Equational and Membership Constraints for Infinite Trees

Joachim Niehren, Andreas Podelski, Ralf Treinen

To cite this version:

Joachim Niehren, Andreas Podelski, Ralf Treinen. Equational and Membership Constraints for Infinite Trees. 5th International Conference on Rewriting Techniques and Applications, 1993, Montreal, Canada. pp.106-120. inria-00536824

HAL Id: inria-00536824
https://inria.hal.science/inria-00536824

Submitted on 20 Nov 2010

Equational and Membership Constraints for Infinite Trees

Joachim Niehren [*] Ralf Treinen [†]
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11, Germany
(niehren/treinen)@dfki.uni-sb.de

Andreas Podelski
Digital Equipment Corporation, Paris Research Laboratory
85, Avenue Victor Hugo, F-92563 Rueil-Malmaison, France
podelski@prl.dec.com

Abstract

We present a new constraint system with equational and membership constraints over infinite trees. It provides for complete and correct satisfiability and entailment tests and is therefore suitable for the use in concurrent constraint programming systems which are based on cyclic data structures.

Our set defining devices are greatest *fixpoint solutions* of regular systems of equations with a deterministic form of union. As the main technical particularity of the algorithms we present a novel memorization technique. We believe that both satisfiability and entailment tests can be implemented in an efficient and incremental manner.

[*] Supported by the Graduierten-Kolleg Informatik der Universität des Saarlandes and by the Hydra project at DFKI.

[†] Supported by the Bundesminister für Forschung und Technology, contract ITW 9105, and by the Esprit working group CCL, contract EP 6028.

Contents

1 Introduction

Concurrent constraint programming (CCP) systems factorize into a constraint system, which may be seen as a parameter to the system, and an extension facility to compute with relations or processes. The constraint system consists of a universal data structure and a set of logical formulae, called constraints, that express relations between the data objects.

There are several computation models for different CCP systems and paradigms, such as AKL [HJ90], ALPS [Mah87], cc-languages [SR91], constraint logic programing (CLP) [JL87, HS88], LIFE [AKP91] and Oz [Smo93, HSW93]. They all require the constraints to be closed under conjunction and raise the need for an efficient and incremental constraint simplification algorithm that yields a test for satisfiability of constraints. All of them use existential quantification of constraints implicitly or explicitly, and most of them require an efficient and incremental entailment test (*i.e.*, a test of of the implication between two constraints). In particular this test is necessary for committed choice mechanisms depending on the satisfaction of guards as in Oz, AKL, LIFE and ALPS.

In many programming languages, *memberships* come in the form of static type assertions. In the CCP context however, it is natural to have memberships as relations. Having definitions for the two sets *Nat* and *NatList* like

$$\begin{aligned} Nat &= 0 \cup succ(Nat) \\ NatList &= nil \cup cons(Nat,\, NatList) \end{aligned}$$

we could of course define according unary predicates *Nat* and *NatList* in the extension facility (for instance as a logic program). The problem is that the extension facility is by design decision in general *in*complete for disjunctive information, while the sort definitions are inherently disjunctive. For instance the conjunction of the atoms $Nat(x) \wedge NatList(x)$ will not be reduced to $\bot$ unless the language provides some kind of backtracking, which often is not the case in CCP systems. Even worse, in the context of the set definitions

$$\begin{aligned} Even &= 0 \cup succ\,(Odd\,) \qquad & Nat &= 0 \cup succ\,(Nat\,) \\ Odd &= succ\,(Even\,) \qquad & Inf &= succ\,(Inf\,) \end{aligned}$$

the computation rules of the extension facility will not detect that the denotation of *Even* is a subset of the denotation of *Nat*, since this requires an inductive argument. Hence, a rule like

$$\texttt{if } Nat(x) \texttt{ then } \cdots$$

will not fire in a context where $Even(x)$ is given. The third reason why we can not employ the extension facility for dealing with memberships is founded in the use of *infinite trees* as the basic data structure. Infinite trees have been introduced in Prolog II [CKC83] in order to model cyclic data structures. With the definition of *Nat* as above, the conjunction $x \doteq succ(x) \wedge Nat(x)$ will forever unfold *Nat*. Again, an inductive reasoning is missing here.

Consequently, we claim that CCP systems will benefit from the incorporation of memberships of some restricted form into the constraint part.[1] This allows to delegate some computation from the extension facility to a possibly complete constraint solver. Hence, our constraint system comprises equational constraints *and* membership constraints. The syntax in BNF style of our constraints is as follows:

$$\gamma \ ::= \ x \mathrel{\dot{\in}} p \mid x \doteq y \mid x \doteq f(y_1, \cdots, y_n) \mid \gamma \wedge \gamma' \mid \exists x\, \gamma \mid \bot \mid \top \ .$$

As defining device we use regular systems of equations with *deterministic* union and its *greatest* fixpoint solution. These equations are not part of the constraint system. Nevertheless, it is possible to extended the system by new equations in the course of computation.

For instance in the definition of *Nat* given above, $x \mathrel{\dot{\in}} Nat$ holds exactly if x is a natural number including ∞. This conforms with the fact that ∞ has an equational representation as the unique solution of $x \doteq succ(x)$.

The union is used in a *deterministic* manner, since the constructors in the different possibilities of an equation are distinct. We use the name determinism for this concept, since the components of the *least* fixpoint solutions of our deterministic regular systems are exactly the sets recognized by deterministic top down tree automata. Without an appropriate restriction of the union like determinism we could not hope for any efficient algorithm. Furthermore, our entailment test relies on the determinism condition.

Our algorithms for testing the satisfiability and entailment are based on a novel technique that we call *memorization*. The correctness of memorization depends mainly on the *greatest* fixpoint solution. We illustrate this technique by proving the entailment:

$$x \doteq succ(x) \ \models x \mathrel{\dot{\in}} Nat \, .$$

By unfolding the definition of *Nat*, we obtain a constraint which simplifies to $x \mathrel{\dot{\in}} Nat$ relatively to $x \doteq succ(x)$.[2] Now, instead of reducing to the same subproblem infinitely often, we memorize all constraints once unfolded, and throw them away when they reappear. In this way $x \mathrel{\dot{\in}} Nat$ is simplified to $\top$, and entailment is proven.

We prove that the step of deleting once unfolded constraints is correct in the greatest fixpoint solution, while it can be wrong in other fixpoints. More technically, we use the

[1] This idea is due to Gert Smolka.

[2] Relative simplification [ST92a, AKPS92] of a constraint ϕ relatively to a constraint ψ means that we transform ϕ into a constraint ϕ' which is equivalent to ϕ modulo ψ (*i.e.*, $\phi \wedge \psi$ is equivalent to $\phi' \wedge \psi$).

fact that the greatest fixpoint solution is obtained by ω iteration steps from 'top'. Note that for arbitrary logic programs this is in general not the case [Llo84].

In order to check the satisfiability of the conjunction of membership constraints, we need to be able to compute the *intersection* of sets. Furthermore, the entailment problem for two membership constraints amounts to the computation of the *subset relation* for the two corresponding sets. Our constraint system provides both computations. Note that we can *not* decide the subset relation $p \subseteq q$ with an emptiness test of $p \cap q^c$, since the family of sets defined by deterministic equation systems is closed under intersection but neither under union nor under complement (either would lead to inefficiency by combinatorial explosion). Instead, we will give a system of transformation rules on conjunctions of subset formulae $p \subseteq q$ according to the equation system, and again apply the memorization technique.

Entailment tests for feature constraints, which refine equational constraints for infinite trees, have been treated in [ST92a, AKPS92]. In most of those contexts rational and infinite trees can not be distinguished by means of logical formulae [BS92, Mah88].

Membership constraints over sets of finite trees have been considered in [CD91, Uri92]. The case of finite feature trees is discussed in [NP93]. In these works (generalized) tree automata or regular equation systems with least fixpoint solutions are used. The proposed simplification algorithms are *not* efficient, since the union in the set defining devices is not restricted such that combinatoric explosion is possible.

As an alternative to the approach chosen here, we could have taken Rabin automata to define sets of infinite trees (*cf.*, [Tho90]). In a constraint system for CCP, however, it would be irrealistic to hope that one could use this theory. The complexity of the algorithms involved is far too high. Clearly, we don't need the expressiveness of the corresponding second-order logic.

The paper is organized as follows. We first introduce general notation and the constraint formulae. In Section 3 we define deterministic equation systems and prove the fact that the greatest fixpoint solution is calculated by ω-iteration from top. In Sections 4 and 5 we introduce the memorization technique and present the satisfiability test, proving its correctness. In Section 6 we present the entailment test. In the last two sections we describe the decision procedure for the test of the subset relation and the computation of the intersection. Finally, we conclude with a discussion of further work.

2 Equational and Membership Constraints

We assume a non empty, finite or infinite, one-sorted signature Σ of function symbols $f, g, \ldots$. $\mathcal{IT}$ denotes the set of all finite and infinite trees over Σ. We also assume an infinite alphabet of variables ranged over by x, y, z and a possibly infinite collection $\mathcal{Q}$ of

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\theta & ::= & x \doteq y \mid \top \mid \theta \wedge \theta' & & & \\
\eta & ::= & x \doteq f(\bar{y}) \mid \top \mid \eta \wedge \eta' & \phi & ::= & \theta \wedge \eta \wedge \mu \mid \bot \\
\mu & ::= & x \mathrel{\dot\in} q \mid \top \mid \mu \wedge \mu' & & &
\end{array}$$

Figure 1: The fragments of constraints without $\exists$.

set expressions ranged over by p, q, r, s. We will be more specific about the set expressions in Section 3.

Finite sequences of set expressions and variables are abbreviated as $\bar{p}$ and $\bar{x}$. We will also use similar notions like $\bar{x} \doteq \bar{y}$ or $\bar{x} \mathrel{\dot\in} \bar{p}$ for finite sequences of formulae.

