Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

**Research Report**
RR-93-06

# On Skolemization in Constrained Logics

**Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux**

**March 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

# On Skolemization in Constrained Logics

**Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux**

DFKI-RR-93-06

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-8903 0 and ITW-9201).

# On Skolemization in Constrained Logics

Hans-Jürgen Bürckert Bernhard Hollunder
Armin Laux
Deutsches Forschungszentrum für künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11, Germany
e-mail: {buerckert, hollunder, laux}@dfki.uni-sb.de

**Abstract**

First-order logics allows one to quantify over all elements of the universe. However, it is often more natural to quantify only over those elements which satisfy a certain condition. Constrained logics provide this possibility by introducing restricted quantifiers $\forall_{X:R} F$ and $\exists_{X:R} F$ where $X$ is a set of variables, and which can be read as "$F$ holds for all elements satisfying the restriction $R$" and "$F$ holds if there exist elements which satisfy $R$", respectively.

In order to test unsatisfiability of a set of such formulas using an extended resolution principle, one needs a procedure which transforms them into a set of constrained clauses. Such a procedure causes more problems than the classical transformation of first-order formulas into a set of clauses. This is due to the fact that quantification over the empty set may occur. Especially, a modified Skolemization procedure has to be used in order to remove restricted existential quantifiers.

In this paper we will give a procedure that transforms formulas with restricted quantifiers into a set of clauses with constraints preserving unsatisfiability. Since restrictions may be given by sorts this procedure can, e.g., be applied to sorted logics where empty sorts may occur. The obtained clauses are of the form $C \,||\, R$ where $C$ is an ordinary clause and $R$ is a restriction, and which can be read as "$C$ holds if $R$ holds". They can be tested on unsatisfiability via constrained resolution. Finally, we introduce so-called constraint unification which can be used for optimization of constrained resolution if certain conditions are satisfied.

# Contents

# 1 Introduction

## 1.1 Motivation

Intelligent problem solving is based on the representation and the use of domain specific knowledge. However, deductive systems based on the classical resolution principle in general do not allow the incorporation of methods for domain specific problem solving: A set of (first-order) formulas is transformed into a set of clauses which then is tested on unsatisfiability by a more or less blind search. Thereby the formulas are transformed into clause form regardless from the fact whether or not there exist special purpose algorithms for solving certain subproblems efficiently. Suppose, for example, the following part of a knowledge base to be given

(A) Elizabeth is a queen
(B) Every queen is a woman
(C) Every woman likes to wear elegant clothes

and we are interested in the query whether Elizabeth likes to wear elegant clothes. Expressing these sentences and the query in clause form we obtain the following result:

(1) *queen* (*Elizabeth*)
(2) $\neg$ *queen* $(x_1) \vee$ *woman* $(x_1)$
(3) $\neg$ *woman* $(x_2) \vee$ *likes-to-wear-elegant-clothes* $(x_2)$
(4) $\neg$ *likes-to-wear-elegant-clothes* (*Elizabeth*).

Of course, it is not a hard problem to prove unsatisfiability of this clause set by applying a few resolution steps. But, in general, large numbers of clauses have to be handled, and the search for a refutation becomes very expensive. Thus it seems to be more appropriate to apply efficient algorithms to treat subproblems, e.g., for dealing with persons and relationships between them. However, the framework of classical resolution does not support the integration of those algorithms; everything has to be proved by a uniform blind search.

Based on this observation an extended resolution principle, so-called constrained resolution, has been developed in [Bür91], [Bür93]. It generalizes a couple of former approaches like resolution with $E$-unification [Plo72] or with sort-unification [Wal88] and enables the integration of domain specific algorithms into the refutation procedure. The main idea is to equip each clause $C$ with a restriction (or "constraint") $R$ which is an open formula with $n$ free variables. For sake of readability we will restrict ourselves to unary restrictions in the introduction. These restrictions represent domain specific knowledge and restrict the possible assignments to the variables occurring in $C$. One then obtains so-called constrained clauses which are written as $C \parallel R$, and which intuitively can be read as "$C$ holds if $R$ holds". In principle, constrained resolution

works as follows: Given two constrained clauses $C_1 \,||\, R_1$ and $C_2 \,||\, R_2$, a new constrained clause $C \,||\, R$ is generated, where $C$ is obtained from $C_1$ and $C_2$ by classical resolution. Therefore, a most general unifier $\sigma$ of the two resolved literals in $C_1$ and $C_2$ is computed. The restriction $R$ is then obtained by applying $\sigma$ to the conjunction of $R_1$ and $R_2$.

Of course, problems are usually not given as sets of constrained clauses. We suppose problems to be given by a restriction theory, which represents domain specific knowledge, and a set of extended first-order formulas where domain specific information can be represented explicitly in restrictions. This is done by generalizing the classical quantifiers $\forall x\ F(x)$ and $\exists x\ F(x)$ to so-called restricted quantifiers $\forall_{x:R(x)}\ F(x)$ and $\exists_{x:R(x)}\ F(x)$, where $R$ is a restriction and $F$ is a formula. The interpretation of such formulas is given by "$F$ holds for all elements satisfying $R$" and "$F$ holds if there exists an element that satisfies $R$", respectively. Here, satisfaction of the restriction $R$ is meant with respect to the ("built-in") restriction theory. These extended formulas are then transformed into a set of constrained clauses which are tested on unsatisfiability by the constrained resolution principle.

The problem of how to transform formulas with restricted quantifiers into constrained clauses is only sketched in [Bür91]. In Section 3 we will present such a transformation algorithm. This algorithm differs from the classical transformation of first-order formulas into clauses since quantification over the empty set may occur. This is the case if a restriction $R$ is interpreted as the empty set in some interpretations $I$, i.e., $R^I(u)$ is false for every element $u$ in the universe of $I$. The transformation algorithm has to handle both the case in which the interpretation of a restriction $R$ is the empty set, as well as the case that the interpretation of $R$ is a non-empty set. Since the truth values of $\forall_{x:R(x)}\ F$ and $\exists_{x:R(x)}\ F$ depend on $F$ in interpretations which interpret $R$ as non-empty set, but do not depend on $F$ in interpretations which interpret $R$ as empty set, this task is not obvious. To overcome this problem, we introduce the method of quantifier splitting which makes the distinction between quantification over empty and non-empty sets explicit (see Subsection 3.1).

For the transformation of formulas with restricted quantifiers into constrained clauses, the restricted existential quantifiers have to be eliminated. But in contrast to classical Skolemization we cannot use a free interpretation of Skolem function symbols, since we have to take the restrictions of the restricted quantifiers and the restriction theory into account. When doing this we obtain tuples of Skolem functions which have $R_1 \times \ldots \times R_n$ as domain if the actual restricted existential quantifier occurs in the scope of restricted universal quantifiers having restrictions $R_1, \ldots, R_n$. The range of such a tuple is given by the restriction $R$ of the existential quantifier to be eliminated. After Skolemization, which is described in Subsection 3.2, the formulas are transformed into a conjunctive normal form. From this normal form constrained clauses can be generated immediately (see Subsection 3.3).

Let us now reconsider the main idea of constrained resolution in more detail: Firstly, a certain part of information is represented separately, namely the part of information for which we possess of special purpose representation and inference algorithms. We assume this information to be given by a set of first-order formulas over some signature $\Delta$, called restriction theory. Formulas over $\Delta$ can then be used as restrictions of constrained clauses. A set $\mathcal{C}$ of constrained clauses is tested on unsatisfiability by successively adding new constrained clauses to $\mathcal{C}$ using the constrained resolution principle. But, in contrast to the classical resolution principle, the derivation of a single empty constrained clause is in general not sufficient to prove unsatisfiability of $\mathcal{C}$. Note, that $\square \,||\, R$ can be read as "there is a contradiction if $R$ holds", i.e., there is a contradiction in each model of the restriction theory which satisfies $R$.

In Section 4 we present a refutation procedure for a set of formulas with restricted quantifiers and some of optimizations. This procedure, given in Subsection 4.1, is based on the constrained resolution principle and the results of Section 3. In [Bür91], [Bür93] it is shown that a constrained clause $C \,||\, R$, whose restrcition $R$ is not satisfied in any model of the restriction theory, cannot contribute to prove unsatisfiability of a constrained clause set. Thus, as a source of optimization, these constrained clauses can be omitted. In Subsection 4.2 we introduce constraint unification which provides a generalization of this optimization if the restriction theory satisfies a certain condition (see Subsection 4.3). It will be shown that this condition is satisfied if the restriction theory does not contain equations, and restrictions do neither contain equations nor disequations.

The results of the present paper have some interesting applications. As argued above, constraint resolution can be used as a method to incorporate domain specific problem solving into the resolution principle. But moreover, since sorts can be seen as unary restrictions, Subsection 3.2 provides an algorithm for Skolemization in sorted logics where empty sorts may occur. Thus, Section 3 gives an algorithm for translating first-order formulas with sort information into a set of constrained clauses. In this case, the sort theory is represented in the restriction theory. Additionally, since we allow the restriction theory to have an arbitrary first-order axiomatization, we can allow operations like, e.g., union and intersection to define sorts. In [BHL93] we show how to use the well-know concept language $\mathcal{ALC}$ (which can be seen as a generalized sorted logic) to define a restriction theory.

## 1.2 Related Work

The idea of clauses with restrictions has already been introduced by Höhfeld and Smolka [HS88] who did not aim at a refutation procedure, but in query answering for logic programming. Thus they do not need Skolemization. The basis of our approach is [Bür91], where a logic with restricted quantifiers and rules for constrained resolution

and constrained factoring have been introduced. However, the problem of how to transform formulas with restricted quantifiers into RQ-clauses has only been sketched there.

As mentioned above, constrained resolution generalizes several approaches of building in theories into resolution based deduction systems (see [Bür91], [Bür93]). For one of them, sorted logics, which have been introduced as a basis for restricted quantification by Oberschelp [Obe62], there are results on Skolemization. The differences to our approach are that sorts are unary predicates, i.e., atomic open formulas that restrict only single variables, and that the constraint theories are just sort hierarchies. Our restrictions can be arbitrary open formulas constraining tuples of variables. In addition, sorts are usually assumed to be non-empty. Hence, Skolemization is not really a problem; it is rather similar to the classical case (cf., for example, [Wal87], [Wal88], [Sch89]).

In [WO90] a many-sorted logic is presented which allows potentially empty sorts. For this logic a transformation of formulas into clause normal form is given. Building upon this work, [Wei92] presents a sound and complete resolution based calculus for a sorted first-order language, where sorts may denote empty sets and conditioned sort declarations are allowed. For this calculus a new unification algorithm is introduced and it is shown that the sort declarations this algorithm is built upon have to be changed dynamically during the deduction process. In order to distinguish between empty and non-empty interpretations of sorts their calculus is equipped with specialized rules. In contrast to that, our quantification splitting approach is based on an *a priori* and explicit case distinction on the interpretations of restrictions as empty and non-empty sets.

Cohn [Coh92] discusses a many-sorted logic with possibly empty sorts, but starts with a set of clauses and thus does not give a Skolemization procedure. Finally, in [FS90] and [FS91], transformation of modal logic sentences into a logic with restrictions is treated. They give a systematic transformation of certain modal logics into constrained first-order logics. But they only address logics which allow a "classical" Skolemization procedure.

The use of terminological logics as restrictions is discussed in [BBH+90]. But in this approach, problems have been assumed to be given as a set of constrained clauses without function symbols, together with a restriction theory. For this case, algorithms for deciding satisfiability and validity of restrictions have been given. In [BHL93] we will show that testing satisfiability and validity becomes more complicated if problems are not given by a constrained clause set but by a set of first-order formulas with restricted quantifiers together with a restriction theory. The reason for this lies in the fact that function symbols may be introduced by Skolemization and we thus need algorithms to test satisfiability and validity of restrictions containing function symbols.

# 2 A Logic with Restricted Quantifiers

In this section we introduce a logic with restricted quantifiers (RQ for short). Firstly, in Subsection 2.1, we give the syntax of RQ-formulas and RQ-clauses. The semantics is given in Subsection 2.2. Finally, in Subsection 2.3, we introduce a resolution principle for RQ-clauses.

## 2.1 Syntax

A **signature** $\Sigma$ consists of three pairwise disjoint sets of symbols: a set $F_\Sigma$ of function symbols, a set $V_\Sigma$ of variables, and a set $P_\Sigma$ of predicate symbols.

Every function symbol $f$ has a nonnegative arity and every predicate symbol $p$ has a positive arity. Function symbols of arity zero are called **constant symbols**. A **term** is either a variable or a string of the form $f(s_1, \ldots, s_n)$, where $f$ is an n-ary function symbol and $s_1, \ldots, s_n$ are terms. A term without variables is called **ground**. The set of all terms w.r.t. $\Sigma$ is denoted by $T_\Sigma(V)$, the set of all ground terms by $T_\Sigma$. An **atom** is a string of the form $p(t_1, \ldots, t_n)$, where $p$ is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms. A **formula** is either an atom or a string of the form $\neg F$, $F \star G$, $\forall x F$ or $\exists x F$, where $F$, $G$ are formulas, $x$ is a variable, and $\star \in \{\vee, \wedge, \leftrightarrow, \rightarrow\}$.

The **scope** of the quantifier $Q \in \{\forall, \exists\}$ in $QxF$ is the subformula $F$ except all subformulas $\forall x G$ or $\exists x G$ of $F$ which start with a leading quantifier over the same variable $x$. A variable $x$ occuring in the scope of a quantifier $QxF$ is **bound**, otherwise it is **free**. A formula without any free variable is **closed**, otherwise it is **open**. Given a formula $F$ with exactly the free variables $x_1, \ldots, x_n$, then $\forall F$ denotes the **universal closure** $\forall x_1 \ldots \forall x_n F$ of $F$ and $\exists F$ denotes the **existential closure** $\exists x_1 \ldots \exists x_n F$ of the formula $F$.

Following our idea of incorporating methods for domain specific problem solving we will distinguish between background knowledge (i.e. domain specific knowledge) and the foreground information. Because of this distinction we will use a restricted quantification system (RQS) to represent the background knowledge and an RQ-signature which extends the RQS by foreground language symbols. An RQS consists of three parts, that is, a signature $\Delta$, a set of (open) $\Delta$-formulas, the admissable restriction formulas, and a restriction theory, which represents the possible interpretations of the restrictions.

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

- a signature $\Delta$ with equality,

- a set of (open) $\Delta$-formluas, the **restriction formulas** or **restrictions** which are closed under conjunction and instantiation of variables, and
- a theory over $\Delta$, the **restriction theory**.

The restriction theory can be given either as a set of axioms or as a set of $\Delta$-structures. Note that in the latter case the restriction theory needs not to have a first-order axiomatization.

A **signature with restricted quantifiers** or an **RQ-signature** $\Sigma$ consists of an RQS $\mathcal{R}$ together with an additional set of predicate symbols $\mathcal{P}_\Sigma$ and an additional set of function symbols $\mathcal{F}_\Sigma$, both disjoint from the symbols of $\Delta$. In order to simplify our notation we will use the prefix "$\Sigma$-" if we denote objects—terms, atoms, formulas, etc.—that are built upon symbols of $\mathcal{F}_\Sigma$ and $\mathcal{P}_\Sigma$, and variables of $V_\Delta$ only.

