Entailment of Atomic Set Constraints is PSPACE-Complete

Joachim Niehren, Martin Müller, Jean-Marc Talbot

Entailment of Atomic Set Constraints is PSPACE-Complete

Joachim Niehren and Martin Müller
Universität des Saarlandes
Saarbrücken, Germany

Jean-Marc Talbot
Max-Planck Institut Informatik
Saarbrücken, Germany

Abstract

The complexity of set constraints has been extensively studied over the last years and was often found quite high. At the lower end of expressiveness, there are atomic set constraints which are conjunctions of inclusions $t_1 \subseteq t_2$ between first-order terms without set operators. It is well-known that satisfiability of atomic set constraints can be tested in cubic time. Also, entailment of atomic set constraints has been claimed decidable in polynomial time. We refute this claim. We show that entailment between atomic set constraints can express validity of quantified boolean formulas and is thus PSPACE hard. For infinite signatures, we also present a PSPACE-algorithm for solving atomic set constraints with negation. This proves that entailment of atomic set constraints is PSPACE-complete for infinite signatures. In case of finite signatures, this problem is even DEXPTIME-hard.

1 Introduction

Set constraints are logical formulas describing relations between sets of trees [2, 5, 6, 13, 16]. Set constraints have received much attention in constraint-based type inference and program analysis for different programming languages [3, 12, 15, 17, 21, 28]. Other applications of set constraints include order-sorted unification [29] and constraint logic programming [20].

Expressiveness and Complexity. Expressiveness and complexity have been widely studied for various classes of set constraint [1, 2, 7, 9, 11, 14, 27]. The complexity of their satisfiability problem was often found to be very high (*e.g.*, NEXPTIME-complete [1, 27] and DEXPTIME-complete [9, 11]). At the lower end of the expressiveness scale, there are atomic set constraints [16] which are conjunctions of inclusions $t_1 \subseteq t_2$ between first-order terms t_1, t_2 without set operators, i.e. terms built from variables x and function symbols f of a given signature Σ. Atomic set constraints are interpreted in the structure of sets of finite trees over Σ (often called ground terms). It is well-known that the satisfiability of an atomic set constraint can be tested in cubic time (see the complete version of [22] for instance).

Entailment. Beyond satisfiability, entailment (the validity of implications $\varphi \models \varphi'$) has raised much interest for various classes of constraints [4, 18, 26]. Entailment is useful for constraint simplification [24], closely related to the treatment of negation (see below), and fundamental for models of concurrent constraint programming [25].

Entailment of atomic set constraints is subsumed by satisfiability of atomic set constraints with negation which is known decidable in NEXPTIME [6, 14]. The precise complexity of entailment of set constraints was first investigated by Charatonik and Podelski. They showed in [9] that entailment of set constraints with intersection (which subsume atomic set constraints) is DEXPTIME-complete for an infinite signature. Beside this, they noted in the same paper with reference to [8] that entailment of atomic set constraints is decidable in polynomial time – again for an infinite signature. We refute this claim and determine the correct complexity.

The reductions in this paper that prove lower complexity bounds are inspired by work of Rehof and Henglein's on entailment of subtype constraints [18]. Following their idea, one can indeed express satisfiability of Boolean formulas in conjunctive normal form by entailment of atomic set constraints (for any non-trivial signature). Since this problem is coNP-complete, entailment of atomic set constraints is coNP-hard – in contrast to the claim of Charatonik and Podelski.

For a finite signature Σ, the situation is even worse: Entailment of atomic set constraints is DEXPTIME-hard since it can express universality of tree automata which is DEXPTIME-complete. The reduction is very simple. Its idea is to consider the set of transition rules of a tree automaton as an atomic set constraint. For instance, the transitions of the automaton $\mathcal{A}$ below corresponds to the atomic set constraint $\varphi_{\mathcal{A}}$:

$$\begin{array}{rl} \mathcal{A}: & f(x,z) \to x,\ \ f(z,x) \to x,\ \ f(z,x) \to z \\ \varphi_{\mathcal{A}}: & f(x,z) \subseteq x \wedge f(z,x) \subseteq x \wedge f(z,x) \subseteq z \end{array}$$

If x is the only final state of $\mathcal{A}$ then $\mathcal{A}$ is universal if f $\varphi_{\mathcal{A}} \wedge \mathcal{T}_\Sigma \subseteq y \models y \subseteq x$ holds in the structure of sets of finite trees over Σ. The formula $\mathcal{T}_\Sigma \subseteq y$ can be expressed by $\bigwedge_{f\in\Sigma} f(y,\ldots,y) \subseteq y$ which is an atomic set constraint but only if Σ is finite.

For an infinite signature, we have argued so far that the complexity of entailment of atomic set constraints is between coNP and DEXPTIME. We show in this paper that this problem is indeed PSPACE-complete. We prove PSPACE-hardness by expressing validity of quantified Boolean formula which is PSPACE complete. The idea is to extend the coNP-hardness proof (which follows [18]) by encoding quantifier prefixes in addition. Note that the PSPACE hardness result of [19] does not carry over to atomic set constraints.

Independence and Negation. Entailment can be used for treating negation since $\psi \models \bigvee_{i=1}^n \psi_i$ holds if and only if $\psi \wedge \bigwedge_{i=1}^n \neg\psi_i$ is unsatisfiable. A constraint language has the independence property [10] if the unsatisfiability of $\psi \wedge \bigwedge_{i=1}^n \neg\psi_i$ is equivalent to that $\psi \models \psi_j$ holds for some $1 \leq j \leq n$. Thus, under the assumption of independence, deciding entailment is equivalent to solving conjunctions of positive and negative constraints.

The independence property does not hold for atomic set constraints since a variable may denote the empty set $\emptyset$. For instance, let ψ be the atomic set constraint $x \subseteq a \wedge y \subseteq a$ which requires that x and y denote the empty set $\emptyset$ or the singleton $\{a\}$. Hence, $\psi \models x \subseteq y \vee y \subseteq x$ is valid but neither $\psi \models x \subseteq y$ nor $\psi \models y \subseteq x$ hold.

As observed by Charatonik and Podelski [8, 9][1], the independence property does hold for Ines constraint [22] – in case of an infinite signature. Ines constraints have the same syntax as atomic set constraints but are interpreted over *non-empty sets* of trees (rather than arbitrary sets).

In this paper, we prove that the following 4 problems are PSPACE-complete for an infinite signature:

1. Entailment of Ines constraints.
2. Satisfiability of Ines constraints with negation.
3. Satisfiability of atomic set constraints with negation.
4. Entailment of atomic set constraints.

Problems (1) and (2) are equivalent since Ines constraints have the independence property. Problem (3) can be reduced to problem (2) in NP (and thus in PSPACE) since we can guess for all variable x whether $x \subseteq \emptyset$ or $x \not\subseteq \emptyset$. Obviously, problem (4) can be reduced to (3). All together, we see that problem (4) is easier than (1) modulo PSPACE reductions.

The easiest problem (4) is PSPACE hard as we argued above. In order to show PSPACE completeness for all problems it remains to show that the hardest problem (1) can be solved in PSPACE.

