Inclusion Constraints over Non-empty Sets of Trees

Martin Müller[1], Joachim Niehren[1] and Andreas Podelski[2]

[1] Programming System Lab,
Universität des Saarlandes, 66041 Saarbrücken, Germany
{mmueller,niehren}@ps.uni-sb.de
[2] Max-Planck-Institut für Informatik,
Im Stadtwald, 66123 Saarbrücken, Germany
podelski@mpi-sb.mpg.de

Abstract. We present a new constraint system called INES. Its constraints are conjunctions of inclusions $t_1 \subseteq t_2$ between first-order terms (without set operators) which are interpreted over non-empty sets of trees. The existing systems of set constraints can express INES constraints only if they include negation. Their satisfiability problem is NEXPTIME-complete. We present an incremental algorithm that solves the satisfiability problem of INES constraints in cubic time. We intend to apply INES constraints for type analysis for a concurrent constraint programming language.

1 Introduction

We propose a new constraint system called INES (Inclusions over Non-Empty Sets) and present an incremental algorithm to decide the satisfiability of INES constraints in time $O(n^3)$. INES constraints are conjunctions of inclusions $t_1 \subseteq t_2$ between first-order terms (without set operators) which are interpreted over the domain of non-empty sets of trees. In this paper we focus on sets of possibly infinite trees. All given results can be easily adapted to finite trees.

An INES-constraint $t_1 \subseteq t_2$ is satisfiable over non-empty sets if and only if $t_1 \not\subseteq \emptyset \wedge t_1 \subseteq t_2$ is satisfiable over arbitrary sets. Note that the constraint $t \not\subseteq \emptyset$ cannot be expressed by positive set constraints only [16]. The expressiveness of INES constraints is subsumed by that of set constraints with negation [9, 16]. In the case of finite trees, the satisfiability problem of set constraints with negation is known to be decidable [1, 13]; it is complete for nondeterministic exponential time [9, 10]. This result implies that the satisfiability problem of INES constraints over sets of finite trees is decidable. The corresponding problem for infinite trees has not been considered before.

We characterize the satisfiability of INES constraints by a set of axioms such that an INES constraint is satisfiable over non-empty sets if and only if it is satisfiable in some model of these axioms. These axioms define a fixpoint algorithm that closes a given input constraint under its consequences with respect to the axioms.

We prove that a constraint φ is satisfiable if and only if the algorithm with input φ does not derive $\bot$ as a consequence of φ. All axioms (for infinite trees) will be discussed later in this introduction.

Sets versus Trees. The satisfiability problems of several classes of first-order formulae interpreted over trees and over non-empty sets of trees are closely related. The following two instances of this observation have inspired our choice of axioms or underly our proofs.

Equality constraints are conjunctions of equations $t_1{=}t_2$ between first-order terms. Over sets, they can be expressed by inclusion constraints due to antisymmetry of set inclusion ($t_1{=}t_2 \leftrightarrow t_1{\subseteq}t_2 \wedge t_2{\subseteq}t_1$). Actually, even the first-order theories of equality constraints over trees and of equality constraints over non-empty sets of trees coincide. This follows from the complete axiomatization of the first-order theory of equality constraints over trees [18, 19, 12] since its axioms also hold over non-empty sets of trees (but don't over possibly empty sets).

There exists a natural interpretation of INES-constraint over tree like structures that we call tree prefixes. In a different context [6] tree prefixes are called Böhm trees (without λ-binders). Tree prefixes come with a natural ordering relation where the empty tree prefix is the greatest element. We prove that an INES constraint is satisfiable over non-empty sets of trees if and only if it is satisfiable over tree prefixes (where the inclusion symbol is interpreted as the inverse of the prefix ordering on tree prefixes).

Axioms. The first two axioms we need postulate the reflexivity and transitivity of the inclusion relation. We also assume the following decomposition axiom (here formulated for a binary function symbol f).

$$f(x,y){\subseteq}f(x',y') \;\rightarrow\; x{\subseteq}x' \wedge y{\subseteq}y'$$

This axiom holds over non-empty sets of trees but not over possibly empty sets, since every variable assignment α with $\alpha(x) = \emptyset$ or $\alpha(y) = \emptyset$ is a solution of $f(x,y){\subseteq}f(x',y')$ but not necessarily of $x{\subseteq}x' \wedge y{\subseteq}y'$. An analogous statement holds for the following clash axiom.

$$f(x,y){\subseteq}g(x',y') \;\rightarrow\; \bot \qquad \text{for } f \neq g$$

These axioms do not suffice to characterize the satisfiability of INES constraints. For instance, the unsatisfiability of the constraint φ given by $x{\subseteq}g(x) \wedge x{\subseteq}g(y) \wedge y{\subseteq}z \wedge z{\subseteq}a$ is not derivable with these axioms alone. We need further axioms that use non-disjointness constraints $t_1 \not\parallel t_2$ defined as $t_1{\cap}t_2{\not\subseteq}\emptyset$. For the nondisjointness relation we require reflexivity and symmetry and a decomposition axiom as for the inclusion relation.