Given such an RQ-signature $\Sigma$ we now define formulas with restricted quantifiers w.r.t. $\Sigma$. Therefore we allow quantifiers to be indexed not only by variables, but by pairs of a variable set $X$ and a restriction formula $R$. These extended quantifiers are written as $\forall_{X:R}$ and $\exists_{X:R}$, and we call them **restricted quantifiers**. Note that the restrictions represent background information and the $\Sigma$-formulas foreground information, respectively. We define **RQ-formulas over** $\Sigma$ by

1. all $\Sigma$-atoms are RQ-formulas,
2. $\forall_{X:R}F$ and $\exists_{X:R}F$ are RQ-formulas, where $F$ is an RQ-formula, $R$ is a restriction, and $X$ contains at least the free variables in $R$,
3. $\forall xF$, $\exists xF$, $F \wedge G$, $F \vee G$, $\neg F$, $F \rightarrow G$, $F \leftrightarrow G$ are RQ-formulas, where $F$, $G$ are RQ-formulas, and $x$ is a variable.

Note that in second definition the formula $F$ may contain free variables of $X$ that are now bounded by the restricted quantifiers $\forall_{X:R}$ or $\exists_{X:R}$. The formula $R$ is called the restriction for the variables of $X$ and can be seen as a sieve that filters out the admissable assignments of elements to these variables.

**Example 2.1** Let $\mathcal{R}$ be an RQS with the predicate symbols $P_\Delta = \{p, q\}$ and the function symbols $F_\Delta = \{f\}$. Assume that the restrictions are given by the $\Delta$-formulas $p(x, y)$ and $q(z)$. Then $p(f(x), y)$ and $q(z_1) \wedge q(f(f(x)))$ are restrictions because restrictions are defined to be closed under instantiation of variables and conjunction. Let us now extend $\mathcal{R}$ to an RQ-signature $\Sigma$, which introduces the additional predicate symbol $\mathcal{P}_\Sigma = \{r\}$ and the additional unary function symbol $\mathcal{F}_\Sigma = \{g\}$. Then $\forall_{x,y,z:p(x,y)} r(g(x))$ and $r(g(x))$ are RQ-formulas, but $\forall_{x:q(g(x))} r(g(x))$ and $p(g(x))$ are *not* since the function symbol $g \in \mathcal{F}_\Sigma$ must not occur in a restriction, and the predicate symbol $p \in P_\Delta$ must only occur in restrictions. ∎

We now introduce **RQ-clauses** (or **constrained clauses**) which consist of a $\Sigma$-clause $C$, the so-called **kernel**, together with a restriction $R$. Such a clause is written as $C \parallel R$ and represents the RQ-formula $\forall_{X:R} C$, where $X$ contains exactly the free variables in $C$ and $R$. If $C$ is empty we call it an **empty RQ-clause**, written as $\square \parallel R$.

In our RQ-signature $\Sigma$ e.g. $r(f(x)) \vee r(f(y)) \parallel p(x,y)$ and $\square \parallel p(x,y) \wedge q(z)$ are RQ-clauses.

Without loss of generality we can assume that the set $\mathcal{F}_\Sigma$ of foreground function symbols is empty. We can always achieve this by modifying an RQS as follows: the first step is to extend the background signature $\Delta$ by the symbols in $\mathcal{F}_\Sigma$. But after doing this we are neither allowed to use these symbols in our foreground language (since $\mathcal{F}_\Sigma$ is empty now), nor to use them in a restriction, because the set of restrictions does not contain any formula over these function symbols up to now.[1] Of course, we want to be able to express the same facts before and after the extension of $F_\Delta$. To guarantee this we use **unfolding**, i.e., we replace every $\Sigma$-term, e.g. $f(x)$, by a new variable $z$, and then we enlarge the set of restrictions by the equation $z = f(x)$. Therefore the second step is to extend the set of restrictions such that it contains in addition all equations of the form $x = t$, where $x$ is a variable and $t$ is a $\Sigma$-term. Finally we expand the restriction theory by the new function symbols.

**Example 2.2** Let us reconsider the RQS $\mathcal{R}$ and the RQ-signature $\Sigma$ of example 2.1, i.e.,

$$P_\Delta = \{p, q\},\ F_\Delta = \{f\} \qquad \text{and} \qquad P_\Sigma = \{r\}\ ,\ F_\Sigma = \{g\}.$$

We already know $\forall_{x,y,z:p(x,y)} r(g(x))$ to be an RQ-formula. After the extension of $F_\Delta$ by the additional function symbol $g$ we obtain

$$P_\Delta = \{p, q\},\ F_\Delta = \{f, g\} \qquad \text{and} \qquad P_\Sigma = \{r\}, F_\Sigma = \emptyset$$

After doing this $\forall_{x,y,z:p(x,y)} r(g(x))$ is no longer an RQ-formula, since $r(g(x))$ is not a $\Sigma$-formula w.r.t. the modified RQ-signature. However, by the extension of the restrictions by $z = f(x)$ and unfolding we obtain the formula $\forall_{x,y,z:p(x,y) \wedge z = f(x)} r(z)$, which is an RQ-formula w.r.t. the modified RQ-signature. ■

## 2.2 Semantics

We first recapitulate the semantics of first-order formulas (with equality) by using $\Sigma$-structures and $\Sigma$-assignments. Then, we extend these $\Sigma$-structures to RQ-structures, which gives a semantics of RQ-formulas.

[1] Note that the original signature $\Delta$ of the RQS did not contain any of these additional function symbols, and restrictions are (open) formulas over this original signature $\Delta$.

Let $\Sigma$ be a signature. A $\Sigma$-**structure** $\mathcal{A}$ consists of a non-empty universe $U^{\mathcal{A}}$ and maps each $n$-ary function (predicate) symbol to an $n$-ary function (relation). A $\Sigma$-**assignment** $\alpha$ maps each variable $x \in V_\Sigma$ to an element $u \in U^{\mathcal{A}}$. This mapping is extended to terms as usual: if $t \equiv f(t_1, \ldots, t_n)$ is an arbitrary term, then we define $\alpha(f(t_1, \ldots, t_n)) := f^{\mathcal{A}}(\alpha(t_1), \ldots, \alpha(t_n))$.

To simplify our notation we will use some abbreviations: $F[x_1, \ldots, x_n]$ denotes a formula $F$ that contains at least the free variables $x_1, \ldots, x_n$. With $F[x \leftarrow t]$ we denote the formula which is obtained from $F$ by replacing every free occurrence of the variable $x$ by the term $t$. Analogously, $F[x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n]$ denotes the replacement of every free variable $x_i$ by the term $t_i$, $i = 1, \ldots, n$. If $u$ is an element of the universe, then $\alpha_{[x \leftarrow u]}$ denotes the $\Sigma$-assignment $\alpha$ with the exception of the explicit assignment of $u$ to $x$. As above, this abbreviation is extended to $\alpha_{[x_1 \leftarrow u_1, \ldots, x_n \leftarrow u_n]}$, where $x_1, \ldots, x_n$ and $u_1, \ldots, u_n$ are variables and elements of the universe, respectively.

Given a $\Sigma$-structure $\mathcal{A}$ and a $\Sigma$-assignment $\alpha$ we define **satisfiability** of a formula recursively as follows:

$$
\begin{array}{lll}
(\mathcal{A}, \alpha) \models p(t_1, \ldots, t_n) & \text{iff} & p^{\mathcal{A}}(\alpha(t_1), \ldots, \alpha(t_n)) \\
(\mathcal{A}, \alpha) \models s = t & \text{iff} & \alpha(s) \text{ und } \alpha(t) \text{ are identical in } \mathcal{A} \\
(\mathcal{A}, \alpha) \models F \vee G & \text{iff} & (\mathcal{A}, \alpha) \models F \text{ or } (\mathcal{A}, \alpha) \models G \\
(\mathcal{A}, \alpha) \models F \wedge G & \text{iff} & (\mathcal{A}, \alpha) \models F \text{ and } (\mathcal{A}, \alpha) \models G \\
(\mathcal{A}, \alpha) \models \neg F & \text{iff} & \text{not } (\mathcal{A}, \alpha) \models F \\
(\mathcal{A}, \alpha) \models F \rightarrow G & \text{iff} & (\mathcal{A}, \alpha) \models F \text{ implies } (\mathcal{A}, \alpha) \models G \\
(\mathcal{A}, \alpha) \models F \leftrightarrow G & \text{iff} & (\mathcal{A}, \alpha) \models F \text{ iff } (\mathcal{A}, \alpha) \models G \\
(\mathcal{A}, \alpha) \models \forall x F & \text{iff} & \text{for every } u \in U^{\mathcal{A}} \text{ holds } (\mathcal{A}, \alpha_{[x \leftarrow u]}) \models F \\
(\mathcal{A}, \alpha) \models \exists x F & \text{iff} & \text{there is a } u \in U^{\mathcal{A}} \text{ such that } (\mathcal{A}, \alpha_{[x \leftarrow u]}) \models F
\end{array}
$$

Furthermore, $(\mathcal{A}, \alpha) \models \mathit{true}$ and $(\mathcal{A}, \alpha) \not\models \mathit{false}$ for every $\Sigma$-structure $\mathcal{A}$ and every $\Sigma$-assignment $\alpha$.

A $\Sigma$-structure $\mathcal{A}$ is a $\Sigma$-**model** of a formula $F$, written $\mathcal{A} \models F$, if and only if $(\mathcal{A}, \alpha) \models F$ holds for every $\Sigma$-assignment $\alpha$. A formula $F$ is called **valid** if and only if every $\Sigma$-structure $\mathcal{A}$ is a $\Sigma$-model of $F$. Two formulas are **equivalent** iff they have exactly the same models.

For the semantics of restricted quantifiers can be given by **relativization**, that is, one can transform any RQ-formula into an equivalent first-order formula by replacing

$$
\begin{array}{lll}
\forall_{X:R} F & \text{by} & \forall x_1 \ldots \forall x_n (R \rightarrow F) \\
\exists_{X:R} F & \text{by} & \exists x_1 \ldots \exists x_n (R \wedge F)
\end{array}
$$

where $X = \{x_1, \ldots, x_n\}$ is a set of variables.

We will use an alternative characterization which maintains the separation of foreground and background symbols. Since an RQ-signature $\Sigma$ is the extension of an under-

lying RQS $\mathcal{R}$, and since we already know the $\Delta$-models given in the restriction theory of $\mathcal{R}$, we only need to interpret additionally the new symbols of the RQ-signature. Therefore we interpret RQ-formulas in structures that expand the $\Delta$-structures of the restriction theory by the new symbols of $\Sigma$. The variables of the restricted quantifiers have of course to be assigned only with elements of those structures which satisfy the restrictions. Let us define this more precisely.

Let $\Sigma$ be an RQ-signature over the RQS $\mathcal{R}$. An **RQ-structure over** $\Sigma$ is a $\Sigma$-structure $\mathcal{A}$ such that the restriction of $\mathcal{A}$ to $\Delta$, written $\mathcal{A}|_\Delta$, is one of the $\Delta$-models in $\mathcal{R}$. As we assumed that $\Sigma$ introduces 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 theses new predicate symbols. If the restriction theory is given by a $\Delta$-axiomatization, RQ-structures are exactly those structures that satisfy the axioms of $\mathcal{R}$, considered as formulas over the extended signature.

Let $\mathcal{A}$ be an RQ-structure over the RQ-signature $\Sigma$, $\alpha$ be a $\Sigma$-assignment, and $X = \{x_1, \ldots, x_n\}$ be a set of variables. The **satisfiability** of an RQ-formula $F$ is defined as an extension of the satisfiablity of first-order formulas by:

$$
\begin{array}{lll}
(\mathcal{A}, \alpha) \models \forall_{X:R} F & \text{iff} & \text{for all } u_1, \ldots, u_n \in U^{\mathcal{A}} \text{ with} \\
 & & (\mathcal{A}|_\Delta, \alpha_{[x_1 \leftarrow u_1, \ldots, x_n \leftarrow u_n]}) \models R \text{ holds} \\
 & & (\mathcal{A}, \alpha_{[x_1 \leftarrow u, \ldots, x_n \leftarrow u_n]}) \models F \\
(\mathcal{A}, \alpha) \models \exists_{X:R} F & \text{iff} & \text{there are } u_1, \ldots, u_n \in U^{\mathcal{A}} \text{ such that} \\
 & & (\mathcal{A}|_\Delta, \alpha_{[x_1 \leftarrow u_1, \ldots, x_n \leftarrow u_n]}) \models R \text{ and} \\
 & & (\mathcal{A}, \alpha_{[x_1 \leftarrow u_1, \ldots, x_n \leftarrow u_n]}) \models F
\end{array}
$$

A closed RQ-formula $F$ is **RQ-satisfiable** if and only if there is an RQ-structure $\mathcal{A}$ such that $(\mathcal{A}, \alpha) \models F$ for each $\Sigma$-assignment $\alpha$. In this case, $\mathcal{A}$ is a **$\Sigma$-model** of $F$, written $\mathcal{A} \models F$. The RQ-formula $F$ is called **RQ-valid** if and only if every RQ-structure $\mathcal{A}$ is a $\Sigma$-model of $F$.

Given a restriction $R$, we say $R$ is **RQ-satisfiable (RQ-valid)** if and only if the existential closure $\exists R$ of $R$ is RQ-satisfiable (RQ-valid).

If we consider RQ-structures as normal structures over the signature $\Sigma \cup \Delta$ we obtain the following obvious relativization proposition.

**Proposition 2.3** *The RQ-formulas $\forall_{\{x_1,\ldots,x_n\}:R} F$ and $\exists_{\{x_1,\ldots,x_n\}:R} F$ are logically equivalent to the formluas $\forall x_1 \ldots \forall x_n (R \rightarrow F)$ and $\exists x_1 \ldots \exists x_n (R \wedge F)$, respectively.*

## 2.3 RQ-Resolution

Given a set $\mathcal{C}$ of RQ-clauses we need an appropriate resolution calculus, which allows one to check $\mathcal{C}$ on unsatisfiability. Such a calculus is given in [Bür91] and consists of the following two rules:

**RQ-resolution rule (RR)**

$$\frac{\begin{array}{l} p(x_1,\ldots,x_n) \vee C_1 \vee \ldots \vee C_k \,||\, R \\ \neg p(y_1,\ldots,y_n) \vee D_1 \vee \ldots \vee D_m \,||\, S \end{array}}{C_1 \vee \ldots \vee C_k \vee D_1 \vee \ldots \vee D_m \,||\, R \wedge S \wedge \Gamma} \quad \text{if } R \wedge S \wedge \Gamma \text{ is RQ-satisfiable}$$

where $\Gamma$ is the conjunction of the equations $x_i = y_i$, $i = 1,\ldots,n$.

That means we can infer the third RQ-clause, which is called **RQ-resolvent**, from the first two RQ-clauses if $R \wedge S \wedge \Gamma$ is RQ-satisfiable.

**RQ-factor rule (FR)**

$$\frac{p(x_1^1,\ldots,x_n^1) \vee \ldots \vee p(x_1^m,\ldots,x_n^m) \vee C_1 \vee \ldots \vee C_k \,||\, R}{p(x_1^1,\ldots,x_n^1) \vee C_1 \vee \ldots \vee C_k \,||\, R \wedge \Gamma} \quad \text{if } R \wedge \Gamma \text{ is RQ-satisfiable}$$

where $\Gamma$ is the conjunction of the equations $x_i^1 = x_i^j$, $i = 1,\ldots,n$ and $j = 2,\ldots,m$.

