



Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

**Research
Report**
RR-93-14

Equational and Membership Constraints for Infinite Trees

Joachim Niehren, Andreas Podelski, Ralf Treinen

April 1993

**Deutsches Forschungszentrum für Künstliche Intelligenz
GmbH**

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Friedrich J. Wendl
Director

Equational and Membership Constraints for Infinite Trees

Joachim Niehren, Andreas Podelski, Ralf Treinen

DFKI-RR-93-14

This work may not be copied or reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the author. This work may be used for personal or internal reference only. All rights reserved. © 2003 DFKI-RR-93-14

A short version of this report will appear in the *Proceedings of the Fifth International Conference on Rewriting Techniques and Applications*, June 16-18, 1993, Montreal, Canada.

This work has been supported in part by a grant from The Bundesminister für Forschung und Technologie (FKZ ITW-9105).

© Deutsches Forschungszentrum für Künstliche Intelligenz 1993

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.

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	3
2	Equational and Membership Constraints	6
3	Set Definitions	7
4	Normal Forms of Constraints	9
5	Correctness of Relative Simplification with Memorization	12
6	The Entailment Check	16
7	Equations for Intersections	21
8	Deciding the Subset Relation	22
9	Conclusion and Further Work	26

Abstract

The present system is a constraint solver with equational and membership constraints over a simple theory. It provides a routine and context sensitive library and matching tests and is therefore suitable for the use in constraint consistent programming systems which are based on cyclic data structures. Our set theory device stores greatest (approx.) solutions of regular systems of equations with a deterministic form of union. As the main technical particularity of the algorithm we present a novel memorization technique. We believe that both satisfiability and entailment tests can be implemented in an efficient and practical manner.

Supported by the Graduate Institute of Information Systems and by the Hybrid project at DLR.
Supported by the Bundesministerium für Forschung und Technologie (contract 11W 0109) and by the Expert Working Group (EP contract EP 0025).

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 programming (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 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

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

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 *incomplete* for disjunctive information, while the sort definitions are inherently disjunctive. For instance the conjunction of the atoms $\textit{Nat}(x) \wedge \textit{NatList}(x)$ will not be reduced to \perp 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

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

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

if *Nat*(*x*) **then** ...

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:

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

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 extend the system by new equations in the course of computation.

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

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:

$$x \dot{=} succ(x) \models x \dot{\in} Nat .$$

By unfolding the definition of *Nat*, we obtain a constraint which simplifies to $x \dot{\in} Nat$ relatively to $x \dot{=} succ(x)$.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 $x \dot{\in} Nat$ is simplified to \top , and entailment is proven.

¹This idea is due to Gert Smolka.

²Relative 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 $\phi' \wedge \psi$).

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 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 unrealistic 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.

$\theta ::= x \doteq y \mid \top \mid \theta \wedge \theta'$	$\phi ::= \theta \wedge \eta \wedge \mu \mid \perp$
$\eta ::= x \doteq f(\bar{y}) \mid \top \mid \eta \wedge \eta'$	
$\mu ::= x \dot{\in} q \mid \top \mid \mu \wedge \mu'$	

Figure 1: The fragments of constraints without \exists .

2 Equational and Membership Constraints

We assume a non empty, finite or infinite, one-sorted signature Σ of function symbols f, g, \dots . \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 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 $\bar{x} \dot{\in} \bar{p}$ 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* $x \dot{\in} p$ and \perp . 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 $x \dot{\in} p$ 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 \perp and \doteq get their standard meaning. As usual, we use the notions of existential (resp. universal) closure, $\exists w$ (resp. $\forall w$), 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 $\models_{\mathcal{A}} \exists w$, 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} ($v \models_{\mathcal{A}} w$) 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 $v \models_{\mathcal{A}}^{\phi} w$ for $\phi \models_{\mathcal{A}} \tilde{\forall}(v \rightarrow w)$ and $v \models_{\mathcal{A}}^{\phi} w$ for $\phi \models_{\mathcal{A}} \tilde{\forall}(v \leftrightarrow w)$.

3 Set Definitions

When simplifying membership constraints such as $x \in p \wedge x \in q$, 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:

$$q = f_1(\bar{q}_1) \cup \dots \cup f_n(\bar{q}_n) \quad \text{or} \quad q = \top. \quad (1)$$

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 \perp , 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

$$x \in q \models_{\mathcal{A}} \exists \bar{y}_1 \dots \exists \bar{y}_n \left((x \doteq f(\bar{y}_1) \wedge \bar{y}_1 \in \bar{q}_1) \vee \dots \vee (x \doteq f(\bar{y}_n) \wedge \bar{y}_n \in \bar{q}_n) \right)$$

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

An equation system \mathcal{E} can be considered as a syntactic characterization of its *IT-models*. Therefore, we identify \mathcal{E} with its *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:

$$x \in p \cap q \models_{\mathcal{E}} x \in p \wedge x \in q.$$

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:

$$x \in q \models_{\mathcal{E}}^{\eta} \bar{y} \in \bar{q}_i.$$

In the rest of this section we discuss computational properties of the greatest *IT-model* \mathfrak{M} and the least *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}_\perp . 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:

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

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}_α , 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 $\mathfrak{m} = \bigcup_{m=0}^{\infty} \mathcal{E}^m(\mathcal{A}_\perp)$ and $\mathfrak{M} = \bigcap_{m=0}^{\infty} \mathcal{E}^m(\mathcal{A}_\top)$.

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 \perp 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 $\phi \models_{\mathfrak{M}} \top$ 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 $x \doteq x \wedge y \in \text{Nat}$. Then $\mathcal{C}(\phi) = \emptyset$, if the signature consists of $\{\text{succ}, 0\}$ 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 $x \in p$ 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 4.2 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 $\phi \models_{\mathfrak{M}} \psi$.