As *atomic constraints* we take *equational constraints* of the form $x \doteq y$ or $x \doteq f(\bar{y})$, *membership constraints* $x \mathrel{\dot\in} p$ and $\bot$. The set of *constraints* is the closure of the set of atomic constraints under conjunction and existential quantification. $\top$ is a constraint standing for the empty conjunction. Note that, without loss of generality, we consider only flat terms $f(\bar{y})$. The symbols for constraints of several restricted forms are given in Figure 1. A membership constraint $x \mathrel{\dot\in} p$ can be seen as a convenient notion for the application $p(x)$ of a unary predicate p to the variable x.

As semantics of this first order language we consider $\mathcal{IT}$–*structures* . These are structures with the domain $\mathcal{IT}$ that interpret the function symbols f of Σ as the pertaining tree constructor $f^{\mathcal{IT}}$. The possible interpretations of the unary predicate symbols of $\mathcal{Q}$ will be restricted in Section 3 by the choice of special $\mathcal{IT}$–structures. It is understood that $\bot$ and $\doteq$ get their standard meaning. As usual, we use the notions of existential (resp. universal) closure, $\tilde{\exists} w$ (resp. $\tilde{\forall} w$), and the set of free variables $\mathcal{V}(w)$ occurring in w.

The notion of a structure $\mathcal{A}$ being a model of a closed formula w ($\models_{\mathcal{A}} w$) is defined as usual. An arbitrary formula w is *satisfiable* in a structure $\mathcal{A}$ if $\models_{\mathcal{A}} \tilde{\exists} w$, otherwise it is *unsatisfiable* in $\mathcal{A}$. A formula v *entails* a formula w in a structure $\mathcal{A}$ ($v \models_{\mathcal{A}} w$) if $\models_{\mathcal{A}} \tilde{\forall}(v \rightarrow w)$. Two formulae v, w are *equivalent* in a structure $\mathcal{A}$ ($v \models\!\mid_{\mathcal{A}} w$) if $\models_{\mathcal{A}} \tilde{\forall}(v \leftrightarrow w)$. The notions of entailment and equivalence can be extended to classes of structures. Sometimes, we furthermore use the notion $v \models^{\phi}_{\mathcal{A}} w$ for $\phi \models_{\mathcal{A}} \tilde{\forall}(v \rightarrow w)$ and $v \models\!\mid^{\phi}_{\mathcal{A}} w$ for $\phi \models_{\mathcal{A}} \tilde{\forall}(v \leftrightarrow w)$.

3 Set Definitions

When simplifying membership constraints such as $x \mathrel{\dot\in} p \wedge x \mathrel{\dot\in} q$, we need set expressions representing intersections. Therefore, we require that the set $\mathcal{Q}$ of set expressions is closed under $\cap$, which is taken to be an associative, commutative and idempotent constructor for set expressions. For instance, $q \cap (p \cap q)$ is identified with $p \cap q$.

The possible interpretations of the unary predicates are described by a given *regular system of equations* $\mathcal{E}$. This is a set of equations of one of the two following forms:

$$q \;=\; f_1(\overline{q_1}) \cup \ldots \cup f_n(\overline{q_n}) \qquad \text{or} \qquad q \;=\; \top\,. \tag{1}$$

We restrict the union in the equations to be *deterministic*, which means that the constructors on the right hand side of an equation have to be pairwise distinct. In particular the empty disjunction, denoted as $\bot$, is allowed.

We say that a set expression is *defined in* $\mathcal{E}$, if it appears on the left hand side of an equation in $\mathcal{E}$. We require that no set expression is defined twice and that each set expression appearing on the right hand side of $\mathcal{E}$ is defined. In the following sections we will often consider a constraint together with an equation system $\mathcal{E}$ and assume that all the set expressions used in the constraint are defined in $\mathcal{E}$.

A structure $\mathcal{A}$ is a *model of* $\mathcal{E}$ if the statement

$$x \mathrel{\dot\in} q \models\!|_{\mathcal{A}} \exists\overline{y_1}\ldots\exists\overline{y_n}\left(\left(x \doteq f(\overline{y_1}) \wedge \overline{y_1} \mathrel{\dot\in} \overline{q_1}\right) \vee \ldots \vee \left(x \doteq f(\overline{y_n}) \wedge \overline{y_n} \mathrel{\dot\in} \overline{q_n}\right)\right)$$

holds for all equations in $\mathcal{E}$ of the first form of (1), and if $x \mathrel{\dot\in} q \models\!|_{\mathcal{A}} \top$ holds in the second case of (1).

An equation system $\mathcal{E}$ can be considered as a syntactic characterization of its $\mathcal{IT}$*–models*. Therefore, we identify $\mathcal{E}$ with its $\mathcal{IT}$–models in notions like $v \models\!|_{\mathcal{E}} w$ and $v \models_{\mathcal{E}} w$.

We restrict ourselves to equation systems $\mathcal{E}$ with *appropriate definitions of compound set expressions*. If p, q and $p \cap q$ are defined in $\mathcal{E}$, then we require:

$$x \mathrel{\dot\in} p \cap q \models\!|_{\mathcal{E}} x \mathrel{\dot\in} p \wedge x \mathrel{\dot\in} q\,.$$

We will often make use of the following observation. If η contains the equation $x \doteq f_i(\bar{y})$, then we get by the determinism condition of $\mathcal{E}$ and the restriction to tree structures:

$$x \mathrel{\dot\in} q \models\!|^{\eta}_{\mathcal{E}} \bar{y} \mathrel{\dot\in} \overline{q_i}\,.$$

In the rest of this section we discuss computational properties of the greatest $\mathcal{IT}$–model $\mathfrak{M}$ and the least $\mathcal{IT}$–model $\mathfrak{m}$ of an equation system $\mathcal{E}$.

The set of $\mathcal{IT}$–structures over the defined set expressions of $\mathcal{E}$ is a complete lattice in its canonical order. We denote its greatest and least elements by $\mathcal{A}_\top$ and $\mathcal{A}_\bot$. The equation system $\mathcal{E}$ defines a monotone operator, also called $\mathcal{E}$, on this lattice. If $q^{\mathcal{A}}$ denotes the interpretation of the unary relation q in the $\mathcal{IT}$–structure $\mathcal{A}$, then the definition of the $\mathcal{IT}$–structure $\mathcal{E}(\mathcal{A})$ is given by:

$$\begin{array}{lcll} q^{\mathcal{E}(\mathcal{A})} & := & \bigcup_{i=1}^{n} f_i^{\mathcal{IT}}(\overline{q_i^{\mathcal{A}}}) & \quad \text{if } q = f_1(\overline{q_1}) \cup \ldots \cup f_n(\overline{q_n}) \text{ in } \mathcal{E} \\ q^{\mathcal{E}(\mathcal{A})} & := & \mathcal{IT} & \quad \text{if } q = \top \text{ in } \mathcal{E}\,. \end{array}$$

Hence, the $\mathcal{IT}$–models $\mathcal{A}$ of $\mathcal{E}$ are exactly the fixpoints of the operator $\mathcal{E}$. By monotonicity and Tarski's fixed point theorem (see for instance [Gue89]) the operator $\mathcal{E}$ has a least and a greatest fixpoint. This proves the existence of $\mathfrak{m}$ and $\mathfrak{M}$.

The operator $\mathcal{E}$ is upward and downward continuous, as the reader easily verifies. This means that $\mathcal{E}$ preserves least upper (greatest lower) bounds of every upward (downward) directed chain $\mathcal{A}_\alpha$, i.e. $\mathcal{E}(\sup \mathcal{A}_\alpha) = \sup \mathcal{E}(\mathcal{A}_\alpha)$ ($\mathcal{E}(\inf \mathcal{A}_\alpha) = \inf \mathcal{E}(\mathcal{A}_\alpha)$). With an application of Kleene's fixed point theorem to the complete lattice of $\mathcal{IT}$ structures and to its dual lattice, we get that the least (resp. greatest) fixed points of $\mathcal{E}$ can be reached in ω iteration steps from bottom (resp. top). This is well known for the least fixed point of $\mathcal{E}$, since $\mathcal{E}$ considered as a logic program defines an upward continuous operator. For the greatest fixed point of $\mathcal{E}$ it is surprising, since it takes in general more than ω steps to iterate the greatest fixed point of a logic program from top [Llo84].

Lemma 3.1 $\quad \mathfrak{m} = \bigcup_{m=0}^{\infty} \mathcal{E}^m(\mathcal{A}_{\bot}) \qquad$ *and* $\qquad \mathfrak{M} = \bigcap_{m=0}^{\infty} \mathcal{E}^m(\mathcal{A}_{\top})$.

We intend to interpret set expressions in greatest $\mathcal{IT}$-models $\mathfrak{M}$. Therefore we call a subset of $\mathcal{IT}$ *definable*, if it is a component of the greatest $\mathcal{IT}$-model of some deterministic equation system.

An example of a non-definable set is $\{f(a,a), f(b,b)\}$, since our equation systems are deterministic.

In general, the restrictions of $\mathcal{IT}$ definable sets to finite trees are exactly those that are recognizable by a deterministic top down tree automaton.

More precisely the restriction of the greatest solution of a deterministic equation system to finite trees is the least solution, and the components of the least solution of deterministic equation systems are exactly the sets recognizable by deterministic tree automata.

4 Normal Forms of Constraints

In order to decide the satisfiability of constraints, we present a transformation of constraints into either $\bot$ or a satisfiable normal form. Since $\exists x \phi$ is satisfiable iff ϕ is, we will restrict ourselves to constraints without existential quantification. These are considered as multisets of atomic constraints. In other words the conjunction is seen to be associative and commutative, but not idempotent.

Since we use membership constraints, all our normal forms are calculated with respect to the maximal model $\mathfrak{M}$ of an equation system $\mathcal{E}$.

