Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-90-02

A Resolution Principle for Clauses with Constraints

Hans-Jürgen Bürckert

March 1990

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, 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, and Siemens. 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

A Resolution Principle for Clauses with Constraints

Hans-Jürgen Bürckert

DFKI-RR-90-02

A short version of this paper has been published in the *Proceedings of 10th Conference on Automated Deduction*, Springer LNCS, 1990

A Resolution Principle for Clauses with Constraints

Hans-Jürgen Bürckert

DFKI, Project Group WINO
Postfach 2080, D-6750 Kaiserslautern, FR Germany
e-mail: buerkert@informatik.uni-kl.de

Abstract: We introduce a general scheme for handling clauses whose variables are constrained by an underlying constraint theory. In general, constraints can be seen as quantifier restrictions as they filter out the values that can be assigned to the variables of a clause (or an arbitrary formulae with restricted universal or existential quantifier) in any of the models of the constraint theory. We present a resolution principle for clauses with constraints, where unification is replaced by testing constraints for satisfiability over the constraint theory. We show that this constrained resolution is sound and complete in that a set of clauses with constraints is unsatisfiable over the constraint theory iff we can deduce a constrained empty clause for each model of the constraint theory, such that the empty clauses constraint is satisfiable in that model. We show also that we cannot require a better result in general, but we discuss certain tractable cases, where we need at most finitely many such empty clauses or even better only one of them as it is known in classical resolution, sorted resolution or resolution with theory unification.

Key words: Resolution, unification, constraint solving, restricted quantification

Table of Contents

1. Introduction

A general observation of Artificial Intelligence (AI) research – although it often has not really been realized in many AI systems – is the fact that the knowledge of an AI system about some problem solving task could be separated into at least two parts:

- A well acquainted part of the problem description:
  Here the system has procedures and facts to treat these problem constraints. In this background part reasoning might be done by calculation and simplification of subgoals, by looking up facts and results in its memory, in a database or anywhere else.
- A more inacquainted part of the problem description:
  Here the system does not dispose of many techniques and facilities of finding a solution. In this foreground part deep reasoning might be necessary, often just by a more or less "blind" search, in order to find solutions to subgoals –perhaps only guided by certain general heuristics.

Now, in the field of Automated Deduction many common automated theorem proving systems or logic programming languages do not support or still worse they completely ignore this separation. They apply standard resolution based deduction methods (with frequently very unsatisfying search procedures) to the whole problem: The description given in a first order predicate logic is transformed into clauses. Hence the structure of the given problem specification is completely destroyed also for those parts, where the system could use standard techniques like integer calculation, linear equation solving etc., if it would know them.

First approaches to separating the knowledge this way are extensions of resolution with paramodulation (Wos & Robinson 1970) or other methods of equality handling, with sort unification (Walther 1983, 1987, Frisch 1989, Schmidt-Schauß 1989), with theory unification (Plotkin 1972, Siekmann 1989) and more general with theory resolution (Stickel 1985, Ohlbach 1986). Sort unification is a very simple way of separating by syntactical phrases rather than by problem oriented reasons; similar for theory unification and paramodulation at least in its use today. Theory resolution provides for a problem oriented separation and takes advantage in allowing special procedures to treat the background parts of the problem. However, it still cuts down the background constraints to literals of clauses, and hence has to apply also search techniques to find the constraints that could be treated by the procedures of the background theories.

In the logic programming area such approaches of replacing "blind" search by computation has also been developed or adapted. However, standard techniques such as theory unification (Gallier & Raatz 1986, Jaffar et al. 1986) or sort unification (Goguen & Meseguer 1984, Smolka et al. 1989) are not exploited in most logic programming languages, especially not in commercial ones. Methods similar to Stickel's theory resolution are proposed

for constraint logic programming languages (Jaffar & Lassez 1986, Dincbas et al. 1988, Höhfeld & Smolka 1988, Smolka 1989). Origins of these ideas can already be found in (Colmerauer 1984). In these approaches, parts of problem descriptions, where very fast solution algorithms are known, are taken as constraints to be solved by such special constraint solving algorithms over distinguished domains – for example, linear equations and inequations over the real numbers.

We argue that this is the right direction for future research: Considering parts of the problem as constraints and restricting their interpretation to a distinguished class of given models. However, the two approaches, theory resolution and constraint logic programming, are not general enough.[1]

Before we describe how to generalize these approaches let us first recall Robinson's Resolution Principle and then develop that new view of considering the unification part as constraint solving.

The Resolution Principle is based on the following inference rule:

From $(A \vee B)$ and $(\neg A \vee C)$ infer $(B \vee C)$

The rule is obviously correct: If $(A \vee B)$ and $(\neg A \vee C)$ are true, then the resolvent $(B \vee C)$ is also true. Since either A is false, then B must be true, or A is true, then C must be true, and hence in any case $(B \vee C)$ must be true.

For predicate logic the two complementary A's are atoms starting with the same predicate symbol, but with potentially different argument terms, say s_i and t_i $(1 \le i \le n)$. Then one has to *unify* corresponding arguments of the literals A and $\neg A$ – that is to find substitutions for the variables that make the corresponding argument terms identical – before the conclusion can be drawn, and the resolvent has to be instantiated with the unifying substitution:

$$\frac{P(s_1,\ldots,s_n) \vee B \qquad \neg P(t_1,\ldots,t_n) \vee C}{\sigma(B \vee C)} \qquad \text{if } \sigma s_i = \sigma t_i \ (1 \le i \le n)$$

Robinson showed that this rule (combined with factorization) provides a complete calculus (Robinson 1965). An analysis of the soundness and completeness proof, however, shows that it is not necessary to unify the terms. It would be enough to test whether they are

[1] Stickel (1985) and Ohlbach (1986) consider theory resolution for *arbitrary* clauses, but with the (implicit) requirement that the background theory has a first order axiomatization – otherwise, their completeness proofs based on the Herbrand Theorem would not be correct. Höhfeld & Smolka (1988) on the other hand allow *arbitrary* classes of models as background theories – and thus generalize the approach of CLP (Jaffar & Lassez 1987) which restricts the constraint theory to be a single model –, but as in CLP they only consider definite clauses over these constraint theories.

unifiable, provided we add a constraint Γ consisting of term equations $s_i = t_i$ (saying "if the terms can be made equal") to the infered resolvent. Whenever such constrained resolvents are now involved in a further resolution step the new resolvent inherits their constraints together with the new argument term equations. Now, these collected constraints have to be tested for unifiability. Thus a resolution step in this modification takes two clauses with such equational constraints and produces a resolvent with a new unifiable equational constraint consisting of the constraints inherited from its parents together with the argument equations for the involved complementary literals.

$$\frac{P(s_1,\ldots,s_n) \vee B \,||\, \Gamma \qquad \neg P(t_1,\ldots,t_n) \vee C \,||\, \Delta}{B \vee C \,||\, \Gamma \wedge \Delta \wedge s_i = t_i} \; \textit{if} \quad \Gamma \wedge \Delta \wedge s_i = t_i \textit{ is unifiable}$$

A more general view is that every clause might have some arbitrary, not necessarily equational constraint and a resolvent of two clauses gets a new constraint that is unsatisfiable whenever one of the constraints of the parents (or the equational constraint of the arguments of the complementary literals) is unsatisfiable.

$$\frac{P(s_1,\ldots,s_n) \vee B \,||\, R \qquad \neg P(t_1,\ldots,t_n) \vee C \,||\, S}{B \vee C \,||\, R \wedge S \wedge s_i = t_i} \; \textit{if} \quad R \wedge S \wedge s_i = t_i \textit{ is satisfiable}$$

It is quite easy to see that this constrained resolution principle is again sound[2], but as in the classical case completeness of a constrained resolution calculus is not straight forward.

The logical view of foreground and background problem descriptions hence could be as follows: There is a set of formulae with a query to be answered or a theorem to be proved or the set itself has to be tested for unsatisfiability. Some of these formulae are axioms of a background theory and parts of the other formulae are constraints that must be solved in the background theory. Here many consequences could be given as well-known or "easy" to verify:[3]

- There are lemmata and theorems proven in the past.
- There are techniques to simplify complex constraints into constraints that are immediate consequences of known results or facts.

---

[2] The argument is quite the same as for classical resolution, if all constraints are true. In the case that any of the constraints is false, the constraint of the resolvent must be false, and hence the resolvent is trivially true (a constrained formula is interpreted as true, when its constraint is false).

[3] Of course this might not always work, as the constraints might state really hard problems of the background theory. However, solving these should no longer be part of the given task, but could open up a new problem perhaps with a new separation into background and foreground knowledge.

➤ There is a tool of standard proof techniques of the field that can be used to prove constraints, which cannot be reduced.

In order to state a little more precisely what we are going to investigate, let us give a formal description, abstracting from (for a practical realization necessary and very interesting, but for theoretical considerations of a constrained resolution principle more or less irrelevant) part of the above presentation, namely, the satisfiability test for the constraints. Furthermore we are going to be a little more general: the background theory need not be given by axioms but just as any class of models.