The infered RQ-clause is called **RQ-factor**.

**Example 2.4** Let $\mathcal{C}$ be the set

$$\begin{array}{lll} \mathcal{C}: & (1) & q(x,x) \,||\, p(x) \\ & (2) & \neg q(z,y) \,||\, p(y) \wedge (z = b) \end{array}$$

of RQ-clauses. Then we obtain the RQ-resolvent

$$(1),(2): \quad \Box \,||\, p(x) \wedge p(y) \wedge (z = b) \wedge (x = z) \wedge (x = y).$$

■

For sake of simplicity we will sometimes use constant symbols in the kernels of RQ-clauses, though we assumed that the foreground language introduces new predicate symbols only.[2] In the above example we would therefore simply write

$$q(x,x) \,||\, p(x) \qquad \text{and} \qquad \neg q(b,y) \,||\, p(y)$$

[2]Note that in Subsection 2.1 we required the set $\mathcal{F}_\Sigma$ of additional foreground function symbols to be empty.

and, by reasoning on the equations as well, $\square \mid\mid p(b)$.

An **RQ-resolution step** $\mathcal{C} \rightarrow \mathcal{C}'$ transforms a set $\mathcal{C}$ of RQ-clauses into a set $\mathcal{C}'$ by either choosing two suitable clauses in $\mathcal{C}$ and adding their RQ-resolvent, or by adding an RQ-factor to $\mathcal{C}$. An **RQ-derivation** is a possibly infinite sequence $\mathcal{C}_0 \rightarrow \mathcal{C}_1 \rightarrow \mathcal{C}_2 \rightarrow \ldots$ of RQ-resolution steps. An **RQ-refutation** of a set $\mathcal{C}_0$ of RQ-clauses is an RQ-derivation which starts with $\mathcal{C}_0$ and satisfies the following condition: For each model $\mathcal{A}$ of the restriction theory there is an RQ-clause set $\mathcal{C}_i$ in the derivation containing an empty clause $\square \mid\mid R$, whose restriction is satisfied by this model, i.e. $\mathcal{A} \models \exists R$. In contrast to the classical resolution principle we need in general more than one empty RQ-clause to prove the unsatisfiablility of an RQ-clause set. To see this let us consider the following example [Bür91].

**Example 2.5** Suppose a restriction theory to be given by the set $\{p(a), p(b) \vee p(c)\}$, and a set $\mathcal{C}$ of RQ-clauses to be given by

$$\begin{array}{lll} \mathcal{C}: & (1) & q(x,x) \mid\mid p(x) \\ & (2) & \neg q(b,y) \mid\mid p(y) \\ & (3) & \neg q(c,z) \mid\mid p(z) \end{array}$$

By RQ-resolution we can derivate exactly the following two clauses:

$$\begin{array}{ll} (1),(2): & \square \mid\mid p(b) \\ (1),(3): & \square \mid\mid p(c) \end{array}$$

Since $\mathcal{A} \models p(b)$ or $\mathcal{A} \models p(c)$ for every RQ-structure $\mathcal{A}$ (because of the restriction theory), we can conclude that $\mathcal{C}$ is unsatisfiable: if $\mathcal{A} \models p(b)$ we have a contradiction because of the first resolvent; if $\mathcal{A} \models p(c)$ we have a contradiction because of the second resolvent. Note that both restrictions, $p(b)$ and $p(c)$, are RQ-satisfiable but not RQ-valid. Therefore, none of both RQ-resolvents respresents a contradiction in all RQ-structures. ■

RQ-resolution has been proved to be sound and complete in the following sense (cf. [Bür91]):

**Theorem 2.6** *A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff for each RQ-structure $\mathcal{A}$ there is an RQ-derivation of an empty RQ-clause $\square \mid\mid R$ from $\mathcal{C}$, such that $\mathcal{A} \models \exists R$.*

Obviously, this theorem is not very satisfactory from a practical point of view when we are interested in an implementation of a theorem proving system for RQ-formulas. For this purpose one has to find answers to the following two questions:

1. As we have seen before, we need in general more than one empty RQ-clause to prove RQ-unsatisfiability of an RQ-clause set $\mathcal{C}$. In which cases it is sufficient to derive a finite number of empty RQ-clauses to prove RQ-unsatifiability ?

2. Let $\mathcal{C}$ be a set of RQ-clauses. Does $\mathcal{C}$ contain for every RQ-structure $\mathcal{A}$ an empty RQ-clause $\square \,||\, R$ such that $\mathcal{A} \models \exists R$ ?

The second question will be investigated in this paper (Section 4). An answer to the first question has already been given in [Bür91]:

**Theorem 2.7** *Let $\Sigma$ be an RQ-signature such that the restriction theory is first-order axiomatizable. Then:*

*A set $\mathcal{C}$ of RQ-clauses is RQ-unsatisfiable iff there exists a finite set of empty RQ-clauses $\square \,||\, R_1, \ldots, \square \,||\, R_n$, deriviable from $\mathcal{C}$, such that $R_1 \vee \ldots \vee R_n$ is RQ-valid, i.e., for each RQ-structure $\mathcal{A}$ we have $\mathcal{A} \models \exists R_1 \vee \ldots \vee \exists R_n$.*

Since we are interested in a refutation procedure for RQ-formulas we will restrict our attention in the following to restriction theories which are first-order axiomatizable. Well-investigated classes of such theories are terminological logics, sort hierarchies, and equational theory. If the restriction theory has a first-order axiomatization, the RQ-unsatisfiability test of an RQ-clause set reduces to the test whether the disjunction of all restrictions of the already derivated empty RQ-clauses is RQ-valid. If this is the case we have found a refutation, otherwise we need further empty RQ-clauses.

# 3 Transformation into RQ-Clauses

In this section we describe a procedure which transforms RQ-formulas into RQ-clauses preserving RQ-satisfiability. The idea behind our procedure is basically the same as for the well-known transformation of first-order formulas into clauses. However, two problems which do not appear in the classical case, have to be solved. On the one hand, *quantification over the empty set* may occur, which is not possible in first-order logics; on the other hand, *typed Skolem functions* have to be introduced by Skolemizing RQ-formulas. Subsection 3.1 is concerned with the first problem, quantification over the empty set; Skolemization of RQ-formulas is discussed in Subsection 3.2. In Subsection 3.3 we show how the Skolemized formulas can be transformed into a set of RQ-clauses. Since Skolemization extends the restriction theory by typed Skolem function symbols and since all variables in RQ-formulas are restricted, it is convenient to introduce typed function declarations for the function symbols occuring in RQ-formulas as well. This is discussed in Subsection 3.4.

## 3.1 Quantifier Splitting

In contrast to first-order logics, RQ-formulas may contain quantification over the empty set. Given a restricted quantifier $\forall_{X:R}$ or $\exists_{X:R}$, the quantification obviously ranges over the empty set in those $\Sigma$-structures $\mathcal{A}$ where $R^{\mathcal{A}} = \emptyset$. The following proposition shows two properties this kind of quantifications have:

**Proposition 3.1** *Let $\Sigma$ be an RQ-signature, $\mathcal{A}$ a $\Sigma$-structure, and $R$ a restriction such that $R^{\mathcal{A}}$ is the empty set. Then $\mathcal{A} \models \forall_{X:R} F$ and $\mathcal{A} \not\models \exists_{X:R} F$ for each RQ-formula $F$.*

The proposition, which can easily be proved via relativization, states that an RQ-formula $\forall_{X:R} F$ ($\exists_{X:R} F$) evaluates to *true* (*false*) for any $F$ in those structures where $R^{\mathcal{A}}$ is the empty set.

In order to transform RQ-formulas into RQ-clauses preserving satisfiability, one has to consider both cases—the structures in which the restriction denotes the empty set as well as a non-empty set. This observation motivates the definition of the **quantifier splitting rules**:

1. $\forall_{X:R} F \rightarrow_{\forall'} (R = \emptyset \rightarrow \mathit{true}) \wedge (R \neq \emptyset \rightarrow \forall_{X:R} F)$
2. $\exists_{X:R} F \dashrightarrow_{\exists} (R = \emptyset \rightarrow \mathit{false}) \wedge (R \neq \emptyset \rightarrow \exists_{X:R} F)$

For sake of readability we use $R = \emptyset \rightarrow F$ and $R \neq \emptyset$ as abbreviations for the RQ-formulas $\forall_{X:\neg R} F$ and $\exists_{X:R} \mathit{true}$. Moreover, we will sometimes use $R = \emptyset$ and $R \neq \emptyset$ as abbreviations for the first-order formulas $\forall x_1 \ldots \forall x_n \neg R[x_1, \ldots x_n]$ if $X = \{x_1, \ldots, x_n\}$ and $\exists x_1 \ldots \exists x_n R[x_1, \ldots x_n]$, respectively. The term $(R = \emptyset \rightarrow \mathit{true})$ in the $\rightarrow_{\forall'}$-rule is obviously redundant; we therefore always use the $\rightarrow_{\forall'}$-rule in a simplified form, defined as:

1′. $\forall_{X:R} F \rightarrow_{\forall} (R \neq \emptyset \rightarrow \forall_{X:R} F)$

On the other hand, we do not simplify the $\rightarrow_{\exists}$-rule—the right hand side of the rule is equivalent to $(R \neq \emptyset \wedge \exists_{X:R} F)$—for the following reason. To obtain a set of RQ-clauses for a given RQ-formula, we transform the formula into conjunctive normal form. The conjunctive normal form of the right hand side of the $\rightarrow_{\exists}$-rule has the form

$$(R \neq \emptyset \vee \mathit{false}) \wedge (R = \emptyset \vee \exists_{X:R} F)$$

In Subsection 3.3 we will see that the first conjunct $(R \neq \emptyset \vee \mathit{false})$ is translated into the RQ-clause $\mathit{false} \,||\, R = \emptyset$, written as $\square \,||\, R = \emptyset$, which can be read as follows: We have derived an empty clause for every structure $\mathcal{A}$ such that $R^{\mathcal{A}}$ is the empty set.

Thus, if we would have simplified the $\rightarrow_\exists$-rule as described before, this contradiction would no longer be represented explicitly by an RQ-clause.

The following proposition, which is an immediate consequence of Proposition 3.1, shows that the application of a quantifier splitting rule to an RQ-formula preserves equivalence.

**Proposition 3.2** *Let $F$ be an RQ-formula. If $F'$ is obtained from $F$ by application of the $\rightarrow_\exists$-rule or the $\rightarrow_\forall$-rule, then $F'$ is equivalent to $F$.*

Both quantifier splitting rules seem to be defined unusually, because the restricted quantifier on the left hand side of a rule occurs also on the right hand side. There is, however, an essential difference between both occurrences: the restricted quantifier on the left side of a rule may quantify over the empty set as well as over a non-empty set. In contrast, the restricted quantifier on the right side of a rule can be treated as if it has to handle the non-empty case only, because quantification over the empty restriction is already expressed by the first conjunct of the right hand side of a rule. This discrimination has, of course, to be done only once for each restricted quantifier.

We are interested in RQ-formulas where each restricted quantifier has been splitted. Let $F$ be an RQ-formula. A **quantifier splitting of** $F$ is an RQ-formula obtained from $F$ by application of the $\rightarrow_\exists$-rule or $\rightarrow_\forall$-rule such that a rule has been applied to every restricted quantifier in $F$ at least once.

Let $F$ be an RQ-formula which contains $n$ restricted quantifier. It is easy to see that a quantifier splitting of $F$ can be obtained by $n$ applications of the quantifier splitting rules. Since each rule application preserves equivalence (Proposition 3.2), every quantifier splitting of $F$ is equivalent to $F$.

## 3.2 Skolemization

Our next task is to eliminate restricted $\exists$-quantifiers in RQ-formulas. We do this by introducing Skolem functions, which, however, have a more complex structure then those in classical first-order logics.

In classical first-order logics Skolemization is established according to the following rule: If an existentially quantified variable occurs in the scope of $\forall$-quantifiers, e.g. $\forall x_1 \ldots \forall x_n \exists y\, F[y]$, then each occurrence of $y$ in $F$ is replaced by the term $f(x_1, \ldots, x_n)$, where $f : U \times \ldots \times U \mapsto U$ is a new $n$-ary function symbol (a so-called Skolem function).

In our logic with restricted quantifiers we obtain Skolem functions which are "typed" in the following sense: Suppose that the restricted $\exists$-quantifier $\exists_{\{y\}:S(y)}\, F[y]$ occurs in the scope of $\forall_{\{x_1\}:R_1(x_1)}, \ldots, \forall_{\{x_n\}:R_n(x_n)}$. Then each occurrence of $y$ in $F$ will be replaced

by a Skolem function $f : R_1 \times \ldots \times R_n \mapsto S$, in other words $\forall x_1 \ldots \forall x_n R_1(x_1) \wedge \ldots \wedge R_n(x_n) \rightarrow S(f(x_1, \ldots, x_n))$.

In contrast to such unary restrictions, we have to generalize this consideration in order to deal with $n$-ary restrictions. Assume the formula $\exists_{\{y_1,\ldots,y_k\}:S(y_1,\ldots,y_k)} F[y_1, \ldots, y_k]$ to appear in the scope of $\forall_{\{x_1^1,\ldots,x_l^1\}:R_1(x_1^1,\ldots,x_l^1)}, \ldots, \forall_{\{x_1^n,\ldots,x_m^n\}:R_n(x_1^n,\ldots,x_m^n)}$. In this case each variable $y_i$ in $F$ has to be replaced by a Skolem function with all these universally quantified variables as arguments. We thus obtain a vector $(f_1, \ldots, f_k)$ of $N$-ary Skolem functions if $x_1^1, \ldots, x_m^n$ are $N$ variables.

These Skolem functions are, of course, not independently from each other; they have to satisfy the following condition (**Cond**):

$$\forall x_1^1 \ldots \forall x_m^n \Big( R_1(x_1^1, \ldots, x_l^1) \wedge \ldots \wedge R_n(x_1^n, \ldots, x_m^n) \Big) \rightarrow \\ S(f_1(x_1^1, \ldots, x_m^n), \ldots, f_k(x_1^1, \ldots, x_m^n))$$

Note that this condition restricts the possible interpretations of the introduced Skolem functions.

By Skolemization such a condition is generated for each restricted $\exists$-quantifier; after Skolemization we therefore only want to consider structures which satisfy them all. Thus, a simple approach seems to be an extension of the restriction theory by these conditions. The following example, however, shows that (Cond) in general is too weak.

**Example 3.3** Let $F$ be the RQ-formula $\exists_{\{x\}:S(x)} (p(x) \vee \neg p(x))$ and let $RT$ be the empty restriction theory. The quantifier splitting of $F$ is given by

$$(S = \emptyset \rightarrow false) \wedge (S \neq \emptyset \rightarrow \exists_{\{x\}:S(x)}(p(x) \vee \neg p(x))).$$

By Skolemization we obtain

$$F' \equiv (S = \emptyset \rightarrow false) \wedge (S \neq \emptyset \rightarrow (p(a) \vee \neg p(a)))$$

and the extended restriction theory $RT' = \{S(a)\}$ where $a$ is a new Skolem constant. That means, $S$ has to be interpreted as a non-empty set in every model of $RT'$ (and we only consider structures which satisfy the restriction theory). In contrast, however, there are models $\mathcal{A}$ of $RT$ with $S^{\mathcal{A}} = \emptyset$. Since these are structures which do not satisfy $F$, we conclude that $F$ is not RQ-valid w.r.t $RT$ while $F'$ is RQ-valid w.r.t. $RT'$. ■

This example shows that such an extension of the restriction theory may influence RQ-satisfiability of RQ-formulas. To overcome this problem, in the above example we do not want that $S$ is interpreted as a non-empty set in *every* RQ-structure w.r.t. $RT'$. We therefore modify (Cond) such that $a^{\mathcal{A}}$ is an element of $S^{\mathcal{A}}$ only in RQ-structures $\mathcal{A}$ such that $S^{\mathcal{A}}$ is non-empty.

