PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

HORN EQUALITY THEORIES AND COMPLETE SETS OF TRANSFORMATIONS

Steffen Hölldobler
Universität der Bundeswehr München, Fakultät für Informatik
Werner-Heisenberg-Weg 39, D-8014 Neubiberg, Germany

ABSTRACT

The idea to combine the advantages of function and logic programming has attracted many researches. Their work ranges from the integration of existing languages over higher-order logic to equational logic languages, where logic programs are augmented with equational theories. Recently, it has been proposed to handle those equational theories by complete sets of transformations. These transformations are extensions of the rules introduced by Herbrand and later used by Martelli and Montanari to compute the most general unifier of two expressions. In this paper we will generalize this idea to complete sets of transformations for arbitrary Horn equality theories, the largest class of equational theories that admit a least Herbrand model. The completeness proof is based on the observation that each refutation with respect to paramodulation and reflection can be modelled by the transformations. As certain conditions imposed on an equational theory restrict the search space generated by paramodulation and reflection we can easily refine our transformations - due to the completeness proof - if the Horn equality theory is ground confluent or canonical.

1 INTRODUCTION

In recent years many proposals have been made to combine function and logic programming [2]. They range from the integration of existing languages, e.g. LOGLISP [39] or QUTE [40], over higher-order logic (e.g. [32]), to equational logic languages, where logic programs are augmented with equational theories, e.g. EQLOG [14]. These equational theories can be handled by E-unification (e.g. [10,35]), by paramodulation or special forms of it (e.g. [8,36,38]), by flattening and SLD-resolution (e.g. [1]), or by complete sets of transformations [11,12,16,31]. These transformations are extensions of the rules introduced by Herbrand [15] and later used by Martelli & Montanari [30] to compute the most general unifier of two expressions.

However, equational theories are not the largest class that admit a least Herbrand model or, equivalently, an initial semantics. This is the class of Horn equality theories [29]. In this paper we will define complete sets of transformations for arbitrary Horn equality theories. To prove the completeness we will make use of the completeness results known for (linear) paramodulation [9] or special forms of it [18]. We will show by a simple proof that each refutation wrt (with respect to) paramodulation and reflection can be modelled by the transformations. This proof allows to refine our transformations if the equational theory is ground confluent or canonical in much the same way as narrowing refines paramodulation. Finally, we will show that for canonical theories rewriting can be applied as a simplification rule.

In the following section we will briefly recall some basic notions and in section 3 we will give an account of the completeness results achieved for paramodulation and special forms of it. The transformations are introduced in section 4 and the completeness proof is given in section 5. In section 6 we will refine the transformation rules and we will finish by comparing our approach with others.

2 PRELIMINARIES

We assume the reader to be familiar with logic programming (e.g. [28]), equations, and rewrite rules (e.g. [20]). Throughout the paper we will make use of the notational conventions laid down in the following table in the sense that, whenever we use x, we implicitely assume that x is a variable. Set operators applied to multisets denote their multiset analogs. Furthermore, **Var(F)** denotes the set of variables occurring in F.

a,b,...	constructors	f,g,...	function symbols
E	equation	s,t,...	terms
EP	equational program	x,y,...	variables
F	multiset of equations	σ,θ,...	substitutions

An **equation** has the form ={s,t} or ={t}. Expressions of the form ={s,t} (resp. ={t}) are interpreted as **non-trivial** (resp. **trivial**) equations s=t (resp. t=t). The "labelled set" notation has been introduced by Sibert [41] and emphasizes that the order in which the terms s and t are written in an equation is immaterial. For notational convenience we will commonly use the more usual form s=t (resp. t=t) to represent ={s,t} (resp. ={t}).

An **equational program** EP consists of a finite set of equational clauses of the form $l \rightarrow r \Leftarrow F$. The arrow in the head of an equational clause emphasizes that equational clauses will be used only from left-to-right. Let $EP^{-1} = \{r \rightarrow l \Leftarrow F \mid l \rightarrow r \Leftarrow F \in EP\}$. To ease our notation we will often omit the curly brackets in the body of an equational clause. The semantics of an equational program EP can be given as the least Herbrand model for EP together with the axioms of equality for EP.