Hence, let us assume the problem description to be given by a class of structures over a first order language[4] and a set of formulae over a foreground language, such that each of these formulae is constrained by some formulae of the background language. These constraints filter out the values from models of the background theory the variables of the foreground formulae can be assigned to.

Thus, in this view our constraints are restrictions of the quantifiers (Hailperin 1957) – this is one way sorted logic has been viewed (Oberschelp 1962, Walther 1987, Frisch 1989, Schmidt-Schauß 1989). Therefore we could also describe it in the following way: We are given a (foreground) language with the logical connectives and a signature of non-logical symbols, but with restricted quantifiers instead of common quantifiers. Their restrictions are given as certain formulae over a (background) language and these restriction formulae have to be interpreted in the background theory.

Example: The formula $\forall_{x,y:parent(x) \wedge child\text{-}of(y,x)}\ loves(x,y)$ might be a formalization of the sentence "All parents love their children", when we have a suitable background theory containing knowledge about parents and their children. The constraint $parent(x) \wedge child\text{-}of(y,x)$ restricts the possible values of the variables x and y to parents and their children.■

We assume that our set of formulae describing the problem is transformed into a set of constrained clauses[5] consisting of literals over the foreground language and constraints restricting the variables of the clause by formulae of the constraint language. Applying the constrained resolution rule given above to this set of constrained clauses we want to prove the unsatisfiability of the clause set – i.e., that no model of the constraint theory will satisfy it.

A closer look at this task shows that this cannot be done by deriving an empty clause as in the classical resolution calculus or its generalizations. The reason is that such an empty clause might still have some constraints, which are only satisfied by *some* of the constraint

---

[4] For instance, this could be the class of all models of some axioms specifying the background theory.

[5] This transformation could of course be tedious, and it provides several problems, essentially stemming from the fact that the restriction of a quantifier could describe an empty set of admissible assignments in some constraint models (empty quantification).

models, but not by all of them. In classical resolution the derivation of the empty clause could be seen as a derivation of *false* from the starting clause set, but here it is just a derivation of *false* within those models that satisfy the constraints of the empty clause.

But, how could we then get a refutation for an unsatisfiable set of constrained clauses? Let us briefly try to analyse, what the notion of unsatisfiability means here: No model of the constraint theory satisfies the clause set. The argumentation for classical resolution was: Suppose we have a model of the clause set, then it is a model of every resolvent and hence of the empty clause, which is impossible. The analogous argument cannot work in our framework, as the supposed model could be one of those that do not satisfy the constraints of the empty clause. Hence, our solution is to construct for each constraint model a suitable empty clause whose restriction is satisfied by that model: A set of constrained clauses is unsatisfiable iff for each constraint model there is a refutation of an empty clause, whose constraint is satisfied by that constraint model.

This result looks very unsatisfactory, and in fact it is in the general case, but we cannot require a better result in general (see section 5). Fortunately, there are tractable cases (as we already know from the past): for example the constraint theory of classical resolution, which is the set of all structures or equivalently (for clause sets) the set of all Herbrand structures. We will discuss this and some other interesting cases later in more detail. Here in the introduction we will just give the general idea: In case of a constraint theory with first order axiomatization we need at most finitely many empty clauses, such that the disjunction of their restriction is true in all constraint models. This is a consequence of the first order axiomatization and follows essentially from compactness of first order logics. Such a refutation could be obtained by *exhaustive* search strategies (cf. Kowalski 1979). Still better, if the constraint theory is given by a set of definite clauses and the constraints are conjunctions of atoms only, then the derivation of a single empty clause provides a refutation. This follows from the least Herbrand model property of definite clauses.

The paper is organized as follows. In section 2 we introduce our language consisting of a restricted quantifier or constraint system, a signature with restricted quantifiers and formulae over this signature. In section 3 we give the semantics of formulae with restricted quantifiers. Section 4 contains our main result, soundness and completeness for resolution with constrained clauses, and in section 5 we discuss some special cases and consequences of our result.

2. Syntax

Following our separation idea we first introduce the background theory and the restriction formulae as a system for restricted quantification. In logic programming applications this is

known as a constraint system, i.e., a constraint theory together with a set of constraints, the restriction formulae (Höhfeld & Smolka 1988).[6]

We require the reader to be familiar with the notions and notations of mathematical logics, automated deduction, and logic programming (cf. Shoenfield 1967, Chang & Lee 1973, Loveland 1978, Kowalski 1979, Lloyd 1984, Gallier 1986, Bläsius & Bürckert 1989). For the notations of unification theory that are essentially based on equational logic and universal algebra, we refer the reader to (Kirchner 1989), especially to the survey of Siekmann (1989).

A ***restricted quantification system (RQS)*** $\mathfrak{R}$ consists of

- a signature Δ with equality
- a theory over Δ, the ***restriction theory***,
- a set of open Δ-formulae, the ***restriction formulae*** or ***restrictions***.

The restrictions must at least be closed under conjunction and under instantiation of variables. The restriction theory can be given as a distinguished class $\mathcal{R}$ of Δ-structures. The theory provides not only the models for the restriction, but also "skeletons" for the models of the whole problem description. Observe, that we do not require $\mathcal{R}$ to be a first order theory, as there might be no first order axiomatization with exactly the Δ-structures of $\mathcal{R}$ as its models. However, $\mathcal{R}$ could of course be given by some Δ-axioms, which are then called ***declarations*** of $\mathcal{R}$. For computational reasons it would be useful to have calculi, preferably decision procedures, in order to detect

- $\mathcal{R}$-(un)satisfiability of the existential closures of the restriction formulae
- $\mathcal{R}$-validity of sets of existentially closed restriction formulae[7].

This might be done by a "simplification" calculus for $\mathcal{R}$: a distinguished set of simple restrictions together with a procedure that can transform non-simple restrictions into simple restrictions for which $\mathcal{R}$-satisfiability or $\mathcal{R}$-validity is known or easy to check.

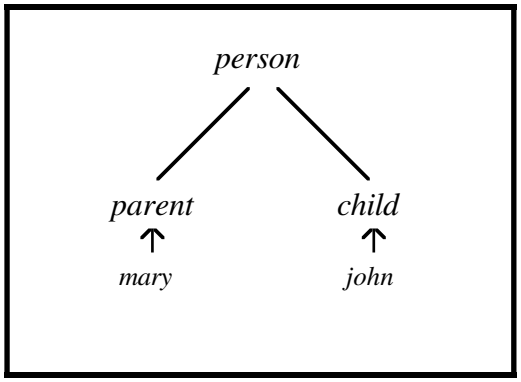
2.1 Examples: (1) An order-sorted signature is an RQS, when we define $\mathcal{R}$ to be the class of all models of the sort declarations (given as a sort theory, cf. Frisch, 1989) and when the set of restrictions contains besides the well-sorted equations the open formulae *S(x)* for each sort symbol *S*, i.e., it consists of conjunctions of equations and sort restrictions *S(x)* for their variables. Notice, that the existential closures of sort restrictions *S(x)* are $\mathcal{R}$-valid, if sorts are required to be non-empty (i.e., if for each sort symbol *S* there is a ground term *t*, such that *S(t)* is explicitely declared or follows from the sort theory).

Suppose we have the following sort hierarchy given by the declarations in the right box:

---

6 We frequently will use these notions synonymously for RQS, restriction theory and restrictions.

7 A set *F* of formulae is valid in a theory $\mathcal{R}$ if for each model of $\mathcal{R}$ there is a formula in the set *F*, that is satisfied by this model. Hence it is the natural generalization of validity of disjunctions of formulae.



$\forall_x \; parent(x) \Rightarrow person(x)$
$\forall_x \; child(x) \Rightarrow person(x)$

$parent(mary)$
$child(john)$

Here the restriction theory $\mathcal{R}$ is the class of all models $\mathcal{A}$, where the denotation $S^{\mathcal{A}}$ of a given unary predicate – i.e. a sort symbol – is a subset of the carrier of $\mathcal{A}$ satisfying the intended subset and element relations ($parent^{\mathcal{A}}$ and $child^{\mathcal{A}}$ are subsets of $person^{\mathcal{A}}$; $mary^{\mathcal{A}}$ is an element of $parent^{\mathcal{A}}$ and $john^{\mathcal{A}}$ is an element of $child^{\mathcal{A}}$). Obviously, the existential closures of sort restrictions are $\mathcal{R}$-valid, as every sort contains a constant.

Sort unification algorithms play the role of simplification calculi, if the equational forms of well-sorted substitutions together with the sort restrictions of their variables are simple restrictions, whose existential closures are trivially $\mathcal{R}$-valid. Obviously, sort unifiability of restrictions (i.e., sorted unification problems) is equivalent to $\mathcal{R}$-validity of their existential closures.

(2) An equational theory $\mathcal{E}$ can be seen as the restriction theory of an RQS, whose restrictions are conjunctions of equations (i.e. unification problems). Here either $\mathcal{R}$ is the class of all models of the theory $\mathcal{E}$ or $\mathcal{R}$ is just the initial algebra or the free algebra of $\mathcal{E}$.[8] An E-unification algorithm would be a suitable simplification calculus with substitutions as simple restrictions. Notice again, that E-unifiability of restrictions is equivalent to $\mathcal{R}$-validity of their existential closures and that idempotent substitutions are always $\mathcal{R}$-valid restrictions.