$$f(y,z) \not\parallel f(y',z') \rightarrow y \not\parallel y' \wedge z \not\parallel z'$$

Finally, we assume a clash axiom similar to the one for inclusion and require nondisjointness to be compatible with inclusion in the following sense.

$$x \not\parallel z \wedge x{\subseteq}y \rightarrow y \not\parallel z$$

Now reconsider the constraint φ given above and observe that we can derive $x \nparallel x$ by reflexivity, then $x \nparallel y$ by decomposition, and $x \nparallel z$ by compatibility. This yields a clash with $x \subseteq g(x) \wedge z \subseteq a$.

Algorithm and Complexity. The above axioms yield an algorithm that adds constraints of the form $x \subseteq y$, $x \nparallel y$ to a given input constraint φ until φ is closed under all axioms or implies $\perp$. The Ines constraint $x \subseteq t_1 \wedge \ldots \wedge x \subseteq t_n$ expresses the n sets denoted by the terms $t_1, \ldots, t_n$ have a non-empty intersection. Fortunately, it is not necessary to add k-ary non-disjointness constraints of the form $x_1 \cap \ldots \cap x_k \not\subseteq \emptyset$ (which can be expressed by the formula $\exists y\, (y \subseteq x_1 \wedge \ldots \wedge y \subseteq x_k))$ of which there are exponentially many. Instead, our algorithm adds at most $O(n^2)$ constraints to the input constraint φ, where n is the number of variables in φ. The addition of a single constraint can be implemented such that it costs time $O(n)$. This yields an implementation of our algorithm with time complexity $O(n^3)$. This implementation can be organized incrementally.

Type Analysis. One application for Ines constraints which we are investigating in [23] is type analysis for concurrent constraint programming [17, 28], in particular Oz [29]. As formal foundations we intend to use the calculi in [25, 26]. There, Ines constraints are used to approximate the set of run-time values for program variables. Since values in Oz include infinite trees, it is important that Ines allows an interpretation over sets of possibly infinite trees. It is considered an error if the set of possible run-time values is empty for some variable. This fact was our initial motivation for the choice of non-empty sets of trees as the interpretation domain for Ines constraints.

Plan of the Paper. In Section 2, we discuss relate work. In Section 3, we define the syntax and semantics of Ines constraints and in Section 4, we present the axioms and the algorithm. In Section 5, we prove the completeness of our algorithm. In Section 6, we compare the interpretations of Ines constraints over tree prefixes and over non-empty sets of trees. Due to space limitations, we omit the details of the proofs in the conference version of the paper.[1]

2 Related Work

Standard Set Constraints. Set constraints as in [2, 5, 10, 15] are inclusions between first-order terms with set operators interpreted over sets of finite trees. Our algorithm can be adapted such that it solves a subclass of set constraints

[1] The full version of this paper [24] contains several further appendixes. We give an example illustrating program analysis for Oz with Ines constraints. We detail the implementation of our the algorithm with incremental $O(n^3)$ complexity. We adapt the algorithm to the finite-tree case and to a subclass of standard set constraints (interpreted over possibly empty sets of finite trees) with explicit non-emptiness constraints $x \not\subseteq \emptyset$. We also prove that satisfiability of atomic set constraints (standard set constraints without set operators and negation) is invariant with respect to the choice of finite or infinite trees.

without set operators in cubic time (see [24]). The general case is nondeterministically exponential time complete as proved in [1, 13]. The subclass that we can solve in cubic time syntactically extends the INES constraints with explicit non-emptiness constraint $x \not\subseteq \emptyset$ (see [24]). Note that the satisfiability of these set constraints depends on the choice of finite or infinite trees (consider $x \subseteq f(x) \wedge x \not\subseteq \emptyset$), which is in contrast to standard set constraints without negation. Our algorithm accounts for finiteness through the occur check.

Atomic Set Constraints. Heintze and Jaffar consider so-called *atomic set constraints* [15] which syntactically coincide with INES constraints but are interpreted over possibly empty sets of finite trees. The satisfiability problem for atomic set constraints is also $O(n^3)$. This result is implicit in the combined results of [14] and [15]. An explicit proof is given in the full version of this paper [24].

