but is not $\mathcal{S}'$. This means that $c[j]$ is in the closure of all other milestones of $\mathcal{S}$, so $\mathcal{S}'$ is generated by $M+n$ terms. Now, if there is a term which is not a milestone in the set $\Phi^{-1}(c)$, we can add it into the set of milestones (and change references in labels of nodes from c[j] to this new milestone) still preserving the estimates of the number of milestones. If there is no such term in $\Phi^{-1}(c)$, then there are no new unsolved local conflicts concerning $c[j]$. The procedure described above will terminate because degrees of forks are bounded by $M+n+1$, so in each equivalence class there is at most $M+n+1$ milestones, which are not u-witnesses. □

Lemma 3.11 *If a system (SC) of set constraints has a solution Φ, then there exists a semi-skeleton of Φ without local conflicts and with at most $M(M+MP)$ milestones, where P is the number of projection expressions occurring in (SC).*

Proof. Let $\mathcal{S}$ be a semi-skeleton. For each node t of $\mathcal{S}$ such that $t \in f^1(E)$, let s be a witness of the respective projection. If there is a milestone $\Phi(s)[i]$ not used in a label of the form $f(*, \Phi(s)[i])$ of edge outgoing from t (or if the successor t' of such an edge belongs to E), then we take the minimal such milestone, and label t by $f(*, \Phi(s)[i]) : \Phi(f(t,s))$.

Repeating this we get either a semi-skeleton without local conflicts or a semi-skeleton with unsolved local conflicts, which can be eliminated using lemma 3.10

It is easy to see that in a semi-skeleton there is no unsolved local conflict, if in each equivalence class either all terms are milestones or there are at least P milestones for projections. □

Using the same arguments we can prove the following

Lemma 3.12 *For any set T of terms there exists its closure to a semi-skeleton without local conflicts with at most $M(|T|+MP)$ milestones.*

Definition 3.13 A semi-skeleton without conflicts between edges and projections is a semi-skeleton without local conflicts such that

- if $t = c_1[i]$ is labeled by $f(*, c_2[j]) : c'$ and $f(c_1[i], *)$ is a label of an edge $s \to s'$, where $s = c_2[j]$, then $\Phi(s') = c'$
- if $t = c_1[i]$ is labeled by $f(c_2[j], *) : c'$ and $f(*, c_1[i])$ is a label of an edge $s \to s'$, where $s = c_2[j]$, then $\Phi(s') = c'$

Lemma 3.14 *If a system (SC) of set constraints has a solution Φ, then there exists a semi-skeleton of Φ without conflicts between edges and projections, and with at most $2M(M+2MP)$ milestones.*

Proof. Let $\mathcal{S}$ be a semi-skeleton without local conflicts. Suppose that $t = c_1[i] \in f^1(E)$ is a node in $\mathcal{S}$ labeled by $f(*, c_2[j]) : c'$, such that for some $t' \in \mathcal{S}$ the edge $c_2[j] \to t'$ is labeled by $f(c_1[i], *)$, and $\Phi(t') \neq c'$. Let s be a witness of $f^1(E)$ for t. Below we show how to solve this conflict by adding some new milestones.

Since $\mathcal{S}$ is a semi-skeleton, we have $t \prec c_2[j]$, and all the terms preceding t in $\Phi^{-1}(c_2)$ are milestones. Therefore we can permute milestones $c_2[1], \ldots, c_2[j-1]$ in the labels of t and in the edges outgoing from t, in such a way that each edge $t \to f(t, x_l)$ is labeled by $f(*, c_2[j_l])$, where $x_l = c_2[j_l]$.

We first assume that $s \prec t$. Therefore s is a milestone, say $s = c_2[k]$. We claim that there is no edge $t \to f(t,s)$. In fact, such an edge could be labeled by $f(*, c_2[k])$, and t could be labeled by $f(*, c_2[k]) : \Phi(f(t,s))$, which would imply that there is no conflict.