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 4.3 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} :

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

<i>decomp</i>	$\frac{x \doteq f(\bar{y}) \wedge x \doteq f(\bar{z}) \wedge \sigma}{x \doteq f(\bar{y}) \wedge \bar{y} \doteq \bar{z} \wedge \sigma}$	
<i>elim</i>	$\frac{x \doteq y \wedge \sigma}{x \doteq y \wedge \sigma [x \leftarrow y]}$	if $x \neq y$ and $x \in \mathcal{V}(\sigma)$
<i>clash 1</i>	$\frac{x \doteq f(\bar{y}) \wedge x \doteq g(\bar{z}) \wedge \sigma}{\perp}$	if $f \neq g$

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

<i>unfold</i>	$\frac{x \dot{\in} q \wedge \mu \square \mu'}{\bar{y} \dot{\in} \bar{p} \wedge \mu \square x \dot{\in} q \wedge \mu'}$	if $x \dot{\in} q$ is not in μ' , $q = \dots \cup f(\bar{p}) \cup \dots$ is in \mathcal{E} , and $x \doteq f(\bar{y})$ is in η .
<i>memo</i>	$\frac{x \dot{\in} q \wedge \mu \square \mu'}{\mu \square \mu'}$	if $x \dot{\in} q$ is in μ'
<i>clash 2</i>	$\frac{x \dot{\in} q \wedge \mu \square \mu'}{\perp \square \mu'}$	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.

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 $\mu_1 \square \mu'_1$ 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 \square 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.

<i>intersect</i>	$\frac{x \dot{\in} p \wedge x \dot{\in} q}{x \dot{\in} p \cap q}$	
<i>empty</i>	$\frac{x \dot{\in} p \wedge \mu}{\perp}$	if $p = \perp$ is in \mathcal{E}

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

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

$$\frac{\frac{x \dot{\in} Inf \square \top}{x \dot{\in} Inf} \text{ unfold}}{\top \square x \dot{\in} Inf} \text{ memo}$$

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.

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

Theorem 4.4 *When started with the constraint ϕ , the above algorithm terminates with \perp 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*, *NatList*, *Even* and *Odd* from the introduction. We compute a normal form of

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

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

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

By applying θ to the membership part we get

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

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

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

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

$$Nat \cap Even \cap Odd = succ(Nat \cap Even \cap Odd)$$

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

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

This normal form is satisfiable since $\infty \in (Nat \cap Even \cap Odd)^{\mathfrak{M}}$. Note that we could replace $z \dot{\in} Nat \cap Even \cap Odd$ by the \mathfrak{M} -equivalent constraint $z \doteq succ(z)$. 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 $\mu \square \mu'$ will be denoted by $\triangleright_{\eta, \mathcal{E}}$ and its reflexive transitive closure by $\triangleright_{\eta, \mathcal{E}}^*$.

Roughly speaking, the following theorem states that the symbol \square 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 5.5 *For each computation*

$$\mu_0 \square \mu'_0 \triangleright_{\eta, \mathcal{E}}^* \mu_1 \square \mu'_1$$

the following two statements are invariants:

$$\mu_0 \wedge \mu'_0 \models_{\mathcal{E}}^{\eta} \mu_1 \wedge \mu'_1 \quad \text{and} \quad \mu'_0 \models_{\mathfrak{M}}^{\eta} \mu_1 \rightarrow \mu'_1.$$

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

Corollary 5.6 (Correctness) *If μ_0 simplifies to μ_1 by memorization relative to η and \mathcal{E} then $\mu_0 \models_{\mathfrak{M}}^{\mathcal{E}} \mu_1$ holds.*

Proof. By definition there is μ'_1 with $\mu_0 \sqcap \top \triangleright_{\eta, \mathcal{E}}^* \mu_1 \sqcap \mu'_1$. The above theorem implies

$$\mu_0 \wedge \top \models_{\mathcal{E}}^{\eta} \mu_1 \wedge \mu'_1 \models_{\mathfrak{M}}^{\eta} \mu_1.$$

□

Theorem 5.5 can be proven with the help of Lemma 5.7, 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 $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ 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

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

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 5.7 (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*

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

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

$$\models_{\mathfrak{M}}^{\eta} x_j \doteq f_j(\bar{x}, \bar{y})$$

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

$$\models_{\mathfrak{M}}^{\eta} \bar{y} \dot{\in} \bar{q} \rightarrow \bar{x} \dot{\in} \bar{p}.$$

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 $\top \models_{\mathfrak{M}}^{\eta} x_1 \dot{\in} p_1 \wedge x_2 \dot{\in} p_2$. 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

$$\alpha(x_j) = f_j^{TT}(\overline{\alpha(x)}, \overline{\alpha(y)}) \quad \text{and} \quad \alpha(y_j) \in q_j^{\mathfrak{m}}$$

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

$$\begin{aligned} \alpha(x_j) &\in p_j^{\mathcal{E}^{m+1}(\mathcal{A}_\top)} \\ &= \dots \cup f_j^{TT}(p_j^{\mathcal{E}^m(\mathcal{A}_\top)}, q_j^{\mathcal{E}^m(\mathcal{A}_\top)}) \cup \dots \end{aligned}$$

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

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

and by assumption

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

Proof of Theorem 5.5. For simplicity we assume $\mu'_0 = \top$, which is sufficient to conclude Corollary 5.6.

We call the expression $\bar{y} \dot{\in} \bar{q} \square \bar{x} \dot{\in} \bar{p}$ *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

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

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

$$\models^n x_j \dot{\in} f_j(\bar{x}, \bar{y}).$$

$\mu_0 \square \top$ 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 $\mu_1 \square \mu'_1$ is appropriate relative to η and \mathcal{E} . The Main Lemma yields

$$\models_{\mathfrak{m}}^{\eta} \mu_1 \rightarrow \mu'_1.$$

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

$$\bar{y} \dot{\in} \bar{q} \wedge x' \dot{\in} p' \sqcap \bar{x} \dot{\in} \bar{p} \triangleright_{n,\varepsilon} \bar{y} \dot{\in} \bar{q} \wedge \bar{y}' \dot{\in} \bar{q}' \sqcap \bar{x} \dot{\in} \bar{p} \wedge x' \dot{\in} p'.$$

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

$$\begin{aligned} p_j &= \dots \cup g_j(\bar{p}, p', \bar{q}, \bar{q}') \cup \dots & \text{and} & \models^n x_j \dot{=} g_j(\bar{x}, x_0, \bar{y}, \bar{y}'), \\ p' &= \dots \cup g'(\bar{p}, p', \bar{q}, \bar{q}') \cup \dots & \text{and} & \models^n x' \dot{=} g'(\bar{x}, x', \bar{y}, \bar{y}'). \end{aligned}$$

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

$$p' = \dots \cup f'(\bar{q}') \cup \dots \quad \text{and} \quad \models^n x' \dot{=} f'(\bar{y}').$$

By the appropriateness of the left hand side of $\triangleright_{n,\varepsilon}$ there are derived constructors \bar{f} with

$$p_j = \dots \cup \bar{f}_j(\bar{p}, \bar{q}, p') \cup \dots \quad \text{and} \quad \models^n x_j \dot{=} \bar{f}_j(\bar{x}, \bar{y}, x')$$

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

$$f_j(\bar{x}, \bar{y}, x') = g_j(\bar{x}, x', \bar{y}, \bar{y}') \quad \text{and} \quad f'(\bar{y}') = g'(\bar{x}, x', \bar{y}, \bar{y}')$$

This implies

$$\bar{f}_j(\bar{p}, \bar{q}, p') = g_j(\bar{p}, \bar{q}, \bar{q}') \quad \text{and} \quad f'(\bar{q}') = g'(\bar{p}, p', \bar{q}, \bar{q}')$$

and proves the appropriateness of the right hand side.