Deciding Entailment. In this paper, we present an PSPACE algorithm that decides entailment of Ines constraint for an infinite signature. Given a judgment $\psi \models x \subseteq y$, this algorithm checks the existence of a term that is both an upper bound of x in ψ and a lower bound of y in ψ. We illustrate reasoning with lower and upper bounds for proving the validity of the following judgment:

$$\psi_1 \wedge \psi_2 \wedge \psi_3 \models x \subseteq y$$

in the structure of non-empty sets of trees, where:

$$\begin{array}{rcl} \psi_1 & = & x \subseteq f(x_1,x_1) \wedge x_1 \subseteq f(a,z) \wedge x_1 \subseteq z \\ \psi_2 & = & x \subseteq f(z,f(z,a)) \\ \psi_3 & = & f(y_1,z) \subseteq y \wedge f(z,y_1) \subseteq y \wedge f(a,a) \subseteq y_1 \end{array}$$

First note that $\psi_1 \models x \subseteq f(z,f(a,z))$ holds, i.e. the term $f(z,f(a,z))$ is an upper bound of x in ψ_1. The reader may notice that there are many more upper bounds for x in ψ_1, for instance $f(x_1,z)$, $f(z,x_1)$,

[1] The independence property for Ines constraints (even with intersections) is proved in [9]. The earlier proof given in [8] is based on an incomplete algorithm for entailment and thus wrong. A counter example can be found in the paper.

$f(z,z)$, $f(f(a,z),x_1)$, $f(f(a,z),z)$. The number of upper bounds may grow exponentially in the size of the constraint. In particular, multiple upper bounds can be combined by "deep shuffle"; as for instance:

$$\begin{array}{rl} & x{\subseteq}f(z,f(a,z)) \\ \wedge & x{\subseteq}f(z,f(z,a)) \end{array} \models x{\subseteq}f(z,f(a,a))$$

This shows that $f(z,f(a,a))$ is an upper bound of x in $\psi_1 \wedge \psi_2$. Furthermore, $\psi_3 \models f(z,f(a,a)){\subseteq}y$ holds, i.e. $f(z,f(a,a))$ is a lower bound of y in ψ_3. Thus, with respect to $\psi_1 \wedge \psi_2 \wedge \psi_3$ there exists a term that is both an upper bound of x and a lower bound of y; this proves the validity of $\psi_1 \wedge \psi_2 \wedge \psi_3 \models x{\subseteq}y$. Our example illustrates an incompleteness of the algorithm from [8] which does not take "deep shuffle" into account.

Plan of the Paper. In Sections 2 and 3 we start with preliminaries and recall results on Ines constraint. In Section 4 we prove the PSPACE hardness for entailment of atomic set constraints. In Sections 5 and 6 we present a PSPACE algorithm for entailment of Ines constraints. Section 7 discusses the independence property of Ines constraints. For lack of space, many proofs are omitted. They can be found in [23].

2 Preliminaries

We assume a set $\mathcal{V}$ of *variables* ranged over by x, y, z and a signature Σ that defines an infinite 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, b. We assume that Σ contains at least one constant and one function symbol of arity ≥ 2.

A *first-order term* t is a variable x or an expression $f(t_1,\ldots,t_n)$ where n is the arity of f. A *ground term* (a *finite tree*) τ is a first-order term without variables. The set of all finite trees over Σ is denoted by $\mathcal{T}_\Sigma$. Given a set S, the powerset of S is denoted by $\mathcal{P}(S)$ and the set of all non-empty subsets of S by $\mathcal{P}^+(S)$. We freely consider the sets $\mathcal{P}(\mathcal{T}_\Sigma)$ and $\mathcal{P}^+(\mathcal{T}_\Sigma)$ as model theoretic structures where a function symbol f of Σ is interpreted as an element-wise tree constructor, *i.e.,* for some sets of finite trees $\sigma_1,\ldots,\sigma_n$, $f(\sigma_1,\ldots,\sigma_n) = \{f(\tau_1,\ldots,\tau_n) \mid \tau_i \in \sigma_i \text{ for all } 1 \leq i \leq n\}$, and the relation symbol $\subseteq$ as the subset relation.

A *path* π is a word of natural numbers. The *empty path* is denoted by ε and the free-monoid concatenation of paths π and π' as $\pi\pi'$. We have $\varepsilon\pi = \pi\varepsilon = \pi$. A path π' is called a *prefix of* π if $\pi = \pi'\pi''$ for some path π''. A *tree domain* is a non-empty prefix closed set of paths. We define the arity of a variable to be 0. A first-order term t can be characterized by a pair (D_t, L_t) consisting of a tree domain D_t and a (total) labeling function $L_t : D_t \to \Sigma \cup \mathcal{V}$ such that for all $\pi \in D_t$ it holds that $\pi i \in D_t$ iff $1 \leq i \leq n$, where n is the arity of $L_t(\pi)$.

Logical Notation. We consider several first-order languages L over our signature Σ of function symbols extended with some relation symbols. For a formula $\Phi \in L$, we denote the set of *free variables* in Φ with $\mathcal{V}(\Phi)$ and the set of function symbols in Φ with $\Sigma(\Phi)$. Given a model theoretic structure $\mathcal{A}$ over the signature Σ of L, a *solution* of $\Phi \in L$ over $\mathcal{A}$ is a variable assignment $\alpha : \mathcal{V} \to \mathcal{A}$ which renders $\alpha(\Phi)$ true. A formula $\Phi \in L$ is *satisfiable* over $\mathcal{A}$ if there exists a solution of Φ over $\mathcal{A}$; it is *valid* over $\mathcal{A}$ if all variable assignments $\alpha : \mathcal{V} \to \mathcal{A}$ are solutions of Φ. We say that Φ entails Φ' if the implication $\Phi \to \Phi'$ is valid over $\mathcal{A}$, and write $\Phi \models_{\mathcal{A}} \Phi'$ in this case, or simply $\Phi \models \Phi'$ if the structure of interest is fixed by the context. The *satisfiability problem of L relative to some structure $\mathcal{A}$* is the problem whether a formula $\Phi \in L$ is satisfiable over $\mathcal{A}$. The *entailment problem of L relative to $\mathcal{A}$* is the problem whether an entailment judgement $\Phi \models_{\mathcal{A}} \Phi'$ holds for two given formulas $\Phi, \Phi' \in L$. The *satisfiability problem of L with negation* relative to $\mathcal{A}$ is whether a conjunction $\bigwedge_{i=1}^n \Phi_i \wedge \bigwedge_{j=1}^m \neg\Phi'_j$ of positive and negative formula in L is satisfiable over $\mathcal{A}$. It is well known that the entailment problem of L over $\mathcal{A}$ is less general than its satisfiability problem with negation, since $\Phi \models_{\mathcal{A}} \Phi'$ if and only if $\Phi \wedge \neg\Phi'$ is unsatisfiable over $\mathcal{A}$.