Set Constraints for Type Analysis. Aiken *et al.* [3, 4] use constraints over specific sets of trees called "types" for the type analysis of FL. There is a minimal type $\mathsf{0}$ which – in terms of constraint solving – behaves just like the empty set in standard set constraints (although it is not an empty set from the types point of view but contains a value denoting non-termination). In contrast to the constraints of this paper, their set constraints provide for union and intersection. One of the optimizations used by Aiken *et al.* is to strengthen the following constraint simplification rule by dropping the disjuncts in brackets [4].

$$f(x, y) \subseteq f(x', y') \rightarrow x \subseteq x' \wedge y \subseteq y' \; [\; \vee x \subseteq \mathsf{0} \vee y \subseteq \mathsf{0} \;]$$

As stated in [4], this optimization does not preserve soundness ($f(a, \mathsf{0}) \subseteq f(b, \mathsf{0})$ holds but $a \subseteq b \wedge \mathsf{0} \subseteq \mathsf{0}$ does not). It might be possible to justify it by using non-empty sets as interpretation domain. This is left to further research.

Entailment and Independence for Ines Constraints. Charatonik and Podelski [11] give an algorithm which decides the entailment problem between INES constraints when interpreted over sets of finite trees. They also decide the satisfiability of INES constraints with negation in the finite tree case. The results in [11] do not include any of the results presented here since they use as an explicit prerequisite the fact that satisfiability of INES constraints is decidable.

Tarskian Set Constraints. MacAllester and Givan [21] give a cubic algorithm which decides satisfiability for a class of Tarskian set constraints [22], and which also contains a non-disjointness constraint. Apart from this syntactic similarity, the two satisfiability problems are rather different problems since Tarskian set constraints are not interpreted over the domain of trees (this is also observed in [22]). A related open question is whether our axioms define a local theory [20, 8], which would also proof the cubic complexity bound of our algorithm.

3 Syntax and Semantics of Ines Constraints

We assume a set of *variables* ranged over by x, y, z and a signature Σ that defines a set of *function symbols* f, g and their respective arity $n \geq 0$. *Constants* (i.e. function symbols of arity 0) are denoted with a and b.

Trees. We base the definition of trees on the notion of paths since we wish to include infinite trees. Paths will turn out central for our proofs in Section 5. A *path* p is a sequence of positive integers ranged over by i, j, n, m. The *empty path* is denoted by ε. We write the free-monoid concatenation of paths p and q as pq; we have $\varepsilon p = p\varepsilon = p$. Given paths p and q, q is called a *prefix of* p if $p = qp'$ for some path p'.

Let τ be a set of pairs $(p,\ f)$ of paths p and function symbols f. We say that τ is *prefix closed*, if $(p,\ f) \in \tau$ and q is a prefix of p implies that there is a g such that $(q,\ g) \in \tau$. It is *path consistent*, if $(p,\ f) \in \tau$ and $(p,\ g) \in \tau$ implies $f{=}g$. We call τ *arity consistent*, if $(p,\ f) \in \tau$, $(pi,\ g) \in \tau$ implies that $i \in \{1, \ldots, n\}$ provided the arity of f is n. Finally, τ is called *arity complete*, if $(p,\ f) \in \tau$, where the arity of f is n, implies for all $i \in \{1, \ldots, n\}$ the existence of a g with $(pi,\ g) \in \tau$.

A (possibly infinite) *tree* τ is a set of pairs $(p,\ f)$ that is non-empty, prefix closed, arity complete, path consistent, and arity consistent. The set of all (possibly infinite) trees over Σ is denoted by Tree and the set of all non-empty sets of trees by $\mathsf{P}^+(\text{Tree})$.

Ines Constraints. An Ines *constraint* $t_1{\subseteq}t'_1 \wedge \ldots \wedge t_n{\subseteq}t'_n$ is a conjunction of inclusions between first-order terms t defined by the following abstract syntax.

$$t \ ::= \quad x \ \mid \ f(\bar{t})$$

Here and throughout the paper, $\bar{t}$ stands for a sequence of terms and we assume implicitly that the length of $\bar{t}$ coincides with the arity of f. We interpret Ines constraints over the structure $\mathsf{P}^+(\text{Tree})$ of non-empty sets of trees. In this structure, a function symbol f of Σ is interpreted as elementwise tree constructor and the relation symbol $\subseteq$ as subset relation. We call a first-order formula over Ines constraint *satisfiable* if it is satisfiable in the structure $\mathsf{P}^+(\text{Tree})$. Two first-order formulae over Ines constraints are called *equivalent* if they are equivalently interpreted in $\mathsf{P}^+(\text{Tree})$.

Flat Ines Constraints. For algorithmic reasons, we use an alternative constraint syntax in the sequel. First, we restrict ourselves to flat terms $f(\overline{x})$ and x instead of possibly deep terms t. Second, we use equalities $x{=}f(\overline{y})$ rather than inclusions $x{\subseteq}f(\overline{y})$ and $f(\overline{y}){\subseteq}x$ (this is a matter of taste). And third, we need binary non-disjointness constraints $x {\not\parallel} y$. Their semantics is given by the equivalence to the formula $x{\cap}y{\not\subseteq}\emptyset$ over sets of trees. Over *non-empty* sets of trees, $x {\not\parallel} y$ is equivalent to $\exists z (z{\subseteq}x \wedge z{\subseteq}y)$. Crucially, however, nondisjointness constraints $x {\not\parallel} y$ avoid explicit existential quantification in our algorithm.