(3) As a special case, remember the empty equational theory of syntactical equality, for which the well-known Robinson unification algorithm is a simplification algorithm. In this case, $\mathcal{R}$ is the class of all algebras over the given signature Δ or the corresponding free or the initial algebra, i.e., the usual term algebra or ground term algebra (i.e. the Herbrand universe). Notice again Example 3.2(1).■

A ***signature with restricted quantifiers*** or an ***RQ-signature*** Σ consists of an RQS $\mathfrak{R}$, together with an additional set of predicate symbols $\mathcal{P}_\Sigma$ and possibly a set of function symbols $\mathcal{F}_\Sigma$ both disjoint from the signature Δ.[9] If the set $\mathcal{F}_\Sigma$ of Σ-function symbols is not

[8] These three RQS are in some sense equivalent. However, this is only true for clauses not for arbitrary formulae, cf. Example 3.2(1).

[9] In order to simplify our notation we will use the prefix "Σ-" if we denote these symbols or objects –terms, atoms, formulae, etc. – that are built up by the additional symbols only.

empty, we have to extend our set of restrictions, such that it contains in addition all equations built up by the new function symbols. Thus we can assume without loss of generality that the new function symbols are already part of the RQS and the RQ-signature contains only new predicate symbols. [10]

Given such an RQ-signature Σ we want to define formulae with restricted quantifiers over this RQ-signature. Therefore we allow quantifiers to be indexed not only by variables, but by pairs of a variable set and a restriction formula (e.g. $\forall_{X:R}$ and $\exists_{X:R}$), and we call them ***restricted quantifiers***. Thus we define ***RQ-formulae over*** Σ by

(1) all Σ-atoms are RQ-formulae

(2) $\forall_{X:R} F$ and $\exists_{X:R} F$ are RQ-formulae, if F is an RQ-formula, R is a restriction, and X is the sequence or set of variables that are not bound in R,

(3) $\forall_X F$, $\exists_X F$, $F \wedge G$, $F \vee G$, $F \Rightarrow G$, $F \Leftrightarrow G$ are RQ-formulae, if F and G are RQ-formulae, and X is the sequence or set of variables that are not bound in F.

Notice, that in definition (2) the formula F might contain free variables of X that are now bound by the restricted quantifiers $\forall_{X:R}$ or $\exists_{X:R}$; the formula R is called the restriction for those variables. We often drop the restricted variables and write $\forall_R F$ instead of $\forall_{X:R} F$.

As in the standard case we will deal with clauses, called ***RQ-clauses*** or ***constrained clauses***; they consist of any (possible empty) finite set C of Σ-literals, i.e., a common Σ-clause, and a restricted quantifier $\forall_{X:R}$ of their variables, frequently written $C \,||\, R$ instead of $\forall_R C$. We call C the matrix or the ***kernel*** of the RQ-clause and R its restriction or ***constraint***. If C is empty we call it an ***empty*** RQ-clause, denoted $\square \,||\, R$ or $\forall_R \square$.

2.2 Examples: (1) Taking the sort signature from Example 2.1(1) and the binary predicate symbols *loves* and *child-of*, the RQ-formula

$$\forall_{x:parent(x)}\ \forall_{y:child(y)}\ \ child\text{-}of(y, x) \Rightarrow loves(x, y)$$

is a formalization of the English sentence “All parents love their children.”

(2) In fact the formula intends a little more, namely, that the children are still “young persons” (i.e., have sort *child*). In order to be closer to the meaning of the sentence, we could replace the sort *child* in the RQS by the binary predicate symbol *child-of* now considered as a symbol of the RQS and perhaps have the declaration in the modified RQS

$$\forall_{x,y}\ \ child\text{-}of(y,x) \Rightarrow parent(x) \wedge person(y)$$

such that our RQ-formula looks like

$$\forall_{x:parent(x)}\ \forall_{y:child\text{-}of(y,x)}\ \ loves(x, y)$$

---

[10] Extending the restrictions with the new function symbols requires also an expansion of the restriction theory with these new symbols, as we have to interpret the extended restriction formulae.

which seems to be a more adequate formalization of "All parents love their children."

(3) If the RQ-signature is given over an RQS defined by an equational theory as in Example 2.1(2), we obtain the RQ-formulae corresponding to the common formulae without restricted quantifiers by *unfolding*: We replace each term t in an atom by a new variable v and replace the quantifier governing the term by a restricted quantifier with the restriction $v = t$. For example, given an equational theory of an associative and commutative function symbol f, a free constant a, and a unary free function symbol g, a formula $\forall_{x,y}\ P(g(f(x, y)), f(x, a))$ with a binary predicate P, becomes the RQ-formula $\forall_{x,y,v,w:\ v=g(f(x,\ y))\ \wedge\ w\ =f(x,\ a)}\ P(v, w)$.■

3. Semantics

The interpretation of the RQ-formulae and especially the restricted quantification causes no deep problems. The semantics of restricted quantifiers can be given by the usual *relativization*, that is, one could transform any RQ-formula into an equivalent formula without restricted quantifiers by

replacing $\forall_{X:R} F$ by $\forall_X R \Rightarrow F$
replacing $\exists_{X:R} F$ by $\exists_X R \wedge F$

However, as we already know the Δ-models given in the RQS, we have to say what our Σ-models should look like. We have to interpret additionally the new predicates of the RQ-signature. Recalling the notion of structure expansions (cf. Shoenfield 1967) we interpret RQ-formulae in structures that expand the Δ-structures of the restriction theory to the new symbols of Σ, such that the variables of a restricted quantifier are assigned by elements of these structures that have to satisfy the restriction. Let us define this more precisely.

An ***RQ-structure over*** Σ is a structure $\mathcal{A}$ such that its reduct $\mathcal{A}|_\Delta$ to Δ is one of the Δ-models in $\mathcal{R}$.[11] As we assumed Σ to introduce only new predicate symbols, but no function symbols, we obtain the different RQ-structures by expanding every model of the restriction theory with all possible interpretations of these new predicate symbols. If the restriction theory $\mathcal{R}$ is given by a Δ-axiomatization, RQ-structures are exactly those structures that satisfy the axioms of $\mathcal{R}$ considered as formulae over the extended signature. Obviously the class of all RQ-structures is a conservative extension of the restriction theory: A restriction is $\mathcal{R}$-valid iff it is satisfied by every RQ-structure.

Given an RQ-structure $\mathcal{A}$, an (open) RQ-formula F, and an assignment $\alpha: V \rightarrow \mathcal{A}$ of the set V of free variables of the formula F, we define satisfiability of F in $\mathcal{A}$ with α (written $(\mathcal{A}, \alpha) \models F$) as usual for atoms and formulae built up by the common junctors. The only

[11] A reduct of a structure to a subsignature is given by forgetting about the denotations of the symbols that are not in the subsignature. Conversely, an expansion of a structure to a supersignature is given by any interpretation of the additional symbols.

cases that are new are formulae with restricted quantifiers and we give their semantics here explicitely:

- $(\mathcal{A}, \alpha) \models \forall_{X:R} F$ iff
  for every assignment $\beta: X \to \mathcal{A}$ with $(\mathcal{A}|_{\Delta}, \alpha \cup \beta) \models R$
  we have $(\mathcal{A}, \alpha \cup \beta) \models F$
- $(\mathcal{A}, \alpha) \models \exists_{X:R} F$ iff
  there is an assignment $\beta: X \to \mathcal{A}$ with $(\mathcal{A}|_{\Delta}, \alpha \cup \beta) \models R$
  such that $(\mathcal{A}, \alpha \cup \beta) \models F$

A closed RQ-formula F is ***RQ-satisfiable***, iff there is an RQ-structure that satisfies F with the empty assignment; otherwise F is called ***RQ-unsatisfiable***. The formula F is ***RQ-valid*** or an ***RQ-tautology***, iff it is satisfied by every RQ-structure. Notice that formulae with $\mathcal{R}$-unsatisfiable restrictions can be viewed as quantified over the empty set, hence such formulae are true in every RQ-structure: they are RQ-tautologies.

3.2 Examples: (1) In case of an RQS given by the empty equational theory (Example 2.1(3)), the RQ-structures are the usual structures considered as the expansions of algebras – the models of equational theories – with predicate symbols.
We could also take the Herbrand universe as restriction theory; the RQ-structures are then just the Herbrand models. However, this semantics has to be used with caution, as Herbrand models can only play the rôle of semantics, when we only consider formulae that are in clause form. Thus, as long as we allow arbitrary RQ-formulae, the RQS for the empty theory given by all algebras cannot equivalently be replaced by the RQS of the Herbrand universe.

(2) If the RQS is given by an arbitrary equational theory (cf. Example 2.1(2)) and the restriction theory consists of the initial algebra of this theory, the RQ-structures are the Herbrand models factored by the congruence, which is induced by the equational theory.