Since all milestones $c_2[1], \ldots, c_2[j]$ are used in labels, and s is one of them, t has a label $f(*, c_2[k]) : c''$. Replacing the labels $f(*, c_2[j]) : c'$ and $f(*, c_2[k]) : c''$ by

$f(*, c_2[j]) : c''$ and $f(*, c_2[k]) : c'$ we solve the conflict, but it may happen that we introduce another one. Repeating this procedure we either solve all conflicts, or get an unsolved problem for a node, which is smaller than its witness. So, let us assume that $t \prec s$.

Now, we consider two cases:

Case 1 There is a term $s' \in \Phi^{-1}(c_2)$ such that, either no edge outgoing from s' is labeled by $f(t, *)$, or an edge with this label has the successor in $\Phi^{-1}(c')$. In this case we can take the minimal such term, add it, if necessary, to the set of milestones and replace the label $f(*, c_2[j]) : c'$ by $f(*, s') : c'$.

Case 2 There is no such term, and in particular s is not such a term. Since $t = c_1[i]$, s is a fork of degree $k \geq i$, and there are k edges outgoing from s to $f(x_1, s), \ldots, f(x_k, s)$ labeled by $f(c_1[j_1], *), \ldots, f(c_1[j_k], *)$ with $i \in \{j_1, \ldots, j_k\}$. Again we consider two cases:

Case 2.1 For some l, we have $\Phi(f(t, s)) = \Phi(f(x_l, s))$, and the term $c_1[j_l]$ is not labeled by $f(*, s) : \Phi(f(x_l, s))$. In this case the conflict is solved if we exchange the labels $f(c_1[i], *)$ with $f(c_1[j_l], *)$. It is easy to check, that we do not replace one conflict by another one.

Case 2.2 For each l we have, either $\Phi(f(t, s)) \neq \Phi(f(x_l, s))$, or $\Phi(f(t, s)) = \Phi(f(x_l, s))$ and the term $c_1[j_l]$ is labeled by $f(*, s) : \Phi(f(x_l, s))$. Since t is not labeled by $f(*, s) : \Phi(f(t, s))$, for each l we have $t \neq x_l$. Note, that there is at least $k + 1$ terms $t, x_1 \ldots, x_k$ in $\Phi^{-1}(c_1)$, preceding s. If, as in Case 2.1, s has is labeled by $f(c_1[l], *) : c'$, the conflict can be solved by permuting milestones in labels. We can label edges using k milestones which precede s and are different from t, and use t for the projection. So, let us assume that $d \neq c'$ for all labels $f(c_1[\ldots], *) : d$ of s. If there is a milestone $c_1[m]$ not used in any label of s, or of an edge outgoing from s, the conflict can be solved using $c_1[m]$. We can label the edges by milestones preceding s which are different from t. So, let us assume that all the milestones in $\Phi^{-1}(c_1)$ have been used.

Let $i = j_l$. Since $\Phi(x_{j_l}) = \Phi(t)$, $x_{j_l} \in f^1(E)$, and there is a term s', such that $f(x_{j_l}, s') \in E$. Note, that $s' \neq s$, because $f(x_{j_l}, s) \notin E$. Let $\Phi(s') = c$.

We consider two subcases.

Case 2.2.1 There exists a milestone $c[m]$ such that the label $f(*, c[m]) : \Phi(f(x_1, s'))$ of t introduces no new conflict between projection and any edge outgoing either from $c''[m]$ or from t. In this case we just label t by this label. Note, that we do not introduce new milestone here, and that this is the only case when we admit a label of the form $f(*, c[m]) : c''$ with no witness of the respective projection in the set $\Phi^{-1}(c)$.