3 Ines and Atomic Set Constraints

An *inclusion constraint* ψ is a conjunction of inclusions between first-order terms:

$$\psi \quad ::= \quad t_1{\subseteq}t_2 \;\mid\; \psi \wedge \psi'$$

The *size* of an inclusion constraint is the number of its symbols (variables and function symbols).

An *atomic set constraint* [16] is an inclusion constraint interpreted in the structure of sets of finite trees $\mathcal{P}(\mathcal{T}_\Sigma)$.

An *Ines constraint* [22] is an inclusion constraint interpreted in the structure $\mathcal{P}^+(\mathcal{T}_\Sigma)$ of non-empty sets of finite trees. In this paper, we do not consider the case of infinite trees.

In the formal parts of this paper, we use a flat syntax for inclusion constraints which restricts the nesting of terms. A *flat inclusion constraint* φ is defined by the following abstract syntax where n is the arity of f:

$$\varphi \quad ::= \quad x{=}f(y_1,\ldots,y_n) \;\mid\; x{\subseteq}y \;\mid\; \varphi \wedge \varphi'$$

Proposition 3.1 *For both models $\mathcal{P}^+(\mathcal{T}_\Sigma)$ and $\mathcal{P}(\mathcal{T}_\Sigma)$ it is linearly equivalent to decide judgements of the form $\Psi \models \Psi'$ or $\varphi \models x{\subseteq}y$.*

Satisfiability. We now recall a result on satisfiability of Ines constraints given in [22]. The analogue result holds for satisfiability of atomic set constraints but this is not needed for the purpose of this paper.

Proposition 3.2 *The satisfiability problem of Ines constraint can be decided in cubic time.*

Without loss of generality we can assume that a flat constraint φ is closed under reflexivity, transitivity, and decomposition in that it satisfies the properties B1-B3 below (we write $\overline{x}$ and $\overline{y}$ for sequences of variables and $\overline{x}{\subseteq}\overline{y}$ for a conjunction of inclusions):

B1 $\quad x{\subseteq}x \in \varphi$ if $x \in \mathcal{V}(\varphi)$

B2 $\quad x{\subseteq}z \in \varphi$ if $x{\subseteq}y \wedge y{\subseteq}z \in \varphi$

B3 $\quad \overline{x}{\subseteq}\overline{y} \in \varphi$ if $z{=}f(\overline{x}) \wedge z{\subseteq}z' \wedge z'{=}f(\overline{y}) \in \varphi$

From a B1-B3 closed constraint one can read of its consequences more easily by syntactic reasoning.

Path Constraints. Given a tree τ and a path π we write as $\tau[\pi]$ the subtree of τ at path π if it exists. Given a set σ of trees and a path π, we define $\sigma[\pi]$ as the π-projection of σ:

$$\sigma[\pi] = \{\tau[\pi] \mid \tau \in \sigma \text{ and } \tau[\pi] \text{ exists}\}$$

Note that $\sigma[\pi]$ is always defined but possibly empty. We need a new class of formulas that we call *path constraints*. These are of the form $x[\pi]{\subseteq}y$ and $x[\pi]{\subseteq}f$ for paths π, variables x, y and constructors f. A variable assignment β solves $x[\pi]{\subseteq}y$ iff $\beta(x)[\pi] \subseteq \beta(y)$. It solves $x[\pi]{\subseteq}f$ if $\beta(x)[\pi] \subseteq f(\mathcal{T}_\Sigma,\ldots,\mathcal{T}_\Sigma)$.

We also need a notion of *syntactic support* for path constraints. We define $\varphi \vdash x[\pi]{\subseteq}y$ and $\varphi \vdash x[\pi]{\subseteq}f$ as follows.

$\varphi \vdash x[\varepsilon]{\subseteq}y$	if $x{\subseteq}y \in \varphi$
$\varphi \vdash x[k]{\subseteq}y_k$	if $x = f(y_1,\ldots,y_k,\ldots,y_n) \in \varphi$
$\varphi \vdash x[\pi\pi']{\subseteq}y$	if exists $z : \varphi \vdash x[\pi]{\subseteq}z$ and $\varphi \vdash z[\pi']{\subseteq}y$
$\varphi \vdash x[\pi]{\subseteq}f$	if exist $y_1,\ldots,y_n,z : \varphi \vdash x[\pi]{\subseteq}z$ and $z = f(y_1,\ldots,y_n) \in \varphi$

This definition of syntactic support is correct for both structures $\mathcal{P}(\mathcal{T}_\Sigma)$ and $\mathcal{P}^+(\mathcal{T}_\Sigma)$. If $\varphi \vdash x[\pi]{\subseteq}y$ then $\varphi \models x[\pi]{\subseteq}y$ and if $\varphi \vdash x[\pi]{\subseteq}f$ then $\varphi \models x[\pi] \subseteq f$.

4 Entailment is PSPACE-hard

We now show that the entailment of atomic set constraints is PSPACE hard since it can express validity of quantified Boolean formulas.

We denote the Booleans with F and T and assume an infinite set of Boolean variables ranged over by u. A *literal* is either a variable u or a negated variable $\neg u$. A *CB-formula* C is a finite set of literals that we write as a conjunction. The empty conjunctive formula is denoted by true. A *DB-formula* D is a finite set of conjunctive formulas that we write as a disjunction $C_1 \vee \ldots \vee C_m$. The empty DB-formula is denoted by false. Let $u_1,\ldots,u_n$ be a sequence of pairwise distinct variables and for all $1 \leq i \leq n$ let Q_i be one of the quantifiers $\exists u_i$ or $\forall u_i$. A *QB-formula* is a closed formula of the following form with $m, n \geq 0$:

$$Q_1 \ldots Q_n \; (C_1 \vee \ldots \vee C_m)$$

We call m, n and the sequence $u_1,\ldots,u_n$ the *parameters* of the above QB-formula. We denote a quantifier prefixes like $Q_1 \ldots Q_n$ with P.

Definition 4.1 QBF *is the validity problem of QB-formulas in the structure of Booleans.*

We call a CB-formula C *normalized* if there is no variable u such that $u \in C$ and $\neg u \in C$. Trivially, a normalized CB-formula is satisfiable. We call a QB-formula $P\ C_1 \vee \ldots \vee C_m$ *normalized* if $m \geq 1$ and

if all C_i's are normalized. Note that a normalized DB-formula is satisfiable since it contains at least one CB-formula and all its CB-formulas are satisfiable.