A **substitution** is defined to be a mapping from the set of variables into the set of terms. Substitutions are extended to morphisms on the set of terms and equations. $\sigma|_V$ denotes the restriction of σ to the set V of variables.

Since we will introduce several new inference rules we assume that derivations and refutations are defined wrt a set of inference rules.

3 PARAMODULATION

Paramodulation has been invented by Robinson & Wos [40] as a substitution rule for first order theories with equality. Furbach et al. [9] have recasted (linear) paramodulation as an inference rule for Horn equality theories: Let G be the goal clause $\Leftarrow F \cup \{E\}$, $P = l \rightarrow r \Leftarrow F^*$ be a new variant of a program clause, s be a subterm of E, and E' be the equation obtained from E by replacing the s by r. If s and l are unifiable with mgu σ, then $G' = \Leftarrow\sigma(F \cup \{E'\} \cup F^*)$ is called **paramodulant** of G and P, in symbols $G \rightarrow_{p(E,s,P,\sigma)} G'$. We will say that paramodulation has been applied upon an element of an equation s=t iff either s or t has been replaced.

To express syntactic equality we have to use the axiom of reflexivity: Let G be the goal clause $\Leftarrow F \cup \{s=t\}$. If s and t are unifiable with mgu σ, then $G' = \Leftarrow\sigma F$ is called **reflectant** of G, in symbols, $G \rightarrow_{r(s=t,\sigma)} G'$.

As the following example will show we need the functional reflexive axioms or, equivalently, an instantiation rule to ensure the completeness of paramodulation: Let G be the goal clause $\Leftarrow F \cup \{E\}$, x be a variable in E, f be an n-ary function symbol, $x_1,\ldots,x_n$ be new variables, and $\sigma = \{x \leftarrow f(x_1,\ldots,x_n)\}$, then $G' = \Leftarrow\sigma G$ is called **instance** of G, in symbols, $G \rightarrow_{i(E,\sigma)} G'$.

For notational convenience we will omit E, s, P, or σ when writing derivations if they can be determined by the context. Furthermore, we will depict the selected subgoal or subterm in boldface.

As an example consider the equational program

FUN: $g \rightarrow a \Leftarrow$ (g)

$f(c(g),c(a)) \rightarrow d(c(g),c(a)) \Leftarrow$ (f)

and the question whether there exists a substitution θ such that $\theta f(x,x) = \theta d(x,x)$ is a logical consequence of FUN. This question can be answered with $\theta = \{x \leftarrow c(g)\}$ by the refutation in figure 1.

$\Leftarrow f(x,x)=d(x,x)$	
$\rightarrow i(\{x \leftarrow c(y)\})$	$\Leftarrow f(c(y),c(y))=d(c(y),c(y))$
$\rightarrow p(g,\{y \leftarrow g\})$	$\Leftarrow \mathbf{f(c(g),c(a))}=d(c(g),c(g))$
$\rightarrow p(f)$	$\Leftarrow d(c(g),c(a))=d(c(g),c(g))$
$\rightarrow p(g)$	$\Leftarrow \mathbf{d(c(g),c(a))}=\mathbf{d(c(g),c(a))}$
$\rightarrow_r$	$\square$

Figure 1

The interested reader may verify that without the instantiation rule a refutation of $\Leftarrow f(x,x)=d(x,x)$ is impossible. Formally, the need for the instantiation rule comes from the lifting lemma, which states that, if there exists a refutation of $EP \cup \{\Leftarrow\sigma F\}$ with computed answer substitution θ, then there exists a refutation of $EP \cup \{\Leftarrow F\}$ and, furthermore, if γ is the computed answer substitution of this refutation, then γ is more general than $\theta\sigma$. In the proof of the lifting lemma one is confronted with the case that in the refutation of $EP \cup \{\Leftarrow\sigma F\}$ paramodulation is applied upon a term s which was introduced by σ. To be able to apply the respective paramodulation step to $\Leftarrow F$ we have to instantiate $\Leftarrow F$. As an example consider the parmodulation step