A variable is called *constrained* (in ϕ) if it appears on the left hand side of an atomic constraint in ϕ which is not equivalent to $\top$. With $\mathcal{C}(\phi)$ we denote the set of all constrained variables in the multiset ϕ. The problem $\phi \models\!\mid_{\mathfrak{M}} \top$ can be decided syntactically. This is trivial for infinite, and a little bit more complicated for finite signatures. For example, let $\mathcal{E}$ contain the definition of Nat from the introduction and let ϕ be the constraint $x \doteq x \wedge y \,\dot{\in}\, Nat$. Then $\mathcal{C}(\phi) = \emptyset$, if the signature consists of $\{succ\,, \mathit{0}\,\}$ only, and $\mathcal{C}(\phi) = \{y\}$ otherwise.

For the case of an infinite signature, x is always constrained in $x \doteq f(\bar{y})$ and constrained in $x \,\dot{\in}\, p$ iff $p = \top$ is not in $\mathcal{E}$ (both statement can be wrong for finite signatures). Note that $x \doteq y$ constrains x if $x \neq y$, but not y.

Definition 4.1 *A constraint ϕ is in* normal form*, iff $\phi = \theta \wedge \eta \wedge \mu$ with*

1. *every variables of ϕ is constrained at most once.*
2. *every variable constrained in θ does not occur in $\eta \wedge \mu$.*
3. *μ is satisfiable in $\mathfrak{M}$.*

ψ is a normal form of *ϕ, if ψ is in normal form and $\phi \models\!\mid_{\mathfrak{M}} \psi$.*

A normal form θ can be considered as an idempotent substitution with domain $\mathcal{C}(\theta)$. The application of this substitution to a formula w is denoted by θw and corresponds exactly to the elimination of the constrained variables of θ in w.

The following proposition implies in particular the satisfiability of normal forms. We will exploit this proposition again for the entailment check.

Proposition 4.2 *If ϕ is in normal form, then every assignment of the non constrained variables of ϕ can be extended to a solution of ϕ in $\mathfrak{M}$:*

$$\models_{\mathfrak{M}} \quad \tilde{\forall} \exists \mathcal{C}(\phi) \phi \,.$$

The proof reduces immediately to the case of equational constraints only, which has been solved in [Mah88].

Normal forms of equational constraints can be obtained by the well known unification rules in Figure 2. We obtain normal forms of arbitrary constraints in four steps. First we calculate a normal form $\theta \wedge \eta$ of the equational part. Second we apply θ to the membership part and call the result μ. Third we simplify μ relative to η and $\mathcal{E}$ by memorization. In the last step we calculate intersections and detect unsatisfiable membership constraints.

The memorization technique is described by schemes of rewrite rules which depend on η and $\mathcal{E}$. It transforms expressions of the form $\mu \,\square\, \mu'$, where $\square$ is a new symbol. We say that μ_0 *simplifies to* μ_1 *relative to* η *and* $\mathcal{E}$, if there is a μ_1' such that $\mu_0 \square \top$ rewrites to

decomp	$\dfrac{x \doteq f(\bar{y}) \wedge x \doteq f(\bar{z}) \wedge \sigma}{x \doteq f(\bar{y}) \wedge \bar{y} \doteq \bar{z} \wedge \sigma}$	
elim	$\dfrac{x \doteq y \wedge \sigma}{x \doteq y \wedge \sigma[x \leftarrow y]}$	if $x \neq y$ and $x \in \mathcal{V}(\sigma)$
clash 1	$\dfrac{x \doteq f(\bar{y}) \wedge x \doteq g(\bar{z}) \wedge \sigma}{\bot}$	if $f \neq g$

Figure 2: Unification rules. Here $\sigma ::= x \doteq y \mid x \doteq f(\bar{y}) \mid \sigma \wedge \sigma'$.

unfold	$\dfrac{x \dot\in q \wedge \mu \,\Box\, \mu'}{\bar{y} \dot\in \bar{p} \wedge \mu \,\Box\, x \dot\in q \wedge \mu'}$	if $x \dot\in q$ is not in μ', $q = \ldots \cup f(\bar{p}) \cup \ldots$ is in $\mathcal{E}$, and $x \doteq f(\bar{y})$ is in η.
memo	$\dfrac{x \dot\in q \wedge \mu \,\Box\, \mu'}{\mu \,\Box\, \mu'}$	if $x \dot\in q$ is in μ'
clash 2	$\dfrac{x \dot\in q \wedge \mu \,\Box\, \mu'}{\bot \,\Box\, \mu'}$	if $x \doteq f(\bar{y})$ is in η the definition of q does not contain f, and $q = \top$ is not in $\mathcal{E}$.

Figure 3: Simplification of Memberships relative to Equational Constraints with Memorization.

$\mu_1 \Box \mu_1'$ relative to η and $\mathcal{E}$. In this case we will prove μ_0 and μ_1 to be equivalent relative to η in $\mathfrak{M}$ (correctness of memorization).

On the right hand side of $\Box$ we memorize the constraints which have already been unfolded. The rules are presented in Figure 3. They forbid multiple unfolding of the same constraint and delete those that have been unfolded before.

The termination of memorization is obvious. The main problem is the correctness of the *memo*-rule, which is proven in Section 5.

A typical example is the simplification of the constraint $x \dot\in Inf$ relative to $x \doteq succ\,(x)$ and $\mathcal{E}$ containing $Inf = succ\,(Inf)$:

$$\dfrac{\dfrac{x \dot\in Inf \;\Box\; \top}{x \dot\in Inf \;\Box\; x \dot\in Inf}\,unfold}{\top \;\Box\; x \dot\in Inf}\,memo$$

The last set of rules handles empty sets in membership constraints and conjunctions of membership constraints for the same variable. It is given in Figure 4. During its execution we want to maintain two invariants. First, each occurring set expression should be defined in $\mathcal{E}$. This means that we have to calculate equations for intersections and to extend the equation system by need without changing the interpretations of previously defined set expressions in $\mathfrak{M}$. This will be done in Section 7.

$$
\begin{array}{lll}
\textit{intersect} & \dfrac{x \,\dot{\in}\, p \wedge x \,\dot{\in}\, q}{x \,\dot{\in}\, p \cap q} & \\[2ex]
\textit{empty} & \dfrac{x \,\dot{\in}\, p \wedge \mu}{\bot} & \text{if } p = \bot \text{ is in } \mathcal{E}
\end{array}
$$

Figure 4: Simplification of Empty Sets and Conjunctions of Membership.

Second, a set expression p should be defined by $p = \bot$ iff $p^{\mathfrak{M}} = \emptyset$. This can easily be done by propagating $\bot$ in $\mathcal{E}$.

Theorem 4.3 *When started with the constraint ϕ, the above algorithm terminates with $\bot$ if ϕ is unsatisfiable in $\mathfrak{M}$, and in a normal form of ϕ otherwise.*

Here is an example that illustrates our algorithm in action. $\mathcal{E}$ contains the equations for Nat, $\mathit{NatList}$, Even and Odd from the introduction. We compute a normal form of

$$x \doteq \mathit{cons}\,(y,x) \wedge x \doteq \mathit{cons}\,(z,x) \wedge x \,\dot{\in}\, \mathit{NatList} \wedge y \,\dot{\in}\, \mathit{Even} \wedge z \,\dot{\in}\, \mathit{Odd} \ .$$

The equational part simplifies to $\theta \wedge \eta$ with

$$\theta \ = \ y \doteq z \wedge x \doteq x, \quad \eta \ = \ x \doteq \mathit{cons}\,(z,x)\,.$$

By applying θ to the membership part we get

$$\mu \ = \ x \,\dot{\in}\, \mathit{NatList} \wedge z \,\dot{\in}\, \mathit{Even} \wedge z \,\dot{\in}\, \mathit{Odd}$$

The memorization algorithm simplifies μ relative to η and $\mathcal{E}$ to

$$\mu_1 \ = \ z \,\dot{\in}\, \mathit{Nat} \wedge z \,\dot{\in}\, \mathit{Even} \wedge z \,\dot{\in}\, \mathit{Odd} \ .$$

This is transformed with the intersection rule to $z \,\dot{\in}\, \mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd}$. The intersection algorithm of Section 7 adds the following equation to $\mathcal{E}$:

$$\mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd} \quad = \quad \mathit{succ}(\mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd}) \ .$$

To be precise it also adds an equation for $\mathit{Even} \cap \mathit{Odd}$, $\mathit{Nat} \cap \mathit{Even}$ or $\mathit{Nat} \cap \mathit{Odd}$ depending on which intersection is calculated first. We get the normal form

$$y \doteq z \wedge x \doteq x \wedge x \doteq \mathit{cons}\,(z,x) \wedge z \,\dot{\in}\, \mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd} \ .$$

This normal form is satisfiable since $\infty \in (\mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd})^{\mathfrak{M}}$. Note that we could replace $z \,\dot{\in}\, \mathit{Nat} \cap \mathit{Even} \cap \mathit{Odd}$ by the $\mathfrak{M}$-equivalent constraint $z \doteq \mathit{succ}\,(z)$. This will be necessary in the entailment check.

5 Correctness of Relative Simplification with Memorization

Since the *clash* rule terminates the rewriting, we can restrict ourselves to the *memo* and *unfold* rule. The relation 'rewrites to in one *memo* or *unfold* step' on expressions of the form $\mu \Box \mu'$ will be denoted by $\rhd_{\eta,\mathcal{E}}$ and its reflexive transitive closure by $\rhd^{*}_{\eta,\mathcal{E}}$.

Roughly speaking, the following theorem states that the symbol $\Box$ can be interpreted as the logical connective $\wedge$ with respect to all $\mathcal{IT}$–models of $\mathcal{E}$, and also as $\rightarrow$ with respect to the greatest $\mathcal{IT}$–model $\mathfrak{M}$.

Theorem 5.1 *For each computation*

$$\mu_0 \Box \mu_0' \ \ \rhd^{*}_{\eta,\mathcal{E}} \ \ \mu_1 \Box \mu_1'$$

the following two statements are invariants:

$$\mu_0 \wedge \mu_0' \ \models\!\!\mid^{\eta}_{\mathcal{E}} \ \mu_1 \wedge \mu_1' \qquad \textit{and} \qquad \mu_0' \ \models^{\eta}_{\mathfrak{M}} \ \mu_1 \rightarrow \mu_1' \, .$$

Only the second statement is not obvious, since the assumption of the implication is weakened by the memorization rule.