These three steps lead us to *flat* Ines *constraints* φ defined as follows.

$$\varphi \ ::= \quad \varphi_1 \wedge \varphi_2 \ \mid \ x{\subseteq}y \ \mid \ x{=}f(\overline{y}) \ \mid \ x {\not\parallel} y$$

We identify flat Ines constraints φ up to associativity and commutativity of conjunction, *i.e.*, we consider φ as a multiset of inclusions $x{\subseteq}y$, equalities $x{=}f(\overline{y})$, and non-disjointness constraints $x {\not\parallel} y$.

From now on, we will consider only flat INES constraints and call them *constraints* for short. This is justified by the following Proposition. Let the *size of a constraint* φ be the number of function symbol occurrences plus variable occurrences in φ.

Proposition 1. *The satisfiability problems of* INES *constraints and of flat* INES *constraints have the same time complexity up to a linear transformation.*

4 Axioms and Algorithm

We present a set of axioms valid for INES-constraints interpreted over non-empty sets of trees. In a second step, we interpret these axioms as an algorithm that solves the satisfiability problem of INES constraints. The correctness and the complexity of this algorithm will be proved in Section 5.

A1. $x \subseteq x$ and $x \subseteq y \wedge y \subseteq z \rightarrow x \subseteq z$

A2. $x = f(\overline{y}) \wedge x \subseteq x' \wedge x' = f(\overline{z}) \rightarrow \overline{y} \subseteq \overline{z}$

A3. $x \subseteq y \rightarrow x \not\parallel y$ and $x \subseteq y \wedge x \not\parallel z \rightarrow y \not\parallel z$ and $x \not\parallel y \rightarrow y \not\parallel x$

A4. $x = f(\overline{y}) \wedge x \not\parallel x' \wedge x' = g(\overline{z}) \rightarrow \perp$ for $f \neq g$

A5. $x = f(\overline{y}) \wedge x \not\parallel x' \wedge x' = f(\overline{z}) \rightarrow \overline{y} \not\parallel \overline{z}$

Table 1. Axioms of INES constraints over non-empty sets of infinite trees

Table 1 contains five rules A1-A5 representing sets of axioms.[2] The union of these sets is denoted by A. For instance, a rule $x \subseteq x$ represents the infinite set of axioms that is obtained by instantiation of the meta variable x with concrete variables. Note that an axiom is either a constraint φ, an implication between constraints $\varphi \rightarrow \psi$, or an implication $\varphi \rightarrow \perp$.

Proposition 2. *The structure* $\mathsf{P}^+(\text{Tree})$ *is a model of the axioms in* A.

Proof. By a routine check. We note that the non-emptiness assumption of $\mathsf{P}^+(\text{Tree})$ is essential for axioms A2 and A3.1. □

[2] Note that these axioms differ from the ones given in the introduction. The constraints used there are not flat and the variable-variable case $x \subseteq y$ and $x \not\parallel y$ are omitted. Indeed, the axioms in the introduction are semantically complete, although this is non-trivial to see and depends on the correctness of the algorithm presented here.

The Algorithm. The set of axioms A can be considered as a (naïve) fixed point algorithm A that, given an input constraint φ, iteratively adds logical consequences of $\mathsf{A}\cup\{\varphi\}$ to φ. More precisely, in every step A inputs a constraint φ and either terminates with $\perp$ or outputs a constraint $\varphi \wedge \psi$. Termination with $\perp$ takes place if there exists $\psi' \in \varphi$ such that $\psi' \rightarrow \perp \in \mathsf{A}$. Output of $\varphi \wedge \psi$ is possible if $\psi \in \mathsf{A}$ or there exists ψ' *in* φ with $\psi' \rightarrow \psi \in \mathsf{A}$.

Example 1. A first type of inconsistency depends on the transitivity of set inclusion. Here is a typical example:

$$x{=}a \wedge x{\subseteq}y \wedge y{\subseteq}z \wedge z{=}b \rightarrow \perp \qquad \text{for } a \neq b$$

Algorithm A may add $x{\subseteq}z$ by $\mathsf{A1.2}$, then $x{\not\parallel}z$ with $\mathsf{A3.1}$, and then terminate with $\perp$ by $\mathsf{A4}$.

Example 2. A second type of inconsistency comes with implicit or explicit non-disjointness requirements. For illustration, we consider:

$$x{=}a \wedge z{\subseteq}x \wedge z{\subseteq}y \wedge y{=}b \rightarrow \perp \text{ for} \qquad \text{for } a \neq b$$

Algorithm A may add $z{\not\parallel}x$ by $\mathsf{A3.1}$, then $x{\not\parallel}z$ via $\mathsf{A3.3}$, then $x{\not\parallel}y$ with $\mathsf{A3.2}$, and finally terminate with $\perp$ via $\mathsf{A4}$.