For the second case we consider an application of the memo rules erasing $x_0 \dot{\in} p_0$:

$$\bar{y} \dot{\in} \bar{q} \wedge x_0 \dot{\in} p_0 \sqcap \bar{x} \dot{\in} \bar{p} \wedge x_0 \dot{\in} p_0 \triangleright_{n,\varepsilon} \bar{y} \dot{\in} \bar{q} \sqcap \bar{x} \dot{\in} \bar{p} \wedge x_0 \dot{\in} p_0.$$

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

$$p_j = \dots \cup \bar{f}_j(\bar{p}, p_0, \bar{q}, p_0) \cup \dots \quad \text{and} \quad \models^n x_j \dot{=} \bar{f}_j(\bar{x}, x_0, \bar{y}, x_0)$$

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

$$\bar{f}_j(\bar{x}, x_0, \bar{y}, x_0) = g_j(\bar{x}, x_0, \bar{y})$$

for all $j \geq 0$. Therefore

$$\bar{f}_j(\bar{p}, p_0, \bar{q}, p_0) = g_j(\bar{p}, p_0, \bar{q}).$$

This shows the appropriateness of the right hand side. \square

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

$$\exists X' \phi' \models_{\mathfrak{M}} \exists X \phi \quad (2)$$

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 4.4 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 $x \dot{\in} p$ 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 $x \dot{\in} p$ 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 $x \dot{\in} p$, 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 $x \dot{\in} p$ is in \mathfrak{M} equivalent to $\exists y (x \dot{=} f(y) \wedge y \dot{=} g(x))$.
- A membership constraint $x \dot{\in} p$, where the definition of p is of the form $p = f(\bar{q})$ and where $p^{\mathfrak{M}}$ is not singleton, is replaced by $\exists \bar{y} (x \dot{=} f(\bar{y}) \wedge \bar{y} \dot{\in} \bar{q})$.

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 5.7, $p^{\mathfrak{M}}$ 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

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

into the existentially quantified branching normal form

$$\exists y \exists z (x \dot{=} \text{succ}(x) \wedge y \dot{=} \text{succ}(z) \wedge z \dot{\in} \text{Even}).$$

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

Lemma 6.8 *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:

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

We can now express an important property of normal forms which is in some sense a counterpart to Proposition 4.3. This lemma has been given in [Mah88] as an axiom of infinite trees.

Lemma 6.9 *For every η we have $\models_{\mathfrak{M}} \tilde{\forall} \hat{\exists} \mathcal{C}(\eta) \eta$.*

Lemma 6.10 (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 4.3 to get a solution of η' which does not satisfy θ .

Let $\mathcal{V}(\theta) \subseteq \mathcal{C}(\eta')$. By Lemma (6.9)

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

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 $\theta\eta'$ 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 4.3

$$\models_{\mathfrak{M}} \tilde{\forall} \exists \mathcal{C}(\theta\eta')(\theta \wedge \theta\eta') \models \tilde{\forall} \exists \mathcal{C}(\theta\eta')(\theta^X \wedge \eta') \models \tilde{\forall} \exists \mathcal{C}(\eta')(\theta' \wedge \eta') \quad (5)$$

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). \square

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 $\phi \models_{\mathfrak{M}} \phi^X \wedge \phi^{-X}$.

Definition 6.11 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,

$$\exists v(v \dot{=} z \wedge x \dot{=} y \wedge y \dot{=} f(y) \wedge z \dot{=} p) \quad (6)$$

is in normal form relative to

$$x \dot{=} f(y) \wedge y \dot{=} f(x) \wedge w \dot{=} h(z) \wedge z \dot{=} q. \quad (7)$$

This does not hold if we drop $x \dot{=} y$ from (6), since then clause 2 of Definition 6.11 is violated.

Theorem 6.12 (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 $x \dot{=} p$ in μ^{-X} there is an $x \dot{=} q$ 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 6.11, $\mathcal{V}(\theta^{-X})$ 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

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \theta^{-X} \quad (8)$$

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \mu^{-X} \quad (9)$$

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \exists X(\theta^X \wedge \eta^{-X} \wedge \eta^X \wedge \mu^X). \quad (10)$$

By Lemma 6.8 and 6.10, (8) is equivalent to condition 1 of the theorem. This relies on clause 2 of Definition 6.11.

<i>r-decomp</i>	$\frac{\theta \wedge \eta}{\bar{y} \doteq \bar{z} \wedge \theta \wedge \eta}$	θ is substitution and $x \doteq f(\bar{y}) \wedge x = f(\bar{z}) \subseteq \eta \wedge \theta\eta'$
<i>r-clim</i>	$\frac{x \doteq y \wedge \theta \wedge \eta}{x \doteq y \wedge (\theta \wedge \eta)[x \leftarrow y]}$	$x \neq y, x \in \mathcal{V}(\theta \wedge \eta)$
<i>r-clash</i>	$\frac{\theta \wedge \eta}{\perp}$	θ is substitution and $x \doteq f(\bar{y}) \wedge x = g(\bar{z}) \subseteq \eta \wedge \theta\eta', f \neq g$
<i>orient</i>	$\frac{x \doteq y \wedge \theta \wedge \eta}{y \doteq x \wedge \theta \wedge \eta}$	$x \notin X, y \in X$

Figure 5: Relative Simplification of Equations.