(3) The models of order-sorted signatures are exactly the RQ-structures over the corresponding RQS. This is an immediate consequence of a relativization theorem in (Schmidt-Schauß 1989).

(4) Take the restriction theory of Example 2.2(2). Then the formula given there has the following semantics: An RQ-structure $\mathcal{A}$ is every structure with a set $parent^{\mathcal{A}}$ and a relation $child\text{-}of^{\mathcal{A}}$ satisfying the declarations of the given restriction theory. Such a structure satisfies that formula, if for each pair a, b of an element a of the $parent^{\mathcal{A}}$ set and an element b that is in the $child\text{-}of^{\mathcal{A}}$ relation to that parent a this pair is also in the relation $loves^{\mathcal{A}}$. That means each structure, where "all parents love their children", is a model of that formula.■

One can see immediately that unrestricted quantifiers are special cases of restricted ones and that we can replace every sequence of restricted universal quantifiers (and analogously of restricted existential quantifiers) by a single one:

$\forall_{X:R}\ \forall_{Y:S}\ F$ is logically equivalent to $\forall_{X \cup Y:R \wedge S}\ F$.[12]

If the restriction is a complex Δ-formula, which contains quantifiers, we can sometimes move these quantifiers out of the restriction and lift it to RQ-formula, for example,

$\forall_{X:R(X)\ \wedge\ \exists_Y\ S(Y)}\ F$ is logically equivalent to $\forall_{X,Y:R(X)\ \wedge\ S(Y)}\ F$.

These, and many further logical equivalent transformations of RQ-formulae can easily be verified for instance via a relativization (Bürckert 1990).

4. Resolution with Restricted Quantifiers

From now on we assume every RQ-formula to be given in RQ-clause form. Of course, it is not trivial to obtain clause form, as the restriction need not be satisfiable in every model of the restriction theory. For example, the restriction theory might have to be expanded with Skolem functions and certain Skolem Δ-formulae; see (Bürckert 1990) for more details on these problems.

Our calculus consists of the following ***RQ-resolution rule*** for any pair of RQ-clauses, such that one of them contains at least one positive literal and the other one contains at least one negative literal each with the same predicate symbol:

$$\frac{\{P(s_{11},\ldots,s_{1n}),\ldots,\ P(s_{k1},\ldots,s_{kn})\} \cup\ C \,||\, R \qquad \{\neg P(t_{11},\ldots,t_{1n}),\ldots,\neg P(t_{m1},\ldots,t_{mn})\} \cup\ D \,||\, S}{C \cup\ D \,||\, R \quad \wedge S \wedge \Gamma \ \ (\text{if } R \wedge S \wedge \Gamma \text{ is } \mathcal{R}\text{-satisfiable})}$$

C and D are the remaining parts of the two clauses and Γ is the conjunction of the (multi)equations $s_{1i} = \ldots = s_{ki} = t_{1i} = \ldots = t_{mi}\ (1 \le i \le n)$. The derived clause is called an ***(RQ-)resolvent*** of the two parent clauses.[13]

An ***RQ-resolution step*** $\mathcal{C} \rightarrow \mathcal{C}'$ transforms an RQ -clause set $\mathcal{C}$ into the set $\mathcal{C}'$ by choosing two suitable clauses in $\mathcal{C}$ and adding their resolvent to $\mathcal{C}$ (after a renaming of all variables of the resolvent consistent with variables that have not been used so far).[14] An ***RQ-derivation*** is a possibly infinite sequence $(\mathcal{C}_n)_{n\ge 0}$ of RQ-resolution steps $\mathcal{C}_n \rightarrow \mathcal{C}_{n+1}$ starting with an initial set $\mathcal{C}_0$ of RQ-clauses. An ***RQ-refutation*** of a set $\mathcal{C}_0$ of RQ-clauses is a (possibly infinite) RQ-derivation $(\mathcal{C}_n)_{n\ge 0}$ starting with that clause set, such that for each model $\mathcal{A}$ of the restriction theory there is an RQ-clause set $\mathcal{C}_n$ in the derivation with an empty

---

[12] This means that the two formulae have exactly the same RQ-models.

[13] Notice, that it is not really necessary to require satisfiability of the constraint of the resolvent. It could be the task of the control strategies to guarantee that not too many tautological resolvents (i.e., resolvents with unsatisfiable restriction) are derived.

[14] We assume that the set of clauses the RQ-resolution step is applied to consists of variable disjoint clauses.

RQ-clause $\square \parallel R$, whose restriction is satisfied by this model, i.e., $\mathcal{A} \models \exists R$. Notice, that an RQ-structure that satisfies the restriction formula of an empty RQ-clause cannot satisfy this empty RQ-clause. Notice further, that an empty RQ-clause with satisfiable, but non-valid restriction formula has RQ-models and hence its derivation cannot terminate a refutation.

RQ-resolution is sound in the sense that every RQ-resolvent is a logical consequence of its RQ-parents:

4.1 Lemma: *Let $\mathcal{C}$ be a set of RQ-clauses, let $\mathcal{C}'$ be derived from $\mathcal{C}$ by an RQ-resolution step and let $\mathcal{A}$ be an RQ -structure. Then: $\mathcal{A} \models \mathcal{C}$ implies $\mathcal{A} \models \mathcal{C}'$.*

Proof: We show that an RQ-structure satisfies the resolvent of two clauses, whenever it satisfies the two parent clauses. For ease of notation we prove this only for the case, where two literals are involved in the resolution. The general case is straight forward.
Let $\mathcal{A} \models \{P(x_1,\ldots,x_n)\} \cup C \parallel R$ and $\mathcal{A} \models \{\neg P(y_1,\ldots,y_n)\} \cup D \parallel S$. If $\mathcal{A}$ does not satisfy the restriction of the resolvent then $\mathcal{A}$ is a model of the resolvent. Hence, let $(\mathcal{A}, \alpha) \models R \wedge S \wedge \Gamma$, then $(\mathcal{A}, \alpha)$ must satisfy the kernel of the two parent clauses. As $(\mathcal{A},\alpha)$ satisfies Γ, obviously $(\mathcal{A},\alpha)$ can only satisfy either $P(x_1,\ldots,x_n)$ or $\neg P(y_1,\ldots,y_n)$, but not both. Hence $(\mathcal{A},\alpha)$ must satisfy either C or D and therefore $\mathcal{A}$ is a model of the resolvent. ■

As our restriction theory is not given by a set of first order axioms we cannot reduce completeness of RQ-resolution to completeness of classical resolution as in the approaches for resolution modulo the equality axioms (Chang & Lee 1973), sort resolution (Walther 1987, Frisch 1989, Schmidt-Schauß 1989) or theory resolution (Stickel 1985, Ohlbach 1986). Instead we use a technique similar to that being used by Jaffar & Lassez (1986) and in a more general framework by Höhfeld & Smolka (1988). The standard proofs for completeness of classical resolution rely on a proof technique that reduces completeness to *ground* completeness of *ground* resolution on a set of *ground* instances of the given clause set. Now, classical resolution could be considered as RQ-resolution with respect to the restriction theory given by the ground term algebra. Hence, if we have an arbitrary RQS, we just have to replace the Herbrand universe by any of the models distinguished by the restriction theory.

4.2 Definition: Let $\mathcal{A} \in \mathcal{R}$ be a model in the restriction theory.
(1) Let C be the kernel of an RQ-clause with variables in X, and let $\alpha: X \to \mathcal{A}$ an assignment. Then we call the triple $(\mathcal{A}, \alpha, C)$ an ***$\mathcal{A}$-clause***.
(2) We call an $\mathcal{A}$-clause $(\mathcal{A}, \alpha, C)$ an ***$\mathcal{A}$-instance*** of an RQ-clause $C \parallel R$ iff $(\mathcal{A}, \alpha) \models R$.
(3) Let $\mathcal{C}$ be a set of RQ-clauses. Then a set $\{(\mathcal{A}, \alpha_i, C_i): i \in I\}$ of $\mathcal{A}$-instances of RQ-clauses $C_i \parallel R_i\ (i \in I)$ of $\mathcal{C}$ is called an ***$\mathcal{A}$-instantiation*** of $\mathcal{C}$, iff $(\mathcal{A},\alpha_i) \models R_i$ for each $i \in I$.[15] The set $\{(\mathcal{A}, \alpha, C): C \parallel R \text{ in } \mathcal{C}, (\mathcal{A}, \alpha) \models R\}$ of all $\mathcal{A}$-instances of all RQ-clauses in $\mathcal{C}$ is called the ***$\mathcal{A}$-base*** of $\mathcal{C}$.

---

[15] Notice that an $\mathcal{A}$-instantiation might contain more than one $\mathcal{A}$-instance of the same RQ -clause.