The idea for encoding Boolean formulas stems from [18]. It is based on the fact that a binary tree of depth n allows to represent the set of solutions of a Boolean formula with n variables, say $u_1, \ldots, u_n$. We identify the Booleans F and T with the integers 1 and 2 respectively. We identify a path $\pi = b_1 \ldots b_n \in \{1,2\}^n$ with a variable assignment into Booleans $\pi : \{u_1, \ldots, u_n\} \to \{1,2\}$ such that $\pi(u_i) = b_i$ for all $1 \leq i \leq n$. Now we can represent every subset of $\{1,2\}^n$ by a binary tree of depth n whose tree domain contains this subset. For $n = 2$ for instance:

$$\{11, 21, 12\} \qquad \Rightarrow \qquad \begin{array}{ccccc} & & 1 & & 2 \\ & 1 & & 1 & \\ & & & & 2 \end{array}$$

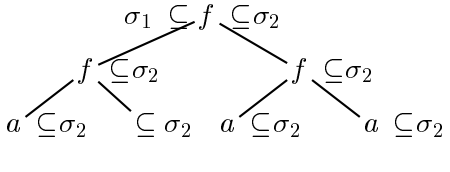
Rather than considering a set of paths as a tree, we represent it by a set of trees. We fix a binary symbol $f \in \Sigma$ and a constant $a \in \Sigma$. For all $\pi \in \{1,2\}^n$ and $\sigma \in \mathcal{P}(\mathcal{T}_\Sigma)$ we define a set of trees $s^\pi_\sigma \in \mathcal{P}(\mathcal{T}_\Sigma)$:

$$s^\pi_\sigma = \begin{cases} a \cap \sigma & \text{if } \pi = \epsilon \\ f(s^{\pi'}_\sigma, \sigma) \cap \sigma & \text{if } \pi = 1\pi' \\ f(\sigma, s^{\pi'}_\sigma) \cap \sigma & \text{if } \pi = 2\pi' \end{cases}$$

The set for $\pi = 21$ and some σ can be depicted by:

$$s^{21}_\sigma = \quad \begin{array}{ccccc} & & f \cap \sigma & & \\ & \sigma & & f \cap \sigma & \\ & & a \cap \sigma & & \sigma \end{array}$$

A set of paths $\Pi \subseteq \{1,2\}^n$ can be represented by a binary predicate $\mathcal{U}_\Pi \subseteq \mathcal{P}_{\mathcal{T}_\Sigma} \times \mathcal{P}_{\mathcal{T}_\Sigma}$ between sets of trees. For all sets $\Pi \subseteq \{1,2\}^n$ let $s^\Pi_\sigma = \cap_{\pi \in \Pi} s^\pi_\sigma$. The predicate $\mathcal{U}_\Pi(\sigma_1, \sigma_2)$ holds iff $\sigma_1 \subseteq s^\Pi_{\sigma_2}$. For instance, $\mathcal{U}_{\{11,21,12\}}(\sigma_1, \sigma_2)$ holds in the following situation:



We encode a BF-formula D by expressing the predicate $\mathcal{U}_{\{\pi | \pi \models D\}}$ with atomic set constraints. This is done by the formula $\mathsf{DBF}_{x,z}(D)$ in Figure 1 which satisfies for all x, z:

$$\beta \models \mathsf{DBF}_{x,z}(D) \quad \text{iff} \quad \mathcal{U}_{\{\pi | \pi \models D\}}(\beta(x), \beta(z))$$

For encoding a quantifier prefix in a QBF-formula, we express another predicate which concerns lower bounds. Given $i \geq 0$ and sets of paths $\Pi_1, \ldots, \Pi_i \subseteq \{1,2\}^n$ we define $\mathcal{L}_{\{\Pi_1,\ldots,\Pi_i\}} \subseteq \mathcal{P}_{\mathcal{T}_\Sigma} \times \mathcal{P}_{\mathcal{T}_\Sigma}$ such that $\mathcal{L}_{\{\Pi_1,\ldots,\Pi_i\}}(\sigma_1, \sigma_2)$ holds if and only if $s^{\Pi_1}_{\sigma_2} \cup \ldots \cup s^{\Pi_i}_{\sigma_2} \subseteq \sigma_1$. For certain sets of sets, this predicate can be expressed by the formula $\mathsf{Pref}_{y,z}(P)$ with variable y, z in Figure 1. For instance, $\beta \models \mathsf{Pref}_{y,z}(\forall u_1 \ldots \forall u_n)$ iff $s^{\{1,2\}^n}_{\beta(z)} \subseteq \beta(y)$, *i.e.*, $\mathcal{L}_{\{\{1,2\}^n\}}(\beta(y), \beta(z))$.

We next define that a set of paths $\Pi \subseteq \{1,2\}^n$ *supports a quantifier prefix* P noted $\Pi \vdash P$: for existential quantifier it holds that $\Pi \vdash \exists u P$ if exists $b \in \{1,2\}$ with $\{\pi | b\pi \in \Pi\} \vdash P$; for universal quantifier $\Pi \vdash \forall u P$ is valid if for all $b \in \{1,2\}$ it holds that $\{\pi | b\pi \in \Pi\} \vdash P$; the empty quantifier prefix is supported by $\{\epsilon\}$, *i.e.*, $\{\epsilon\} \vdash \epsilon$. Given this definition it holds that:

$$\beta \models \mathsf{Pref}_{y,z}(P) \quad \text{iff} \quad \begin{cases} \text{exists } \Gamma \subseteq \{\Pi \mid \Pi \vdash P\} : \\ \mathcal{L}_\Gamma(\beta(y), \beta(z)) \end{cases}$$

The complete encoding of a normalized QBF-formula PD with parameters $n \geq 0$, $m \geq 1$, and $u_1, \ldots, u_n$ is given in Figure 1: it is the formula $\mathsf{QBF}_{x,y,z}(PD)$ which conjoins $\mathsf{DBF}_{x,z}(D)$ and $\mathsf{Pref}_{y,z}(P)$. Beside of x, y, z the fresh variables $x^1_1, \ldots, x^m_{n+1}$ and $y_1, \ldots, y_{n+1}$ are used (but only of local interest).

Proposition 4.2 *Let D be normalized. A QB-formula PD is valid iff the judgment $\mathsf{QBF}_{x,y,z}(PD) \models x \subseteq y$ for atomic set constraints holds where x, y, z are fresh variables.*

The proof is given in the long version of the paper. We next illustrate the encoding at an example. We consider the QB-formula PD with parameters $n = 2$, $m = 2$, and sequence u_1, u_2:

$$\begin{array}{ll} C_1 = \neg u_2, & D = C_1 \vee C_2, \\ C_2 = u_1 \wedge u_2, & P = \exists u_1 \forall u_2 \end{array}$$

It is not difficult to see that PD is valid since D evaluates to true if one chooses $u_1 = \mathsf{T}$ and u_2 arbitrarily.