Using clause 3 of Definition 6.11 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

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \eta^{-X} \quad (11)$$

$$\eta' \wedge \mu' \models_{\mathfrak{M}} \exists X(\theta^X \wedge \eta^X \wedge \mu^X) \quad (12)$$

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 6.11, the formula $\theta^X \wedge \eta^X \wedge \mu^X$ is in normal form. Hence, by Proposition 4.3, (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 $x = f(\bar{z}) \in \theta\eta'$. \square

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. These rules are equivalence transformations relative to η' in all \mathcal{IT} structures. The rules terminate with either \perp or with $\theta \wedge \eta$, such that the clauses 1 and 2 of Definition 6.11 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 $x \doteq p \wedge x \doteq q$ and check for membership in empty sets, as explained in Section 4. If this does not lead to \perp , we arrive at a relative normal form.

<i>int 1</i>	$\frac{\mathcal{E}}{\mathcal{E} \cup \{e\}}$	$p = f_1(\bar{p}_1) \cup \dots \cup f_n(\bar{p}_n) \cup f_{n+1}(\bar{p}_{n+1}) \cup \dots$ in \mathcal{E} $q = f_1(\bar{q}_1) \cup \dots \cup f_n(\bar{q}_n) \cup g_{n+1}(\bar{q}_{n+1}) \cup \dots$ in \mathcal{E} with $f_j \neq g_k$ for all $j, k \geq n + 1$. e is $p \cap q = f_1(\bar{p}_1 \cap \bar{q}_1) \cup \dots \cup f_n(\bar{p}_n \cap \bar{q}_n)$
<i>int 2</i>	$\frac{\mathcal{E}}{\mathcal{E} \cup \{e\}}$	if $q = \top$ and $p = def_p$ are contained in \mathcal{E} and e is the equation $p \cap q = def_p$.

Figure 6: Computation of Intersections

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 $x \in p \cap q \models_{\mathcal{E}'} x \in p \wedge x \in q$ 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 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 *Even* \cap *Nat*. First the first rule adds the

equation

$$Even \cap Nat = \emptyset \cup succ(Odd \cap Nat).$$

A further application the same rule adds

$$Odd \cap Nat = succ(Even \cap Nat).$$

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:

$$\Gamma ::= p \dot{\subset} q \mid \top \mid \Gamma \wedge \Gamma'.$$

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

Without loss of generality we make two assumptions on \mathcal{E} . First we assume that $p^{\mathcal{A}} = \emptyset$ iff $p = \perp$ 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.

Theorem 8.13 (Correctness and Completeness) *The rewrite system of Figure 7 terminates. If Γ_0 simplifies to Γ_1 relative to \mathcal{E} then $\Gamma_0 \models_{\mathcal{A}} \Gamma_1$ holds. A constraint $\Gamma_1 \neq \perp$ 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 $Even \dot{\subset} Nat$ holds in \mathfrak{M} as well as in \mathfrak{m} . We assume that the signature contains a constructor different from *succ* and \emptyset :

$$\frac{\frac{Even \dot{\subset} Nat \sqcap \top}{Odd \dot{\subset} Nat \sqcap Even \dot{\subset} Nat} \text{unfold1}}{Even \dot{\subset} Nat \sqcap Even \dot{\subset} Nat \wedge Odd \subset Nat} \text{memo1}}{\top \sqcap Even \dot{\subset} Nat \wedge Odd \dot{\subset} Nat}$$

<i>unfold1</i>	$\frac{p \dot{\subset} q \wedge \Gamma \sqsupset \Gamma'}{\Gamma_1 \wedge \Gamma \sqsupset p \dot{\subset} q \wedge \Gamma'}$	if $p \dot{\subset} q$ is not in Γ' , the equations $p = f_1(\bar{p}_1) \cup \dots \cup f_n(\bar{p}_n)$ $q = f_1(\bar{q}_1) \cup \dots \cup f_n(\bar{q}_n) \cup \dots$ are in \mathcal{E} and $\Gamma_1 = \bar{p}_1 \dot{\subset} \bar{q}_1 \wedge \dots \wedge \bar{p}_n \dot{\subset} \bar{q}_n$.
<i>memo1</i>	$\frac{p \dot{\subset} q \wedge \Gamma \sqsupset \Gamma'}{\Gamma \sqsupset \Gamma'}$	if $p \dot{\subset} q$ is in Γ' .
<i>clash3</i>	$\frac{p \dot{\subset} q \wedge \Gamma \sqsupset \Gamma'}{\perp \sqsupset \Gamma}$	if $p = \dots \cup f(\bar{p}) \cup \dots$ is in \mathcal{E} , but the definition of q in \mathcal{E} is not of form $q = \dots \cup f(\bar{q}) \cup \dots$ or $q = \top$.
<i>clash4</i>	$\frac{p \dot{\subset} q \wedge \Gamma \sqsupset \Gamma'}{\perp \sqsupset \Gamma}$	if $p = \top$ is in \mathcal{E} , but the definition of q is not $q = \top$.

Figure 7: Deciding the Subset Relation with Memorization

As a second example we consider $Inf \dot{\subset} Zero$ with \mathcal{E} containing $Zero = 0$. In the case of \mathfrak{m} the definition of Inf is replaced by $Inf = \perp$ and the subset relation holds. In the case of \mathfrak{M} we do not replace the definition of Inf . The *clash3* rule applies, and the subset relation is refuted.

In order to prove Theorem 8.13 analogously to Section 5, we have to exchange the constraint $x \dot{\subset} q$ by $p \dot{\subset} q$, the Main Lemma by Lemma 8.15 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 $\triangleright_{\mathcal{E}}$ and its reflexive and transitive closure by $\triangleright_{\mathcal{E}}^*$. The following theorem is symmetric to Theorem 5.5 but holds for \mathfrak{m} and \mathfrak{M} .

Theorem 8.14 (Correctness) *For every computation:*

$$\Gamma_0 \sqsupset \Gamma'_0 \triangleright_{\mathcal{E}}^* \Gamma_1 \sqsupset \Gamma'_1$$

the following two invariants are valid:

$$\Gamma_0 \wedge \Gamma'_0 \models_{\mathcal{E}} \Gamma_1 \wedge \Gamma'_1 \quad \text{and} \quad \Gamma'_0 \models_{\{\mathfrak{m}, \mathfrak{M}\}} \Gamma_1 \rightarrow \Gamma'_1.$$

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

Lemma 8.15 *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} :*

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

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

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

There are models where this lemma is wrong. For instance consider the equation system $p = f(p)$ and $r = f(r)$ with the model $r^{\mathcal{A}} = \{f(f(f(\dots)))\}$ and $p^{\mathcal{A}} = \emptyset$. Then $\top \models_{\mathcal{A}} r \dot{\subset} p$ 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 $\bar{s}^{\mathfrak{M}} \subset \bar{q}^{\mathfrak{M}}$ and proof by induction $\bar{r}^{\mathfrak{M}} \subset \bar{p}^{\mathfrak{M}(\mathcal{A}_\top)}$ for each $m \geq 0$.