Generalizing this idea a vector $(f_1, \ldots, f_k)$ of Skolem functions has to satisfy the condition

$$\exists z_1 \ldots \exists z_k \; S(z_1, \ldots, z_k) \rightarrow (Cond)$$

which will be added to the restriction theory as the **Skolem declaration**

$$(S \neq \emptyset) \rightarrow (f_1, \ldots, f_k) : R_1 \times \ldots \times R_n \mapsto S$$

Summing up, one **Skolemization step** eliminates the first restricted $\exists$-quantifier in an RQ-formula as follows: Firstly, it enlarges the signature $\Delta$ by the new Skolem function symbol(s), then it extends the restriction theory by the required Skolem declaration, removes the current restricted $\exists$-quantifier, and finally replaces each occurrence of its variables by the corresponding vector of Skolem functions. In the above example we would obtain the Skolemized formula $F' \equiv (S = \emptyset \rightarrow false) \wedge (S \neq \emptyset \rightarrow (p(a) \vee \neg p(a)))$ and the Skolem declaration $(S \neq \emptyset) \rightarrow a : \mapsto S$.

The following theorem states that, given an RQ-formula with splitted quantifiers, such a Skolemization step preserves RQ-satisfiability.

**Theorem 3.4** *Let $\Sigma$ be an RQ-signature and let $F$ be an RQ-formula with splitted quantifiers, and let $\Sigma'$, $F'$ be obtained from $\Sigma$, $F$ by one Skolemization step. Then there exists an RQ-structure over $\Sigma$ which satisfies $F$ iff there exists an RQ-structure over $\Sigma'$ which satisfies $F'$.*

*Proof:* We will show that for each $\Sigma$-model $\mathcal{A}$ of $F$ there exists a $\Sigma'$ model $\mathcal{A}'$ of $F'$, and vice versa by constructing $\mathcal{A}'$ from $\mathcal{A}$ in an appropriate manner. That means, $\mathcal{A}'$ is an RQ-structure over $\Sigma'$ (i.e. $\mathcal{A}'$ is a model of the restriction theory within $\Sigma'$) iff $\mathcal{A}$ is an RQ-structure over $\Sigma$.

Suppose $G \equiv \exists_{\{y_1,\ldots,y_k\}:S(y_1,\ldots,y_k)} \hat{G}[y_1, \ldots, y_k]$ contains the first restricted $\exists$-quantifier in $F$. Furthermore suppose the formula $G$ to occur in the scope of the universal quantifiers $\forall_{\{x_1^1,\ldots,x_l^1\}:R_1(x_1^1,\ldots,x_l^1)}, \ldots, \forall_{\{x_1^n,\ldots,x_m^n\}:R_n(x_1^n,\ldots,x_m^n)}$.

Let $\mathcal{A}$ be a $\Sigma$-structure and let $\mathcal{A}'$ be a $\Sigma'$-structure such that $(i)$ $U^{\mathcal{A}} = U^{\mathcal{A}'}$ and $(ii)$ $\mathcal{A}'$ is an extension of $\mathcal{A}$ to $\Sigma'$, i.e., $\mathcal{A}'|_\Sigma = \mathcal{A}$. That means, $\mathcal{A}'$ interprets in addition to $\mathcal{A}$ the new Skolem function symbols only. Note that every $\Sigma$-assignment also is a $\Sigma'$-assignment, and vice versa (since $U^{\mathcal{A}} = U^{\mathcal{A}'}$ and $V_\Sigma = V_{\Sigma'}$).

Since the interpretation of the new Skolem function symbols in $\mathcal{A}'$ depends on the interpretation of the restrictions $R_1, \ldots, R_n$ and $S$ in $\mathcal{A}$, we distingiush the following four cases:

*Case 1:* $S^{\mathcal{A}} \neq \emptyset$ and each $R_i^{\mathcal{A}} \neq \emptyset$. Because of $\mathcal{A}'|_\Sigma = \mathcal{A}$ we also have $S^{\mathcal{A}'} \neq \emptyset$ and each $R_i^{\mathcal{A}'} \neq \emptyset$. If $\alpha$ is an arbitrary $\Sigma$-assignment we define

$$(f_1,\ldots,f_k)^{\mathcal{A}'}(\alpha(x_1^1),\ldots,\alpha(x_m^n)) := \begin{cases} (b_1^\alpha,\ldots,b_k^\alpha), & \text{if } R_1^{\mathcal{A}'}(\alpha(x_1^1),\ldots,\alpha(x_l^1)), \\ & \quad\vdots \\ & R_n^{\mathcal{A}'}(\alpha(x_1^n),\ldots,\alpha(x_m^n)), \text{ and} \\ & \text{a tuple } (b_1^\alpha,\ldots,b_k^\alpha) \text{ exists with} \\ & S^{\mathcal{A}'}(b_1^\alpha,\ldots,b_k^\alpha) \text{ and} \\ & (\mathcal{A}', \alpha_{[y_1 \leftarrow b_1^\alpha,\ldots,y_k \leftarrow b_k^\alpha]}) \models \hat{G} \\ (c_1^\alpha,\ldots,c_k^\alpha), & \text{otherwise} \\ & \text{such that } S^{\mathcal{A}'}(c_1^\alpha,\ldots,c_k^\alpha) \text{ holds} \end{cases}$$

If there is more than one possible value of $(f_1,\ldots,f_k)^{\mathcal{A}'}(\alpha(x_1^1),\ldots,\alpha(x_m^n))$, we choose one of them arbitrary, but fixed.

We have to show that $\mathcal{A}$ is a $\Sigma$-model of $F$ iff $\mathcal{A}'$ is a $\Sigma'$-model of $F'$. Given an arbitrary $\Sigma$-assignment $\alpha$ we distinguish whether or not $(\mathcal{A},\alpha)$ satisfies all restrictions $R_i(x_1^i,\ldots x_j^i)$ of the universal quantifiers.

If there is an $i$ such that $(\mathcal{A},\alpha) \not\models R_i(x_1^i,\ldots,x_j^i)$ the truth value of

$$\forall_{\{x_1^i,\ldots,x_j^i\}:R_i(x_1^i,\ldots,x_j^i)}\, H$$

does not depend on the truth value of the subformula $H$ (see Proposition 3.1). Therefore, the syntactical transformation which is performed within $H$ by Skolemization cannot influence the truth value of the RQ-formula $F$.

Otherwise, since the variables $x_1^1,\ldots,x_m^n$ are all universally quantified, it is sufficient to show that

$$(\mathcal{A},\alpha) \models \exists_{\{y_1,\ldots,y_k\}:S(y_1,\ldots,y_k)}\hat{G}[y_1,\ldots,y_k]$$
$$\text{iff}$$
$$(\mathcal{A}',\alpha) \models \hat{G}[y_1 \leftarrow f_1(x_1^1,\ldots,x_m^n),\ldots,y_k \leftarrow f_k(x_1^1,\ldots,x_m^n)]$$

where $\alpha$ is a $\Sigma$-assignment such that $(\mathcal{A},\alpha) \models R_i(x_1^i,\ldots,x_{m_i}^i)$ for $i = 1,\ldots,n$.

Suppose $(\mathcal{A},\alpha) \models \exists_{\{y_1,\ldots,y_k\}:S(y_1,\ldots,y_k)}\hat{G}[y_1,\ldots,y_k]$. In this case, there exist elements $u_1,\ldots,u_k$ in $U^{\mathcal{A}}$ $(= U^{\mathcal{A}'})$ such that $S^{\mathcal{A}}(u_1,\ldots,u_k)$ holds, i.e., $(\mathcal{A}, \alpha_{[y_1 \leftarrow u_1,\ldots,y_k \leftarrow u_k]}) \models \hat{G}[y_1,\ldots,y_k]$. Summing up we know

1. $R_1^{\mathcal{A}'}(\alpha(x_1^1),\ldots,\alpha(x_l^1)),\ldots,R_n^{\mathcal{A}'}(\alpha(x_1^n),\ldots,\alpha(x_m^n))$ because of our choice of $\alpha$.
2. A tuple $(u_1,\ldots,u_k)$ exists with
   (a) $S^{\mathcal{A}'}(u_1,\ldots,u_k)$ and

(b) $(A', \alpha_{[y_1 \leftarrow u_1, \ldots, y_k \leftarrow u_k]}) \models \hat{G}$

and thus $(\mathcal{A}', \alpha_{[y_1 \leftarrow b_1^\alpha, \ldots, y_k \leftarrow b_k^\alpha]}) \models \hat{G}[y_1, \ldots, y_k]$ if $b_i^\alpha = f_i^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n))$ because of the definition of the vector $(f_1, \ldots, f_k)^{\mathcal{A}'}$. From this we can conclude that $(\mathcal{A}', \alpha) \models \hat{G}[y_1 \leftarrow f_1(x_1^1, \ldots, x_m^n), \ldots, y_k \leftarrow f_k(x_1^1, \ldots, x_m^n)]$.

Now suppose that $(\mathcal{A}', \alpha) \models \hat{G}[y_1 \leftarrow f_1(x_1^1, \ldots, x_m^n), \ldots, y_k \leftarrow f_k(x_1^1, \ldots, x_m^n)]$. Since we assumed $S^{\mathcal{A}'}$ to be non-empty, there are elements $u_1, \ldots, u_k$ in $U^{\mathcal{A}'}$ such that $S^{\mathcal{A}'}(u_1, \ldots, u_k)$ holds. Especially we know that

$$S^{\mathcal{A}'}(f_1^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n)), \ldots, f_k^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n)))$$

holds because of the definition of the Skolem functions. Summing up, if $d_i$ is an abbreviation for $f_i^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n))$, we have

1. $(\mathcal{A}', \alpha_{[y_1 \leftarrow d_1, \ldots, y_k \leftarrow d_k]}) \models \hat{G}[y_1, \ldots, y_k]$ and
2. $S^{\mathcal{A}'}(d_1, \ldots, d_k)$

and from this we immediately obtain $(\mathcal{A}, \alpha) \models \exists_{\{y_1, \ldots, y_k\}: S(y_1, \ldots, y_k)} \hat{G}[y_1, \ldots, y_k]$.

In the next cases we abbreviate $\exists_{\{y_1, \ldots, y_k\}: S(y_1, \ldots, y_k)} \hat{G}[y_1, \ldots, y_k]$ by $G$, and $\hat{G}[y_1 \leftarrow f_1(x_1^1, \ldots, x_m^n), \ldots, y_k \leftarrow f_k(x_1^1, \ldots, x_m^n)]$ by $G'$.

*Case 2:* $S^{\mathcal{A}} = \emptyset$ and each $R_i^{\mathcal{A}} \neq \emptyset$. If $\alpha$ is an arbitrary $\Sigma$-assignment, then we define

$$(f_1, \ldots, f_k)^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n)) := (b_1, \ldots, b_k)$$

where the $b_i$ are in $U^{\mathcal{A}}$ arbitrary, but fixed.

We have to show that $\mathcal{A} \models F$ iff $\mathcal{A}' \models F'$. We know that the $\exists_{\{y_1, \ldots, y_k\}: S(y_1, \ldots, y_k)}$ quantifier has been splitted by the $\rightarrow_\exists$-rule. Therefore, $G$ occurs on the right hand side of the implication $S \neq \emptyset \rightarrow G$, and $G'$ occurs on the right hand side of $S \neq \emptyset \rightarrow G'$. Since $S^{\mathcal{A}} = S^{\mathcal{A}'} = \emptyset$, we have $\mathcal{A} \models S \neq \emptyset \rightarrow G$ and $\mathcal{A}' \models S \neq \emptyset \rightarrow G'$. Since $F'$ is obtained from $F$ by replacing $G$ by $G'$ only, we can conclude that $\mathcal{A} \models F$ iff $\mathcal{A}' \models F'$.

*Case 3:* $S^{\mathcal{A}} \neq \emptyset$ and $R_i^{\mathcal{A}} = \emptyset$ for some $i$. We define

$$(f_1, \ldots, f_k)^{\mathcal{A}'}(\alpha(x_1^1), \ldots, \alpha(x_m^n)) := (b_1, \ldots, b_k)$$

where the $b_i$ are arbitrary, but fixed in $U^{\mathcal{A}}$ such that $S^{\mathcal{A}'}(b_1, \ldots, b_k)$ holds.

Then $G$ as well as $G'$ occur in the scope of universal quantification over the empty set. Therefore the truth value of $F$ and $F'$ does not depend on $G$ and $G'$, respectively (Proposition 3.1). Thus $\mathcal{A} \models F$ iff $\mathcal{A}' \models F'$ follows immediately.

*Case 4:* $S^{\mathcal{A}} = \emptyset$ and $R_i^{\mathcal{A}} = \emptyset$ for some $i$: This case can be treated similarly to (3).

Up to now we have shown that there exists a $\Sigma'$-model $\mathcal{A}'$ of $F'$ for each $\Sigma$-model $\mathcal{A}$ of $F$, and vice versa. We finally show that $\mathcal{A}$ is an RQ-structure over $\Sigma$, i.e. $\mathcal{A}$ is a model of the restriction theory within $\Sigma$, iff $\mathcal{A}'$ is an RQ-structure over $\Sigma'$.

Let $\mathcal{T}$ be the restriction theory of $\Sigma$, and let $\mathcal{T}'$ be the restriction theory of $\Sigma'$. We firstly show that $\mathcal{A}'$ is an RQ-stucture over $\Sigma'$ if $\mathcal{A}$ is an RQ-stucture over $\Sigma$. Note that $\mathcal{T}'$ contains additionally to $\mathcal{T}$ the Skolem declarations only. Since $\mathcal{A}'|_\Sigma = \mathcal{A}$ and $\mathcal{A}$ is an RQ-structure over $\Sigma$ we know that $\mathcal{A}'$ satisfies $\mathcal{T}$. Therefore we only have to show that $\mathcal{A}'$ satisfies the Skolem declarations. Assume that $S \neq \emptyset \rightarrow (f_1, \ldots, f_k) : R_1 \times \ldots \times R_n \mapsto S$ is a Skolem declaration in $\mathcal{T}'$. Again, we distinguish the interpretations of $S$ and $R_i$. If $S^{\mathcal{A}'} = \emptyset$, then the declaration is obviously satisfied by $\mathcal{A}'$.