(4) A set of $\mathcal{A}$-clauses $\{(\mathcal{A}, \alpha_i, C_i): i \in I\}$ is satisfied by a Σ-expansion $\mathcal{A}^*$ of $\mathcal{A}$ iff $(\mathcal{A}^*, \alpha_i) \models C_i$ for each $i \in I$; it is unsatisfiable, if it is not satisfied by a Σ-expansion of $\mathcal{A}$. ■

Our definition is the suitable generalization of the classical notion of a ground instance of a clause, which can be seen as a triple consisting of the Herbrand universe, a ground assignment, and the clause to be instantiated. Ohlbach (1986) introduces abstract clauses as a scheme for sets of ground clauses (e.g., classical clauses are schemes for the sets of all their ground instances). Our RQ-clauses can be seen as schemes for sets of $\mathcal{A}$-clauses.

As in the classical case we get that unsatisfiability of a set of RQ-clauses implies unsatisfiability of some $\mathcal{A}$-instantiation (for each model $\mathcal{A}$ of the restriction theory).

4.3 Theorem: (Herbrand Theorem for RQ-clauses)
A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff for each model $\mathcal{A}$ in the restriction theory $\mathcal{R}$ there is a finite $\mathcal{A}$-instantiation of $\mathcal{C}$ that is unsatisfiable.

Proof: For the one direction let $\mathcal{C}$ be RQ-unsatisfiable. Suppose there is an $\mathcal{A} \in \mathcal{R}$ such that each finite $\mathcal{A}$-instantiation of $\mathcal{C}$ is satisfiable. By the Compactness Theorem for First Order Logics we have that the $\mathcal{A}$-base for $\mathcal{C}$ is satisfiable.[16] Hence there is an expansion $\mathcal{A}^*$, such that for each $C \,||\, R$ in $\mathcal{C}$ and each assignment α with $(\mathcal{A}, \alpha) \models R$ we have $(\mathcal{A}^*, \alpha) \models C$; a contradiction to the RQ-unsatisfiability of $\mathcal{C}$. The other direction is obvious. ■

In order to prove unsatisfiability of an $\mathcal{A}$-instantiation we introduce the following ***$\mathcal{A}$-resolution*** rule for $\mathcal{A}$-clauses.

$$\frac{(\mathcal{A}, \alpha, \{P(s_{11},\ldots,s_{1n}),\ldots, P(s_{k1},\ldots,s_{kn})\} \cup C) \quad (\mathcal{A}, \alpha, \{\neg P(t_{11},\ldots,t_{1n}),\ldots,\neg P(t_{m1},\ldots,t_{mn})\} \cup D)}{(\mathcal{A}, \alpha, C \cup D) \qquad \text{if } (\mathcal{A},\alpha) \models \Gamma \text{ } (\Gamma \text{ as in the RQ-resolution rule})}$$

which can be used to deduce the empty $\mathcal{A}$-clause from every unsatisfiable set of $\mathcal{A}$-clauses.

4.4 Proposition: *If a finite $\mathcal{A}$-instantiation $\mathcal{D}$ of a set of RQ-clauses is unsatisfiable, then there is a finite derivation of an empty $\mathcal{A}$-clause via $\mathcal{A}$-resolution.*

Proof: The proof is analogous to the usual case when we have ground instantiations instead of $\mathcal{A}$-instantiations. The only difference is that instead of syntactical equality of ground instances we have here *semantical* equality of arguments under the assignment.
1. If the empty $\mathcal{A}$-clause is already in $\mathcal{D}$, we are finished.

---

16 The satisfiability of sets of $\mathcal{A}$-clauses is a first order problem, more precisely it is a propositional logic one: We can consider it as the problem of satisfiability of ground clauses over the Σ-predicate symbols and the elements of the carrier of $\mathcal{A}$ considered as constants. Each $\mathcal{A}$-clause $(\mathcal{A}, \alpha, C)$ corresponds then to the ground clause αC – the assignments are ground substitutions replacing the variables by the constants, i.e., the elements of the carrier of $\mathcal{A}$.

2. We proceed by induction on the number N of *excess literals* in $\mathcal{D}$ (number of literal occurrences in $\mathcal{D}$ minus number of clauses in $\mathcal{D}$).
$N = 0$: (Then no non-unit clause is in $\mathcal{D}$).[17] Hence, as $\mathcal{D}$ is unsatisfiable, there must be two complementary units $(\mathcal{A}, \alpha, C)$, $(\mathcal{A}, \alpha, D)$ in $\mathcal{D}$ (i.e., the corresponding arguments of the two $\mathcal{A}$-clauses are assigned to the same elements in $\mathcal{A}$ by α). Otherwise let $\mathcal{A}^*$ be any expansion of $\mathcal{A}$, such that $p^{\mathcal{A}^*}(\alpha x_1,\ldots,\alpha x_n)$ is true in $\mathcal{A}^*$, whenever the unit $(\mathcal{A}, \alpha, \{p(x_1,\ldots,x_n)\})$ is in $\mathcal{D}$, and $p^{\mathcal{A}^*}(\alpha x_1,\ldots,\alpha x_n)$ is false in $\mathcal{A}^*$, otherwise. Then $\mathcal{A}^*$ satisfies $\mathcal{D}$ by definition. Hence the empty $\mathcal{A}$-clause can be derived.
$N > 0$: Let the lemma be true for all $\mathcal{D}$ with fewer than N excess literals. Then there is at least one non-unit clause, say $(\mathcal{A}, \alpha, C)$. (There is at least one excess literal, as $N > 0$). Now, let $D := C \setminus \{L\}$ for some literal L in C. Then the set $\mathcal{D}' := (\mathcal{D} \setminus \{(\mathcal{A}, \alpha, C)\}) \cup \{(\mathcal{A}, \alpha, D)\}$ is also an unsatisfiable set of $\mathcal{A}$-clauses. Hence, as it contains one fewer excess literal, by the induction hypothesis there is a deduction of the empty $\mathcal{A}$-clause for this set. Similarily, the set $\mathcal{D}'' := (\mathcal{D} \setminus \{(\mathcal{A}, \alpha, C)\}) \cup \{(\mathcal{A}, \alpha, \{L\})\}$ is unsatisfiable and there is a deduction of the empty $\mathcal{A}$-clause by induction hypothesis. If L is not used in the deduction for $\mathcal{D}'$, it works also for $\mathcal{D}$. Otherwise we can construct a deduction of the empty $\mathcal{A}$-clause from $\mathcal{D}$ as follows. We add L back into D and all its descendants in the deduction for $\mathcal{D}'$ in order to get a deduction for $\mathcal{D}$. Now, either this modified deduction contains the empty $\mathcal{A}$-clause or it contains the unit clause $(\mathcal{A}, \alpha, \{L\})$ resulting from the empty $\mathcal{A}$-clause after adding L to it. In the latter case we get a deduction of the empty $\mathcal{A}$-clause from $\mathcal{D}$ by appending the deduction of the empty $\mathcal{A}$-clause from $\mathcal{D}''$ onto the end of this modified deduction.■

The following Lifting Lemma is a modification of the usual one. It says that an $\mathcal{A}$-resolvent of $\mathcal{A}$-instances of two RQ-clauses is an $\mathcal{A}$-instance of the RQ-resolvent of the two RQ-clauses. However, it is more trivial as we do not simplify the restrictions (this is computing the most general unifier of the arguments of the complementary literals in classical resolution), but keep them as constraints that are only tested for satisfiability (i.e., unifiability in the classical case).

4.5 Lifting Lemma: *Let $C_1 \parallel R_1$ and $C_2 \parallel R_2$ be two constrained clauses and let $(\mathcal{A}, \alpha, C_1)$, $(\mathcal{A}, \alpha, C_2)$ be $\mathcal{A}$-instances of the two RQ-clauses. Then an $\mathcal{A}$-resolution step on the two instances can be lifted to an RQ-resolution step on the RQ-clauses, such that the $\mathcal{A}$-resolvent $(\mathcal{A}, \alpha, C)$ is an $\mathcal{A}$-instance of the RQ-resolvent $C \parallel R$.*

Proof: By the definition of $\mathcal{A}$-instances and $\mathcal{A}$-resolvents $(\mathcal{A}, \alpha) \models R$ with $R := R_1 \wedge R_2 \wedge \Gamma$.■

Corollary: *If a set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable, then for each $\mathcal{A} \in \mathcal{R}$ there is a finite RQ-derivation of an empty RQ-clause whose restriction is satisfiable by $\mathcal{A}$.*

---

[17] A unit clause is a clause with one literal. Here we use this notion analoguously for $\mathcal{A}$-clauses.

Proof: If $\mathcal{C}$ is RQ-unsatisfiable, then for each $\mathcal{A} \in \mathcal{R}$ there must be a finite $\mathcal{A}$-instantiation which is unsatisfiable (Herbrand Theorem). Hence there is a finite derivation of the empty clause for this set of $\mathcal{A}$-clauses (Prop. 4.4). By the Lifting Lemma it can be lifted to $\mathcal{C}$, such that $\mathcal{A}$ satisfies the corresponding restriction of the empty clause.■

Now we are ready to formulate and prove correctness and completeness of our constrained resolution calculus. The result is only of relative merit as we did not give a calculus for proving satisfiability of constraints. Still worse, as the constraint theory is given by a class of models, it need not to be a first order theory (that means that there need not exist a first order axiomatization, e.g., it could be the theory of real numbers), and hence we cannot have a complete calculus for constrained clauses in the strong sense in general.[18]

4.6 Theorem: (Soundness and Completeness of RQ-resolution)
A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff for each $\mathcal{A} \in \mathcal{R}$ there exists an RQ-derivation from $\mathcal{C}$ containing an empty RQ-clause $\square \,||\, R$, such that $\mathcal{A} \models \exists R$.