Corollary 5.2 (Correctness) *If μ_0 simplifies to μ_1 by memorization relative to η and $\mathcal{E}$ then $\mu_0 \ \models\!\!\mid^{\mathcal{E}}_{\mathfrak{M}} \ \mu_1$ holds.*

Proof. By definition there is μ_1' with $\mu_0 \Box \top \ \ \rhd^{*}_{\eta,\mathcal{E}} \ \mu_1 \Box \mu_1'$. The above theorem implies

$$\mu_0 \ \wedge \ \top \ \models\!\!\mid^{\eta}_{\mathcal{E}} \ \mu_1 \ \wedge \ \mu_1' \ \models\!\!\mid^{\eta}_{\mathfrak{M}} \ \mu_1 \, .$$

□

Theorem 5.1 can be proven with the help of Lemma 5.3, which reflects an important property of the greatest $\mathcal{IT}$–model $\mathfrak{M}$.

In order to be general enough we need the concept of *derivable constructors*. If $f \in \Sigma$ is a constructor with arity n, m a natural number and $\sigma : \{1, \cdots, n\} \rightarrow \{1, \cdots, m\}$ an arbitrary mapping, then the pair f_σ is called derivable constructor with arity m. The interpretation of f_σ in a Σ–structure $\mathcal{I}$ is defined by

$$f_\sigma^{\mathcal{I}}(d_1, \ldots, d_m) \ = \ f^{I}(d_{\sigma(1)}, \ldots, d_{\sigma(n)})$$

for all elements d_i of the domain of $\mathcal{I}$. Each constructor is itself a derivable constructor, since we may chose σ to be the identity. We will freely use derivable constructors as abbreviations in terms. For example $f_\sigma(x,y)$ stands for $f(y,y,y)$ if σ is the mapping $\sigma(1) = \sigma(2) = \sigma(3) = 2$. In the sequel we will not distinguish between constructors and derivable constructors.

Using this notion in the rest of this section we will always assume finite sequence of objects to have the form $\bar{o} = (o_i)_i$.

Lemma 5.3 (Main) *Let $\mathcal{E}$ be an equation system, $\bar{p} = (p_i)_i$ and $\bar{q} = (q_i)_i$ finite sequences of set expressions, $\bar{x} = (x_i)_i$ and $\bar{y} = (y_i)_i$ finite sequences of variables, $\bar{f} = (f_i)_i$ a finite sequence of derivable constructors and η a constraint. We assume that for all j the equations*

$$p_j \;=\; \ldots \cup\, f_j(\bar{p},\bar{q}) \,\cup\, \ldots$$

are contained in $\mathcal{E}$ and that the statement

$$\models^{\eta}_{\mathfrak{M}} \; x_j \;\dot{=}\; f_j(\bar{x},\bar{y})$$

is valid. Then the following implication relative to η and the greatest model $\mathfrak{M}$ of $\mathcal{E}$ holds:

$$\models^{\eta}_{\mathfrak{M}} \; \bar{y}\,\dot{\in}\,\bar{q} \;\rightarrow \bar{x}\,\dot{\in}\,\bar{p}\,.$$

In order to illustrate the contents of the Main Lemma, let η be $x_1 \doteq f(x_1,x_2) \wedge x_2 \doteq f(x_1,x_2)$ and let $\mathcal{E}$ contain the equations $p_1 \doteq f(p_1,p_2)$ and $p_2 \doteq f(p_1,p_2)$. The main lemma implies $\top \models^{\eta}_{\mathfrak{M}} x_1 \,\dot{\in}\, p_1 \wedge x_2 \,\dot{\in}\, p_2$. Note that this does not hold in any other solution of $\mathcal{E}$, i.e. for $p_1^{\mathfrak{m}} = \emptyset$ and $p_2^{\mathfrak{m}} = \emptyset$.

Proof of the Main Lemma. We proof that each solution of $\eta \wedge \bar{y}\,\dot{\in}\,\bar{q}$ over $\mathcal{M}$ is a solution of $\bar{x}\,\dot{\in}\,\bar{p}$. Suppose $\alpha : Var \rightarrow \mathcal{IT}$ to be a solution of $\eta \wedge \bar{y}\,\dot{\in}\,\bar{q}$ in $\mathcal{M}$. This implies

$$\alpha(x_j) \;=\; f_j^{\mathcal{IT}}(\overline{\alpha(x)},\overline{\alpha(y)}) \qquad \text{and} \qquad \alpha(y_j) \;\in\; q_j^{\mathfrak{M}}$$

for all j (with $\overline{\alpha(x)} \;=\; (\alpha(x_i))_i$).

By the representation of $\mathfrak{M}$ in Lemma 3.1, it suffices to show:

$$\alpha(x_j) \;\in\; p_j^{\mathcal{E}^m(\mathcal{A}_\top)}$$

for all j and all $m \geq 0$. This can be done by induction over m. The case $m = 0$ is trivial. For the induction step we have to show

$$\begin{array}{rcl}\alpha(x_j) & \in & p_j^{\mathcal{E}^{m+1}(\mathcal{A}_\top)} \\ & = & \ldots \cup\, f_j^{\mathcal{IT}}(\overline{p^{\mathcal{E}^m(\mathcal{A}_\top)}},\, \overline{q^{\mathcal{E}^m(\mathcal{A}_\top)}}) \,\cup \ldots\end{array}$$

for all j. But $\alpha(x_j)$ is contained in the right hand side. Indeed $\alpha(x_j) \;=\; f_j^{\mathcal{IT}}(\overline{\alpha(x)},\overline{\alpha(y)})$, by induction hypothesis

$$\overline{\alpha(x)} \in \overline{p^{\mathcal{E}^m(\mathcal{A}_\top)}}$$

and by assumption

$$\overline{\alpha(y)} \in \overline{q^{\mathfrak{M}}} \subset \overline{q^{\mathcal{E}^m(\mathcal{A}_\top)}} .$$

□

Proof of Theorem 5.1. For simplicity we assume $\mu_0' = \top$, which is sufficient to conclude Corollary 5.2.

We call the expression $\bar{y} \mathrel{\dot{\in}} \bar{q} \mathbin{\Box} \bar{x} \mathrel{\dot{\in}} \bar{p}$ *appropriate with respect to* η *and* $\mathcal{E}$ iff there is a finite sequence of derived constructors $\bar{f}$ and a finite sequence of variables $\bar{y}$ such that for all j the equation

$$p_j \;=\; \ldots \cup f_j(\bar{p}, \bar{q}) \cup \ldots .$$

is in $\mathcal{E}$ and the following statement holds:

$$\models^{\eta} \; x_j \doteq f_j(\bar{x}, \bar{y}) .$$

$\mu_0 \mathbin{\Box} \top$ is appropriate even for arbitrary η and $\mathcal{E}$. We will show that $unfold$ and $memo$ steps relative to η and $\mathcal{E}$ maintain appropriateness relative to η and $\mathcal{E}$. Therefore $\mu_1 \mathbin{\Box} \mu_1'$ is appropriate relative to η and $\mathcal{E}$. The Main Lemma yields

$$\models^{\eta}_{\mathfrak{M}} \;\; \mu_1 \;\rightarrow\; \mu_1' .$$

It remains to prove that the $unfold$ and $memo$ rule maintain appropriateness. First we consider the $unfold$ rule that reduces $x' \mathrel{\dot{\in}} p'$ to $\overline{y'} \mathrel{\dot{\in}} \overline{q'}$ in

$$\bar{y} \mathrel{\dot{\in}} \bar{q} \wedge x' \mathrel{\dot{\in}} p' \mathbin{\Box} \bar{x} \mathrel{\dot{\in}} \bar{p} \;\; \rhd_{\eta,\mathcal{E}} \; \bar{y} \mathrel{\dot{\in}} \bar{q} \wedge \overline{y'} \mathrel{\dot{\in}} \overline{q'} \mathbin{\Box} \bar{x} \mathrel{\dot{\in}} \bar{p} \wedge x' \mathrel{\dot{\in}} p' .$$

To prove the appropriateness of the right hand side we will find constructors g' and $\bar{g}$ with

$$\begin{array}{rcllll} p_j & = & \ldots \cup g_j(\bar{p}, p', \bar{q}, \overline{q'}) \cup \ldots & \text{and} & \models^{\eta} & x_j \doteq g_j(\bar{x}, x_0, \bar{y}, \overline{y'}) , \\ p' & = & \ldots \cup g'(\bar{p}, p', \bar{q}, \overline{q'}) \cup \ldots & \text{and} & \models^{\eta} & x' \doteq g'(\bar{x}, x', \bar{y}, \overline{y'}) . \end{array}$$

By the application condition of the $unfold$ rule, there is a constructor f' with

$$p' \;=\; \ldots \cup f'(\overline{q'}) \cup \ldots \qquad \text{and} \qquad \models^{\eta} \;\; x' \doteq f'(\overline{y'}) \; .$$

By the appropriateness of the left hand side of $\rhd_{\eta,\mathcal{E}}$ there are derived constructors $\bar{f}$ with

$$p_j \;=\; \ldots \cup f_j(\bar{p}, \bar{q}, p') \cup \ldots \qquad \text{and} \qquad \models^{\eta} \;\; x_j \doteq f_j(\bar{x}, \bar{y}, x')$$

We can now find definable constructors that are essentially like f_j, but take possibly more arguments in a possibly permuted order. Formally there are definable constructors $\bar{g}$ and g' with

$$f_j(\bar{x}, \bar{y}, x') \;=\; g_j(\bar{x}, x', \bar{y}, \overline{y'}) \qquad \text{and} \qquad f'(\overline{y'}) \;=\; g'(\bar{x}, x', \bar{y}, \overline{y'}) \; .$$