Now suppose $S^{\mathcal{A}'} \neq \emptyset$, and therefore $\mathcal{A}' \models$ (Cond) is to show. Let $\alpha$ be an arbitrary $\Sigma$-assignment. If there is an $i$ such that $(\mathcal{A}', \alpha) \not\models R_i(x_1^1, \ldots, x_j^i)$ there is nothing to show. If, on the other hand, $(\mathcal{A}', \alpha)$ satisfies each $R_i(x_1^1, \ldots, x_j^i)$, then $(\mathcal{A}', \alpha)$ has to satisfy $S(f_1(x_1^1, \ldots, x_m^n), \ldots, f_k(x_1^1, \ldots, x_m^n))$ as well. By the definition of the Skolem functions in *Case 1* $\mathcal{A}'$ does satisfy this condition.

Conversely, suppose $\mathcal{A}'$ satisfies $\mathcal{T}'$. Then $\mathcal{A}'$ satisfies $\mathcal{T}$, and therefore $\mathcal{A} = \mathcal{A}'|_\Sigma$ obviously satisfies $\mathcal{T}$ since $\mathcal{T}'$ contains additionally to $\mathcal{T}$ the Skolem declarations only. Thus, the theorem is proved. □

## 3.3 RQ-clause form

In the previous subsections we have shown how to transform RQ-formulas into RQ-formulas without restricted $\exists$-quantifier preserving satisfiability. In this subsection we finally describe how to obtain RQ-clauses from the latter RQ-formulas. Remember that an RQ-clause is of the form $C \parallel R$, where $C$ is a $\Sigma$-clause and $R$ is a restriction, and represents the RQ-formula $\forall_{X:R} C$.

In order to achieve the transformation we need the following two rules, which show how restricted $\forall$-quantifiers can be moved within RQ-formulas.

- $(\forall_{X:R} F) \vee G \rightarrow_\vee \forall_{X:R} (F \vee G)$
  if no $x \in X$ occurs free in $G$.
- $\forall_{X:R} (F \wedge G) \rightarrow_\wedge (\forall_{Y:R} F) \wedge (\forall_{Z:R} G)$
  if each $x \in X$ occurring in $F$ ($G$) is replaced by a new variable $y \in Y$ ($z \in Z$).

It is straightforward to prove that applications of both rules preserve equivalence of RQ-formulas.

Observe that, in contrast to first-order logics, the rule $(\forall_{X:R} F) \wedge G \rightarrow \forall_{X:R} (F \wedge G)$ does not preserve equivalence of RQ-formulas, e.g., $\forall_{\{x\}:R(x)} p(x) \wedge \mathit{false}$ is obviously

RQ-unsatisfiable, while $\forall_{\{x\}:R(x)}(p(x) \wedge \mathit{false})$ is RQ-satisfiable. Therefore, we cannot transform an RQ-formula into an equivalent RQ-formula which is in prenex form. The following proposition, however, states that an RQ-formula can be transformed into another useful normal form.

**Proposition 3.5** *Let $F$ be an RQ-formula obained by quantifier splitting and Skolemization. Then $F$ can be transformed into an equivalent formula which is a conjunction of RQ-formulas of the form*

$$\forall_{X_1:R_1} \ldots \forall_{X_n:R_n} (F_1 \vee \ldots \vee F_m)$$

*where each $F_i$ is either a $\Sigma$-literal or of the form $R = \emptyset$ or $R \neq \emptyset$ for some restriction $R$.*

*Proof:* Let $F$ be an RQ-formula obtained from quantifier splitting and Skolemization. We prove the claim by induction on the number of restricted $\forall$-quantifiers occurring in $F$. If $F$ does not contain any restricted $\forall$-quantifier, then $F$' can be tranformed into the required form by applying the distributivity law.

Now suppose that $F$ contains $n+1$ restricted $\forall$-quantifiers. Then $F$' has one of the following forms:

1. $R = \emptyset \vee \forall_{X:R} F'$,
2. $(R = \emptyset \vee \forall_{X:R} F') \wedge G$,
3. $(R = \emptyset \vee \forall_{X:R} F') \vee G$.

where $R$ is a restriction, and $F'$, $G$ together contain $n$ restricted $\forall$-quantifiers. By the induction hypothesis we can assume that $F'$ and $G$ are already of the required form; i.e., $F'$ and $G$ are of the form $\mathsf{M}_1 F'_1 \wedge \ldots \wedge \mathsf{M}_n F'_n$ and $\mathsf{N}_1 G_1 \wedge \ldots \wedge \mathsf{N}_k G_k$, respectively, where each $\mathsf{M}_i$, $\mathsf{N}_i$ represents restricted $\forall$-quantifiers, and each $F'_i$, $G_i$ is a disjunction.

Without loss of generality we can assume that the restricted $\forall$-quantifiers contain pairwise disjoint variables, what can be achieved by variable renaming.

(1) Assume that $F$ has the form $R = \emptyset \vee \forall_{X:R} F'$. Then, by the induction hypothesis, $F$ can be rewritten as $R = \emptyset \vee \forall_{X:R}(\mathsf{M}_1 F'_1 \wedge \ldots \wedge \mathsf{M}_n F'_n)$. By applying the $\rightarrow_\vee$-rule, $\rightarrow_\wedge$-rule, and the distributivity law, we can obtain the RQ-formula

$$\forall_{X:R} \mathsf{M}_1(R = \emptyset \vee F'_1) \wedge \ldots \wedge \forall_{X:R} \mathsf{M}_n(R = \emptyset \vee F'_n).$$

Obviously, this RQ-formula is RQ-satisfiable if and only if $F$ is RQ-satisfiable.

(2) Assume that $F$ has the form $(R = \emptyset \vee \forall_{X:R} F') \wedge G$ or $G \wedge (R = \emptyset \vee \forall_{X:R} F')$. Then, as described in (1), $(R = \emptyset \vee \forall_{X:R} F')$ can be transformed into the RQ-formula

$\forall_{X:R} \mathsf{M}_1(R = \emptyset \vee F_1') \wedge \ldots \wedge \forall_{X:R} \mathsf{M}_n(R = \emptyset \vee F_n')$, and, by the induction hypothesis, $G$ can be transformed into the form $\mathsf{N}_1 G_1 \wedge \ldots \wedge \mathsf{N}_k G_k$. Consequently, $F$ can be rewritten as

$$\forall_{X:R} \mathsf{M}(R = \emptyset \vee F_1') \wedge \ldots \wedge \forall_{X:R} \mathsf{M}(R = \emptyset \vee F_n') \wedge \mathsf{N}_1 G_1 \wedge \ldots \wedge \mathsf{N}_k G_k.$$

(3) Assume that $F$ has the form $(R = \emptyset \vee \forall_{X:R} F') \vee G$ or $G \vee (R = \emptyset \vee \forall_{X:R} F')$. Then, as described in (1) and by the induction hypotheses, $F$ can be rewritten as

$$\forall_{X:R} \mathsf{M}_1(R = \emptyset \vee F_1') \wedge \ldots \wedge \forall_{X:R} \mathsf{M}_n(R = \emptyset \vee F_n') \vee (\mathsf{N}_1 G_1 \wedge \ldots \wedge \mathsf{N}_k G_k)$$

By applying the $\rightarrow_\vee$-rule, $\rightarrow_\wedge$-rule, and the distributivity law, we can obtain an RQ-formula

$$\forall_{X:R} \mathsf{M}_1 \mathsf{N}_1(R = \emptyset \vee F_1' \vee G_1) \wedge \ldots \wedge \forall_{X:R} \mathsf{M}_n \mathsf{N}_k(R = \emptyset \vee F_n' \vee G_k)$$

□

The following proposition shows that the thus obtained RQ-formulas can immediately be translated into RQ-clauses.

**Proposition 3.6** *Let $F$ be an RQ-formula of the form $\forall_{X_1:R_1} \ldots \forall_{X_n:R_n}(F_1 \vee \ldots \vee F_m)$, where each $F_i$ is either a $\Sigma$-literal or of the form $R = \emptyset$ or $R \neq \emptyset$ for some restriction $R$. Then $F$ is equivalent to the RQ-clause*

$$F_{i_1} \vee \ldots \vee F_{i_k} \,||\, R_1 \wedge \ldots \wedge R_n \wedge F_{j_1} \ldots \wedge F_{j_l}$$

*where the $F_{i_{i'}}$ are exactly the $\Sigma$-literals in $F$, the $F_{j_{j'}}$ are of the form $R = \emptyset$ or $R \neq \emptyset$, and the $R_i$ are the restrictions of the universal quantifiers.*

*Proof:* An RQ-clause $C \,||\, R$ is represents the RQ-formula $\forall_{X:R} C$, where $X$ contains exactly the free variables in $R$ and $C$. By relativization, $\forall_{X:R} C$ is equivalent to $\forall X(R \rightarrow C)$ and thus it is easy to see, that the RQ-formula $\forall_{X_1:R_1} \ldots \forall_{X_n:R_n}(F_1 \vee \ldots \vee F_m)$ can be written as $F_1 \vee \ldots \vee F_m \,||\, R_1 \wedge \ldots \wedge R_n$.[3]

Because of the application of the quantifier splitting procedure the $F_i$ may contain restriction-information of the form $R = \emptyset$ and $R \neq \emptyset$, respectively. Since $G_1 \rightarrow (G_2 \vee G_3)$ is logically equivalent to $(G_1 \wedge \neg G_2) \rightarrow G_3$ for arbitrary formulas $G_i$, we can transform

$$R = \emptyset \vee F_1 \vee \ldots \vee F_m \,||\, R_1 \wedge \ldots \wedge R_n \quad \text{into} \quad F_1 \vee \ldots \vee F_m \,||\, R \neq \emptyset \wedge R_1 \wedge \ldots \wedge R_n,$$
$$R \neq \emptyset \vee F_1 \vee \ldots \vee F_m \,||\, R_1 \wedge \ldots \wedge R_n \quad \text{into} \quad F_1 \vee \ldots \vee F_m \,||\, R = \emptyset \wedge R_1 \wedge \ldots \wedge R_n.$$

[3]Remember that restrictions are assumed to be closed under conjunction.

With the help of these rules the formula $F$ can be transformed into an RQ-clause of the required form. Since both rules transform RQ-formulas into equivalent RQ-formulas the proposition is proved. □

With this result we are now able to transform an arbitrary RQ-formula $F$ into a set $\mathcal{C}$ of RQ-clauses such that $\mathcal{C}$ is RQ-satisfiable if and only if $F$ is RQ-satisfiable. We still have the possibility of **simplification** of the obtained RQ-clauses. This can be done by the following three rules:

$$\begin{array}{rcl} C_1 \vee \ldots \vee C_n \vee \mathit{false} \,||\, R & \rightarrow & C_1 \vee \ldots \vee C_n \,||\, R \\ \mathit{false} \,||\, R & \rightarrow & \square \,||\, R \\ C_1 \vee \ldots \vee C_n \,||\, R \wedge S \wedge (S \neq \emptyset) & \rightarrow & C_1 \vee \ldots \vee C_n \,||\, R \wedge S \end{array}$$

where $R$, $S$ are restrictions.

The first two rules are obvious. We will give a proof sketch why the third rule preserves equivalence as well. By definition the RQ-clause $C_1 \vee \ldots \vee C_n \,||\, R \wedge S \wedge S \neq \emptyset$ represents the RQ-formula $\forall_{X:R \wedge S \wedge \exists y_1 \ldots \exists y_m S(y_1,\ldots,y_m)}\, C_1 \vee \ldots \vee C_n$, where $X$ contains exactly the free variables in $R$, $S$, and in the $C_i$, and the $y_i$ are not in $X$. By relativization we obtain $\forall X(R \wedge S \wedge \exists y_1 \ldots \exists y_m S(y_1,\ldots,y_m)) \rightarrow C_1 \vee \ldots \vee C_n$. By classical first-order transformations this formula is equivalent to $\forall X \forall y_1 \ldots \forall y_m (R \wedge S \wedge S(y_1,\ldots,y_m)) \rightarrow C_1 \vee \ldots \vee C_n$. Since the $y_i$ do not appear in $X$ we can obtain the equivalent formula $\forall X(R \wedge S) \rightarrow C_1 \vee \ldots \vee C_n$, which represents the RQ-clause $C_1 \vee \ldots \vee C_n \,||\, R \wedge S$.

We have now reached the aim of this section: we are able to transform an arbitrary RQ-formula $F$ into a set $\mathcal{C}$ of RQ-clauses such that $\mathcal{C}$ is RQ-satisfiable if and only if $F$ is. To perform this transformation we first have to replace $\leftrightarrow$ and $\rightarrow$, and to shift negation inside. The second step is to compute the quantifier splitting. To the thus obtained formula we apply our Skolemization procedure and obtain an RQ-formula $F'$ without existential quantifiers, and a modified RQ-signature $\Sigma'$. The final step is to transform $F'$ into the form of Proposition 3.6 and to translate each conjunction into an RQ-clause.

We have seen, that $F$ is RQ-satisfiable w.r.t. $\Sigma$ if and only if $\mathcal{C}$ is RQ-satisfiable w.r.t. $\Sigma'$. Building upon this will in the following give a procedure for testing RQ-satisfiabilty of $\mathcal{C}$ w.r.t. $\Sigma'$.

## 3.4 Function Declarations

If we transform arbitrary RQ-formulas into RQ-clauses, these RQ-clauses may contain function symbols because of Skolemization. For example, the transformation of the RQ-formula

$$\forall_{\{x\}:R(x)} \exists_{\{y\}:S(y)} p(x,y)$$

results in the RQ-clause set

$$\begin{array}{rcl} p(x,y) & || & R(x) \wedge y = f_{Skolem}(x) \wedge S \neq \emptyset \\ \square & || & R(z) \wedge S = \emptyset \end{array}$$

and an extension of the restriction theory by the Skolem declaration

$$(S \neq \emptyset) \rightarrow f_{Skolem} : R \mapsto S.$$

Analogously, RQ-clauses may contain function symbols which have been introduced in RQ-formulas as, e.g., in

$$\forall_{\{x\}:human} male(father(x))$$

where *human* is a unary restriction, *male* is a predicate, and *father* a function symbol. Up to now we interpreted these function symbols free, i.e., we assumed a $\Sigma$-structure $\mathcal{A}$ to map each $n$-ary function symbol $f$ to a function $f^{\mathcal{A}} : U^{\mathcal{A}} \times \ldots \times U^{\mathcal{A}} \mapsto U^{\mathcal{A}}$, where $U^{\mathcal{A}}$ is the universe of $\mathcal{A}$. Indeed, since all variables in RQ-formulas are constrained, it is convenient to take restrictions into account when interpreting function symbols. In the above example it is more intuitive to define the unary function symbol *father* to map from *human* to *human* instead of mapping arbitrary elements of the universe to the universe.

Analogously to the Skolem function declarations one therefore can extend the restriction theory by a function declaration for each function symbol occurring in a set of RQ-clauses (RQ-formulas). If $f$ is an $n$-ary function symbol and $R_1, \ldots, R_n, R$ are restrictions, a **function declaration** is of the form

$$f : R_1 \times \ldots \times R_n \mapsto R.$$

A straightforward semantics of these function declaration for the case that $R$ is a unary restriction is characterized by the following property:

$$\begin{array}{c} \forall x_1^1 \ldots \forall x_m^n \Big( R_1(x_1^1, \ldots, x_l^1) \wedge \ldots \wedge R_n(x_1^n, \ldots, x_m^n) \Big) \rightarrow \\ R(f(x_1^1, \ldots, x_m^n), \ldots, f_k(x_1^1, \ldots, x_m^n)) \end{array}$$

There are, of course, a lot of possible extensions of this straight-forward semantics for function declarations. There are two aspects that have to be taken into consideration when choosing a semantics. Firstly, for a concrete application some semantics may be more intuitive than another one. And, secondly, finding algorithms to decide satisfiability and validity of restrictions may be more or less expensive (or even impossible) depending on the choosen semantics.