Proof: 1. Soundness ($\Leftarrow$): Assume $\mathcal{C}$ were RQ-satisfiable. Then there exists an $\mathcal{A} \in \mathcal{R}$ such that $\mathcal{A}^* \models \mathcal{C}$ for some expansion of $\mathcal{A}$. By Lemma 4.1 $\mathcal{A}^* \models \forall_R\ C$ for each RQ-clause $\forall_R\ C$ that can be derived by RQ-resolution from $\mathcal{C}$. By the precondition an empty RQ-clause $\forall_R\ \square$ with $\mathcal{A} \models \exists R$ is derivable from $\mathcal{C}$. Hence there is an assignment α with $(\mathcal{A}^*, \alpha) \models \square$. This is a contradiction.
2. Completeness ($\Rightarrow$): Let $\mathcal{A}$ be a model in $\mathcal{R}$. As $\mathcal{C}$ is RQ-unsatisfiable, by the Herbrand Theorem 4.3 there is an unsatisfiable $\mathcal{A}$-instantiation of $\mathcal{C}$ and we can deduce the empty $\mathcal{A}$-clause by $\mathcal{A}$-resolution collecting the restrictions in $R_{\mathcal{A}}$. By the Lifting Lemma this refutation can be lifted and hence the collected restriction $R_{\mathcal{A}}$ of the empty clause is satisfied by $\mathcal{A}$. ■

Remark: It is enough to consider for each $\mathcal{A}$ just those clauses of $\mathcal{C}$ whose restrictions are satisfiable by this distinguished $\mathcal{A}$; the same holds for the resolvents that are derived.

Corollary: *Let $\mathcal{C}$ be a set of RQ-clauses. If there is an empty RQ-clause with an $\mathcal{R}$-valid restriction derivable by RQ-resolution, then $\mathcal{C}$ is RQ-unsatisfiable.*

5. Consequences and Discussion

Obviously the Completeness Theorem is not very satisfactory from the practical point of view, in particular not when we are interested in an actual implementation of a theorem proving

[18] Of course for applications the theory has to be first order in the sense that there should be a way – possibly with a different signature as usual – of first order axiomatization, e.g, linear equation solving over the reals could be coded in first order, although the real numbers are not a first order theory.

system for constrained clauses. The question arises if and how we can guarantee that RQ-derivation terminates when only finitely many empty clauses are needed for an RQ-refutation. Remembering that for classic resolution there exist exhaustive refutation strategies, we observe that with such a strategy there is a finite RQ-refutation terminating with an RQ-clause set that contains finitely many empty RQ-clauses such that each restriction model satisfies the restriction of one of these empty clauses, whenever finitely many empty clauses provide an RQ-refutation.

As a first immediate consequence of the Compactness Theorem of First Order Logics we have that if $\mathcal{R}$ is a first order theory (that is, if there exists a first order axiomatization with exactly the models given in $\mathcal{R}$), then for every RQ-unsatisfiable set of RQ-clauses $\mathcal{C}$ there are finitely many empty RQ-clauses derivable from $\mathcal{C}$ by RQ-resolution such that the disjunction of their restrictions is a logical consequence of the restriction theory. Hence, in this case exhaustive refutation strategies provide correct and complete RQ-resolution calculi, if we have a calculus for proving $\mathcal{R}$-validity of disjunctions of restrictions.

5.1 Theorem: *Let $\mathfrak{R}$ be an RQS with a first order restriction theory $\mathcal{R}$. A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff there are finitely many empty RQ-clauses with constraints $R_1, \ldots, R_n$ RQ-derivable from $\mathcal{C}$ such that $\mathcal{R} \models \exists R_1 \vee \ldots \vee \exists R_n$.*

Proof: Let $Th(\mathcal{R})$ be a first order axiomatization of $\mathcal{R}$. Then the Completeness Theorem implies that RQ-unsatisfiability of $\mathcal{C}$ is equivalent to: for each $\mathcal{A}$ with $\mathcal{A} \models Th(\mathcal{R})$ there is some empty RQ-clause $\square \,||\, R_{\mathcal{A}}$ derivable from $\mathcal{C}$, such that $\mathcal{A} \models \exists R_{\mathcal{A}}$.
This in turn is equivalent to the unsatisfiability of the possibly infinite set of formulae

$$Th(\mathcal{R}) \cup \{\neg \exists R_{\mathcal{A}} : \mathcal{A} \models Th(\mathcal{R})\} .$$

By the Compactness Theorem there is a finite subset that is unsatisfiable, hence there exist finitely many empty RQ-clauses derivable from $\mathcal{C}$ with $\mathcal{R} \models \exists R_1 \vee \ldots \vee \exists R_n$.■

Corollary: *Let $\mathfrak{R}$ be as in the theorem above. If the set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable then there are finitely many models $\mathcal{A}_1, \ldots, \mathcal{A}_n$ of the restriction theory and finitely many empty RQ-clauses $\square \,||\, R_1, \ldots, \square \,||\, R_n$ derivable from $\mathcal{C}$ via RQ-resolution such that $\mathcal{A}_i \models \exists R_i$ $(1 \leq i \leq n)$.*

The following example shows that we need the derivation of more than one empty RQ-clause in general, even if the initial clause set constains only clauses with *valid* restrictions.

5.2 Example: Suppose that we have an RQS given with the restriction theory

$$\mathcal{R} := \{Pa,\ Pb \vee Pc\}$$

Let us consider the following set of three constrained clauses:

$$\begin{array}{lll} \mathcal{C}: & (1) & Q(x,x) \parallel Px \\ & (2)\neg & Q(b,y) \parallel Py \\ & (3)\neg & Q(c,z) \parallel Pz \end{array}$$

Then we can derive the following two resolvents

from clause (1) and (2) $\square \parallel Pb$
from clause (1) and (3) $\square \parallel Pc$

Obviously none of the subsets consisting of clauses (1) and (2) or of clauses (1) and (3) is unsatisfiable with respect to the restriction theory. This is mirrored in the fact that the two empty RQ-clauses don't have an $\mathcal{R}$-valid restriction.[19] But the two empty clauses together provide a refutation of the whole input clause set $\mathcal{C}$, as $\mathcal{R} \models Pb \vee Pc$.■

A further interesting consequence is obtained for an RQS that has a generic model for its distinguished restrictions. This means that there is a model $\mathcal{G}$ such that each restriction R of the RQS is satisfiable in $\mathcal{G}$ iff R is satisfied by every model of the restriction theory. In that case, for every RQ-unsatisfiable set of RQ-clauses $\mathcal{C}$ there exists a single empty RQ-clause derivable from $\mathcal{C}$ whose restriction is $\mathcal{R}$-valid. Special examples are RQS whose restriction theory is given by a single model, and RQS, whose restriction theory is a definite clause theory[20] and whose restrictions are conjunctions of atoms.

Let $\mathfrak{R}$ be an RQS with restriction theory $\mathcal{R}$ and let $\mathcal{G}$ be a Δ-structure. We call $\mathcal{G}$ ***generic*** for $\mathfrak{R}$ if for all restrictions R in $\mathfrak{R}$

$$\mathcal{G} \models \exists R \text{ iff } \mathcal{R} \models \exists R.$$

5.3 Theorem: *Let $\mathfrak{R}$ be an RQS with generic model $\mathcal{G}$, and let $\mathcal{C}$ be a set of RQ -clauses. Then: $\mathcal{C}$ is RQ-unsatisfiable iff there exists an empty RQ-clause $\square \parallel R$ derivable from $\mathcal{C}$ via RQ -resolution such that $\mathcal{G} \models \exists R$, and hence $\mathcal{R} \models \exists R$.*

Proof: Without loss of generality we can assume that the generic model $\mathcal{G}$ is also in $\mathcal{R}$ (otherwise by the genericity property the class $\mathcal{R}' := \mathcal{R} \cup \{\mathcal{G}\}$ is an equivalent restriction theory for the RQS in the sense that $\mathcal{R}' \models \exists R$ iff $\mathcal{R} \models \exists R$ for all restrictions R).

Let $\mathcal{C}$ be RQ-unsatisfiable. By the Soundness and Completeness Theorem for each model $\mathcal{A}$ in $\mathcal{R}$ and hence especially for the generic model $\mathcal{G}$ there is an empty RQ-clause with restriction R derivable from $\mathcal{C}$ such that the model satisfies $\exists R$.

---

[19] Notice, that the restrictions of the input clauses are all $\mathcal{R}$-valid, because of the first axiom of the restriction theory. That is an RQ-resolution rule, where only resolvents with $\mathcal{R}$-valid restrictions are derived, is not complete. However, see Theorem 4.6.4.