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

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

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

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

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

The induction step is done by

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

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

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

□

Proof of Theorem 8.14. For simplicity we assume $\Gamma'_0 = \top$, which is sufficient to conclude Theorem 8.13.

We call the expression $\bar{s} \dot{\subset} \bar{q} \square \bar{r} \dot{\subset} \bar{p}$ *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

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

are in \mathcal{E} . $\Gamma_0 \square \top$ 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 $\Gamma_1 \square \Gamma'_1$ is appropriate relative to \mathcal{E} and Lemma 8.15 yields

$$\models_{\{m, \mathfrak{M}\}} \Gamma_1 \rightarrow \Gamma'_1.$$

It remains to prove that the *unfold1* and *memo1* rule maintain appropriateness. First we consider the *unfold1* rule. It reduces $r' \dot{\subset} p'$ to $\bar{s}' \dot{\subset} \bar{q}'$ in

$$\bar{s} \dot{\subset} \bar{q} \wedge r' \dot{\subset} p' \square \bar{r} \dot{\subset} \bar{p} \triangleright_{\mathcal{E}} \bar{s}' \dot{\subset} \bar{q}' \wedge \bar{s}' \dot{\subset} \bar{q}' \square \bar{r} \dot{\subset} \bar{p} \wedge r' \dot{\subset} p'.$$

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

$$\begin{aligned} r_j &= \bigcup_k g_j^k(\bar{r}, r', \bar{s}, \bar{s}') & \text{and} & & p_j &= \bigcup_k g_j^k(\bar{p}, p', \bar{q}, \bar{q}') \cup \dots \\ r' &= \bigcup_k g'^k(\bar{r}, r', \bar{s}, \bar{s}') & \text{and} & & p' &= \bigcup_k g'^k(\bar{p}, p', \bar{q}, \bar{q}') \cup \dots \end{aligned}$$

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

$$r' = \bigcup_k f'^k(\bar{s}') \quad \text{and} \quad p' = \bigcup_k f'^k(\bar{q}') \cup \dots$$

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

$$r_j = \bigcup_k f_j^k(\bar{r}, \bar{s}, r') \quad \text{and} \quad p_j = \bigcup_k f_j^k(\bar{p}, \bar{q}, p') \cup \dots$$

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

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

This implies

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

which proves the appropriateness of the right hand side. The considerations for the *memo1* rule are similar. \square

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

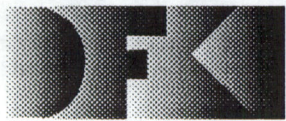
We are grateful to Gert Smolka. He inspired this work and contributed ideas during the whole development. We would like to thank Hassan Aït-Kaci and Hubert Comon for stimulating questions and fruitful discussions.

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 *Tapsoft*, 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. Forthcoming.
- [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.



**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

**DFKI
-Bibliothek-
PF 2080
D-6750 Kaiserslautern
FRG**

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse bezogen werden.

Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Research Reports

RR-92-21

Jörg-Peter Mohren, Jürgen Müller
Representing Spatial Relations (Part II) -The Geometrical Approach
25 pages

RR-92-22

Jörg Würtz: Unifying Cycles
24 pages

RR-92-23

Gert Smolka, Ralf Treinen:
Records for Logic Programming
38 pages

RR-92-24

Gabriele Schmidt: Knowledge Acquisition from Text in a Complex Domain
20 pages

RR-92-25

Franz Schmalhofer, Ralf Bergmann, Otto Kühn, Gabriele Schmidt: Using integrated knowledge acquisition to prepare sophisticated expert plans for their re-use in novel situations
12 pages

RR-92-26

Franz Schmalhofer, Thomas Reinartz, Bidjan Tschaitshian: Intelligent documentation as a catalyst for developing cooperative knowledge-based systems
16 pages

RR-92-27

Franz Schmalhofer, Jörg Thoben: The model-based construction of a case-oriented expert system
18 pages

RR-92-29

Zhaohui Wu, Ansgar Bernardi, Christoph Klauck: Skeletal Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

DFKI Publications

The following DFKI publications or the list of all published papers so far can be ordered from the above address.

The reports are distributed free of charge except if otherwise indicated.

RR-92-30

Rolf Backofen, Gert Smolka:
A Complete and Recursive Feature Theory
32 pages

RR-92-31

Wolfgang Wahlster:
Automatic Design of Multimodal Presentations
17 pages

RR-92-33

Franz Baader: Unification Theory
22 pages

RR-92-34

Philipp Hanschke: Terminological Reasoning and Partial Inductive Definitions
23 pages

RR-92-35

Manfred Meyer:
Using Hierarchical Constraint Satisfaction for Lathe-Tool Selection in a CIM Environment
18 pages

RR-92-36

Franz Baader, Philipp Hanschke:
Extensions of Concept Languages for a Mechanical Engineering Application
15 pages

RR-92-37

Philipp Hanschke: Specifying Role Interaction in Concept Languages
26 pages

RR-92-38

Philipp Hanschke, Manfred Meyer:
An Alternative to H-Subsumption Based on Terminological Reasoning
9 pages

RR-92-40

Philipp Hanschke, Knut Hinkelmann: Combining Terminological and Rule-based Reasoning for Abstraction Processes
17 pages

RR-92-41

Andreas Lux: A Multi-Agent Approach towards Group Scheduling
32 pages

RR-92-42

John Nerbonne:
A Feature-Based Syntax/Semantics Interface
19 pages

RR-92-43

Christoph Klauck, Jakob Mauss: A Heuristic driven Parser for Attributed Node Labeled Graph Grammars and its Application to Feature Recognition in CIM
17 pages

RR-92-44

Thomas Rist, Elisabeth André: Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP
15 pages

RR-92-45

Elisabeth André, Thomas Rist: The Design of Illustrated Documents as a Planning Task
21 pages

RR-92-46

Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, Wolfgang Wahlster: WIP: The Automatic Synthesis of Multimodal Presentations
19 pages

RR-92-47

Frank Bomarius: A Multi-Agent Approach towards Modeling Urban Traffic Scenarios
24 pages