# 4 The General Refutation Framework

In the previous section a method for transforming RQ-formulas into a set of RQ-clauses preserving RQ-satisfiability has been described. To these RQ-clauses we can apply RQ-resolution (see 4.1). In Subsection 4.2 we introduce constraint unification which can be used to reduce the number of derived RQ-clauses. However, this optimization can only be applied if the RQS satisfies a certain condition introduced in 4.3.

Some of the proofs in this section need techniques from mathematical logics and model theory. We will give them in the appendix.

## 4.1 The Refutation Procedure

To prove RQ-unsatisfiablity of an RQ-clause set $\mathcal{C}$ we can use the RQ-resolution and the RQ-factor rule which succesively add new RQ-clauses to $\mathcal{C}$. This process is iterated until a set of empty RQ-clauses $\square \parallel R_1, \ldots, \square \parallel R_n$ is derived such that $R_1 \vee \ldots \vee R_n$ is RQ-valid (see Lemma 2.6).

By definition, the RQ-resolution as well as the RQ-factor rule only add new RQ-clauses to $\mathcal{C}$ whose restrictions are RQ-satisfiable. It is easy to show that this is an optimization which reduces the search space, but which does not affect refutation completeness of the RQ-resolution principle: If we apply the RQ-resolution or the RQ-factor rule to an RQ-clause with an RQ-unsatisfiable restriction, then the restriction of the resulting RQ-clause is RQ-unsatisfiable as well. Furthermore, if we have derived empty RQ-clauses $\square \parallel R_1, \ldots, \square \parallel R_n$, and $\square \parallel R$ is an empty RQ-clause with an RQ unsatisfiable restriction $R$, then obviously $R_1 \vee \ldots \vee R_n \vee R$ is RQ-valid iff $R_1 \vee \ldots \vee R_n$ is RQ-valid. Thus, an RQ-clause with an RQ-unsatisfiable restriction $R$ is redundant when testing RQ-unsatisfiability of a clause set $\mathcal{C}$.

This idea can be generalized as follows: If a concrete restricted quantification system is given it may be the case that there are more RQ-clauses redundant for testing RQ unsatisfiability than those having an RQ-unsatisfiable restriction. For example, in the next subsection we will introduce a technique, called constraint unification, and we will show that RQ-clauses whose restrictions are not constraint unifiable are redundant for proving RQ-unsatisfiability if the restricted quantification system satisfies a certain condition. In the general refutation procedure below we will therefore use a slightly modified version of the RQ-resolution and the RQ-factor rule. Thereby we will use the predicate **redundant** which has an RQ-clause $C \parallel R$ as input and is interpreted w.r.t. the actual RQS as follows: *redundant* $(C \parallel R)$ is true iff the inconsistency of a clause set $\mathcal{C}$ can be proved without adding $C \parallel R$ to $\mathcal{C}$. For example, an RQ-clause $C \parallel R$ is redundant if $R$ is RQ-unsatisfiable (cf. definition of the RQ-resolution and the

RQ-factor rule). Thus, we obtain the following generalized versions (RR') and (FR') of the RQ-resolution and the RQ-factor rule

**RQ-resolution rule (RR')**

$$\frac{\begin{array}{l} p(x_1,\ldots,x_n) \vee C_1 \vee \ldots \vee C_k \,||\, R \\ \neg p(y_1,\ldots,y_n) \vee D_1 \vee \ldots \vee D_m \,||\, S \end{array}}{C_1 \vee \ldots \vee C_k \vee D_1 \vee \ldots \vee D_m \,||\, R \wedge S \wedge \Gamma} \quad \text{if not } \mathit{redundant}\,(C||R \wedge S \wedge \Gamma)$$

where $C$ is $C_1 \vee \ldots \vee C_k \vee D_1 \vee \ldots \vee D_m$, and $\Gamma$ is the conjunction of the equations $x_i = y_i$, $i = 1,\ldots,n$.

**RQ-factor rule (FR')**

$$\frac{p(x_1^1,\ldots,x_n^1) \vee \ldots \vee p(x_1^m,\ldots,x_n^m) \vee C_1 \vee \ldots \vee C_k \,||\, R}{p(x_1^1,\ldots,x_n^1) \vee C_1 \vee \ldots \vee C_k \,||\, R \wedge \Gamma} \quad \text{if not } \mathit{redundant}\,(C||R \wedge \Gamma)$$

where $C$ is $p(x_1^1,\ldots,x_n^1) \vee C_1 \vee \ldots \vee C_k$, and $\Gamma$ is the conjunction of the equations $x_i^1 = x_i^j$, $i = 1,\ldots,n$ and $j = 2,\ldots,m$.

Obviously, if we only know these RQ-clauses to be redundant whose restrictions are RQ-unsatisfiable we obtain the RQ-resolution and the RQ-factor rule of Subsection 2.3.

The **general refutation procedure** has a set $\mathcal{S}$ of RQ-formulas as input and works as follows: Firstly, $\mathcal{S}$ is transformed into a set $\mathcal{C}$ of RQ-clauses preserving RQ satisfiability (according to the results of the previous section). Then non-redundant RQ-clauses are derived by the rules (RR') and (FR') until the RQ-unsatisfiability of $\mathcal{C}$ has been proved, i.e., until a set $\Box \,||\, R_1, \ldots, \Box \,||\, R_n$ has been derived such that $R_1 \vee \ldots \vee R_n$ is RQ-valid. This procedure, given in Figure 1, returns *inconsistent* iff the input is not RQ-satisfiable, and either returns *consistent* or does not terminate else.

Correctness and refutation completeness of this procedure follow immediately from Theorems 2.6 and 2.7.

## 4.2 Constraint Unification

In this subsection we introduce constraint unification. This unification procedure can be used as an instance of the redundant predicate in the general refutation procedure if the restricted quantification system satisfies a certain condition, called (TM) which is introduced in the next subsection.

To give a definition of constraint unifiability we need the notion of an **equational restriction** which is an equation of the form $s = t$ where $s$ and $t$ are terms. Restrictions

**Input:** A set $\mathcal{S}$ of RQ-formulas and
a background theory $\mathcal{T}$ which has a first-order axiomatization
**Output:** *inconsistent* iff $\mathcal{S}$ is not RQ-satisfiable
*consistent* or the algorithm does not terminate, else

**Initializing**

Transform $\mathcal{S}$ into a set $\mathcal{C}$ of RQ-clauses while preserving RQ-satisfiability. Delete each RQ-clause $C \parallel R$ in $\mathcal{C}$ such that *redundant* $(C \parallel R)$ is true.

**Testing**

1. If $\Box \parallel R_1, \ldots, \Box \parallel R_n$ are the empty RQ-clauses in $\mathcal{C}$, and $R_1 \vee \ldots \vee R_n$ is RQ-valid w.r.t. $\mathcal{T}$, then return *inconsistent*.

2. If there is an RQ-clause to which the RQ-factor rule (FR') is applicable, but has not yet been applied, then apply the RQ-factor rule to this RQ-clause and add the RQ-factor to $\mathcal{C}$.

3. Find two RQ-clauses which can be resolved against each other by the RQ-resolution rule (RR') (of course the two RQ-clauses have to be choosen by a fair strategy). If there does not exist such a pair of RQ-clauses, return *consistent*. Otherwise, add the RQ-resolvent to $\mathcal{C}$ (after an appropriate variable renaming) and goto 1.

Figure 1: The general refutation procedure.

which are not equational restrictions will be called **non-equational restrictions**. Let now $R = E_1 \wedge \ldots \wedge E_n \wedge N_1 \wedge \ldots \wedge N_m$ be a restriction, where $E_1, \ldots, E_n$ are the equational restrictions in $R$. Then $R$ is **constraint unifiable with substitution** $\sigma$ iff there exists an RQ-structure $\mathcal{A}$ and a $\Sigma$-assignment $\alpha$ such that

1. $E_1 \wedge \ldots \wedge E_n$ is unifiable with $\sigma$, and

2. $(\mathcal{A}, \alpha) \models \sigma(N_1) \wedge \ldots \wedge \sigma(N_m)$.

If the restriction $R$ is constraint unifiable with $\sigma$ we call $\sigma$ a **constraint unifier** of $R$. If a constraint unifier $\sigma$ is a most general unifier of the equational restrictions in $R$ we call $\sigma$ a **constraint mgu** of $R$. We say $R$ is **constraint unifiable** iff there is a substitution $\sigma$ such that $R$ is constraint unifiable with $\sigma$. If $\sigma$ and $\sigma'$ are unifiers we say $\sigma \leq \sigma'$ ($\sigma$ is **more general** than $\sigma'$) iff there exists a substitution $\lambda$ such that $\sigma'(x) = \lambda(\sigma(x))$ for all variables $x$. We say $\sigma < \sigma'$ iff $\sigma \leq \sigma'$, but $\sigma' \neq \sigma$.

**Example 4.1** Let $A$ and $B$ be predicate symbols of a background signature $\Delta$, and let $R$ be the restriction $(y = f(x)) \wedge A(x) \wedge B(y)$, where $f$ is a function symbol. Obviously, the only equational restriction in $R$, $y = f(x)$, is unifiable with mgu $\sigma = \{y \leftarrow f(x)\}$.

1. Let $f$ have the function declaration $f : A \mapsto \neg B$. Since function declarations are part of the restriction theory, each RQ-structure $\mathcal{A}$ has to satisfy $f : A \mapsto \neg B$, i.e., $f^{\mathcal{A}}(u) \notin B^{\mathcal{A}}$ if $u \in A^{\mathcal{A}}$. By definition, $R$ is constraint unifiable with $\sigma$ iff there exists an RQ-structure $\mathcal{A}$ and a $\Delta$-assignment $\alpha$ such that $(\mathcal{A}, \alpha) \models \sigma(A(x)) \wedge \sigma(B(y))$. This is the case iff $\alpha(x) \in A^{\mathcal{A}}$ and $f^{\mathcal{A}}(\alpha(x)) \in B^{\mathcal{A}}$. Because of the function declaration such a pair $(\mathcal{A}, \alpha)$ cannot exist, i.e., $R$ is not constraint unifiable with $\sigma$.

2. If $f$ has the function declaration $f : A \mapsto A$, then $R$ is constraint unifiable with $\sigma$ iff there exists an RQ-structure $\mathcal{A}$ and a $\Delta$-assignment $\alpha$ such that $\alpha(x) \in A^{\mathcal{A}}$ and $f^{\mathcal{A}}(\alpha(x)) \in B^{\mathcal{A}}$. Because of the function declaration we know $f^{\mathcal{A}}(\alpha(x)) \in A^{\mathcal{A}}$ if $\alpha(x) \in A^{\mathcal{A}}$. Thus $R$ is constraint unifiable with $\sigma$ if there exists an RQ-structure $\mathcal{A}$ such that $A^{\mathcal{A}} \cap B^{\mathcal{A}} \neq \emptyset$. ■

By definition of constraint unifiability the test whether or not a restriction $R$ is constraint unifiable depends on finding an arbitrary constraint unifier $\sigma$ of $R$. The following proposition states that it is sufficient to test constraint unifiability of $R$ only with constraint mgu's of $R$.

**Proposition 4.2** *A restriction $R$ is constraint unifiable iff there exists a constraint mgu of $R$.*

*Proof:* We will show the following: If $R$ is constraint unifiable with $\sigma'$ which is not an mgu of the equational restrictions in $R$, then there exists a unifier $\sigma$ such that $\sigma < \sigma'$ and $R$ is constraint unifiable with $\sigma$.

Let $R$ be given by $E_1 \wedge \ldots \wedge E_n \wedge N_1 \wedge \ldots \wedge N_m$, where $E_1, \ldots, E_n$ are the equational restrictions in $R$. Furthermore, let $\lambda$ be a substitution such that $\sigma' = \lambda \circ \sigma$, $\sigma < \sigma'$, and $\sigma$ is a unifier of $E_1 \wedge \ldots \wedge E_n$. Such a $\lambda$ exists because $\sigma'$ is a unifier, but not an mgu of the equational restrictions $E_i$. Since $R$ is constraint unifiable with $\sigma'$ we know that there exists a pair $(\mathcal{A}, \alpha)$ such that $(\mathcal{A}, \alpha) \models \sigma'(N_i)$ for $i = 1, \ldots, m$, and because of $\sigma' = \lambda \circ \sigma$ we thus obtain $(\mathcal{A}, \alpha \circ \lambda) \models \sigma(N_i)$. Thus we know $(i)$ $E_1 \wedge \ldots \wedge E_n$ is unifiable with $\sigma$ (by the construction of $\lambda$), and $(ii)$ there exists an RQ-structure $\mathcal{A}$ and a $\Delta$-assignment $\beta = \alpha \circ \lambda$ such that $(\mathcal{A}, \beta) \models \sigma(N_i)$ for $i = 1, \ldots, m$. By definition, that means $R$ is constraint unifiable with substitution $\sigma$. □

## 4.3 Using Constraint Unification as Optimization

In this subsection we show that constraint unification can be used to define the redundant predicate in the general refutation procedure, provided that a certain condition, called (TM), holds. To formulate this condition we need the notion of a term-model. If $\mathcal{S}$ is a set of formulas over some signature $\Delta$, then $\mathcal{A}$ is a **term-model** of $\mathcal{S}$ over $\Delta$ iff $\mathcal{A} \models \mathcal{S}$, all elements in the universe $U^{\mathcal{A}}$ are interpretations of $\Delta$-ground terms, and two different $\Delta$-ground terms denote different elements in $U^{\mathcal{A}}$.

The property (TM) the RQS has to satisfy is given by: Let $\mathcal{T}$ be a satisfiable background theory and let $R_1, \ldots, R_n$ be a set of restrictions, then (TM) is defined by

(TM)
$$\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$$
iff
there exists a term-model $\mathcal{A}$ of $\mathcal{T}$ such that $\mathcal{A} \models \exists(R_1 \vee \ldots \vee R_n)$

For technical reasons we will sometimes use an equivalent formulation of this property, called (TM'), which is given by

(TM')
$$\mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\} \text{ is satisfiable}$$
iff
$$\mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\} \text{ has a term-model}$$

The following theorem gives a sufficient condition which guarantees property (TM): A background theory $\mathcal{T}$ and a set $R_1, \ldots, R_n$ of restrictions satisfy (TM) if

1. $\mathcal{T}$ does not contain equations, neither explicitly nor implicitly, and
2. each restriction $R_i$ can be written as $E_i \wedge N_i$, where $E_i$ is a conjunction of equations and $N_i$ does not contain equations nor disequations, neither explicitly nor implicitly.

This condition is, e.g., satisfied if the background theory $\mathcal{T}$ is a sort hierarchy (i.e., given by formulas of the form $S(a)$ and $\forall x S(x) \rightarrow T(x)$), and restrictions are conjunctions of equations and of formulas $S(t)$ only (where $S$ is a sort and $t$ is a term). The concept language $\mathcal{ALC}$ satisfies this condition as well (see [BHL93]).