Case 2.2.2 For each milestone $c[m]$ in $\Phi^{-1}(c)$ the label $f(*, c[m]) : \Phi(f(x_1, s')$ of t introduces new conflict between projection and edge outgoing either from $c[m]$ or from t. In this case we introduce a new milestone in the set $\Phi^{-1}(c_1)$. We can do it, because the number of terms in this set is greater than the number of milestones. In fact, otherwise the conflict could be solved by a permutation which labels $s \to f(x, s)$ by $f(x, *)$, and labels s by $f(x_m, *) : \Phi(f(x_m, s))$, where x_m are witnesses of projections. However, in this case we do not take as the new milestone the first (in the ordering $\prec$) term from the set $\Phi^{-1}(c_1)$ which is not a milestone, but first such term which either has a witness of the projection $f^1(E)$ different from s, or is a node x of $\mathcal{S}$ labeled by $f(*, s) : \Phi(f(x, s))$. This prevents us from replacing one conflict between edge and projection with another one. Note that the terms $x_1, \ldots, x_k$ and all chosen witness of projections for s have this property, and the last (according to $\prec$) of them is not a milestone.

This ends the consideration of cases. Note, that in the cases 1 and 2.2.2 we introduce new milestones, so we have to close the graph to a semi-skeleton without local conflicts. After this step some of milestones introduced in the proof of this lemma may no longer solve conflicts between edges and projections, but arguments here is very similar to that of the lemma 3.10. If $\mathcal{S}$ is a closure of $M + MP + n$ milestones and one of them no longer solves the problem it should solve, then this one is obtained in the procedure of closing, so $\mathcal{S}$ is a closure of $M + MP + n - 1$ milestones and we can add a new one without changing the estimates. More explanation will be given after the proof of lemma 3.16.

Now, we shall estimate the number of milestones that have to be introduced. Note, that if in this proof we have introduced P milestones in $\Phi^{-1}(c_2)$, then we certainly do not have to introduce another one. This can be shown just as case 1 with the chosen term being a milestone. Therefore Case 1 introduces at most MP new milestones. Thus if we add new milestones only in the Case 1, then degrees of forks are bounded by $M + 2MP$. Now, we shall show that introducing a milestone in the case 2.2.2 does not increase estimation of degrees of forks.

Let us recall this estimate. We estimated that in $\mathcal{S}$ there are at most $M + 2MP$ terms of depth $d + 1$, where d is the depth of s. s is a fork of degree k, so s is the only subterm of k such terms. Then we introduce at most $k-1$ new terms of depth $\leq d$ (the milestones used in the labels of edges outgoing from s). But we know that there is a conflict between some projection and some edge outgoing either from t or from $c[1]$. If the conflict concerns the edge outgoing from t, then t is a subterm of some other term in $\mathcal{S}$, so t is not introduced as a new term of depth $\leq d$ and we introduced there at most $k-2$ terms. Since $t \prec x_k$ and x_k is of depth $d-1$, after introducing the new milestone we still have at most $M+2MP$ terms of the same depth. If the conflict concerns the edge outgoing from $c[1]$, then $t \prec c[1]$, $c[1]$ is a fork of degree $\geq i$ (remember that $t = c_1[i]$), so we have at most $M-1+2MP$ terms of depth $max(depth(s), depth(c[1])$ (because $c[1]$ is not maximal term in $\mathcal{S}$ according to subterm ordering) and at most $M-1+2MP-(i-1) = M+2MP-i$ terms of depth lower than $max(depth(s), depth(c[1]))$, because both $c[1]$ and s are forks of degree $\geq i$ and $c_1[1], \ldots, c_1[i]$ are of depth lower than $min(depth(s), depth(c[1])$. Again after introducing a new milestone we have at most $M+2MP$ terms of the same depth, so degrees of forks are bounded by $M+2MP$ and the total number of milestones in $\mathcal{S}$ is bounded by $M(M+2MP)$ plus the number of milestones added in the case 2.2.2.