Example 3. Inconsistencies of the above two types may be detected by structural reasoning with $\mathsf{A2}$. Consider:

$$x{=}f(x) \wedge x{=}f(z) \wedge z{=}a \rightarrow \perp$$

Algorithm A may add $x{\subseteq}x$ by $\mathsf{A1.1}$, then $x{\subseteq}z$ with $\mathsf{A2}$, then $x{\not\parallel}z$ by $\mathsf{A3.1}$, and finally terminate with $\perp$ with $\mathsf{A4}$.

Example 4. We need another structural argument based on $\mathsf{A5}$ for deriving the unsatisfiability of the following constraint.

$$x{=}f(x) \wedge z{\subseteq}x \wedge z{\subseteq}y \wedge y{=}f(x') \wedge x'{=}a \rightarrow \perp$$

Algorithm A may add $x{\not\parallel}y$ after several steps as shown in Example 2. Then it may proceed with $x{\not\parallel}x'$ via $\mathsf{A5}$ and terminate with $\perp$ via $\mathsf{A4}$.

Termination. Algorithm A can be organized in a terminating manner by adding a simple *control*. Given an input constraint φ, we add only such constraints $x{\not\parallel}y$ and $x{\subseteq}y$ to φ which are not contained in φ. We also restrict reflexivity of inclusion $x{\subseteq}x$ to such variables x occurring in φ. Given a subset S of A, a constraint φ is called A'-*closed*, if algorithm A under the given control and restricted to the axioms in A' cannot proceed. (Note that constraints do not contain $\perp$ by definition.) This defines the notion of A-closedness but also of $\mathsf{A1}$-closedness, $\mathsf{A2}$-closedness, *etc.*, which will be needed later on.

Example 5. Our control takes care of termination in presence of cycles like $x{=}f(x)$. For instance, the following constraint is A-closed.

$$x{=}f(x) \wedge x{\subseteq}y \wedge y{=}f(x) \wedge x{\subseteq}x \wedge y{\subseteq}y \wedge x \not\parallel x \wedge y \not\parallel y \wedge x \not\parallel y \wedge y \not\parallel x$$

In particular, A2 and A5 do not loop through the cycle $x{=}f(x)$ infinitely often.

Proposition 3. *If φ is a constraint with m variables then algorithm* A *with input φ terminates under the above control in at most $2 \cdot m^2$ steps.* □

Proof. Since A does not introduce new variables, it may add at most m^2 non-disjointness constraints $x \not\parallel y$ and m^2 inclusions $x{\subseteq}y$. □

Proposition 4. *Every* A*-closed constraint φ is satisfiable over* $\mathsf{P}^+(\mathrm{Tree})$.

The proof of this statement is the subject of Section 5 and detailed in [24]. There, we construct the greatest solution for a satisfiable constraint (Lemma 9). Note that constraints in general do not have a smallest solution (consider $x{\subseteq}f(x\ y)$).

Theorem 5. *The satisfiability of* INES *constraints can be decided in time $O(n^3)$ (offline and online) where n is the constraint size.*

Proof. Proposition 2 shows that φ is unsatisfiable if A started with φ terminates with $\perp$. Proposition 4 proves that φ is satisfiable if A started with φ terminates with a constraint. Since A terminates for all input constraints under the above control (Proposition 3), this yields a effective decision procedure. The complexity statement is proved with Proposition 14 in [24]. The main idea is that every step of algorithm A can be implemented in time $O(n)$ and that there are $O(n^2)$ steps (Proposition 3). [3] In the proof of Proposition 14 [24], we present an incremental implementation of algorithm A. It exploits that algorithm A leaves the order unspecified in which axioms in A are applied. □

There is a class of constraints on which algorithm A indeed takes cubic time, namely the inclusions cycles $x_1{\subseteq}x_2 \wedge \ldots \wedge x_{n-1}{\subseteq}x_n \wedge x_n{\subseteq}x_1$ where $n \geq 1$. The closure under A is the full transitive closure $\bigwedge\{x_i{\subseteq}x_j \mid i, j \in \{1 \ldots n\}\}$ plus the corresponding non-disjointness constraints.

5 Completeness

The goal of this Section is to prove the completeness of our algorithm as stated in Proposition 4. We have to construct a solution for every A-closed constraint. The idea is to construct solution in a substructure of $\mathsf{P}^+(\mathrm{Tree})$ the structure of tree prefixes.

[3] Every step of algorithm A costs time $O(n)$ only with respect to an amortized time analysis, which we do not make explicit in our complexity proof in [24].