RR-92-48

Bernhard Nebel, Jana Koehler: Plan Modifications versus Plan Generation: A Complexity-Theoretic Perspective
15 pages

RR-92-49

Christoph Klauck, Ralf Legleitner, Ansgar Bernardi: Heuristic Classification for Automated CAPP
15 pages

RR-92-50

Stephan Busemann:
Generierung natürlicher Sprache
61 Seiten

RR-92-51

Hans-Jürgen Bürckert, Werner Nutt: On Abduction and Answer Generation through Constrained Resolution
20 pages

RR-92-52

Mathias Bauer, Susanne Biundo, Dietmar Dengler, Jana Koehler, Gabriele Paul: PHI - A Logic-Based Tool for Intelligent Help Systems
14 pages

RR-92-54

Harold Boley: A Direkt Semantic Characterization of RELFUN
30 pages

RR-92-55

John Nerbonne, Joachim Laubsch, Abdel Kader Diagne, Stephan Oepen: Natural Language Semantics and Compiler Technology
17 pages

RR-92-56

Armin Laux: Integrating a Modal Logic of Knowledge into Terminological Logics
34 pages

RR-92-58

Franz Baader, Bernhard Hollunder: How to Prefer More Specific Defaults in Terminological Default Logic
31 pages

RR-92-59

Karl Schlechta and David Makinson: On Principles and Problems of Defeasible Inheritance
13 pages

RR-92-60

Karl Schlechta: Defaults, Preorder Semantics and Circumscription
19 pages

RR-93-02

Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profillich, Thomas Rist: Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03

Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profillich, Enrico Franconi: An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-04

Christoph Klauck, Johannes Schwagereit: GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-05

Franz Baader, Klaus Schulz: Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-06

Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: On Skolemization in Constrained Logics
40 pages

RR-93-07

Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux: Concept Logics with Function Symbols
36 pages

RR-93-08

Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer: COLAB: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-09

Philipp Hanschke, Jörg Würtz: Satisfiability of the Smallest Binary Program
8 Seiten

RR-93-10

Martin Buchheit, Francesco M. Donini, Andrea Schaerf: Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-11

Bernhard Nebel, Hans-Juergen Buerckert: Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-12

Pierre Sablayrolles: A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-13

Franz Baader, Karl Schlechta: A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-14

Joachim Niehren, Andreas Podelski, Ralf Treinen: Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-15

Frank Berger, Thomas Fehle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster: PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-16

Gert Smolka, Martin Henz, Jörg Würtz: Object-Oriented Concurrent Constraint Programming in Oz
17 pages

DFKI Technical Memos**TM-91-12**

Klaus Becker, Christoph Klauck, Johannes Schwagereit: FEAT-PATR: Eine Erweiterung des D-PATR zur Feature-Erkennung in CAD/CAM
33 Seiten

TM-91-13

Knut Hinkelmann: Forward Logic Evaluation: Developing a Compiler from a Partially Evaluated Meta Interpreter
16 pages

TM-91-14

Rainer Bleisinger, Rainer Hoch, Andreas Dengel: ODA-based modeling for document analysis
14 pages

TM-91-15

Stefan Busemann: Prototypical Concept Formation An Alternative Approach to Knowledge Representation
28 pages

TM-92-01

Lijuan Zhang: Entwurf und Implementierung eines Compilers zur Transformation von Werkstückrepräsentationen
34 Seiten

TM-92-02

Achim Schupeta: Organizing Communication and Introspection in a Multi-Agent Blocksworld
32 pages

TM-92-03

Mona Singh: A Cognitive Analysis of Event Structure
21 pages

TM-92-04

Jürgen Müller, Jörg Müller, Markus Pischel, Ralf Scheidhauer: On the Representation of Temporal Knowledge
61 pages

TM-92-05

Franz Schmalhofer, Christoph Globig, Jörg Thoben: The refitting of plans by a human expert
10 pages

TM-92-06

Otto Kühn, Franz Schmalhofer: Hierarchical skeletal plan refinement: Task- and inference structures
14 pages

TM-92-08

Anne Kilger: Realization of Tree Adjoining Grammars with Unification
27 pages

TM-93-01

Otto Kühn, Andreas Birk: Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

DFKI Documents**D-92-11**

Kerstin Becker: Möglichkeiten der Wissensmodellierung für technische Diagnose-Expertensysteme
92 Seiten

D-92-12

Otto Kühn, Franz Schmalhofer, Gabriele Schmidt: Integrated Knowledge Acquisition for Lathe Production Planning: a Picture Gallery (Integrierte Wissensakquisition zur Fertigungsplanung für Drehteile: eine Bildergalerie)
27 pages

D-92-13

Holger Peine: An Investigation of the Applicability of Terminological Reasoning to Application-Independent Software-Analysis
55 pages

D-92-14

Johannes Schwagereit: Integration von Graph-Grammatiken und Taxonomien zur Repräsentation von Features in CIM
98 Seiten

D-92-15

DFKI Wissenschaftlich-Technischer Jahresbericht 1991
130 Seiten

D-92-16

Judith Engelkamp (Hrsg.): Verzeichnis von Softwarekomponenten für natürlichsprachliche Systeme
189 Seiten

D-92-17

Elisabeth André, Robin Cohen, Winfried Graf, Bob Kass, Cécile Paris, Wolfgang Wahlster (Eds.): UM92: Third International Workshop on User Modeling, Proceedings
254 pages

Note: This document is available only for a nominal charge of 25 DM (or 15 US-\$).