**Theorem 4.3** *Let $\mathcal{T}$ be a satisfiable background theory, and let $R_1, \ldots, R_n$ be a set of restrictions such that*

1. *$\mathcal{T}$ does not contain (explicitly or implicitly) equations.*
2. *Each restriction $R_i$ can be written as $E_i \wedge N_i$, where $E_i$ is a conjunction of equations and $N_i$ does neither contain (explicitly or implicitly) equations nor disequations.*

*Then $\mathcal{T}$ and $R_1, \ldots, R_n$ satisfy condition (TM).*

*Proof:* By presumption, $\mathcal{T}$ does not contain equations such that for all ground terms $s$ and $t$ obviously $HT \models s = t$ iff $s$ and $t$ are identical. Analogously, if $\mathcal{T}' := \mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\}$, then $\mathcal{T}' \models s = t$ iff $s$ and $t$ are identical, since $\mathcal{T}'$ does not contain any equations as well. Thus Theorem A.3 in the appendix can be applied.

We will now show that propery (TM') holds, i.e., $\mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\}$ is satisfiable iff $\mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\}$ has a term-model. If $\mathcal{T}' := \mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\}$ is not satisfiable, this set obviously does not have a term-model. Conversely, if $\mathcal{T}'$ is satisfiable, then $\mathcal{T}'$ has a term-model over the signature $\Delta \cup C$, if $\Delta$ is the background signature and $C$ is a countable infinite set of new constants (see Theorem A.3 in the appendix). Since properties (TM) and (TM') are equivalent the theorem is proved. □

To show that constraint unification can be used to define the redundant predicate, provided that condition (TM) holds, we have to show two facts:

1. If a restriction $R$ is RQ-satisfiable, then $R$ is constraint unifiable as well.
2. RQ-clauses whose restriction is not constraint unifiable are not needed to prove RQ-unsatisfiability of an RQ-clause set.

It is easy to see that the constraint unification test subsumes the RQ-satisfiability test of restrictions, i.e., if a restriction $R$ is constraint unifiable, then $R$ is RQ satisfiable as well. To see this, let us reconsider the definition of constraint unifiability: $R$ is constraint unifiable iff the equational restrictions in $R$ are unifiable with some substitution, say $\sigma$, and if there exists an RQ-structure $\mathcal{A}$ and a $\Delta$-assignment $\alpha$ such that $(\mathcal{A}, \alpha) \models \sigma(N_1) \wedge \ldots \sigma(N_m)$, where the $N_i$ are the non-equational restrictions in $R$. If $(\mathcal{A}, \alpha) \models \sigma(N)$ for some non-equational restriction $N$, then $(\mathcal{A}, \alpha \circ \sigma) \models N$. Furthermore, if $E$ is an equational restriction in $R$, we already know $(\mathcal{A}, \alpha) \models \sigma(E)$ for each RQ-structure $\mathcal{A}$ and for each $\Delta$-assignment $\alpha$ (since $\sigma$ is a unifier of $E$). That means, $(\mathcal{A}, \alpha \circ \sigma) \models R$, i.e., $R$ is RQ-satisfiable.

Now we will show that a restriction $R$ is not needed to prove RQ-unsatisfiabilty of an RQ-clause set if $R$ is not constraint unifiable.

**Theorem 4.4** *Let $\Delta$ be a signature, let $\mathcal{T}$ be a satisfiable set of a $\Delta$-formulas, and let $R_1, \ldots, R_n$ be restrictions such that property (TM) holds. Then $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$ iff $\mathcal{T} \models \exists(R_{U_1} \vee \ldots \vee R_{U_m})$, where $R_{U_1}, \ldots, R_{U_m}$ are the constraint unifiable restrictions in $R_1, \ldots, R_n$.*

*Proof:* If $\mathcal{T} \models \exists(R_{U_1} \vee \ldots \vee R_{U_m})$ then obviously $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$. Conversely, let now $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$, i.e., the set

$$\mathcal{T}' := \mathcal{T} \cup \{\forall \neg R_i \mid i = 1, \ldots, n\}$$

is unsatisfiable. But let us assume that $\mathcal{T} \not\models \exists(R_{U_1} \vee \ldots \vee R_{U_m})$, i.e., the set

$$\mathcal{T}'_U := \mathcal{T} \cup \{\forall \neg R_{U_j} \mid j = 1, \ldots, m\}$$

is satisfiable. Since $\mathcal{T}$ and $R_1, \ldots, R_n$ satisfy (TM), for all ground terms $s$, $t$ holds $\mathcal{T} \cup \{\forall \neg R_{U_j} \mid j = 1, \ldots, m\} \models s = t$ iff $s$ and $t$ are identical. Thus $\mathcal{T}'_U$ has a term-model $\mathcal{A}$ over the signature $\Delta_C$ which extends $\Delta$ by a countable infinite set $C$ of new constants (see Theorem A.3 in the appendix). We will show that in this case $\mathcal{T}'$ also has a term-model over $\Delta_C$ and thus is satisfiable, what contradicts $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$. Thereby, note that each restriction $R_i$ is a conjunction of equational restrictions $E_1, \ldots, E_{k_i}$ and non-equational restrictions $C_1, \ldots, C_{l_i}$ (with $k_i + l_i \geq 1$).

If $\alpha_0$ is an arbitrary but fixed $\Delta_C$-assignment then $(\mathcal{A}, \alpha_0) \models \mathcal{T}'_U$, and thus especially $(\mathcal{A}, \alpha_0) \models \neg E_j \vee \neg C_j$ for each $j \in \{1, \ldots, m\}$. Let now $R_N = E_N \wedge C_N$, $1 \leq N \leq m$, be a restriction which is *not* constraint unifiable. By definition of constraint unifiability, there are two possibilities: Firstly, let $E_N$ not be unifiable. Since unifiability of a set $E$ of equations is equivalent to satisfiability of the existential closure of $E$ in each term-model, in this case we have $(\mathcal{A}, \alpha) \models \neg E_N$ for each $\Delta_C$-assignment $\alpha$, and thus especially $(\mathcal{A}, \alpha_0) \models \neg E_N$. That means, $(\mathcal{A}, \alpha_0) \models \neg E_N \vee \neg C_N$. Secondly, let $E_N$ be unifiable, say with mgu $\sigma$, but let $\sigma C_N$ be unsatisfiable. If $(\mathcal{A}, \alpha_0) \not\models E_N$, then obviously $(\mathcal{A}, \alpha_0) \models \neg E_N \vee \neg C_N$. If, on the other hand, $(\mathcal{A}, \alpha_0) \models E_N$, then $\alpha_0$ is a unifier of $E_N$ and hence $\alpha_0 = \beta \circ \sigma$ for some $\beta$. If $(\mathcal{A}, \alpha_0) \models C_N$ as well, then $(\mathcal{A}, \beta \circ \sigma) \models C_N$ and thus $(\mathcal{A}, \beta) \models \sigma C_N$ which contradicts the unsatisfiability of $\sigma C_N$. That means, $(\mathcal{A}, \alpha_0) \models \neg C_N$, and therefore $(\mathcal{A}, \alpha_0) \models \neg E_N \vee \neg C_N$.

Summing up, $(\mathcal{A}, \alpha_0) \models R_N$ if $R_N$ is a restriction which is not constraint unifiable, and $\mathcal{A} \models \mathcal{T}$ since $\mathcal{A} \models \mathcal{T}'_U$. Since $\alpha_0$ is an arbitrary but fixed $\Delta_C$-assignment, $\mathcal{A}$ is a $\Delta_C$-model of $\mathcal{T}'$, and therefore $\mathcal{T}'$ has a $\Delta$-model (since $\mathcal{T}'$ contains $\Delta$-formulas only). This contradicts the presumption $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$ and thus we can conclude $\mathcal{T} \models \exists(R_{U_1} \vee \ldots \vee R_{U_m})$. □

The next theorem shows us how to use the constraint unifiable restrictions in order to prove the unsatisfiability of a set of RQ-clauses.

**Theorem 4.5** *Let $\Delta$ be a signature, let $\mathcal{T}$ be a satisfiable set of $\Delta$-formulas, and let $R_1, \ldots, R_n$ be restrictions such that condition (TM) is satisfied. If each restriction $R_i$ is given by $E_{i_1} \wedge \ldots \wedge E_{i_k} \wedge C_{i_1} \wedge \ldots \wedge C_{i_l}$, where $E_{i_1}, \ldots, E_{i_k}$ are the equational restrictions in $R_i$, and if each conjunction $E_{i_1} \wedge \ldots \wedge E_{i_k}$ is unifiable with the mgu $\sigma_i$, then $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$ iff $\mathcal{T} \models \exists(\sigma_1 C_1 \vee \ldots \vee \sigma_n C_n)$.*

*Proof:* Let $R$ be the restriction $E \wedge C$, where $E$ and $C$ are abbreviations for the conjunction of the equational restrictions and the conjunction of the non-equational restrictions in $R$, respectively. Furthermore, let $R$ be constraint unifiable. Firstly, we will show: If $E$ is unifiable with the mgu $\sigma$, then $\mathcal{T} \models \exists R$ iff $\mathcal{T} \models \exists \sigma C$.

Let $\mathcal{T} \models \exists R$, i.e., the set

$$T' := \mathcal{T} \cup \{\forall \neg E \vee \neg C\}$$

is unsatisfiable. Suppose $\mathcal{T} \not\models \exists \sigma C$, i.e., the set

$$T'_\sigma := \mathcal{T} \cup \{\forall \neg \sigma C\}$$

is satisfiable. Then, since for all ground terms $s$, $t$ obviously $\mathcal{T}'_\sigma \models s = t$ iff $s$ and $t$ denote the same ground term, $\mathcal{T}'_\sigma$ has a term-model $\mathcal{A}$ over the signature $\Delta_C$ which extends $\Delta$ by a countable infinite set of new constants (Theorem A.3 in the appendix). We will show that in this case $\mathcal{T}'$ has a term-model over $\Delta_C$ and thus is satisfiable.

If $\alpha_0$ is an arbitrary but fixed $\Delta_C$-assignment, then $(\mathcal{A}, \alpha_0) \models \neg \sigma C$ and thus $(\mathcal{A}, \alpha_0 \circ \sigma) \models \neg C$. If, on the one hand, $(\mathcal{A}, \alpha_0) \models \neg E$, then $(\mathcal{A}, \alpha_0) \models \neg E \vee \neg C$. If, on the other hand, $(\mathcal{A}, \alpha_0) \models E$, then $\alpha_0$ is a unifier of $E$ since $\mathcal{A}$ is a term-model. It is a well-known fact that

$$(\star) \qquad \alpha_0 = \alpha_0 \circ \sigma$$

if $\alpha_0$ is a unifier and $\sigma$ is the mgu of a set $E$ of equations.[4] Because of $(\mathcal{A}, \alpha_0 \circ \sigma) \models \neg C$ we obtain therefore $(\mathcal{A}, \alpha_0) \models \neg C$, i.e., $(\mathcal{A}, \alpha_0) \models \neg E \vee \neg C$. Summing up, if $\mathcal{T} \models \exists R$ then $\mathcal{T}'_\sigma$ is satisfiable and thus $\mathcal{T} \models \exists \sigma C$.

Conversely, let $\mathcal{T} \models \exists \sigma C$, i.e., the set $T'_\sigma := \mathcal{T} \cup \{\forall \neg \sigma C\}$ is unsatisfiable. Suppose $\mathcal{T} \not\models \exists R$, i.e., the set $\mathcal{T}' := \mathcal{T} \cup \{\forall \neg E \vee \neg C\}$ is satisfiable. Analogously to the first case we then know that $\mathcal{T}'$ has a $\Delta_C$-term model, say $\mathcal{A}$, and we will show that then $\mathcal{T}'_\sigma$ has a term-model over $\Delta_C$ as well and thus is satisfiable.

If $\alpha_0$ is an arbitrary but fixed $\Delta_C$-assignment, then $(\mathcal{A}, \alpha_0) \models \neg E \vee \neg C$. Again we distinguish two cases. If, on the one hand, $(\mathcal{A}, \alpha_0) \models E$ then obviously $(\mathcal{A}, \alpha_0) \models \neg C$. But because of $(\star)$ we can conclude $(\mathcal{A}, \alpha_0 \circ \sigma) \models \neg C$, and thus $(\mathcal{A}, \alpha_0) \models \sigma C$. Let, on the other hand, $(\mathcal{A}, \alpha_0) \not\models E$ and suppose $(\mathcal{A}, \alpha_0) \models \sigma C$. Since $\sigma$ is an mgu of $E$ we know $(\mathcal{A}, \alpha_0) \models \sigma E$.[5] That means $(\mathcal{A}, \alpha_0) \models \sigma E \wedge \sigma C$. But since we assumed $\mathcal{A}$ to be a term-model of $\mathcal{T} \cup \{\forall \neg E \vee \neg C\}$ we know $(\mathcal{A}, \alpha) \models \neg E \vee \neg C$ for each $\Delta_C$-assignment $\alpha$. This contradicts our assumption $(\mathcal{A}, \alpha_0) \models \sigma C$. Summing up, if $\mathcal{T} \models \exists \sigma C$, then the set

[4] Without loss of generality, one can assume that mgu's are idempotent, i.e., $\sigma \circ \sigma = \sigma$ for each mgu $\sigma$ of a set $E$ of equations. Therefore $\alpha_0 = \beta \circ \sigma = \beta \circ \sigma \circ \sigma = \alpha_0 \circ \sigma$ if $\sigma$ is the mgu and $\alpha_0 = \beta \circ \sigma$ is some unifier of $E$.

[5] If $\sigma$ is a mgu of a set $E$ of equations, then $\mathcal{A} \models \forall \sigma E$ for each structure $\mathcal{A}$, i.e., $(\mathcal{A}, \alpha) \models \sigma E$ for all assignments $E$.

$\mathcal{T}'$ is unsatisfiable, and that means $\mathcal{T} \models \exists R$. Thus we have shown that $\mathcal{T} \models \exists(E \wedge C)$ iff $\mathcal{T} \models \exists\sigma C'$.

We still have to show that $\mathcal{T} \models \exists(R_1 \vee \ldots \vee R_n)$ iff $\mathcal{T} \models \exists(\sigma_1 C_1 \vee \sigma_n C_n)$ if each $R_i$ is given by the conjunction of equational restrictions $E_i$ and non-equational restrictions $C_i$, and $\sigma_i$ is a mgu of $E_i$. This is easy to verify: Obviously, $\mathcal{T} \models \exists(E_1 \wedge C_1) \vee \ldots \vee (E_n \wedge C_n)$ iff for each $\Delta$-structure $\mathcal{A}$ with $\mathcal{A} \models \mathcal{T}$ there exists an index $i_{\mathcal{A}}$ such that $\mathcal{A} \models \exists(E_{i_{\mathcal{A}}} \wedge C_{i_{\mathcal{A}}})$. This is the case iff $\mathcal{A} \models \exists\sigma_{i_{\mathcal{A}}} C_{i_{\mathcal{A}}}$ for each $\Delta$-structure $\mathcal{A}$ that satisfies $\mathcal{T}$, i.e., $\mathcal{T} \models \exists(\sigma_1 C_1 \vee \ldots \vee \sigma_n C_n)$. □

The theorem shows that the *redundant* predicate can be instantiated by testing constraint unifiability if the given RQS satsifies condition (TM). By this optimization, search space of the RQ-resolution principle may be decreased largely.