Tree Prefixes. A *tree prefix* τ is a set of pairs $(p,\ f)$ that is prefix closed, path consistent, and arity consistent. Note that every tree is a tree prefix. The set of all tree prefixes is denoted by Prefix. We can naturally interpret INES constraints over tree prefixes such that Prefix becomes a structure. Function symbols $f \in \Sigma$ are interpreted as tree prefix constructors (generalizing tree constructors). The inclusion symbol $\subseteq$ is interpreted as the *inverted* subset relation on tree prefixes that we denote with $\leq$ (*i.e.*, $\tau_1 \leq \tau_2$ iff $\tau_1 \supseteq \tau_2$). The relation $\tau_1 \nparallel \tau_2$ holds over Prefix iff $\tau_1 \cup \tau_2$ is path consistent (and hence a tree prefix).

Proposition 6. Prefix *is a substructure of* P^+(Tree) *with respect to the embedding Trees* : Prefix $\to \mathsf{P}^+$(Tree) *given by:*

$$\mathit{Trees}(\tau) = \{\tau' \mid \tau' \text{ is a tree such that } \tau' \leq \tau\}$$

Proof. The mapping *Trees* is a homomorphism with respect to function symbols $f \in \Sigma$ and the relation symbols $\subseteq$ and $\nparallel$. $\square$

Corollary 7. *If a constraint is satisfiable over* Prefix *then it is satisfiable over* P^+(Tree).

Proof. For constraints $x{\subseteq}y$, $x{=}f(\overline{y})$, and $x \nparallel y$, this follows from Proposition 6. A conjunction of such constraints is satisfiable if all conjuncts are satisfiable. $\square$

Path Reachability. We introduce the path reachability relations $\overset{\varphi}{\rightsquigarrow}_p$ and the notion of path consistency with respect to constraints. For all paths p and constraint φ, we define a binary relation $\overset{\varphi}{\rightsquigarrow}_p$, where $x \overset{\varphi}{\rightsquigarrow}_p y$ reads as "y is reachable from x over path p in φ":

$$\begin{array}{lll} x \overset{\varphi}{\rightsquigarrow}_\varepsilon y & \text{if} & x{\subseteq}y \ \mathit{in}\ \varphi \\ x \overset{\varphi}{\rightsquigarrow}_i y_i & \text{if} & x{=}f(y_1 \ldots y_i \ldots y_n) \ \mathit{in}\ \varphi, \\ x \overset{\varphi}{\rightsquigarrow}_{pq} y & \text{if} & x \overset{\varphi}{\rightsquigarrow}_p u \text{ and } u \overset{\varphi}{\rightsquigarrow}_q y. \end{array}$$

We define relations $x \overset{\varphi}{\rightsquigarrow}_p f$ meaning "f can be reached from x via path p in φ":

$$x \overset{\varphi}{\rightsquigarrow}_p f \ \text{ if } \ x \overset{\varphi}{\rightsquigarrow}_p y \text{ and } y{=}f(\bar{u}) \ \mathit{in}\ \varphi,$$

For example, if φ is the constraint $x{\subseteq}y \wedge y{=}f(u,z) \wedge z{=}g(x)$ then the following reachability from x relationships hold: $x \overset{\varphi}{\rightsquigarrow}_\varepsilon y$, $x \overset{\varphi}{\rightsquigarrow}_2 z$, $x \overset{\varphi}{\rightsquigarrow}_{21} x$, $x \overset{\varphi}{\rightsquigarrow}_{21} y$, etc., as well as $x \overset{\varphi}{\rightsquigarrow}_\varepsilon f$, $x \overset{\varphi}{\rightsquigarrow}_2 g$, $x \overset{\varphi}{\rightsquigarrow}_{21} f$, etc.

Definition 8 Path Consistency. We call a constraint φ *path consistent* if the following two conditions hold for all x, y, p, f, and g.

1. If $x \overset{\varphi}{\rightsquigarrow}_p g$, $x{\subseteq}x$, and $x \overset{\varphi}{\rightsquigarrow}_p f$ then $f = g$.
2. If $x \overset{\varphi}{\rightsquigarrow}_p g$, $x \nparallel y$, and $y \overset{\varphi}{\rightsquigarrow}_p f$ then $f = g$.

Lemma 9. *Every* A1-A2*-closed and path consistent constraint is satisfiable over* Prefix.

Lemma 10. *Every* A3-A5*-closed constraint is path consistent.*

Proof of Proposition 4. We have to show that every A-closed constraint φ is satisfiable. φ is path consistent by Lemma 10, satisfiable in Prefix by Lemma 9, and hence satisfiable in $\mathsf{P}^+(\mathrm{Tree})$ by Corollary 7. □

6 Non-Empty Sets versus Trees

We discuss interpretations of INES constraints over tree prefixes and over non-empty sets of trees. For the fragment of equality constraints we also consider an interpretation over trees.