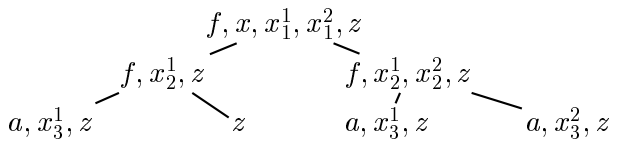
$$
\begin{array}{lcl}
\mathsf{QBF}_{x,y,z}(PD) & \equiv & \mathsf{DBF}_{x,z}(D) \wedge \mathsf{Pref}_{y,z}(P) \\
\mathsf{DBF}_{x,z}(C_1 \vee \ldots \vee C_m) & \equiv & \bigwedge_{i=1}^{m}(x{\subseteq}x_1^i \wedge \mathsf{CF}_z(i,C_i) \wedge x_{n+1}^i{=}a \wedge \bigwedge_{j=2}^{n+1} x_j^i{\subseteq}z) \\
\mathsf{Pref}_{y,z}(Q_1 \ldots Q_n) & \equiv & a{\subseteq}y_{n+1} \wedge \bigwedge_{j=1}^{n} \mathsf{Qu}_z(Q_j) \wedge y_1{\subseteq}y \\
\mathsf{CF}_z(i,C) & \equiv & \bigwedge_{j=1}^{n} \mathsf{Lit}_z(i,C,u_j) \\
\mathsf{Lit}_z(i,C,u_j) & \equiv & \left\{\begin{array}{ll} x_j^i{=}f(z,x_{j+1}^i) & \text{if } u_j \text{ in } C \\ x_j^i{=}f(x_{j+1}^i,z) & \text{if } \neg u_j \text{ in } C \\ x_j^i{=}f(x_{j+1}^i,x_{j+1}^i) & \text{otherwise} \end{array}\right. \\
\mathsf{Qu}_z(Q) & \equiv & \left\{\begin{array}{ll} f(y_{j+1},y_{j+1}){\subseteq}y_j & \text{if } Q = \forall u_j \\ f(y_{j+1},z){\subseteq}y_j \wedge f(z,y_{j+1}){\subseteq}y_j & \text{if } Q = \exists u_j \end{array}\right.
\end{array}
$$

Figure 1: Encoding a normalized QB-formula with parameters $n \geq 0$, $m \geq 1$, and $u_1, \ldots, u_n$.

We fix variables $x, z, x_1^1, \ldots, x_3^2$ and y, y_1, y_2, y_3. The encoding $\mathsf{DBF}_{x,z}(D)$ is the following constraint which up to some minor simplifications and variable renamings was also considered in the introduction:

$$
\begin{array}{lcl}
\mathsf{DBF}_{x,z}(D) & = & x{\subseteq}x_1^1 \wedge x{\subseteq}x_1^2 \wedge x_3^1{=}a \wedge x_3^2{=}a \wedge \\
 & & x_2^1{\subseteq}z \wedge x_2^2{\subseteq}z \wedge x_3^1{\subseteq}z \wedge x_3^2{\subseteq}z \wedge \\
 & & \mathsf{CF}_z(1,C_1) \wedge \mathsf{CF}_z(2,C_2) \\
\mathsf{CF}_z(1,C_1) & = & x_1^1{=}f(x_2^1,x_2^1) \wedge x_2^1{=}f(x_3^1,z) \\
\mathsf{CF}_z(2,C_2) & = & x_1^2{=}f(z,x_2^2) \wedge x_2^2{=}f(z,x_3^2)
\end{array}
$$

The set of all upper bounds for x in $\mathsf{DBF}_{x,z}(D)$ can be depicted by the following tree T. Each node of T is labeled by a set of $\mathcal{P}(\Sigma \cup V)$ such that for all t, π if t is an upper bound of x at π then the label of T at path π contains the root symbol of t, *i.e.*, $L_T(\pi)$ is the set $\{L_t(\epsilon) \mid \mathsf{DBF}_{x,z}(D) \vdash x[\pi]{\subseteq}t\}$.



Notice that the set of path π satisfying $\mathsf{DBF}_{x,z}(D) \vdash x[\pi]{\subseteq}a$ is equal to $\{11, 21, 22\}$ and thus corresponds exactly to the set of solutions of D. Next, we consider the translation of $\mathsf{QBF}_{x,y,z}(PD)$:

$$
\begin{array}{lcl}
\mathsf{QBF}_{x,y,z}(D) & = & \mathsf{DBF}_{x,z}(D) \wedge \mathsf{Pref}_{y,z}(P) \\
\mathsf{Pref}_{y,z}(P) & = & f(y_2,z){\subseteq}y_1 \wedge f(z,y_2){\subseteq}y_1 \wedge \\
 & & f(y_3,y_3){\subseteq}y_2 \wedge a{\subseteq}y_3 \wedge y_1{\subseteq}y
\end{array}
$$

Hence, the subset of lower bounds of y with variables in $\mathsf{DBF}_{x,z}(D)$ is the following set of trees:

$$\{f(z, f(a,a)), f(f(a,a), z)\}$$

This reflects that $\{\{11, 12\}, \{21, 22\}\} \vdash \exists u_1 \forall u_2$ and proves that $\mathsf{QBF}_{x,y,z}(PD)$ entails $x{\subseteq}f(z, f(a,a)){\subseteq}y$.

Theorem 4.3 *For all signatures with a least one constant and one binary function symbol, the entailment problem of atomic set constraints is PSPACE hard.*

This follows immediately from Proposition 4.2 since the validity problem of quantified boolean formulas (QBF) is PSPACE complete and the size of the encoding is clearly polynomial with respect to the size of the quantified Boolean formula.

5 Characterization of Entailment

We now give a syntactic characterization of entailment for Ines constraints (see Proposition 5.5) on which our decision procedure in Section 6 is based. We note that the characterization is complete for satisfiable B1-B3 closed constraints only.

Singletons. Entailment can depend on the fact that some term has to denote a singleton, *i.e.*, a set with exactly one element. For example, notice that the following entailment is valid for Ines:

$$x{\subseteq}f(a,v') \wedge x{\subseteq}f(u,u') \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} a{\subseteq}u \qquad (1)$$

For every solution β of the left hand side, $\beta(u) \cap \beta(a) \neq \emptyset$ holds. And since a denotes the singleton $\{a\}$, it entails $a{\subseteq}u$. Of course, there are other ways to constrain a variable (or a term) to denote a singleton. Our general idea for the recognition of singletons is to test for

ground upper bounds of variables. For completeness, we must respect the "deep shuffling" of upper bounds, as illustrated in the following example.

$$\begin{array}{ll} & x{\subseteq}f(g(a,v),v') \\ \wedge & x{\subseteq}f(g(u,a),u') \\ \wedge & x{\subseteq}f(w,a) \end{array} \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} x{\subseteq}f(g(a,a),a) \quad (2)$$

We introduce a predicate symbol $\mathsf{com}(V)$ of arity 0 for every finite non-empty set V of variables: a variable assignment β solves $\mathsf{com}(V)$ if $\bigcap_{v\in V}\beta(v)\neq\emptyset$, i.e. if there exists a common tree in the denotations of all variables in V.

Definition 5.1 *Let φ be a constraint and $V \subseteq \mathcal{V}(\varphi)$ be a non-empty set of variables. Then*

$$\varphi \vdash \mathsf{com}(V) \text{ if } \begin{cases} \text{exists } z \in \mathcal{V}(\varphi) \text{ and } \pi \text{ such that} \\ \text{for all } v \in V:\ \varphi \vdash z[\pi]{\subseteq}v \end{cases}$$

This definition is correct in that if $\varphi \vdash \mathsf{com}(V)$, then $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} \mathsf{com}(V)$.