This implies

$$f_j(\bar{p}, \bar{q}, p') \;=\; g_j(\bar{p}, q, \bar{q}, \overline{q'}) \qquad \text{and} \qquad f'(\overline{q'}) \;=\; g'(\bar{p}, p', \bar{q}, \overline{q'})$$

and proves the appropriateness of the right hand side.
For the second case we consider an application of the memo rules erasing $x_0 \mathrel{\dot{\in}} p_0$:

$$\bar{y} \mathrel{\dot{\in}} \bar{q} \wedge x_0 \mathrel{\dot{\in}} p_0 \,\square\, \bar{x} \mathrel{\dot{\in}} \bar{p} \wedge x_0 \mathrel{\dot{\in}} p_0 \;\; \rhd_{\eta,\mathcal{E}} \; \bar{y} \mathrel{\dot{\in}} \bar{q} \,\square\, \bar{x} \mathrel{\dot{\in}} \bar{p} \wedge x_0 \mathrel{\dot{\in}} p_0 \,.$$

The appropriateness of the left hand side guarantees the existence of derivable constructors $\bar{f}$ and f_0 with

$$p_j \;=\; \ldots \cup f_j(\bar{p}, p_0, \bar{q}, p_0) \cup \ldots \qquad \text{and} \qquad \models^{\eta} \;\; x_j \doteq f_j(\bar{x}, x_0, \bar{y}, x_0)$$

for all $j \geq 0$. Now we can find derivable constructors that take x_0, resp. p_0 only once. To be precise, we define $\bar{g}$ and g_0 with

$$f_j(\bar{x}, x_0, \bar{y}, x_0) \;=\; g_j(\bar{x}, x_0, \bar{y})$$

for all $j \geq 0$. Therefore

$$f_j(\bar{p}, p_0, \bar{q}, p_0) \;=\; g_j(\bar{p}, p_0, \bar{q})\,.$$

This shows the appropriateness of the right hand side.

□

6 The Entailment Check

In this section, we show how to decide entailment between existentially quantified constraints in the greatest model $\mathfrak{M}$.

> For the purpose of this section we assume that the signature contains at least two elements, since otherwise the domain of the models under consideration will be singleton, and hence *every* equation will hold.

Initially, we are given the question whether

$$\exists X'\phi' \models_{\mathfrak{M}} \exists X\phi \tag{2}$$

holds, where X, X' are finite sets of variables. We may assume without loss of generality that ϕ' is satisfiable in $\mathfrak{M}$, since otherwise (2) holds vacuously. Hence by Theorem 4.3 we can assume ϕ' to be in normal form. For the purpose of entailment checks it is convenient to exclude certain forms of degenerate membership constraints. Hence, we furthermore require that for all membership constraints $x \mathrel{\dot\in} p$ of ϕ' the definition of p is disjunctive. A normal form meeting this additional condition is called a *branching normal form.* Note that a branching normal form contains only membership constraints $x \mathrel{\dot\in} p$ for which $p^{\mathfrak{M}}$ is not singleton. We can always transform a normal form into an equivalent branching normal form by introducing new existentially quantified variables:

- A membership constraint $x \mathrel{\dot\in} p$, where $p^{\mathfrak{M}}$ is singleton, is equivalent to a corresponding equation. For example, if $\mathcal{E}$ contains $p = f(q)$ and $q = g(p)$, then $x \mathrel{\dot\in} p$ is in $\mathfrak{M}$ equivalent to $\exists y(x \doteq f(y) \wedge y \doteq g(x))$.
- A membership constraint $x \mathrel{\dot\in} p$, where the definition of p is of the form $p = f(\bar{q})$ and where $p^{\mathfrak{M}}$ is not singleton, is replaced by $\exists \bar{y}(x \doteq f(\bar{y}) \wedge \bar{y} \mathrel{\dot\in} \bar{q})$.

Both rules maintain normal forms (modulo existential quantification) and terminate, since the second rule applies only when $p^{\mathfrak{M}}$ is not a singleton. Note that by Lemma 5.3, $p^{\mathfrak{M}}$ is non-singleton iff the definition of p depends on a definition which is disjunctive, or which is $\top$.

Taking the definition of *Inf*, *Even* and *Odd* as given in the introduction, we transform the existentially quantified normal form

$$\exists y(x \mathrel{\dot\in} \mathit{Inf} \wedge y \mathrel{\dot\in} \mathit{Odd})$$

into the existentially quantified branching normal form

$$\exists y \exists z(x \doteq \mathit{succ}(x) \wedge y \doteq \mathit{succ}(z) \wedge z \mathrel{\dot\in} \mathit{Even})\,.$$

The next lemma states that membership constraints in a branching normal form can not contribute to equalities:

Lemma 6.1 *Let $\eta' \wedge \mu'$ be in branching normal form. Then $\eta' \wedge \mu' \models_{\mathfrak{M}} \theta$ iff $\eta' \models_{\mathfrak{M}} \theta$.*

Hence, we assume without loss of generality in (2) that ϕ' is in branching normal form. Since we may in (2) assume without loss of generality that X' is disjoint to $\mathcal{V}(\phi)$,we may drop the existential quantifier on the left hand side. The normal form ϕ' can be written as $\theta' \wedge \eta' \wedge \mu'$. Since θ' is an idempotent substitution and since we may assume X to be disjoint to $\mathcal{V}\theta'$, we arrive at the problem

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \exists X \theta' \phi$$

where $\eta' \wedge \mu'$ is in branching normal form.

Before we state the entailment theorem we consider the special case of a right hand side consisting of equations only. We say that some θ is *complete* for some η if

$$\theta \models x \doteq y \quad \text{and} \quad x \doteq f(\bar{x}) \in \eta \quad \text{and} \quad y \doteq f(\bar{y}) \in \eta \quad \text{implies} \quad \theta \models \bar{x} \doteq \bar{y}\,.$$

For instance, $x \doteq v \wedge y \doteq v$ is complete for $x \doteq f(x,y) \wedge y \doteq f(v,x) \wedge v \doteq f(y,x)$.

We define the quantor $\hat{\exists}x\phi$ (read: there is *at most* one x such that ϕ) as an abbreviation for:

$$\forall y_1 \forall y_2 (\phi[x \leftarrow y_1] \wedge \phi[x \leftarrow y_2] \rightarrow y_1 \doteq y_2)$$

The generalization $\hat{\exists}X$ to a finite set X of variables is straightforward. This quantor has the important property that:

$$\tilde{\forall}\hat{\exists}X\phi \wedge \tilde{\forall}\exists X(\phi \wedge \psi) \models \tilde{\forall}(\phi \rightarrow \psi)\,. \tag{3}$$

We can now express an important property of normal forms which is in some sense a counterpart to Proposition 4.2. This lemma has been given in [Mah88] as an axiom of infinite trees.

Lemma 6.2 *For every η we have $\models_{\mathfrak{M}} \tilde{\forall}\hat{\exists}\mathcal{C}(\eta)\,\eta$.*

Lemma 6.3 (Determined Equations) *Let θ be complete for η', let θ contain no trivial equation $x \doteq x$ and let $\theta \wedge \eta'$ be satisfiable in $\mathfrak{M}$. Then $\eta' \models_{\mathfrak{M}} \theta$ iff $\mathcal{V}(\theta) \subseteq \mathcal{C}(\eta')$.*

For instance,

$$x \doteq f(x,y) \wedge y \doteq f(v,x) \wedge v \doteq f(y,x) \models_{\mathfrak{M}} x \doteq v \wedge y \doteq v\,.$$

This does not hold, if we drop the third equation form the assumption.

Proof. If $\mathcal{V}(\theta) \not\subseteq \mathcal{C}(\eta')$, then we can find a valuation which satisfies η' but not θ as follows: If θ contains $x \doteq y$, where both x and y are not constrained in η', then we may choose arbitrary different values for x and y. If θ contains $x \doteq y$, where one variable (say x) is not constrained and the other (say y) is constrained in η' by, say, $y = f(\bar{y})$, then we choose for x a value which has a root symbol different from f. This is always possible since we assume our signature to contain at least two elements. In both cases, we use Proposition 4.2 to get a solution of η' which does not satisfy θ.

Let $\mathcal{V}(\theta) \subseteq \mathcal{C}(\eta')$. By Lemma (6.2)

$$\models_{\mathfrak{M}} \tilde{\forall}\hat{\exists}\mathcal{C}(\eta')\,\eta'\,. \tag{4}$$

We may assume without loss of generality that θ is arranged to be an idempotent substitution. We now show that $\theta\eta'$ is in normal form up to multiple occurrences of atomic constraints. Assume that

$$x \doteq f(\bar{y}) \wedge x \doteq g(\bar{z}) \subseteq \theta\eta'\,.$$

Since $\theta \wedge \eta'$ is satisfiable, f equals g. Since θ is complete for η', we obtain $\theta \models \bar{y} \doteq \bar{z}$. Since θ is an idempotent substitution and since $\bar{y}, \bar{z}$ are in the codomain of θ this implies $\bar{y} = \bar{z}$. Hence by Proposition 4.2

$$\models_{\mathfrak{M}} \tilde{\forall}\exists\mathcal{C}(\theta\eta')(\theta \wedge \theta\eta') \models\!\!\mid \tilde{\forall}\exists\mathcal{C}(\theta\eta')(\theta' \wedge \eta') \models \tilde{\forall}\exists\mathcal{C}(\eta')(\theta' \wedge \eta') \tag{5}$$

The last implication is justified by $\mathcal{V}(\theta) \subseteq \mathcal{C}(\eta')$, hence $\mathcal{V}(\theta\eta') \subseteq \mathcal{C}(\eta')$. Now, the claim follows by (3) from (4) and (5). □

A proof of a more general lemma (in the context of feature constraints) has been given in [ST92b].

Before we state the entailment theorem we have to introduce some more notation. We call θ X-*directed* if θ contains no equation $x \doteq y$ with $x \notin X$ and $y \in X$.

For a constraint ϕ we define ϕ^X to be the subset of atomic constraints which constrain only variables from X, and ϕ^{-X} to be the subset of atomic constraints which constrain only variables alien to X. Since every constraint is either equivalent to $\top$ or constrains a variable, we have $\phi \models\!\!\mid_{\mathfrak{M}} \phi^X \wedge \phi^{-X}$.