Theorem 11. *Given an* INES *constraints φ, the following three statements are equivalent:*

1. *φ is satisfiable (over* $\mathsf{P}^+(\mathrm{Tree})$*).*
2. φ is satisfiable over Prefix.
3. φ is satisfiable in some model of the axioms in A.

Proof. 1) to 3). If φ is satisfiable over $\mathsf{P}^+(\mathrm{Tree})$, then it is satisfiable in some model of A, since $\mathsf{P}^+(\mathrm{Tree})$ is a model of A by Proposition 2.

3) to 2). Let φ be satisfiable in some model of A. Algorithm A terminates when started with φ by Proposition 3. It outputs a constraint ψ (and not $\perp$) that is equivalent to φ in all models of A. ψ is A-closed and hence satisfiable over Prefix by Lemmata 9 and 10.

2) to 1). If φ is satisfiable over Prefix then it is satisfiable by Corollary 7. □

An *equality constraint* is a conjunction of equalities $x{=}y$ and $x{=}f(\overline{y})$. Over $\mathsf{P}^+(\mathrm{Tree})$, equalities can be expressed by inclusions since the inclusion ordering is antisymmetric ($x{=}y \;\leftrightarrow\; x{\subseteq}y \wedge y{\subseteq}x$).

Theorem 12. *The three first-order theories of equality constraints over non-empty sets of trees, over tree prefixes, and over trees coincide (i.e., of the structures* $\mathsf{P}^+(\mathrm{Tree})$*,* Prefix *and* Tree*).*[4]

Proof. This follows from the fact that all axioms of the complete axiomatization of trees [18, 19, 12] are valid for non-empty sets of trees. This holds for the axioms of the form $\forall\overline{y}\exists!\overline{x}(x_1{=}f_1(\overline{x}\;\overline{y}) \wedge \ldots \wedge x_n{=}f_n(\overline{x}\;\overline{y}))$. Validity of the other axioms is immediate since they are already contained in A with inclusion replaced for equality. □

[4] Independently, A. Colmerauer observed this for $\mathsf{P}^+(\mathrm{Tree})$ and Tree (pers. comm.).

In contrast, first-order formulae over inclusion constraints can distinguish the structures $\mathsf{P}^+(\text{Tree})$ and Prefix. A formula that holds over Prefix but not over $\mathsf{P}^+(\text{Tree})$ is given by

$$\forall x (a \subseteq x \wedge b \subseteq x \rightarrow \forall y (y \subseteq x))$$

where $a \neq b$. Another formula distinguishing both structures comes with a constraint-based reformulation of the coherence property (defined for complete partial orders in [6]).

We say that an ordering relation satisfies the *coherence property* if it satisfies the following formulae for all finite sets I (where inclusion symbol is interpreted as the given ordering).

$$\bigwedge_{i,j \in I} \exists z (z \subseteq x_i \wedge z \subseteq x_j) \rightarrow \exists z (\bigwedge_{i \in I} z \subseteq x_i)$$

This formula states that for all variable assignment α the elements from the finite set $\{\alpha(x_i) \mid i \in I\}$ have a common lower bound if every two of its elements $\alpha(x_i), \alpha(x_j)$ have $(i, j, \in \{1, \ldots, n\})$. For inclusion over non-empty sets this property does not hold. There it states the non-emptiness of an n-intersection $t_1 \cap \ldots \cap t_n$ if all pairwise intersections $t_i \cap t_j$ are non-empty $(i, j \in \{1 \ldots n\})$, which is refuted by the example $I = \{1, 2, 3\}$ and $\alpha(x_1) = \{a, b\}$, $\alpha(x_2) = \{a, c\}$, $\alpha(x_3) = \{b, c\}$ for distinct constants a, b, c.

Proposition 13. *The tree prefix ordering $\leq$ satisfies the coherence property.*

Proof. For some finite index set $J \subseteq I$ and variable assignment α into Prefix, note that α is a solution of $\exists z (\bigwedge_{i \in J} z \subseteq x_i)$ iff $\bigcup_{i \in J} \alpha(x_i)$ is path consistent. If α is a solution of all $\exists z (z \subseteq x_i \wedge z \subseteq x_j)$ then all pairwise unions $\alpha(x_i) \cup \alpha(x_j)$ are path consistent such that the union $\bigcup_{i \in I} \alpha(x_i)$ is path consistent. Hence α is a solution of $\exists z (\bigwedge_{i \in I} z \subseteq x_i)$. □

Acknowledgements. We would like to thank David Basin, Denys Duchier, Witold Charatonik, Harald Ganzinger, Gert Smolka, Ralf Treinen and Uwe Waldmann, as well as the anonymous referees for valuable comments on drafts of this paper. The research reported in this paper has been supported by the the Esprit Working Group CCL II (EP 22457) and the Deutsche Forschungsgemeinschaft through the Graduiertenkolleg Kognitionswissenschaft and the SFB 378 at the Universität des Saarlandes.