D-92-18

Klaus Becker: Verfahren der automatisierten Diagnose technischer Systeme
109 Seiten

D-92-19

Stefan Ditttrich, Rainer Hoch: Automatische, Deskriptor-basierte Unterstützung der Dokumentanalyse zur Fokussierung und Klassifizierung von Geschäftsbriefen
107 Seiten

D-92-21

Anne Schauder: Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars
57 pages

D-92-22

Werner Stein: Indexing Principles for Relational Languages Applied to PROLOG Code Generation
80 pages

D-92-23

Michael Herfert: Parsen und Generieren der Prolog-artigen Syntax von RELFUN
51 Seiten

D-92-24

Jürgen Müller, Donald Steiner (Hrsg.): Kooperierende Agenten
78 Seiten

D-92-25

Martin Buchheit: Klassische Kommunikations- und Koordinationsmodelle
31 Seiten

D-92-26

Enno Tolzmann: Realisierung eines Werkzeugauswahlmoduls mit Hilfe des Constraint-Systems CONTAX
28 Seiten

D-92-27

Martin Harm, Knut Hinkelmann, Thomas Labisch: Integrating Top-down and Bottom-up Reasoning in COLAB
40 pages

D-92-28

Klaus-Peter Gores, Rainer Bleisinger: Ein Modell zur Repräsentation von Nachrichtentypen
56 Seiten

D-93-01

Philipp Hanschke, Thom Frühwirth: Terminological Reasoning with Constraint Handling Rules
12 pages

D-93-02

Gabriele Schmidt, Frank Peters, Gernod Laufkötter: User Manual of COKAM+
23 pages

D-93-03

Stephan Busemann, Karin Harbusch(Eds.): DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