[20] A definite clause contains exactly one positive literal and may be some negative literals. Equational theories or logic programs without negations are definite clause theories.

Conversely, if there is an empty RQ-clause derivable from $\mathcal{C}$, such that its restriction R is satisfied by the generic model: $\mathcal{G} \models \exists R$. Then each model of the restriction theory satisfies this restriction, i.e. $\mathcal{R} \models \exists R$. Hence by the Completeness Theorem the RQ-clause set is RQ-unsatisfiable.■

Corollary: *If the RQS is given by a single model $\mathcal{A}$, then: $\mathcal{C}$ is RQ-unsatisfiable iff there exists an empty RQ-clauses $\square \,||\, R$ derivable from $\mathcal{C}$ such that $\mathcal{A} \models \exists R$.*

For such an RQS with a generic model, a refutation will be obtained whenever an empty RQ-clause is derived whose restriction is satisfied by the generic model. Hence we can modify our resolution rule such that we only allow resolvents whose restrictions are satisfied in the generic model. By the genericity property this means that we allow only resolvents whose restrictions are $\mathcal{R}$-valid, i.e., satisfiable in *all* models of the restriction theory instead of *some* model of the restriction theory. This modified resolution calculus is still complete for RQS with generic models. Of course any refutation of an RQ-unsatisfiable clause set needs only clauses whose restrictions are satisfiable by the generic model and hence $\mathcal{R}$-valid.

Corollary: *Let $\mathfrak{R}$ be an RQS with generic model $\mathcal{G}$ and let $\mathcal{C}$ be a set of RQ-clauses. Let $\mathcal{C}'$ be the subset of $\mathcal{C}$, whose RQ -clauses are $\mathcal{R}$-valid. Then:*
$\mathcal{C}$ is RQ -unsatisfiable iff $\mathcal{C}'$ is RQ-unsatisfiable iff there exists an empty RQ -clauses $\square \,||\, R$ derivable from $\mathcal{C}'$ via RQ-resolution steps with $\mathcal{R}$-valid restrictions only, such that $\mathcal{R} \models \exists R$.

It is well-known that definite clause theories have a least Herbrand model, which is a generic model for queries that are conjunctions of atoms (see for example Lloyd 1984, Theorem 6.6). Hence we can apply our result to constraint theories that have a definite clause axiomatization.

5.4 Theorem: *If the restriction theory $\mathcal{R}$ is given by a definite clause theory and the distinguished restrictions are conjunctions of atoms only, then:*
A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff there exists an empty RQ-clause derivable from $\mathcal{C}$ that has an $\mathcal{R}$-valid restriction.

This last result demonstrates why resolution with sorted clauses (for sort theories with least Herbrand models as e.g. sort hierarchies, Frisch 1989) or with E-unification works analogously to classical resolution: In the first case only initial clauses and resolvents have to be considered whose sort restrictions are non-empty in *all* models of the sort hierarchy (or equivalently in the generic least Herbrand model of the sort declarations). This is for example guaranteed when the sort hierarchy satisfies the condition that for each sort there has to exist a constant of this sort. In the case of an RQS given by an equational theory, Plotkin (1972) shows that resolution with E-unification is complete, when we have clauses with negativ

equality literals only. This corresponds to positive equational restrictions in the RQS view. Our result demonstrates once more from a somewhat different perspective that it is enough to consider clauses whose negativ equality literals are E-unifiable: Plotkin's E-trivialization rule is resolution of negative equality literals with the reflexitivity clause, or equivalently, E-unification of the corresponding restriction viewed as unification problem.

In fact, our completeness result shows a little more in the two cases above. We do not need a sort unification or an E-unification procedure. It is enough to have procedures that are able to decide unifiability. This is similar to lazy-unification as proposed by Ohlbach (1986) in order to avoid the problem that, with non-finitary unification, a resolution step need not terminate; see also (Bürckert 1986). Lazy-unification for E-unification and sort unification was inspired by constrained resolution as proposed by Huet (1978) in his thesis.[21] Nevertheless, unification algorithms provide satisfiability test procedures that have several advantages. The main one is their *incrementality*. For instance testing unifiability of the conjunction of two equational constraints $f(x) = f(f(a))$ and $g(h(x)) = g(h(f(y)))$ is equivalent to testing unifiability of the conjunction of the equational forms of the two unifiers $x = f(a)$ and $x = f(y)$ of these two problems. The latter one is obviously simpler and faster to check then the original one. Thus unification algorithms provide a satisfiability test that works incrementally by reducing the original problems to often much simpler ones.

6. Related Work

As already mentioned our approach is related to theory resolution of Stickel (1985) and to constraint logic programming approaches (Jaffar & Lassez 1986, Dincbas et al. 1988, Höhfeld & Smolka 1988). It follows a tradition of automated deduction research that is concerned with the question of how can we replace “blind” search with more directed search, or still better, with deterministic computation (sort unification, constraint solving). Or to phrase it differently: how can we integrate semantic information about the problem, in order to prune the search space (paramodulation, E-unification, theory resolution).

In fact, all of these improvements are special forms of constrained resolution. Here we only sketch how the two most general approaches, Stickel’s theory resolution calculus and constraint logic programming of Höhfeld and Smolka fit into our framework. Stickel already demonstrated how other less general approaches are instances of his theory resolution, Höhfeld and Smolka do the same for constraint logic programming approaches.

---

[21] Huet used higher order equational constraints and postponed unification until an empty clause with such constraints is derived, in order to avoid the problem that, because of the undecidability of higher order unification, already single resolution steps need not terminate.

Stickel's total theory resolution extends resolution by a generalization of the notion of complementary literals. A total theory resolution step takes a couple of clauses, and from each clause a subclause is chosen. If this set of subclauses is complementary in the sense that it is unsatisfiable with respect to the given theory T[22], then the remaining parts of the chosen clauses are joined together to form a total theory resolvent:

$$\frac{\begin{array}{l} C_1 \vee D_1 \\ C_2 \vee D_2 \\ \ldots \\ C_n \vee D_n \end{array}}{D_1 \vee \ldots \vee D_n} \quad \text{if } C_1 \wedge \ldots \wedge C_n \text{ is T-unsatisfiable}$$

If the literals of these C_i are literals of the background theory, we could see their negation $\neg C_i$ as constraints R_i.[23] If we further modify constrained resolution by a second deduction rule, that takes any couple of constrained clauses and adds as a "theory resolvent" the clause obtained by the union of the literals of the parents and whose restriction is the union of the parent's constraints and is valid with respect to the constraint theory, then this is equivalent to a total theory resolution step.

$$\frac{\begin{array}{l} D_1 \parallel R_1 \\ D_2 \parallel R_2 \\ \ldots \\ D_n \parallel R_n \end{array}}{D_1 \vee \ldots \vee D_n} \quad \text{if } \exists\, R_1 \vee \ldots \vee R_n \text{ is T-valid}$$ [24]

However, as our result shows, such theory resolution steps could be delayed until the end, which means they have only to be applied to constrained empty clauses.

Höhfeld and Smolka investigate constrained (SLD-)resolution for definite clauses with constraints as a goal reduction procedure. As their application in logic programming requires, they are only interested in answers to queries, but not in refutation proofs. In fact, their approach cannot serve as a theorem proving system since they will stop whenever an answer goal – a constrained empty clause – is returned. But in general this does not prove the query to be a theorem of the constrained logic program. This implies only that the query holds just in those models of the constraint theory that satisfy the answer constraints. Our result shows that, if we want to use their approach as a theorem proving procedure, we have to derive

---

22 A set of clauses is T-unsatisfiable iff it has no T-models.

23 Remember that $C \vee D$ is equivalent to $\neg C \Rightarrow D$, which in turn is equivalent to $D \parallel \neg C$, as constraints can be seen as preconditions.

24 A formula is T-valid iff it is a logical consequence of T. Hence T-validity of $\exists\, R_1 \vee \ldots \vee R_n$ is equivalent to T-unsatisfiability of $C_1 \wedge \ldots \wedge C_n$.

"enough" answers in order to have, for each model of the constraint theory, an answer constraint that is satisfied by this model. If the constraint theory is just one model, as in common constraint logic programming approaches (Jaffar & Lassez 1986, Dincbas et al. 1988), then of course a single answer determines also a proof for the query.

7. Conclusion

We have introduced a general method to handle clauses over constraints, or equivalently, clauses whose variables are bound by restricted quantifiers. Our approach generalizes well-known methods of restricted quantification like sorted resolution and highlights constraint logic programming languages in the theorem proving context. It provides a new view of E-unification and other more general ways of building in theories, as for example in theory resolution. In fact, it generalizes theory resolution in that it allows the specification of theories not necessarily by a first order axiomatization but by any other form that describes a class of models.