When we know the estimates of degrees of forks, we can estimate the number of milestones added in the case 2.2.2. It is easy to see that if in the case 2.2.2 we introduce $M+2MP$ new milestones ($k \leq M+2MP$ is the degree of s) in the set $\Phi^{-1}(c_1)$ then we solve all conflicts between projections for terms in $\Phi^{-1}(c_1)$ and edges outgoing from s (and any other fork) – this is because adding a milestone solves the problem for one term (one milestone in $\Phi^{-1}(c_1)$) and the degree of s remains bounded by $M+2MP$. Hence we have to introduce at most $M+2MP$ milestones in this set and the total number of milestones added in the case 2.2.2 is bounded by $M(M+2MP)$. □

Definition 3.15 A skeleton of a solution Φ of a system (SC) of set constraints is a semi-skeleton without conflicts between edges and projections such that

- if $t = c_1[i]$ is labeled by $f(*, c_2[j]) : c'$, and $s = c_2[j]$ is labeled by $f(c_1[i], *) : c''$, then $c' = c''$

Lemma 3.16 *If a system (SC) of set constraints has a solution Φ then there exists a skeleton of Φ with at most $M(M+3MP) + M(M+2MP)$ milestones.*

Proof. Let $\mathcal{S}$ be a semi-skeleton of Φ whose existence is provided by the previous lemma. Suppose that there is a conflict between two projections, i.e. there is a term $t = c_1[i]$ labeled by $f(*, c_2[j]) : c'$, and a term $s = c_2[j]$ labeled by $f(c_1[i], *) : c''$, with $c' \neq c''$.

Let us consider two cases:

Case i) the label $f(*, c_2[j]) : c'$ of t was defined in the case 2.2.1 of the proof of the previous lemma. This was the only case when we admitted a label of the form $f(*, c[m]) : c''$ with no witness of the respective projection in the set $\Phi^{-1}(c)$.

In this case we can find in the set $\Phi^{-1}(c_1)$ a term which is not a milestone (arguments for this is given at the beginning of the case 2.2.2 in the previous lemma), so we can introduce a new milestone $c_1[m]$, and label s by $f(c_1[m], *) : c''$, which solves the conflict.

Case ii) both labels were defined at some other point of proof, so we have desired witness in respective sets. We consider two sequences of terms: $f(t, y_1), \ldots, f(t, y_m)$ and $f(x_1, s), \ldots, f(x_n, s)$ with $\Phi(y_1) = \ldots = \Phi(y_m) = \Phi(s) = c_2$ and $\Phi(x_1) = \ldots = \Phi(x_n) = \Phi(t) = c_1$. These are the successors of edges outgoing from t and s, and witnesses of respective projections. There are two possibilities:

Case ii.1 $f(t, s)$ is a member of both these sequences. In this case we can permute milestones in labels in such a way that we get labels $f(*, c_2[j]) : \Phi(t, s)$ and $f(c_1[i], *) : \Phi(t, s)$.

Case ii.2 $f(t, s)$ is not a member of one of these sequences. Without loss of generality we can assume it is not a member of the first one. So we have $m+1$ different terms $y_1, \ldots, y_m, s$ with $\Phi(y_1) = \ldots = \Phi(y_m) = \Phi(s) = c_2$, so we can find in $\Phi^{-1}(c_2)$ a term which is neither used as a milestone in any label of t, nor in any label of an edge outgoing from t. We use it to solve the conflict and, if necessary, we add it to the set of milestones.

The same arguments as in previous lemmas shows that it suffices to add at most P milestones in $\Phi^{-1}(c)$, for each c, which gives at most MP new milestones. Then we have to close $\mathcal{S}$ with these new milestones to a semi-skeleton without conflict between edges and projections which again does not increase the estimates of the number of milestones (see below). □

Estimation of the number of milestones.