Definition 5.2 (Ground Upper Bounds) *Let φ be a constraint and $V \subseteq \mathcal{V}(\varphi)$. If $\varphi \vdash \mathsf{com}(V)$ then we define the set $\mathcal{U}^{\mathrm{grd}}_\varphi(V)$ of* ground upper bounds of V in φ *to be the set of all trees $\tau \in D_\tau$ such that there exists $x \in V$ with $\varphi \vdash x[\pi]{\subseteq}L_\tau(\pi)$.*

Note that $L_\tau(\pi)$ is a function symbol and not a variable since τ is a tree (which do not contain variables in contrast to terms). The definition of $\mathcal{U}^{\mathrm{grd}}_\varphi(V)$ is correct in that if $\tau \in \mathcal{U}^{\mathrm{grd}}_\varphi(V)$ then for all solution β of φ over $\mathcal{P}^+(\mathcal{T}_\Sigma)$ it holds that $\bigcap_{x\in V}\beta(x) \subseteq \{\tau\}$. In particular, for a singleton $\{x\}$, $\tau \in \mathcal{U}^{\mathrm{grd}}_\varphi(\{x\})$ implies the validity of $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} x{\subseteq}\tau$. Hence, for every satisfiable constraint φ and set $V \neq \emptyset$ with $\varphi \vdash \mathsf{com}(V)$ there exists at most one ground upper bound for V in φ.

Also note that the definition of ground upper bounds can deal with "deep shuffling" for satisfiable B1-B3 closed constraints. For example in (2), the term $f(g(a,a),a)$ is a ground upper bound of the set $\{x\}$ in a B1-B3 closure of a flattened version of the left hand side of (2). For the following definitions we introduce the auxiliary notation:

$$D(V,\pi) \;=\; \{w \mid \text{exists } v \in V : \varphi \vdash v[\pi]{\subseteq}w\}$$

Definition 5.3 (Upper Bounds) *Let φ be a constraint and $V \subseteq \mathcal{V}(\varphi)$. If $\varphi \vdash \mathsf{com}(V)$ then we define the set $\mathcal{U}_\varphi(V)$ of* upper bound of V in φ *to be the set of all terms t which satisfy for all $\pi \in D_t$:*

1. *exists $v \in V$ with $\varphi \vdash v[\pi]{\subseteq}L_t(\pi)$, or*
2. *exists $Z \subseteq \mathcal{V}(\varphi)$ such that $L_t(\pi) \in Z$, $\varphi \vdash \mathsf{com}(Z)$, and $\mathcal{U}^{\mathrm{grd}}_\varphi(Z) \cap \mathcal{U}^{\mathrm{grd}}_\varphi(D(V,\pi)) \neq \emptyset$.*

The definition of $\mathcal{U}_\varphi(V)$ is correct: for all terms $t \in \mathcal{U}_\varphi(V)$ it holds that $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} \bigcap_{v\in V} v{\subseteq}t$. In particular, for a singleton $\{x\}$, $t \in \mathcal{U}_\varphi(\{x\})$ implies $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} x{\subseteq}t$. Notice that $\mathcal{U}^{\mathrm{grd}}_\varphi(\{x\}) = \mathcal{U}_\varphi(\{x\}) \cap \mathcal{T}_\Sigma$ holds for all x and φ.

Also note that our notion of upper bounds respects deep shuffle for satisfiable B1-B3 closed constraints. The less straightforward part of definition 5.3 is case 2. Let us illustrate this case on the constraint φ_3:

$$\begin{array}{ll} \varphi_3: & z{\subseteq}g(z_1) \wedge z_1{\subseteq}f(a,v_1) \wedge \\ & z{\subseteq}g(z_2) \wedge z_2{\subseteq}f(v_2,a) \wedge \\ & y{\subseteq}h(x) \wedge \\ & y{\subseteq}h(x_1) \wedge x_1{\subseteq}f(a,w_1) \wedge \\ & y{\subseteq}h(x_2) \wedge x_2{\subseteq}f(w_2,a) \end{array}$$

We next argue for φ_3 that $g(x)$ is a an upper bound for z, that is $g(x) \in \mathcal{U}_{\varphi_3}(\{z\})$ and thus $\varphi_3 \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} z{\subseteq}g(x)$. We apply Definition 5.3 with $V = \{z\}$ and verify condition 2 for the path 1. Note first that $D(\{z\},1)$ is equal to $\{z_1,z_2\}$ and that $f(a,a)$ is a ground upper bound for $\{z_1,z_2\}$ and also for $Z = \{x,x_1,x_2\}$. Note also that $\varphi_3 \vdash \mathsf{com}(Z)$ holds.

Next, we define the set of lower bounds that a constraint φ provides for a variable x. We use a kind of tree automaton that uses ground upper bounds in its complex start condition.

Definition 5.4 (Lower Bounds) *Given a constraint φ we define the set $\mathcal{L}_\varphi(x)$ of* lower bounds of variables x in φ *recursively as follows:*

$$\begin{array}{ll} \tau \in \mathcal{L}_\varphi(x) & \text{if exists } X \subseteq \mathcal{V}(\varphi) : x \in X \\ & \text{and } \mathcal{U}^{\mathrm{grd}}_\varphi(X) = \{\tau\} \\ x \in \mathcal{L}_\varphi(x) & \text{if } x \in \mathcal{V}(\varphi) \\ t \in \mathcal{L}_\varphi(x) & \text{if } y{\subseteq}x \in \varphi, \text{ and } t \in \mathcal{L}_\varphi(y) \\ f(\bar{t}) \in \mathcal{L}_\varphi(x) & \text{if } x = f(\bar{x}) \in \varphi \text{ and } \bar{t} \in \mathcal{L}_\varphi(\bar{x}) \end{array}$$

In the last line, we denote by $\bar{t}$ a sequence of terms $t_1, \ldots, t_n$, by $\overline{x}$ a sequence of variables $x_1, \ldots, x_n$, and by $\bar{t} \in \mathcal{L}_\varphi(\overline{x})$ the conditions $t_1 \in \mathcal{L}_\varphi(x_1)$, ..., $t_n \in \mathcal{L}_\varphi(x_n)$.

The definition of $\mathcal{L}_\varphi(x)$ is correct in that for all t, x if $t \in \mathcal{L}_\varphi(x)$ then $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} t \subseteq x$ holds. The first statement reflects the fact that ground upper bounds also define lower bounds (since $x \subseteq \tau$ implies $\tau \subseteq x$). As an illustration, for the previously given constraint φ_3, we prove that $\varphi_3 \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} f(a,a) \subseteq x$. We have already mentioned that $f(a,a)$ is a ground upper bound for the set $\{x, x_1, x_2\}$. Hence, this allows us to conclude that $f(a,a)$ is a lower bound for x.

Proposition 5.5 (Characterization) *If φ is satisfiable and B1-B3 closed and $x, y \in \mathcal{V}(\varphi)$, then $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} x \subseteq y$ iff $\mathcal{U}_\varphi(\{x\}) \cap \mathcal{L}_\varphi(y) \neq \emptyset$.*

Proof. The direction from right to left follows trivially from the correctness of the definitions of upper and lower bounds. The inverse direction is technically involved and can be found in [23]. □

6 Entailment is in PSPACE

Theorem 6.1 *Given an infinite signature, the entailment problem of Ines constraints is decidable in PSPACE.*

By the characterization described in Section 5, an entailment judgement $\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} x \subseteq y$ holds for a satisfiable B1-B3 closed Ines constraint φ if and only if $\mathcal{U}_\varphi(\{x\}) \cap \mathcal{L}_\varphi(y) \neq \emptyset$ holds. Slightly generalizing this property for x, y we define a predicate $A(V, x)$ for V, y.