Definition 6.4 *Let $\eta' \wedge \mu'$ be in branching normal form. The constraint $\exists X(\theta \wedge \eta \wedge \mu)$ is in* normal form relative to $\eta' \wedge \mu'$ *if*

1. *θ is X-directed,*
2. *θ is complete for $\eta' \wedge \eta$, and $\theta \wedge \eta'$ is satisfiable in $\mathfrak{M}$,*
3. *$\mathcal{C}(\eta')$ and $\mathcal{C}(\mu)$ are disjoint,*
4. *$\theta \wedge \eta \wedge \mu$ is in normal form.*

For instance,

$$\exists v(v \doteq z \wedge x \doteq y \wedge y \doteq f(y) \wedge z \,\dot{\in}\, p) \tag{6}$$

is in normal form relative to

$$x \doteq f(y) \wedge y \doteq f(x) \wedge w \doteq h(z) \wedge z \,\dot{\in}\, q\,. \tag{7}$$

This does not hold if we drop $x \doteq y$ from (6), since then clause 2 of Definition 6.4 is violated.

Theorem 6.5 (Entailment) *Let $\eta' \wedge \mu'$ be in branching normal form, let X be disjoint to $\mathcal{V}(\eta' \wedge \mu')$ and let $\exists X(\theta \wedge \eta \wedge \mu)$ be in normal form relative to $\eta' \wedge \mu'$. Then $\eta' \wedge \mu' \models_{\mathfrak{M}} \exists X(\theta \wedge \eta \wedge \mu)$ iff the three following statements hold:*

1. *$\mathcal{V}(\theta^{-X}) \subseteq \mathcal{C}(\eta')$,*

2. *for every $x \mathrel{\dot\in} p$ in μ^{-X} there is an $x \mathrel{\dot\in} q$ in μ' with $q^{\mathfrak{M}} \subseteq p^{\mathfrak{M}}$,*

3. *$\eta^{-X} \subseteq \theta\eta'$.*

For instance (7) $\models_{\mathfrak{M}}$ (6) holds provided that $q^{\mathfrak{M}} \subseteq p^{\mathfrak{M}}$.

Proof. By clause (1) of Definition 6.4, $\mathcal{V}(\theta^{-X})$ is disjoint from X. Since furthermore $\mathcal{V}(\mu^{-X})$ is disjoint from X, $\eta' \wedge \mu' \models_{\mathfrak{M}} \exists X(\theta \wedge \eta \wedge \mu)$ is equivalent to the conjunction of the three statements

$$\begin{aligned} \eta' \wedge \mu' &\models_{\mathfrak{M}} \theta^{-X} && (8)\\ \eta' \wedge \mu' &\models_{\mathfrak{M}} \mu^{-X} && (9)\\ \eta' \wedge \mu' &\models_{\mathfrak{M}} \exists X(\theta^{X} \wedge \eta^{-X} \wedge \eta^{X} \wedge \mu^{X})\,. && (10)\end{aligned}$$

By Lemma 6.1 and 6.3, (8) is equivalent to condition 1 of the theorem. This relies on clause 2 of Definition 6.4.

Using clause 3 of Definition 6.4 and the fact that X is disjoint to $\mathcal{V}(\eta' \wedge \mu')$, (9) is equivalent to condition 2 of the theorem.

Since θ is X-directed, $\mathcal{V}(\theta\eta')$ is disjoint from X. Hence condition (3) of the theorem implies that $\mathcal{V}(\eta^{-X})$ is disjoint from X, and hence (10) is equivalent to the conjunction of

$$\begin{aligned} \eta' \wedge \mu' &\models_{\mathfrak{M}} \eta^{-X} && (11)\\ \eta' \wedge \mu' &\models_{\mathfrak{M}} \exists X(\theta^{X} \wedge \eta^{X} \wedge \mu^{X}) && (12)\end{aligned}$$

If condition 3 and 1 of the theorem hold, then by (8) we have $\eta' \wedge \mu' \models_{\mathfrak{M}} \eta' \wedge \theta^{-X} \models \eta^{-X}$. By clause (4) of Definition 6.4, the formula $\theta^{X} \wedge \eta^{X} \wedge \mu^{X}$ is in normal form. Hence, by Proposition 4.2, (12) holds.

On the other hand, assume that (10) holds and that $x \doteq f(\bar{y}) \in \eta^{-X}$. If $\theta\eta'$ does not contain an equation for x, or contains an equation $x = g(\bar{z})$ with $g \neq f$, then the same holds for η' by condition 1 of the theorem and since $\theta \wedge \eta'$ is satisfiable in $\mathfrak{M}$. This contradicts our assumption that entailment holds. Hence, there is an $x \doteq f(\bar{z})$ in $\theta\eta'$. Since θ is complete for $\eta \wedge \eta'$, it is also complete for $\eta \wedge \theta\eta'$. Hence $\theta \models \bar{y} \doteq \bar{z}$, that is since θ is an idempotent substitution, $\theta\bar{y} = \theta\bar{z}$. Since $\bar{y}$, $\bar{z}$ are in the codomain of θ and since θ is idempotent, $\bar{y} = \bar{z}$, hence $x = f(\bar{z}) \in \theta\eta'$. □

Next we show how to transform a constraint $\exists X\phi$ into normal form relative to $\eta' \wedge \mu'$. First, we transform the equational part of ϕ using the rules of Figure 5. This rules are equivalence transformations relative to η' in all $\mathcal{IT}$ structures. The rules terminate with

r-decomp	$\dfrac{\theta \wedge \eta}{\bar{y} \doteq \bar{z} \wedge \theta \wedge \eta}$	θ is substitution and $x \doteq f(\bar{y}) \wedge x = f(\bar{z}) \subseteq \eta \wedge \theta\eta'$
r-elim	$\dfrac{x \doteq y \wedge \theta \wedge \eta}{x \doteq y \wedge (\theta \wedge \eta)[x \leftarrow y]}$	$x \neq y, x \in \mathcal{V}(\theta \wedge \eta)$
r-clash	$\dfrac{\theta \wedge \eta}{\bot}$	θ is substitution and $x \doteq f(\bar{y}) \wedge x = g(\bar{z}) \subseteq \eta \wedge \theta\eta'$, $f \neq g$
orient	$\dfrac{x \doteq y \wedge \theta \wedge \eta}{y \doteq x \wedge \theta \wedge \eta}$	$x \notin X, y \in X$

Figure 5: Relative Simplification of Equations.

either $\bot$ or with $\theta \wedge \eta$, such that the clauses 1 and 2 of Definition 6.4 hold. Let μ be the membership part of ϕ. Now we simplify $\theta\mu$ relative to $\eta \wedge \eta'$ as explained in Section 4. Finally, we simplify constraints of the form $x \,\dot{\in}\, p \wedge x \,\dot{\in}\, q$ and check for membership in empty sets, as explained in Section 4. If this does not lead to $\bot$, we arrive at a relative normal form.

As an example of equational simplification, the constraint

$$\exists v(x \doteq v \wedge v \doteq f(v))$$

simplifies relative to $x \doteq f(y) \wedge y \doteq f(z) \wedge z \doteq f(x)$ to

$$\exists v(v \doteq z \wedge x \doteq z \wedge y \doteq z \wedge z \doteq f(z))\,.$$

7 Equations for Intersections

We need an algorithm that extends a deterministic equation system $\mathcal{E}$ containing definitions of p and q by an appropriate definition for $p \,\cap\, q$.

In the terminology of model theory, we will extend the formula $\mathcal{E}$ to a formula $\mathcal{E}'$, such that every $\mathcal{IT}$–model of $\mathcal{E}$ extends conservatively to a $\mathcal{IT}$–model of $\mathcal{E}'$, and such that $x \,\dot{\in}\, p \cap q \models\!\!|_{\mathcal{E}'}\; x \,\dot{\in}\, p \wedge x \,\dot{\in}\, q$ holds for all new set expressions $p \cap q$. Thereby appropriateness of definitions of set expressions inherits from $\mathcal{E}$ to $\mathcal{E}'$.

This extension can be achieved by iterated applications of the non-deterministic rewrite rules in Figure 6 that are easily proven correct in the above sense. Note that the rules maintain determinism of equation systems.

The first rule possibly creates the need for adding further equations to $\mathcal{E}$ in order to have definitions for all set expressions which appear on the right hand sides in $\mathcal{E}$. This completion process can be organised in a terminating manner, by adding $p \cap q$ to $\mathcal{E}$ only under the assumption that p, q and all set expressions on the right hand sides of $\mathcal{E}$ are

int 1	$\dfrac{\mathcal{E}}{\mathcal{E} \cup \{e\}}$	$p \;= f_1(\overline{p_1}) \cup \ldots \cup f_n(\overline{p_n}) \cup f_{n+1}(\overline{p_{n+1}}) \cup \ldots$ in $\mathcal{E}$ $q \;= f_1(\overline{q_1}) \cup \ldots \cup f_n(\overline{q_n}) \cup g_{n+1}(\overline{q_{n+1}}) \cup \ldots$ in $\mathcal{E}$ with $f_j \neq g_k$ for all $j, k \geq n+1$. e is $p \cap q \;=\; f_1(\overline{p_1} \cap \overline{q_1}) \cup \ldots \cup f_n(\overline{p_n} \cap \overline{q_n})$
int 2	$\dfrac{\mathcal{E}}{\mathcal{E} \cup \{e\}}$	if $q = \top$ and $p = \mathit{def}_p$ are contained in $\mathcal{E}$ and e is the equation $p \cap q \;=\; \mathit{def}_p$.

Figure 6: Computation of Intersections

defined in $\mathcal{E}$. More precisely, only binary intersections of set expressions on the right hand side have to be added. These are at most n^2 equations, where n is the number of defined set expressions in $\mathcal{E}$.