The following procedure ($skeleton(\Phi)$) computes, for given solution Φ of a system of set constraints, a skeleton of this solution. This procedure does not solve systems of set constraints. It is only used to prove, that if a system has a solution, then there exists a skeleton with bounded number of milestones.

```
procedure close(S)
begin
    S:=closure of the graph S to a semi-skeleton
            (* use Lemma 3.3 *)
    while (there is an edge $t \to s$ labeled by $f(*, c[i])$
            (or $f(c[i], *)$) and a label $f(*, c[i]) : c'$
            (or $f(c[i], *) : c'$) of $t$, with $\Phi(s) \neq c'$ )
    do
        erase the label $f(*, c[i]) : c'$
            (respectively $f(c[i], *) : c'$)
end

procedure solve-local-conflicts(S)
begin
    repeat
        close(S)
        label the nodes in S (* use Lemma 3.11 *)
        if there exists unsolved local conflict then
            $\mathcal{S}$:=$\mathcal{S} \cup \{newmilestone\}$
                (* use Lemma 3.10 *)
    until there is no local conflict in S
end

procedure solve-conflicts-edges-projections(S)
begin
    repeat
        solve-local-conflicts(S)
        if there exists a conflict between edge and
                projection then
            either $\mathcal{S}$:=$\mathcal{S} \cup \{newmilestone\}$
                (* use Lemma 3.14, case 1 or case 2.2.2 *)
            or solve the conflict (* use
                Lemma 3.14, cases different from 1
and 2.2.2 *)
    until there are no conflicts between edges
                and projections in S
end

procedure skeleton(Φ)
begin
    $\mathcal{S} := \emptyset$
    for each $c \in \Phi(T(\Sigma))$ do $\mathcal{S}$:=$\mathcal{S} \cup$ minimal term
                from $\Phi^{-1}(c)$
    repeat
        solve-conflicts-edges-projections(S)
        if there exists a conflict between two
                projections then
            $\mathcal{S}$:=$\mathcal{S} \cup \{newmilestone\}$
                (* use Lemma 3.16 *)
    until there are no conflicts in S
end
```

We classify the milestones in $\mathcal{S}$ as follows. We say that a milestone $c[i]$

- is a fork milestone if $i = 1$, or it is used in the procedure *close* (i.e. if there is a fork t of degree greater than or equal to i with outgoing edges labeled by $f(c[\ldots], *)$ or $f(*, c[\ldots])$)
- is a u-witness if it is not a fork milestone and is used in the procedure *solve-local-conflicts* (equivalently there is a fork of degree k_1 with edges labeled $f(*, c[\ldots])$ (or $f(c[\ldots], *)$), with k_2 labels of the form $f(*, c[\ldots]) : c'$) (respectively $f(c[\ldots], *) : c'$), and $k_1 + k_2 \geq i$)
- is a 2.2.2-witness if it is not of above types and is used in the procedure *solve-conflicts-edges-projections* in case 2.2.2 (equivalently it is neither fork milestone, nor u-witness and there is an edge labeled by $f(*, c[\ldots])$ or $f(c[\ldots], *)$)
- is a 1-witness if it is not of above types and is used in the procedure *solve-conflicts-edges-projections* in case 1
- is a s-witness (special witness) if it is not of above types and is used in the procedure *skeleton*

Note that

- each milestone is of some type
- during the execution of *skeleton* the type of a milestone can change
- in each moment of the execution:
  - the number of fork milestones is bounded by M multiplied by the degrees of forks
  - the number of u-witnesses is bounded by MP
  - the number of 1-witnesses is bounded by MP
  - the number of 2.2.2-witnesses is bounded by $M + 2MP$
  - the number of s-witnesses is bounded by MP
  - degrees of forks are bounded by $M + 3MP$ (i.e. the number of fork milestones, u-witnesses, 1-witnesses, and s-witnesses)
- if in any moment of execution there is a conflict in $\mathcal{S}$, then the number of milestones of respective type is strictly lower then indicated above (e.g. if we have to add some new term to $\mathcal{S}$ to solve a conflict between two projections, and after adding this term there is a new edge which causes a conflict with some projection, then the number of milestones solving conflicts between edges and projections is strictly lower than $M + 3MP$)

Definition 3.17 A *pre-germ* of a solution Φ is any labeled graph $\mathcal{S}$ such that

- $\mathcal{S}$ satisfies all conditions from the definition of skeleton except of point 8 of Definition 3.1.
- the collapsing function on $\mathcal{S}$ is 1 to 1

Lemma 3.18 *If there exists a skeleton of a solution Φ of a system (SC) of set constraints, with at most T milestones, then there exists a pre-germ with T milestones and at most $2MT$ nodes.*

Proof. Let $\mathcal{S}$ be a skeleton with T milestones. Note that since all leaves in $\mathcal{S}$ are milestones and $\mathcal{S}$ is a forest, there are at most T nodes of outdegree greater than one.