Definition 6.2 *$\varphi \vdash A(V, y)$ holds for V, y, φ if and only if both $\varphi \vdash \mathsf{com}(V)$ and $\mathcal{U}_\varphi(V) \cap \mathcal{L}_\varphi(y) \neq \emptyset$ are valid.*

Lemma 6.3 (Correctness) *For all φ, V, y such that $\varphi \vdash \mathsf{com}(V)$ holds, the statement $\varphi \vdash A(V, y)$ is equivalent to the disjunction of the following three properties:*

1. *exists $v \in V$ such that $\varphi \vdash v[\epsilon] \subseteq y$*
2. *exists $Y \subseteq \mathcal{V}(\varphi)$ such that $y \in Y$ and $\mathcal{U}^{\mathrm{grd}}_\varphi(D(V, \varepsilon)) \cap \mathcal{U}^{\mathrm{grd}}_\varphi(Y) \neq \emptyset$*
3. *exists $v \in V$, y', f and $y_1, \ldots, y_n$ such that $\varphi \vdash v[\epsilon] \subseteq f$, $y' \subseteq y$ in φ, $y' = f(y_1, \ldots, y_n)$ in φ, and $\varphi \vdash A(D(V, i), y_i)$ holds for all $1 \leq i \leq n$.*

Proof. can be found in [23]. □

Lemma 6.4 (Complexity) *Let φ be a satisfiable B1-B3 closed constraint, V be non-empty set of variables such that $\varphi \vdash \mathsf{com}(V)$ and y a variable. Deciding whether $\varphi \vdash A(V, y)$ holds is in PSPACE.*

Testing whether $\varphi \vdash A(V, y)$ holds can done by recursively checking the properties of Lemma 6.3.

Property 1 is equivalent to that $y \in D(V, \epsilon)$ holds and this can be checked (by computing $D(V, \epsilon)$) in polynomial time in the size of φ. Property 2 is equivalent to that for all path π for which there exist a variable v in $D(V, \varepsilon)$ and a variable v' such $\varphi \vdash v[\pi] \subseteq v'$, there exist f and $w \in D(Y, \varepsilon)$ such that $\varphi \vdash v[\pi] \subseteq f$ and $\varphi \vdash w[\pi] \subseteq f$. Clearly, for a fixed Y this property is in coNP. Thus for variable Y the property is in Σ^{p}_2, thus in PSPACE. For testing Property 3 one chooses non-deterministically a constraint $y' = f(y_1, \ldots, y_n)$ *in* φ such that $y' \subseteq y$ *in* φ and then test recursively that $\varphi \vdash A(D(V, i), y_i)$ holds for all $1 \leq i \leq n$.

The complete computation for testing $\varphi \vdash A(V, y)$ (where $V \cup \{y\} \subseteq \mathcal{V}(\varphi)$) can be described by an and-or-tree whose root is labeled with $A(V, y)$. Up to a reordering of edges the tree for $\varphi \vdash A(V, y)$ is uniquely determined by φ, V and y.

The and-or tree for testing $\varphi \vdash A(W, x)$ has the following form: there are three kinds of nodes, or-nodes, and-nodes and leaves. The root of the tree is an or-node labeled by $A(W, x)$. Its leaves are either labeled with T or F. An or-node is labeled with a term of the form $A(V, y)$ for some V and y. We now define the set of sons of an or-node N with label $A(V, y)$: if Properties 1 or 2 hold for y and V then N has a unique son, a leaf labeled by T. Otherwise, we consider the set M which contains all terms of the form $V \subseteq f(y_1, \ldots, y_n)$ such that $n \geq 1$ and there exists y' satisfying $y' = f(y_1, \ldots, y_n)$ *in* φ and $y' \subseteq y$ *in* φ. If

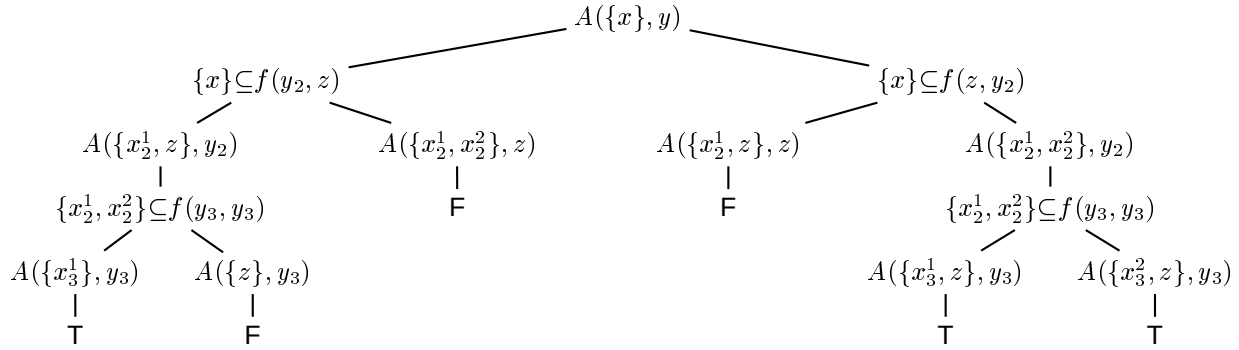


Figure 2: The And-Or Tree proving $\varphi_{\mathsf{e}} \vdash A(\{x\}, y)$

$$
\begin{array}{llll}
x\subseteq x_1^1 \wedge x_1^1{=}f(x_2^1, x_2^1) \wedge & & \wedge\, y_1'{=}f(y_2, z) \;\wedge\; y_1'\subseteq y \wedge & \\
\quad \wedge\, x_2^1{=}f(x_3^1, z) \;\wedge x_3^1{=}a \;\wedge & & \wedge\, y_1''{=}f(z, y_2) \;\wedge\; y_1''\subseteq y \wedge & x_1^1\subseteq z \wedge x_2^1\subseteq z \wedge x_3^1\subseteq z \;\wedge \\
x\subseteq x_1^2 \wedge x_1^2{=}f(z, x_2^2) \;\wedge & y_2'{=}f(y_3, y_3) \wedge y_2'\subseteq y_2 \wedge & & x_1^2\subseteq z \wedge x_2^2\subseteq z \wedge x_3^2\subseteq z \\
\quad \wedge\, x_2^2{=}f(z, x_3^2) \;\wedge x_3^2{=}a \wedge & \quad y_3{=}a \wedge & &
\end{array}
$$

Figure 3: The Constraint φ_{e}

$M = \emptyset$ then the unique son of N is a leaf labeled by F. If $M \neq \emptyset$ then the set of sons of N is the set of and-nodes built from the labels in M. The sons of an and-node with label $V\subseteq f(y_1, \ldots, y_n)$ are the or-nodes built from the labels $A(D(V, i), y_i)$ where $1 \leq i \leq n$.