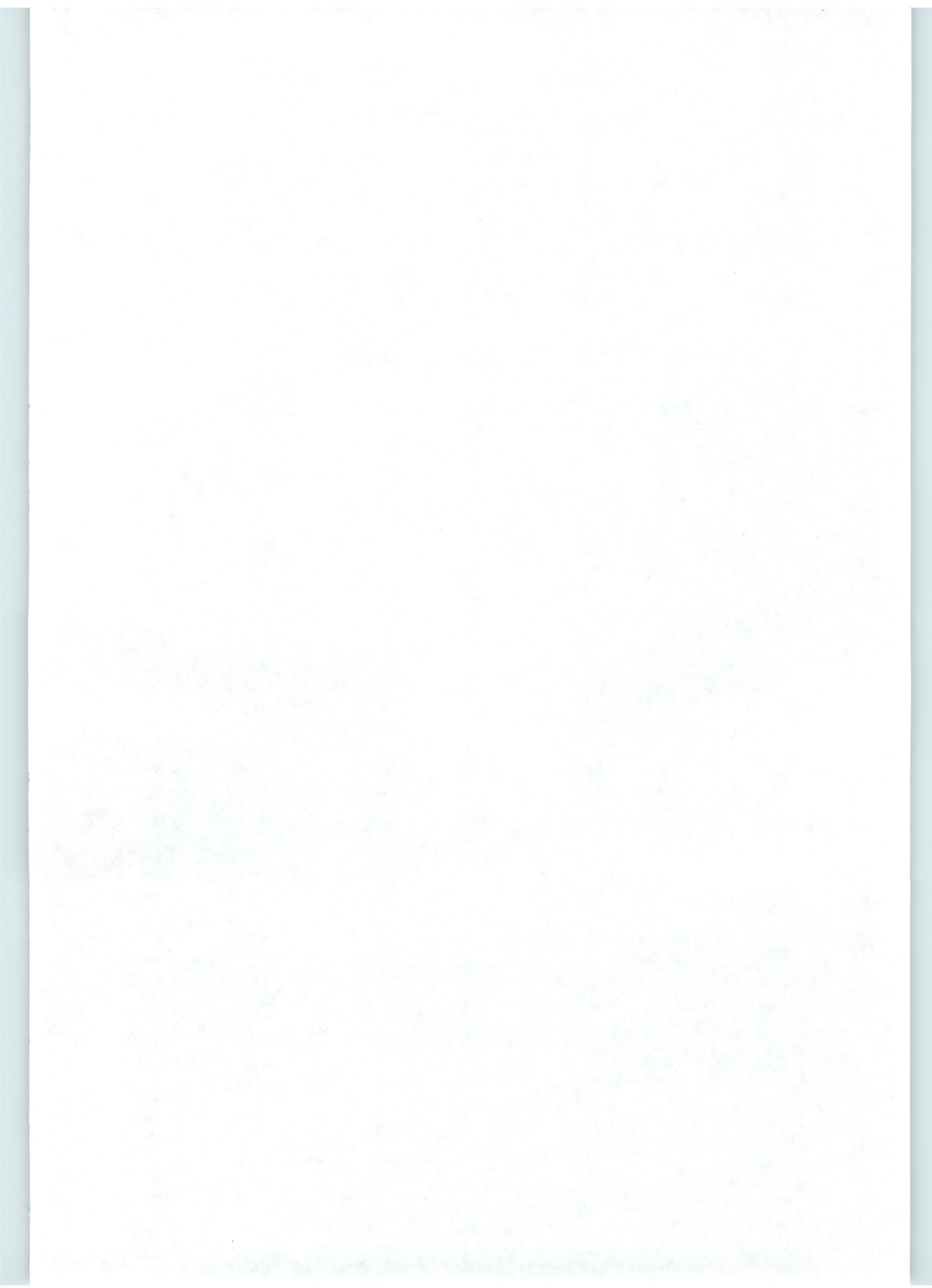
D-93-04

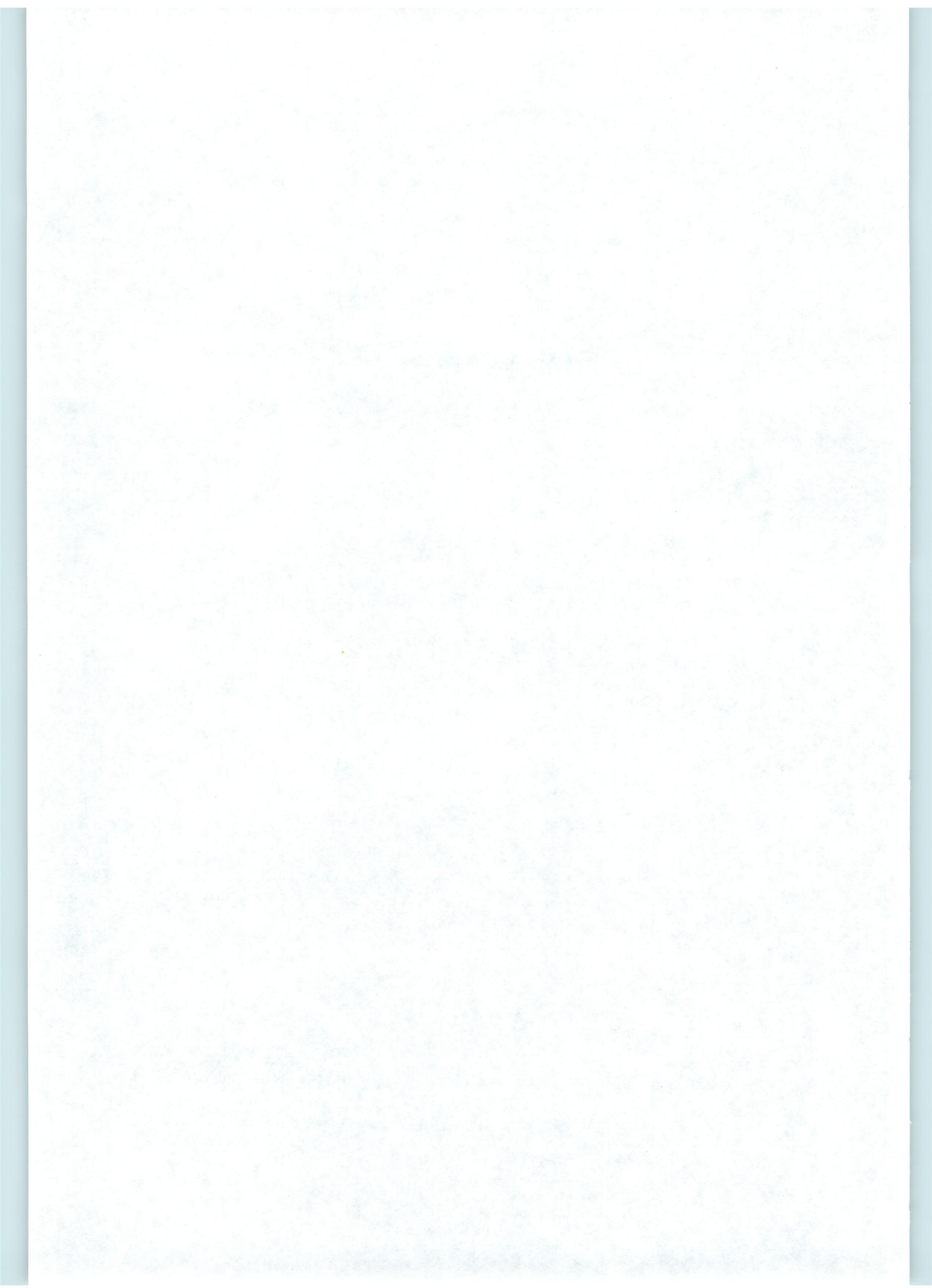
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

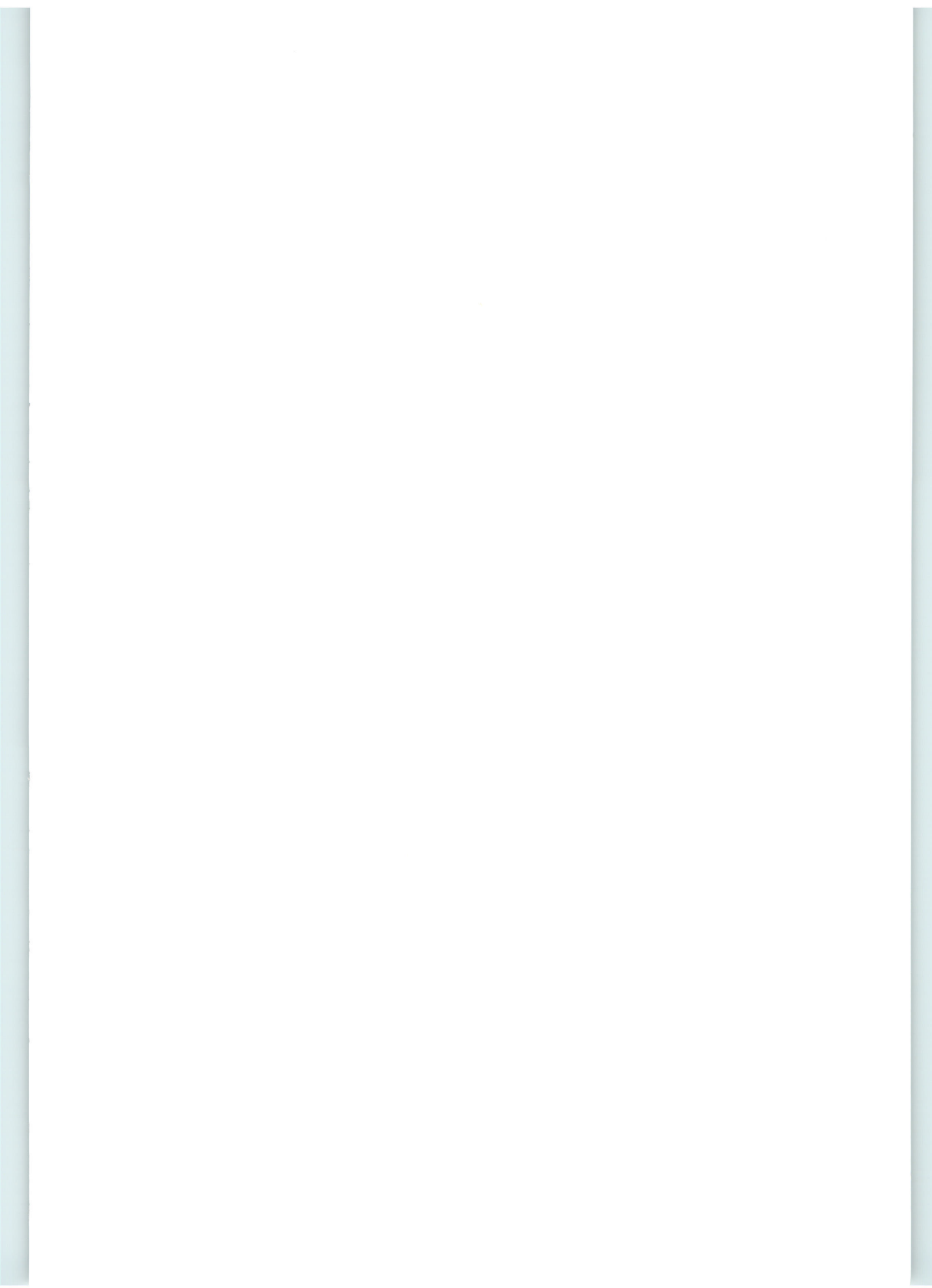
D-93-06

Jürgen Müller (Hrsg.): Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz Saarbrücken 29.-30. April 1993
235 Seiten

Note: This document is available only for a nominal charge of 25 DM (or 15 US-\$).







Equational and Membership Constraints for Infinite Trees

Joachim Niehren, Andreas Podelski, Ralf Treinen

RR-93-14

Research Report