Suppose that $\mathcal{S}$ has more than $2MT$ nodes. Then, there exists a path $u_1 \to \ldots \to u_m$ with $m \geq M$ 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 $\mathcal{S}_1$. The collapsing function remains 1 to 1. In fact, if a term s' in $\mathcal{S}_1$ is a collapse of two nodes s_1, s_2 then $s'[u_i/u_j]$ in $\mathcal{S}$ 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. Repeating this procedure we get a pre-germ with at most $2MT$ nodes. $\square$

Lemma 3.19 *The collapse of a pre-germ $\mathcal{S}$ of a solution Φ of a system (SC) of set constraints can be extended to a (perhaps different, regular) solution Φ' of (SC).*

Proof. We define Φ' as follows:

First, for each node t of $\mathcal{S}$, we define $\Phi'(\mathit{collapse}(t)) = \Phi(t)$. This definition is correct because the collapsing function is one-to-one.

Second, for each node t of $\mathcal{S}$ and each label $f(*, c[i]) : c'$ (and for $f(c[i], *) : c'$ symmetrically) we define $\Phi'(f(\mathit{collapse}(t), \mathit{collapse}(c[i])) = c'$ (and for $f(c[i], *) : c'$ symmetrically). This definition is correct, because $\mathcal{S}$ is a skeleton without conflicts concerning projections.

Third, for each pair of milestones $c[i], c'[j]$ and each binary function symbol f, if $\Phi' f(\mathit{collapse}(c[i]), \mathit{collapse}(c'[j]))$ is not defined yet, then we define it as $\Phi(f(c[i], c'[j]))$

Fourth, we repeat the following step until it does not extend Φ' to new terms: for each term t (which is not a node of $\mathcal{S}$) such that $\Phi'(t) \in \Phi(f^1(E))$ (and symmetrically for $f^2(E)$) for some projection expression occurring in (SC) we define $\Phi'(f(t, \mathit{collapse}(c[i])) = c'$, where $f(*, c[i]) : c'$ is the proper label of $\Phi'(t)[1]$.

Fifth, if Φ' is not defined on $T(\Sigma)$ yet, then we take the minimal (according to $\prec$) term t for which $\Phi'(t)$ is not defined, and define it according to the following rules:

- if t is a constant symbol, then $\Phi'(t) = \Phi(t)$
- if $t = f(s)$, then $\Phi'(t) = \Phi(f(\Phi'(s)[1]))$
- if $t = f(s, s')$ then, $\Phi'(t) =$ $\Phi(f(\mathit{collapse}(\Phi'(s)[1]), \mathit{collapse}(\Phi'(s')[1])))$

Fourth and fifth steps are repeated until Φ' is defined on $T(\Sigma)$.

It is easy to see that for Φ' so defined, the sentence ψ is satisfied (Φ' is a run of $\mathcal{A}$), and that for each term belonging to a set defined by a projection expression, there exists a corresponding witness. Hence Φ' is a solution of (SC). $\square$

As a consequence of Lemma 3.16 and Lemma 3.19 we get the final result.

Theorem 3.20 The problem if a system of set constraints with projections has a solution is in NEXPTIME.

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 *Computer Science Logic'93*, LNCS 832, pages 1–17. Springer-Verlag, 1994.

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

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

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

[9] 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.

[10] 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.*

[11] 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.

[12] 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.

[13] 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.

[14] D. Kozen. Logical aspects of set constraints. In *Computer Science Logic'93*, volume 832 of *LNCS*, pages 175–188, 1994.

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

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

[17] 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.

[18] 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.