For example we can extend an equations system $\mathcal{E}$ containing the above definitions for *Even* and *Nat* with an equation for $\mathit{Even} \cap \mathit{Nat}$. First the first rule adds the equation

$$\mathit{Even} \cap \mathit{Nat} \;=\; \mathit{0} \;\cup\; \mathit{succ}(\mathit{Odd} \cap \mathit{Nat})\,.$$

A further application the same rule adds

$$\mathit{Odd} \cap \mathit{Nat} \;=\; \mathit{succ}(\mathit{Even} \cap \mathit{Nat})\,.$$

8 Deciding the Subset Relation

We will decide the subset relation $p^{\mathcal{A}} \subset q^{\mathcal{A}}$ for $\mathcal{A} = \mathfrak{m}$ or $\mathcal{A} = \mathfrak{M}$ using the memorization technique. Note that this includes a subset check for sets recognized by deterministic top down tree automata as well as for $\mathcal{IT}$ definable sets.

Therefore we define the following fragment of new constraints:

$$\Gamma \;::=\; p \,\dot{\subset}\, q \,|\, \top \,|\, \Gamma \wedge \Gamma'\,.$$

The memorization technique is carried out by rewriting expressions of the form $\Gamma \Box \Gamma'$. We say that Γ_0 simplifies to Γ_1 relative to $\mathcal{E}$ if there is a Γ'_1 such that $\Gamma_0 \Box \top$ rewrites to $\Gamma_1 \Box \Gamma'_1$ relative to $\mathcal{E}$.

Without loss of generality we make two assumptions on $\mathcal{E}$. First we assume that $p^{\mathcal{A}} = \emptyset$ iff $p = \bot$ in $\mathcal{E}$, and that set expressions which are used on the right hand side of $\mathcal{E}$ do not denote the empty set in $\mathcal{A}$. Second, we assume that $p^{\mathcal{A}} = \mathcal{IT}$ iff $p = \top$ in $\mathcal{E}$. Both conditions can be assured for $\mathfrak{M}$ as well as for $\mathfrak{m}$, for finite as well as for infinite signatures.

The rules of the rewrite system are presented in Figure 7.

$unfold1$	$\dfrac{p \,\dot\subset\, q \wedge \Gamma \,\Box\, \Gamma'}{\Gamma_1 \,\wedge\, \Gamma \,\Box\, p \,\dot\subset\, q \,\wedge\, \Gamma'}$	if $p \,\dot\subset\, q$ is not in Γ', the equations $p \;=\; f_1(\overline{p_1}) \,\cup\, \ldots \,\cup\, f_n(\overline{p_n})$ $q \;=\; f_1(\overline{q_1}) \,\cup\, \ldots \,\cup\, f_n(\overline{q_n}) \,\cup\, \ldots$ are in $\mathcal{E}$ and $\Gamma_1 \;=\; \overline{p_1} \,\dot\subset\, \overline{q_1} \wedge \ldots \wedge \overline{p_n} \,\dot\subset\, \overline{q_n}$.
$memo1$	$\dfrac{p \,\dot\subset\, q \wedge \Gamma \,\Box\, \Gamma'}{\Gamma \,\Box\, \Gamma'}$	if $p \,\dot\subset\, q$ is in Γ'.
$clash\,3$	$\dfrac{p \,\dot\subset\, q \wedge \Gamma \,\Box\, \Gamma'}{\bot \,\Box\, \Gamma}$	if $p \;=\; \ldots \,\cup\, f(\bar{p}) \,\cup\, \ldots$ is in $\mathcal{E}$, but the definition of q in $\mathcal{E}$ is not of form $q \;=\; \ldots \,\cup\, f(\bar{q}) \,\cup\, \ldots$ or $q \;=\; \top$.
$clash\,4$	$\dfrac{p \,\dot\subset\, q \wedge \Gamma \,\Box\, \Gamma'}{\bot \,\Box\, \Gamma}$	if $p \;=\; \top$ is in $\mathcal{E}$, but the definition of q is not $q \;=\; \top$.

Figure 7: Deciding the Subset Relation with Memorization

Theorem 8.1 (Correctness and Completeness) *The rewrite system of Figure 7 terminates. If Γ_0 simplifies to Γ_1 relative to $\mathcal{E}$ then $\Gamma_0 \models\!\mid_{\mathcal{A}} \Gamma_1$ holds. A constraint $\Gamma_1 \neq \bot$ that can not be simplified is valid.*

Termination and the last statement are trivial. The clash rules are correct by the assumptions on $\mathcal{E}$. It remains to show that the rules $unfold1$ and $memo1$ are correct. This can be done in analogy to Section 5.

In the following example we apply memorization to prove that $\mathit{Even} \,\dot\subset\, \mathit{Nat}$ holds in $\mathfrak{M}$ as well as in $\mathfrak{m}$. We assume that the signature contains a constructor different from *succ* and *0*:

$$\begin{array}{c}
\underline{\mathit{Even} \,\dot\subset\, \mathit{Nat} \,\Box\, \top}\;{}_{\mathit{unfold1}} \\
\underline{\mathit{Odd} \,\dot\subset\, \mathit{Nat} \,\Box\, \mathit{Even} \,\dot\subset\, \mathit{Nat}}\;{}_{\mathit{unfold1}} \\
\underline{\mathit{Even} \,\dot\subset\, \mathit{Nat} \,\Box\, \mathit{Even} \,\dot\subset\, \mathit{Nat} \wedge \mathit{Odd} \,\subset\, \mathit{Nat}}\;{}_{\mathit{memo1}} \\
\top \,\Box\, \mathit{Even} \,\dot\subset\, \mathit{Nat} \wedge \mathit{Odd} \,\dot\subset\, \mathit{Nat}
\end{array}$$

As a second example we consider $\mathit{Inf} \,\dot\subset\, \mathit{Zero}$ with $\mathcal{E}$ containing $\mathit{Zero} \;=\; 0$. In the case of $\mathfrak{m}$ the definition of Inf is replaced by $\mathit{Inf} \;=\; \bot$ and the subset relation holds. In the case of $\mathfrak{M}$ we do not replace the definition of Inf. The $clash\,3$ rule applies, and the subset relation is refuted.

In order to prove Theorem 8.1 analogously to Section 5, we have to exchange the constraint $x \,\dot\in\, q$ by $p \,\dot\subset\, q$, the Main Lemma by Lemma 8.3 and the assumptions about x in η by assumptions for the definitions of p in $\mathcal{E}$.

The relation 'rewrites to in one $unfold1$ or $memo1$ step' is denoted as $\rhd_{\mathcal{E}}$ and its reflexive and transitive closure by $\rhd_{\mathcal{E}}^{*}$. The following theorem is symmetric to Theorem 5.1 but holds for $\mathfrak{m}$ and $\mathfrak{M}$.

Theorem 8.2 (Correctness) *For every computation:*

$$\Gamma_0 \Box \Gamma_0' \ \rhd_{\mathcal{E}}^{*} \ \Gamma_1 \Box \Gamma_1'$$

the following two invariants are valid:

$$\Gamma_0 \wedge \Gamma_0' \ \mathrel{\models\!\!\!\dashv}_{\mathcal{E}} \ \Gamma_1 \wedge \Gamma_1' \qquad \textit{and} \qquad \Gamma_0' \ \models_{\{\mathfrak{m},\mathfrak{M}\}} \ \Gamma_1 \to \Gamma_1'.$$

Only the second statement requires a proof. Therefore we claim the following lemma that is symmetric to the Main Lemma.

Lemma 8.3 *Let $\mathcal{E}$ be an equation system, $\bar{p}$, $\bar{q}$, $\bar{r}$ and $\bar{s}$ finite sequences of set expressions and $(f_j^k)_k$ finite sequences of derivable constructors. We assume for all j the following equations in $\mathcal{E}$:*

$$r_j \ = \ \bigcup_k f_j^k(\bar{r},\bar{s}) \qquad \textit{and} \qquad p_j \ = \ \bigcup_k f_j^k(\bar{p},\bar{q}) \ \cup \ \ldots$$

In this case the following entailment with respect to the least and the greatest model of $\mathcal{E}$ hold:

$$\models_{\{\mathfrak{m},\mathfrak{M}\}} \ \bar{s} \mathrel{\dot{\subset}} \bar{q} \ \to \ \bar{r} \mathrel{\dot{\subset}} \bar{p}\,.$$

There are models where this lemma is wrong. For instance consider the equation system $p \ = \ f(p)$ and $r \ = \ f(r)$ with the model $r^{\mathcal{A}} \ = \ \{f(f(f(\ldots)))\}$ and $p^{\mathcal{A}} \ = \ \emptyset$. Then $\top \models_{\mathcal{A}} r \mathrel{\dot{\subset}} p$ does not hold.

Proof. We mainly use the representations of $\mathfrak{m}$ and $\mathfrak{M}$ from Lemma 3.1. For the case of $\mathfrak{M}$ we assume $\overline{s^{\mathfrak{M}}} \subset \overline{q^{\mathfrak{M}}}$ and proof by induction $\overline{r^{\mathfrak{M}}} \subset \overline{p^{\mathcal{E}^m(\mathcal{A}_\top)}}$ for each $m \geq 0$.

The induction base $m = 0$ is trivial. The induction step is done by

$$\begin{array}{lcl} p_j^{\mathcal{E}^{m+1}(\mathcal{A}_\top)} & \supset & \cup_k f_j^k(\overline{p^{\mathcal{E}^m(\mathcal{A}_\top)}}, \overline{q^{\mathcal{E}^m(\mathcal{A}_\top)}}) \\ & \supset & \cup_k f_j^k(\overline{r^{\mathfrak{M}}}, \overline{s^{\mathfrak{M}}}) \\ & = & \overline{r^{\mathfrak{M}}} \end{array}$$

for all j. The second inclusion holds by induction hypothesis and assumption:

$$\begin{array}{lclcl} \overline{p^{\mathcal{E}^m(\mathcal{A}_\top)}} & \supset & \overline{r^{\mathfrak{M}}} & & \\ \overline{q^{\mathcal{E}^m(\mathcal{A}_\top)}} & \supset & \overline{q^{\mathfrak{M}}} & \supset & \overline{s^{\mathfrak{M}}} \end{array}$$

For the case of $\mathfrak{m}$ we assume $\overline{s}^{\mathfrak{m}} \subset \overline{q}^{\mathfrak{m}}$. By induction we prove $\overline{r^{\mathcal{E}^m(\mathcal{A}^{\perp})}} \subset \overline{p}^{\mathfrak{m}}$ for each $m \geq 0$.