To illustrate this construction, we consider the constraint φ_{e} given in Figure 3 which up to flattening is essentially the same as considered in Section 4. The constraint φ_{e} is B1-B3 closed up to trivial constraints that do not matter here. The computation tree for $\varphi_{\mathsf{e}} \vdash A(\{x\}, y)$ is given in Figure 2.

Since φ is satisfiable, it guarantees for all pairs of or-nodes on the same branch with labels $A(V, y)$ and $A(V', y')$ that $V \cap V' = \emptyset$ holds. So, the length of a branch in the and-or-tree is lineary bounded in the size of φ.

As usual, an and-or-tree can be evaluated to a Boolean value. For a satisfiable B1-B3 closed constraints φ and V, z the tree for $\varphi \vdash A(V, z)$ evaluates to T if and only if $\varphi \vdash A(V, z)$ holds. Note that the tree for $\varphi_{\mathsf{e}} \vdash A(\{x\}, y)$ evaluates to T since its right subtree does.

For constructing and evaluating a computation tree on the fly, it is sufficient to memorize the information along a single branch only. Hence, it follows that entailment of Ines constraints is in PSPACE.

7 Ines versus Atomic Set Constraints

It may seem difficult to show that the entailment problems of atomic set constraints and of Ines constraints are of the same complexity. Under the assumption of an infinite signature, however, the problem can be settled due to the independence property of Ines constraints[2].

Theorem 7.1 (Independence) *Ines constraints have the independence property in case of an infinite signature. For all $\varphi, \varphi_1, \ldots, \varphi_n$:*

$$\varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} \bigvee_{i=1}^{n} \varphi_i \quad \textit{iff } \exists 1 \leq j \leq n : \; \varphi \models_{\mathcal{P}^+(\mathcal{T}_\Sigma)} \varphi_j$$

Corollary 7.2 *For an infinite signature, the following 4 problems are PSPACE complete: 1) Entailment of Ines constraints. 2) Satisfiability of Ines constraints with negation. 3) Satisfiability of atomic set constraints with negation. 4) Entailment of atomic set constraints.*

Proof. From Theorems 4.3, 6.1, and 7.1 □

Acknowledgments The authors would like to thank Phillipe Devienne, Sophie Tison, Marc Tommasi, and Ralf Treinen for inspiring discussions and comments on all our attemps to solve the

[2]This theorem has been first claimed in [8] but the proof given there is wrong. A correct proof is given in [9] and in the long version of the present paper [23].

problem. This research has been supported by the Esprit Working Group CCL II (EP 22457) and the SFB 378 at the Universität des Saarlandes.

References

[1] A. Aiken, D. Kozen, M. Vardi, and E. Wimmers. The Complexity of Set Constraints. *7^{th} Conf. on CSL*, volume 832 of *LNCS*, pages 1–17, 1993.

[2] A. Aiken, D. Kozen, and E. Wimmers. Decidability of systems of set constraints with negative constraints. *Information and Computation*, 122(1):30–44, 1995.

[3] A. Aiken and E. Wimmers. Type inclusion constraints and type inference. In *6^{th} ACM Conf. on Functional Programming and Computer Architecture*, pages 31–41, 1993.

[4] H. Aït-Kaci, A. Podelski, and G. Smolka. A feature-based constraint system for logic programming with entailment. *TCS*, 122(1–2):263–283. 1994.

[5] L. Bachmair, H. Ganzinger, and U. Waldmann. Set constraints are the monadic class. In *8^{th} LICS*, pages 75–83. 1993.

[6] W. Charatonik and L. Pacholski. Negative set constraints with equality. In *9^{th} LICS*, 128–136. 1994.

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

[8] W. Charatonik and A. Podelski. The independence property of a class of set constraints. In *2^{nd} Int. Conf. on Principles and Practice of Constraint Progr.*, volume 1118 of *LNCS*, pages 76–90. 1996.

[9] W. Charatonik and A. Podelski. Set constraints with intersection. In *12^{th} LICS*, pages 352–361, 1997.

[10] A. Colmerauer. Equations and inequations on finite and infinite trees. In *2nd Int. Conf. on Fifth Generation Computer Systems*, pages 85–99, 1984.

[11] P. Devienne, J.-M. Talbot, and S. Tison. Solving classes of set constraints with tree automata. In *3^{rd} Int. Conf. on Principles and Practice of Constraint Progr.*, volume 1330 of *LNCS*, pages 62–76. 1997.

[12] T. Frühwirth, E. Shapiro, M. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *6^{th} LICS*, pages 300–309. 1991.

[13] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints using tree automata. In *10^{th} Symp. on Theoretical Aspects of Computer Software*, volume 665 of *LNCS*, pages 505–514. 1993.

[14] R. Gilleron, S. Tison, and M. Tommasi. Solving systems of set constraints with negated subset relationships. In *34^{th} Symp. on Found. of Computer Sc.*, pages 372–380. 1993.

[15] N. Heintze. *Set Based Program Analysis*. PhD thesis, Carnegie Mellon University, Oct. 1992.

[16] N. Heintze and J. Jaffar. A decision procedure for a class of set constraints. In *5^{th} LICS*. 1990.

[17] N. Heintze and J. Jaffar. A finite presentation theorem for approximating logic programs. In *17^{th} ACM S. on Princ. of Programming Languages*, 197–209. 1990.

[18] F. Henglein and J. Rehof. The complexity of subtype entailment for simple types. In *12^{th} LICS*, pages 362–372, 1997.

[19] F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *25^{th} Int. Conf. on Automata, Languages, and Programming*, LNCS, 1998.

[20] D. Kozen. Set constraints and logic programming. *Information and Computation*, 142(1):2–25, 1998.

[21] M. Müller. *Set-based Failure Diagnosis for Concurrent Constraint Programming*. Dissertation. Universität des Saarlandes, Saarbrcken, Jan. 1998.

[22] M. Müller, J. Niehren, and A. Podelski. Inclusion constraints over non-empty sets of trees. In *Theory and Practice of Software Development.*, volume 1214 of *LNCS*, pages 345–356, 1997.

[23] M. Müller, J. Niehren, and J.-M. Talbot. Entailment of atomic set constraints is PSpace-complete, 1999. long version at `www.ps.uni-sb.de/Papers`.

[24] F. Pottier. Simplifying subtyping constraints. In *ACM SIGPLAN Int. Conf. on Functional Programming*, pages 122–133. 1996.

[25] V. A. Saraswat, M. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In *ACM POPL*, pages 333–352. 1991.

[26] G. Smolka and R. Treinen. Records for logic programming. *Journal of Logic Programming*, 18(3):229–258, Apr. 1994.

[27] K. Stefansson. Systems of set constraints with negative constraints are NEXPTIME-complete. In *9^{th} LICS*, pages 137–141. 1994.

[28] J.-M. Talbot. *Contraintes Ensemblistes Définies et Co-définies : Extensions et Applications.* PhD thesis, Université de Lille, July, 1998.

[29] T. E. Uribe. Sorted Unification Using Set Constraints. In *Conf. on Autom. Deduction*, pages 163–177, 1992.