# 5 Conclusion

In this paper we have presented a general refutation procedure for testing a set of RQ-formulas on RQ-unsatisfiability. For this procedure we extended the work of Bürckert [Bür91] by a transformation procedure of RQ-formulas into RQ-clauses and by a generalization of the RQ-factor and the RQ-resolution rule.

The first part of the paper is concerned with the question of how to transform a set of RQ-formulas into a set of RQ-clauses preserving RQ-satisfiability. It turned out that this task is not obvious since quantification over the empty set may occur, i.e., restrictions may be interpreted as empty set in some RQ-structures. Since the truth values of the RQ-formulas $\forall_{X:R} F$ and $\exists_{X:R} F$ depend on $F$ in RQ-structures which interpret $R$ as non-empty set, and do not depend on $F$ in RQ-structures which interpret $R$ as empty set we introduced the method of quantifier splitting. Thereby, the possible interpretations of restrictions as empty and as non-empty set are made explicit.

Building upon this, we investigated how to eliminate restricted $\exists$-quantifiers in RQ-formulas via Skolemization. Thereby, we obtained tuples of Skolem functions together with Skolem declarations which are added to the restriction theory. Since this extension of the restriction theory must not influence its models these declarations also have to distinguish between empty and non-empty interpretation of restrictions. After quantifier splitting and Skolemization, the transformation of RQ-formulas into RQ-clauses does no longer cause deep problems. This transformation procedure can, e.g., be used in sorted logics even if empty sorts are alloed. This is due to the fact that a sort hierarchy can be seen as restriction theory and sorts as unary restrictions.

We have introduced constraint unification, and it was shown that this unification procedure can be used to define the redundancy predicate if the RQS satisfies condition (TM). We proved that filtering out RQ-clauses whose restrictions are not constraint

unifiable reduces the search space without affecting refutation completeness (provided that condition (TM) holds).

This result is utilized in [BHL93] where the concept language $\mathcal{ALC}$ is taken to define a background theory. There, it has been shown that the resulting RQS satisfies condition (TM) and thus restrictions are tested on constraint unifiability rather than on RQ-satisifiabilty. Furthermore, an idea is presented how to use the general refutation procedure for abductive reasoning.

Within DFKI a prototypical implementation of a constrained resolution prover exists, and we are still working on optimizations to handle restricted quantification systems which satisfy condition (TM).

# A Theories with Term-models

In the following we will give proofs for the theorems on term-models which have been used in Section 4. Most of the techniques we use are adapted from Chapter 2 of [CK90].

Basis for our argumentation is the notion of a witness. Let $\mathcal{F}$ be a set of $\Delta$-formulas, and let $\exists x R(x)$ be a $\Delta$-formula, where $x$ is the only free variable in $R$. Then a ground term $t$ is called a **witness** for $R$ in $\mathcal{F}$ over $\Delta$ iff $\mathcal{F} \models \exists x R(x) \rightarrow R(t)$. We say $\mathcal{F}$ has witnesses over $\Delta$ iff for every $\Delta$-formula $\exists x R(x)$, where $x$ is the only free variable in $R$, there exists a witness for $R$ over $\Delta$. We will abbreviate the set of all ground terms over a signature $\Delta$ by $T_\Delta$.

**Lemma A.1** *Let $\mathcal{F}$ be a satisfiable set of $\Delta$-formulas, let $C$ be a countable infinite set of new constants, and $\Delta_C := \Delta \cup C$. Then $\mathcal{F}$ can be extended to a satisfiable (possibly infinite) set $\mathcal{F}_C$ of $\Delta_C$-formulas which has witnesses over $\Delta_C$.*

*Proof:* Let $R_1, R_2, R_3, \ldots$ be an enumeration of all $\Delta_C$-formulas with exactly one free variable, say $x_i$ in $R_i$. Then we define a sequence $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \ldots$ of sets of $\Delta_C$-formulas by

1. $\mathcal{F}_0 := \mathcal{F}$
2. $\mathcal{F}_i := \mathcal{F}_{i-1} \cup \{\exists x_i R_i(x_i) \rightarrow R_i(t_i)\}$, where $x_i$ is the free variable in $R_i$, such that
   (a) if there already exists a witness $t \in T_{\Delta_C}$ for $R_i$ in $\mathcal{F}_{i-1}$ over $\Delta_C$, then let $t_i := t$.
   (b) if there does not exist a witness for $R_i$ in $\mathcal{F}_{i-1}$ over $\Delta_C$, let $c_i$ be a new constant (i.e. $c_i \in C$, but $c_i$ does neither occur in $\mathcal{F}_i$ nor in $R_i$), and $t_i := c_i$.

Furthermore we define

$$\mathcal{F}_C := \bigcup_i \mathcal{F}_i$$

Then by construction $\mathcal{F}_C$ has witnesses over $\Delta_C$, since for each $\Delta_C$-formula $\exists x R(x)$, where $x$ is the only free variable in $R$, there exists a witness for $R$ in $\mathcal{F}_C$ over $\Delta_C$, namely $t_i$, if $R$ is the $i$-th formula in the above enumeration of all $\Delta_C$-formulas with exactly one free variable.

We still have to show that $\mathcal{F}_C$ is a satisfiable set of $\Delta_C$-formulas. This can be done by induction over $i$: Obviously, $\mathcal{F}_0 = \mathcal{F}$ is a satisfiable set of $\Delta_C$-formulas since $\mathcal{F}$ is a satisfiable set of $\Delta$-formulas and $\Delta_C$ is an extension of the signature $\Delta$. For the induction step we thus assume $\mathcal{F}_{i-1}$ to be a satisfiable set of $\Delta_C$-formulas.

By definition $\mathcal{F}_i$ is computed from $\mathcal{F}_{i-1}$ by adding the $\Delta_C$-formula $\exists x_i R_i(x_i) \rightarrow R_i(t_i)$. If, for case (a), $t_i$ is already a witness for $R_i$ in $\mathcal{F}_{i-1}$ over $\Delta_C$, i.e. $\mathcal{F}_{i-1} \models \exists x_i R_i(x_i) \rightarrow R_i(t_i)$, then $\mathcal{F}_i$ obviously is a satisfiable set of $\Delta_C$-formulas.

Let, for case (b), $t_i = c_i$ be a new constant. If $\mathcal{F}_i$ is unsatisfiable, then

$$\mathcal{F}_{i-1} \models \neg(\exists x_i R_i(x_i) \rightarrow R_i(t_i)).$$

This is equivalent to

$$\mathcal{F}_{i-1} \models \exists x_i R_i(x_i) \wedge \neg R_i(t_i).$$

As the witness $t_i$ is a fresh constant, i.e. it does neither occur in $\mathcal{F}_{i-1}$ nor in $R_i$, this is equivalent to

$$\mathcal{F}_{i-1} \models \forall x_i(\exists x_i R_i(x_i) \wedge \neg R_i(x_i)), \text{and thus}$$

$$\mathcal{F}_{i-1} \models \exists x_i R_i(x_i) \wedge \neg \exists x_i R_i(x_i)),$$

which contradicts the satisfiability of the set $\mathcal{F}_{i-1}$ of $\Delta_C$-formulas. □

For the next lemma we need the notion of a term-model. If $\mathcal{F}$ is a set of formula over some signature $\Delta$, then $\mathcal{A}$ is a **term-model** of $\mathcal{F}$ over $\Delta$ iff $\mathcal{A} \models \mathcal{F}$, all elements in the universe $U^{\mathcal{A}}$ are interpretations of $\Delta$-ground terms, and two different $\Delta$-ground terms denote different elements in $U^{\mathcal{A}}$.

**Lemma A.2** *Let $\Delta$ be a signature with equality, and let $\mathcal{F}$ be a satisfiable set of $\Delta$-formulas which has witnesses over $\Delta$. Then:*

1. *$\mathcal{F}$ has a model $\mathcal{A}$ such that each element in the universe $U^{\mathcal{A}}$ is an interpretation of a ground term over $\Delta$.*

2. *If, in addition, $\mathcal{F} \models s = t$ iff $s$ and $t$ denote the same ground term over $\Delta$, then $\mathcal{F}$ has a term-model over $\Delta$.*

*Proof:* Let $\mathcal{A}$ be a $\Delta$-structure such that the universe $U^{\mathcal{A}}$ is the set of ground terms over $\Delta$, the algebra reduct[6] of $\mathcal{A}$ is the ground term algebra, and

$$(\star) \qquad p^{\mathcal{A}}(t_1, \ldots, t_n) \text{ iff } \mathcal{F} \models p(t_1, \ldots, t_n)$$

for all $t_i \in T_\Delta$ and for all predicates $p \in \Delta$ (including equality). We will show that $\mathcal{A} \models \mathcal{F}$ by induction over the number of connectives and quantifiers in $\Delta$-formulas. Therefore we have to show that $\mathcal{A} \models F$ iff $\mathcal{F} \models F$ for each $\Delta$-formula $F$. If $F$ does neither contain a connective nor a quantifier, then $F$ is of the form $p(t_1, \ldots, t_n)$ where

[6]The algebra reduct of a $\Delta$-structure $\mathcal{A}$ is the part of $\mathcal{A}$ which does not interpret the predicate symbols in $\Delta$.

each $t_i \in T_\Delta$ and $p$ is a predicate in $\Delta$. Because of $(\star)$ we then know $\mathcal{A} \models F$ iff $\mathcal{F} \models F$. We thus assume that $\mathcal{A} \models F$ iff $\mathcal{F} \models F$ for all $\Delta$-formulas $F$ with $n$ connectives and quantifiers.

The induction step for formulas of the form $\neg F$ and $F_1 \wedge F_2$ is trivial, and here is nothing to show for the connective $\vee$ and the quantifier $\forall$, since they can be expressed by negation together with $\wedge$ and $\exists$, respectively. Let now $F$ be of the form $\exists x R(x)$, where $R$ contains $n$ connectives and quantifiers. If $\mathcal{A} \models \exists x R(x)$ then there exists a $\Delta$-assignment $\alpha$ such that $\mathcal{A}, \alpha \models R(x)$, and $\alpha(x) = t$ for some ground term $t$. Hence $\mathcal{A} \models R(t)$ and, by induction hypothesis, $\mathcal{F} \models R(t)$. Since $R(t) \rightarrow \exists x R(x)$ is a tautology, we obtain $\mathcal{F} \models \exists x R(x)$.

If, on the other hand, $\mathcal{F} \models \exists x R(x)$. Then $\mathcal{F} \models R(t)$ for some ground term $t$ since $\mathcal{F}$ has witnesses over $\Delta$. By induction hypothesis we obtain $\mathcal{A} \models R(t)$ and thus $\mathcal{A} \models \exists x R(x)$.

Suppose now $\mathcal{F} \models s = t$ iff $s$ and $t$ denote the same ground terms. Because of $(\star)$ for the equality predicate we then know that $\mathcal{A} \models s = t$ iff $\mathcal{F} \models s = t$ iff $s$ and $t$ denote the same ground terms. That means, $\mathcal{A}$ is a term-model of $\mathcal{F}$. □

**Remark:** The precondition of 2. of the above lemma holds, for instance, if equality has at most negative occurrences in $\mathcal{F}$. In this case just the trivial equations can be deduced from $\mathcal{F}$.

**Theorem A.3** *Let $\mathcal{F}$ be a satisfiable set of formulas over a signature $\Delta$ such that $\mathcal{F} \models s = t$ iff $s$ and $t$ denote the same ground term over $\Delta$. If $C$ is a countable infinite set of new constants, then $\mathcal{F}$ has a $\Delta$-model iff $\mathcal{F}$ has a term-model over $\Delta \cup C$.*

*Proof:* Let $\Delta_C$ be the signature which consists of $\Delta$ and the constants in $C$. If $\mathcal{A}$ is a term-model of $\mathcal{F}$ over $\Delta_C$, then $\mathcal{A}$ is a $\Delta_C$-model of $\mathcal{F}$. Since $\mathcal{F}$ is a set of $\Delta$-formulas thus $\mathcal{A}$ also is a $\Delta$-model of $\mathcal{F}$.

If $\mathcal{F}$ has a $\Delta$-model, then $\mathcal{F}$ is a satisfiable set of $\Delta$-formulas, and thus $\mathcal{F}$ can be extended to a satisfiable (possibly infinite) set $\mathcal{F}_C$ of $\Delta_C$-formulas which has witnesses over $\Delta_C$ (Lemma A.1). Obviously, if we construct $\mathcal{F}_C$ from $\mathcal{F}$ as we did in the proof of Lemma A.1, $\mathcal{F}_C \models s = t$ iff $\mathcal{F} \models s = t$, i.e. $\mathcal{F}_C \models s = t$ iff $s$ and $t$ denote the same ground terms over $\Delta_C$. Thus $\mathcal{F}_C$ has a term-model over $\Delta_C$ because of Lemma A.2. Since $T \subseteq T_C$ therefore $\mathcal{F}$ has a term-model over $\Delta_C$. □

# References


[BBH+90] F. Baader, H.-J. Bürckert, B. Hollunder, W. Nutt, and J. H. Siekmann. Concept logics. In Lloyd, editor, *Proceedings of Symposium on Computational Logic*, ESPRIT Basic Research Series, pages 177–201. Springer, 1990.

[BHL93] H.-J. Bürckert, B. Hollunder, and A. Laux. Concept logics with function symbols. Research Report RR-93-07, DFKI Saarbrücken, 1993.

[Bür91] H.-J. Bürckert. *A Resolution Principle for a Logic with Restricted Quantifiers*, volume 568 of *LNAI*. Springer, 1991.

[Bür93] H.-J. Bürckert. A Resolution Principle for Constrained Logics. *Artificial Intelligence*, 1993. To appear.

[CK90] C. C. Chang and H. J. Keisler. *Model Theory*. North-Holland, 1990.

[Coh92] A.G. Cohn. A many sorted logic with possibly empty sorts. In *Proceedings of the 11th Conference on Automated Deduction*, number 607 in LNAI, pages 633–647. Springer, 1992.

[FS90] A. Frisch and R. Scherl. A constraint logic approach to modal deduction. In J. Eijck, editor, *Logics in AI, European Workshop JELIA '90*, pages 234–250. Springer, 1990.

[FS91] A. Frisch and R. Scherl. A general framework for modal deduction. In Allen, Fikes, and Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proceeding of the Second International Conference*, San Mateo, 1991.

[HS88] M. Höhfeld and G. Smolka. Definite relations over constraint languages. LILOG–Report 53, IBM Deutschland, 1988.

[Obe62] A. Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik (in german). *Mathematische Annalen*, 145:297 333, 1962.

[Plo72] G. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[Sch89] M. Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *LNAI*. Springer, 1989.

[Wal87] C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Research Notes in Artificial Intelligence. Pitman Publishing, London, 1987.

[Wal88] C. Walther. A many-sorted unification. *JACM*, 35(1):1–17, 1988.

[Wei92] C. Weidenbach. A new sorted logic. In *Proceedings of the GWAI-92*, 1992.

[WO90] C. Weidenbach and H.-J. Ohlbach. A resolution calculus with dynamic sort structures and partial functions. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 688–693. Pitman Publishing, London, August 1990.

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 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.

### 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 Tschaitschian:* 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:*
Skeletel Plans Reuse: A Restricted Conceptual Graph Classification Approach
13 pages

**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 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

## 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 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

## 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 Dittrich, 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-$).

# On Skolemization in Constrained Logics

**Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux**

**RR-93-06**
Research Report