Hence our results can be applied to build in any kind of theory, especially, when we are not interested in general formulae but in a restricted class of formulae that can be seen as constraints or quantifier restrictions, i.e., that is at least closed under conjunction. However, it could also be used as a method to reason about theories on the meta-level of building strong theorem provers, as it might be possible to replace a restricted quantification system – restriction theory and restrictions – by an equivalent one. For example, in (Buntine & Bürckert 1989, Bürckert 1988) we argue that AC-unification is equivalent to AC1-disunification.[25] This can be justified by considering the following two RQS that are equivalent in the sense that their constraints have the same solutions with respect to their constraint theory: one is the equational theory AC of associative and commutative function symbols (more exactly the corresponding free AC-algebra) together with equational constraints (the AC-unification problems), the other is the theory AC with additional unit *1* (more exactly the free AC1-algebra) together with equations and certain negated equations (AC1-disunification problems) as constraints.

Our approach could also be used to combine knowledge representation languages of the KL-ONE family (Brachman & Schmolze 1985, Nebel 1989) with predicate logic. In (Bürckert 1990) we sketch a logical framework for such a concept logic. Here RQ-resolution, where the satisfiability test for restrictions is just the consistency check for concept descriptions, could be a suitable inference method for such a hybrid knowledge representation system based on concept logic.

---

[25] AC-unification may return millions of most general unifiers, which are represented by a few solutions of a corresponding AC1-disunification problem (Bürckert et al. 1988).

We like to emphasize that our approach is not only useful when algorithms that test satisfiability of the restrictions are available. But the method may also be used if there is some knowledge base for the background theory, that provides a couple of results like theorems etc. for this theory. We could then use RQ-resolution in order to reduce the given problem to restriction formulae – the restrictions of the empty clauses – that could be found in the knowledge base.

Obviously there remain many open problems: How should several constraint theories be combined? Do there exist (incremental) satisfiability tests for their constraints? And of course, a most pressing problem is to gain experience with some actual implementations of constraint theories. In (Bürckert 1990) we investigate some general properties of constraint reduction or simplification systems that can be used as incremental satisfiability test systems.

Acknowledgements. I like to thank Norbert Eisinger, Werner Nutt, Hans Jürgen Ohlbach, and Gert Smolka for their support and intensive discussions that helped me to understand what's going on in constrained resolution. This research was supported by the German Bundesministerium für Forschung und Technologie under grant ITW 8903 0.

References

K.H. Bläsius, H.-J. Bürckert (eds.). *Deduction Systems in Artificial Intelligence*. Horwood, Chichester, 1989

R.J. Brachman, J.G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science 9(2)*, 171-216, 1985.

W. Buntine, H.-J. Bürckert. On Solving Equations and Disequations. SEKI-Report SR-89-03, Universität Kaiserslautern, 1989

H.-J. Bürckert. Lazy Theory Unification in Prolog: An Extension of the Warren Abstract Machine. *Proc. of German Workshop on Artificial Intelligence, Springer Informatik-Fachberichte 124*, 277-288, 1986

H.-J. Bürckert. Lazy E-Unification – A Method to Delay Alternative Solutions, SEKI-Report SR-87-07, Universität Kaiserslautern, 1987

H.-J. Bürckert. Solving Disequations in Equational Theories. *Proc. of Conf. on Automated Deduction, Springer LNCS 310*, 517-526, 1988

H.-J. Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*. Dissertation, in preparation, 1990

H.-J. Bürckert, A. Herold, D. Kapur, J.H. Siekmann, M.E. Stickel, M. Tepp, H. Zhang. Opening the AC-Unification Race, *J. of Automted Reasoning 4*, 465-474, 1988

C.-L. Chang, R.C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York – London, 1973

A. Colmerauer. Equations and Inequations on Finite and Infinite Trees. *Proc. of Conf. on Fifth Generation Computer Systems*, 85-99, 1984

M. Dincbas, P. van Hentenryck, H. Simonis, A. Aggoun, T. Graf, F. Berthin. The Constraint Logic Programming Language CHIP. *Proc. of Conf. on Fifth Generation Computer Systems*, 1988

A. Frisch. A General Framework for Sorted Deduction: Fundamenrtal Results on Hybrid Reasoning. *Conf. on Principles of Knowledge Representation and Reasoning*, 126-136, 1989

J. Gallier, S. Raatz. SLD-Resolution Methods for Horn Clauses with Equality Based on E-Unification. *Proc. of Symp. on Logic Programming*, 1986

J. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Harper and Row, 1986

M.R. Genesereth, N.J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987

J.A. Goguen, J. Meseguer. Equalities, Types, Modules, and Generics for Logic Programming. *J. of Logic Programming 1*, 179-210, 1984

M. Höhfeld, G. Smolka. Definite Relations over Constraint Languages. LILOG-Report 53, IBM Deutschland, 1988

G. Huet. *Constrained Resolution – A Complete Method for Higher Order Logic*. Ph.D. Thesis, Case Western University, 1972

J. Jaffar, J.-L. Lassez. Constrained Logic Programming. *Proc. of ACM Symp. on Principles of Programming Languages*, 111-119, 1987

J. Jaffar, J.-L. Lassez, M. Maher. Logic Programming Scheme. In: *Logic Programming: Functions, Relations, Equations* (eds. D. De Groot, G. Lindstrom), Prentice Hall, 1986

C. Kirchner (ed.). Special Issue on Unification. *J. of Symbolic Computation*, 1989

R. Kowalski. *Logic for Problem Solving*. North-Holland, 1979

J.W. Lloyd. *Foundations of Logic Programming*. Springer, 1984

D.W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland, 1978

B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Dissertation, Universität Saarbrücken, 1989

A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik (in German). *Mathematische Annalen 145*, 297-333, 1962

H.J. Ohlbach. The Semantic Clause Graph Procedure – A First Overview. *Proc. of German Workshop on Artificial Intelligence, Springer Informatik-Fachberichte 124*, 218-229, 1986

G. Plotkin. Building in Equational Theories. *Machine Intelligence 7*, 1972

J.A. Robinson. A Machine Oriented Logic Based on the Resolution Principle. *Journal of the ACM 12(1)*, 1965

M. Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Springer LNAI 395, 1989

J.R. Shoenfield. *Mathematical Logic*. Addison Wesley, 1967

J.H. Siekmann. Unification Theory – A Survey. In: Special Issue on Unification (ed. C. Kirchner), *J. of Symbolic Computation*, 1989

G. Smolka. *Logic Programming over Polymorphically Order-Sorted Types*. Dissertation, Universität Kaiserslautern, 1989

G. Smolka, W. Nutt, J.A. Goguen, J. Meseguer. Order-Sorted Equational Computation. In: *Resolution of Equations in Algebraic Structures* (eds. H. Ait-Kaci, M. Nivat). Prentice Hall, 1989

M.E. Stickel. Automated Deduction by Theory Resolution. *J. of Automated Reasoning 1(4)*, 1985

C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Pitman & Morgan Kaufmann Publishers, Research Notes in Artificial Intelligence, 1987

L. Wos, G.A. Robinson. Paramodulation and the Set of Support. *Proc. of Symp. on Automated Demonstration*, Springer, 1970

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
67608 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymem ftp von ftp.dfki.uni-kl.de (131.246.241.100) unter pub/Publications bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or per anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) under pub/Publications.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

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-92-53
Werner Stephan, Susanne Biundo:
A New Logical Framework for Deductive Planning
15 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 Profitlich, Thomas Rist:
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, 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 Fehrle, 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

RR-93-17
Rolf Backofen: Regular Path Expressions in Feature Logic
37 pages

RR-93-18
Klaus Schild: Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-20
Franz Baader, Bernhard Hollunder: Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-22
Manfred Meyer, Jörg Müller: Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy: Comparative Study of Connectionist Simulators
20 pages

RR-93-24
Rainer Hoch, Andreas Dengel: Document Highlighting — Message Classification in Printed Business Letters
17 pages

RR-93-25
Klaus Fischer, Norbert Kuhn: A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-26
Jörg P. Müller, Markus Pischel: The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-27
Hans-Ulrich Krieger:
Derivation Without Lexical Rules
33 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker: Feature-Based Allomorphy
8 pages

RR-93-33
Bernhard Nebel, Jana Koehler:
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-34
Wolfgang Wahlster:
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.):
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt: Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-38
Stephan Baumann: Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf:
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-41
Winfried H. Graf: LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-42
Hubert Comon, Ralf Treinen:
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

DFKI Technical Memos

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 Cognitiv 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

TM-93-02
Pierre Sablayrolles, Achim Schupeta:
Conlfict Resolving Negotiation for COoperative Schedule Management
21 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer:
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

DFKI Documents

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-93-05
Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster:
PPP: Personalized Plan-Based Presenter
70 pages

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-$).

D-93-07
Klaus-Peter Gores, Rainer Bleisinger:
Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-08
Thomas Kieninger, Rainer Hoch: Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer:
TDL ExtraLight User's Guide
35 pages

D-93-10
Elizabeth Hinkelman, Markus Vonerden,Christoph Jung: Natural Language Software Registry
(Second Edition)
174 pages

D-93-11
Knut Hinkelmann, Armin Laux (Eds.):
DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein:
RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-14
Manfred Meyer (Ed.): Constraint Processing – Proceedings of the International Workshop at CSAM'93, July 20-21, 1993
264 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).