The induction step is done by

$$\begin{array}{rcl} r_j^{\mathcal{E}^{m+1}(\mathcal{A}_\perp)} & = & \cup_k f_j^k(\overline{r^{\mathcal{E}^m(\mathcal{A}_\perp)}}, \overline{s^{\mathcal{E}^m(\mathcal{A}_\perp)}}) \\ & \subset & \cup_k f_j^k(\overline{p}^{\mathfrak{m}}, \overline{q}^{\mathfrak{m}}) \\ & \subset & p^{\mathfrak{m}} \end{array}$$

for all j. The second inclusion holds by induction hypothesis and assumption:

$$\begin{array}{ccccc} \overline{r^{\mathcal{E}^m(\mathcal{A}_\perp)}} & \subset & \overline{p}^{\mathfrak{m}} & & \\ \overline{s^{\mathcal{E}^m(\mathcal{A}_\perp)}} & \subset & \overline{s}^{\mathfrak{m}} & \subset & \overline{q}^{\mathfrak{m}} \end{array}$$

□

Proof of Theorem 8.2. For simplicity we assume $\Gamma_0' = \top$, which is sufficient to conclude Theorem 8.1.

We call the expression $\bar{s} \dot{\subset} \bar{q} \Box \bar{r} \dot{\subset} \bar{p}$ *appropriate with respect to* $\mathcal{E}$ iff there are finite sequences of derivable constructors $(f_j^k)_k$ such that for all j the equations

$$r_j \;=\; \bigcup_k f_j^k(\bar{r}, \bar{s}) \qquad \text{and} \qquad p_j \;=\; \bigcup_k f_j^k(\bar{p}, \bar{q}) \cup \ldots$$

are in $\mathcal{E}$. $\Gamma_0 \Box \top$ is appropriate even for arbitrary $\mathcal{E}$. We will show that *unfold* and *memo* steps relative to $\mathcal{E}$ maintain appropriateness relative to $\mathcal{E}$. Therefore $\Gamma_1 \Box \Gamma_1'$ is appropriate relative to $\mathcal{E}$ and Lemma 8.3 yields

$$\models_{\{\mathfrak{m},\mathfrak{M}\}} \;\Gamma_1 \rightarrow \Gamma_1' \,.$$

It remains to prove that the *unfold*1 and *memo*1 rule maintain appropriateness. First we consider the *unfold*1 rule. It reduces $r' \dot{\subset} p'$ to $\overline{s}' \dot{\subset} \overline{q}'$ in

$$\bar{s} \dot{\subset} \bar{q} \wedge r' \dot{\subset} p' \Box \bar{r} \dot{\subset} \bar{p} \;\;\rhd_{\mathcal{E}}\; \bar{s} \dot{\subset} \bar{q} \wedge \overline{s}' \dot{\subset} \overline{q}' \Box \bar{r} \dot{\subset} \bar{p} \wedge r' \dot{\subset} p' \,.$$

We will show that there are constructors g_j^k and g'^k with

$$\begin{array}{rclcrcl} r_j & = & \bigcup_k g_j^k(\bar{r}, r', \bar{s}, \overline{s}') & \quad \text{and} \quad & p_j & = & \bigcup_k g_j^k(\bar{p}, p', \bar{q}, \overline{q}') \;\cup\; \ldots \\ r' & = & \bigcup_k g'^k(\bar{r}, r', \bar{s}, \overline{s}') & \quad \text{and} \quad & p' & = & \bigcup_k g'^k(\bar{p}, p', \bar{q}, \overline{q}') \;\cup\; \ldots \end{array}$$

By the application condition of the *unfold* rule, there is a sequence of derivable constructors $(f'^k)_k$ with

$$r' \;=\; \bigcup_k f'^k(\overline{s}') \qquad \text{and} \qquad p' \;=\; \bigcup_k f'^k(\overline{q}') \cup \ldots \;\;.$$

By the appropriateness of the left hand side there are sequences of derivable constructors $(f_j^k)_k$ with

$$r_j \;=\; \bigcup_k f_j^k(\bar{r}, \bar{s}, r') \qquad \text{and} \qquad p_j \;=\; \bigcup_k f_j^k(\bar{p}, \bar{q}, p') \cup \ldots \,.$$

It is easy to find definable constructors g_j^k and g'^k with

$$\begin{array}{rcl} f_j^k(\bar{r}, \bar{s}, r') & = & g_j^k(\bar{r}, r', \bar{s}, \overline{s'}) \\ f'^k(\overline{s'}) & = & g'^k(\bar{r}, r', \bar{s}, \overline{s'}) \,. \end{array}$$

This implies

$$\begin{array}{rcl} f_j^k(\bar{p}, \bar{q}, p) & = & g_j^k(\bar{p}, p, \bar{q}, \overline{q'}) \\ f'^k(\overline{q'}) & = & g'^k(\bar{p}, p', \bar{q}, \overline{q'}) \;, \end{array}$$

which proves the appropriateness of the right hand side. The considerations for the *memo* 1 rule are similar. □

9 Conclusion and Further Work

We have presented a rule-based algorithm which allows for satisfiability and entailment tests of equational and membership constraints. The development of an abstract machine in the style of [ST92b] and the calculation of precise complexity bounds is up to further research.

The constraint system presented here can possibly be extended in various directions. One immediate question is the decidability of the first-order theory of a deterministic equation system with maximal fixpoint solution; *i.e.*, the decidability of *arbitrary* first-order formulae built up from equational and membership constraints. We conjecture a positive answer, encouraged by the decidability result [CD91] for the first-order theory of tree automata.

Another extension of this work could increase the expressive power of the equation systems by weakening the restriction that they be deterministic. The relaxation of the determinism condition will cause problems in the entailment check.

Finally, it will be interesting to apply the methods developed here to the other formalism modeling cyclic data structures: feature trees [ST92a, BS92, AKPS92].

Acknowledgments

We are grateful to Gert Smolka. He inspired this work and contributed ideas during the whole development. We would like to thank Hassan Aït-Kaci and Hubert Comon for stimulating questions and fruitful discussions.

References

[AKP91] Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of LIFE. In Jan Maluszyński and Martin Wirsing, editors, *Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming (Passau, Germany)*, pages 255–274. Springer-Verlag, LNCS 528, August 1991.

[AKPS92] Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. In *Proceedings of the 5th International Conference on Fifth Generation Computer Systems*, pages 1012–1022, Tokyo, Japan, June 1992. ICOT.

[BS92] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. Research Report RR-92-30, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany, September 1992.

[CD91] Hubert Comon and Catherine Delor. Equational formulae with membership constraints. Rapport de Recherche 649, LRI, Université de Paris Sud, Orsay,France, March 1991. To appear in *Information and Computation*, November 1993.

[CKC83] A. Colmerauer, H. Kanoui, and M. Van Caneghem. Prolog, theoretical principles and current trends. *Technology and Science of Informatics*, 2(4):255–292, 1983.

[Col84] A. Colmerauer. Equations and inequations on finite and infinite trees. In *Proceedings of the 2nd International Conference on Fifth Generation Computer Systems*, pages 85–99, 1984.

[Gue89] Irène Guessarian. Some fixpoint techniques in algebraic structures and applications to computer science. In Aït-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 1 (Algebraic Techniques), pages 263–292. Academic Press, 1989.

[HJ90] Seif Haridi and Sverker Janson. Kernel andorra prolog and its computation model. In D. Warren and P. Szeredi, editors, *Logic Programming, 7th International conference*, pages 31–48, Cambridge, June 1990. MIT Press.

[HS88] Markus Höhfeld and Gert Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Postfach 80 08 80, 7000 Stuttgart 80, Germany, October 1988.

[HSW93] Martin Henz, Gert Smolka, and Jörg Würtz. Oz - a programming language for multi-agent systems. In *13th International Joint Conference on Artificial Intelligence*, Chambéry, France, August 1993. Morgan Kaufmann.

[JL87] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of 14th ACM Conference on Principles of Programming Languages*, pages 111–119, Munich, Germany, January 1987. ACM.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[Mah87] Michael J. Maher. Logic semantics for a class of committed-choice programs. In Jean-Louis Lassez, editor, *Proceedings of the Fourth International Conference on Logic Programming*, pages 858–876. MIT Press, 1987.

[Mah88] Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proceedings of the Third Annual Symposium on Logic in Computer Science*, pages 348–357. IEEE Computer Society, 1988.

[NP93] Joachim Niehren and Andreas Podelski. Feature automata and recognizable sets of feature trees. In *Tapssoft*, April 1993.

[Smo93] Gert Smolka. A calculus for higher-order concurrent constraint programming. Research report, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, D-W-6600 Saarbrücken, Germany, 1993. Forthcomming.

[SR91] Vijay Saraswat and Martin Rinard. Semantic foundations of concurrent constraint programming. In *Proceedings of 18th Symposium on Principles of Programming Languages*, pages 333–351, Orlando, FL, January 1991. ACM.

[ST92a] Gert Smolka and Ralf Treinen. Records for logic programming. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 240–254, Washington, USA, 1992. The MIT Press.

[ST92b] Gert Smolka and Ralf Treinen. Records for logic programming. Research Report RR-92-23, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, D-W-6600 Saarbrücken, Germany, August 1992.

[Tho90] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B - Formal Models and Semantics, chapter 4, pages 133–191. Elsevier Science Publishers and The MIT Press, 1990.

[Uri92] Thomás E. Uribe. Sorted unification using set constraints. In D. Kapur, editor, *11th International Conference on Automated Deduction*, Lecture Notes in Computer Science vol. 607, pages 163–177, Saratoga Springs, NY, June 1992. Springer-Verlag.