References

1. A. Aiken, D. Kozen, and E. Wimmers. Decidability of Systems of Set Constraints with Negative Constraints. *Information and Computation*, 1995.
2. A. Aiken and E. Wimmers. Solving Systems of Set Constraints. In *Proc. 7th LICS*, pp. 329–340. IEEE, 1992.
3. A. Aiken and E. Wimmers. Type Inclusion Constraints and Type Inference. In *Proc. 6th FPCA*, pp. 31–41. 1993.
4. A. Aiken, E. Wimmers, and T. Lakshman. Soft Typing with Conditional Types. In *Proc. 21st POPL*. ACM, 1994.

5. L. Bachmair, H. Ganzinger, and U. Waldmann. Set Constraints are the Monadic Class. In *Proc. 8^{th} LICS*, pp. 75–83. IEEE, 1993.
6. H. P. Barendregt. *The Type Free Lambda Calculus*. In Barwise [7], 1977.
7. J. Barwise, ed. *Handbook of Mathematical Logic*. Number 90 in Studies in Logic. North–Holland, 1977.
8. D. Basin and H. Ganzinger. Automated Complexity Analysis Based on Ordered Resolution. In *11^{th} LICS*. IEEE, 1996.
9. W. Charatonik and L. Pacholski. Negative set constraints with equality. In *Proc. 9^{th} LICS*, pp. 128–136. 1994.
10. W. Charatonik and L. Pacholski. Set constraints with projections are in NEXP-TIME. In *Proc.* 35^{th} *FOCS*, pp. 642–653. 1994.
11. W. Charatonik and A. Podelski. The Independence Property of a Class of Set Constraints. In *Proc. 2nd CP*. LNCS 1118, Springer, 1996.
12. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7:371–425. 1989.
13. R. Gilleron, S. Tison, and M. Tommasi. Solving Systems of Set Constraints with Negated Subset Relationships. In *Proc. 34^{nd} FOCS*, pp. 372–380. 1993.
14. N. Heintze. Set Based Analysis of ML Programs. Technical Report CMU–CS–93–193, School of Computer Science, Carnegie Mellon University. July 1993.
15. N. Heintze and J. Jaffar. A Decision Procedure for a Class of Set Constraints (Extended Abstract). In *Proc. 5^{th} LICS*, pp. 42–51. IEEE, 1990.
16. D. Kozen. Logical aspects of set constraints. In *Proc. CSL*, pp. 175–188. 1993.
17. M. J. Maher. Logic semantics for a class of committed-choice programs. In J.-L. Lassez, ed., *Proc. 4^{th} ICLP*, pp. 858–876. The MIT Press, 1987.
18. M. J. Maher. Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees. In *Proc. 3^{rd} LICS*, pp. 348–457. IEEE, 1988.
19. A. I. Malc'ev. Axiomatizable Classes of Locally Free Algebras of Various Type. In *The Metamathematics of Algebraic Systens: Collected Papers 1936-1967*, ch. 23, pp. 262–281. North–Holland, 1971.
20. D. McAllester. Automatic Recognition of Tractability in Inference Relations. *Journal of the ACM*, 40(2), Apr. 1993.
21. D. McAllester and R. Givan. Taxonomic Syntax for First-Order Inference. *Journal of the ACM*, 40(2), Apr. 1993.
22. D. McAllester, R. Givan, D. Kozen, and C. Witty. Tarskian Set Constraints. In *Proc. 11^{th} LICS*. IEEE, 1996.
23. M. Müller. *Type Analysis for a Higher-Order Concurrent Constraint Language*. Doctoral Dissertation. Universität des Saarlandes, Technische Fakultät, 66041 Saarbrücken, Germany. In preparation.
24. M. Müller, J. Niehren, and A. Podelski. Inclusion Constraints over Non-Empty Sets of Trees. Full version: `http://www.ps.uni-sb.de/Papers/ines97.html`.
25. J. Niehren. Functional Computation as Concurrent Computation. In *23^{rd} POPL*, pp. 333–343. ACM, 1996.
26. J. Niehren and M. Müller. Constraints for Free in Concurrent Computation. In *Proc. 1^{st} ASIAN*, LNCS 1023, pp. 171–186. Springer, 1995.
27. *The Oz Programming System*. Programming Systems Lab, Universität des Saarlandes. Available at `http://www.ps.uni-sb.de/www/oz/`.
28. V. A. Saraswat. *Concurrent Constraint Programming*. The MIT Press, 1993.
29. G. Smolka. The Oz Programming Model. In J. van Leeuwen, ed., *Computer Science Today*, LNCS 1000, pp. 324–343. Springer, 1995.