$\Leftarrow\{x \leftarrow c(g)\}(f(x,x)=d(x,x)) = \Leftarrow f(c(g),c(g))=d(c(g),c(g))$
$\rightarrow_p \Leftarrow f(c(g),c(a))=d(c(g),c(g))$

which was lifted in figure 1 using an instantiation and a paramodulation step.

This use of the instantiation rule suggest to define a new inference rule **instantiation and paramodulation** ($\rightarrow_{ip}$): $G \rightarrow_{ip(\sigma)} G'$ iff G' has been obtained from G be a (possibly empty) finite sequence of instantiation steps followed by a single paramodulation step and, if $\sigma_1,\ldots,\sigma_n$ are the substitutions used in this derivation, then $\sigma = \sigma_n \ldots \sigma_1$. E.g.,

$\Leftarrow f(x,x)=d(x,x) \rightarrow_{ip(\{x \leftarrow c(g)\})} \Leftarrow f(c(g),c(a))=d(c(g),c(g))$.

Note, an instantiation and paramodulation step corresponds to a paramodulation step using a "prefixed axiom" in [34].

The completeness of reflection, instantiation and paramodulation follows immediately from [9]:

Theorem 1: (Completeness of $\{\rightarrow_{ip},\rightarrow_r\}$)
If θ is a correct answer substitution for EP and $\Leftarrow F$ then there exists a computed answer substitution σ obtained by a refutation of $EP \cup EP^{-1} \cup \{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$ such that σ is more general than θ.

Clauses from EP^{-1} are needed in theorem 1, since we cannot assume in general that arbitrary equational programs are ground confluent (see FUN). In analogy to the respective result for SLD-resolution (e.g. [28]) it can easily be proved that refutations wrt reflection, instantiation and paramodulation are independent of a **computation rule**, i.e. a function which applied to a non-empty goal clause always selects an equation from that clause.

Theorem 2: (Independence of the computation rule)
Let R and R' be computation rules. If there exists a refutation of $EP \cup \{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substi-

tution σ, and via R, then there exists a refutation of EP∪{⇐F} wrt {→ip,→r}, via R', and, if σ' is the computed answer substitution of the refutation via R', then σ and σ' are variants. Furthermore, in both refutations paramodulation has been applied the same number of times.

Remark: If the computation rule selects a subgoal of the form x=y, where x and y are variables, then we cannot only apply reflection, but we have also to apply instantiation and paramodulation. However, due to theorem 2, we may choose a computation rule that never selects an equation of the form x=y if it has another choice. In such a refutation we will eventually encounter a goal clause of the form $\Leftarrow\{x_i=y_i \mid 1\le i\le n\}$ and it is easy to see that the completeness of reflection, instantiation and paramodulation is retained even if we apply only reflection to solve such a goal clause.

Of course, the search space generated by reflection, instantiation and paramodulation contains far to many redundant and irrelevant inferences and it has been proposed at first by Slagle [41] and Lankford [27] to impose certain conditions on equational theories such that paramodulation need not to be applied at variable occurrences. This restricted form of paramodulation is often called **narrowing** (e.g [21]). Obviously, instantiation is no longer needed if it suffices to apply paramodulation upon non-variable terms.

In [18] these refinements of paramodulation have formally been developed for Horn equational theories. It has been shown that clauses from EP^{-1} are no longer needed if the equational program is ground confluent. Furthermore, (conditional) narrowing ($\rightarrow_n$) can be applied instead of instantiation and paramodulation if the equational program is a collapse-free and ground confluent term rewriting system and the answer substitution is normalizable, where an equational program is said to be **collapse-free** iff it does not contain a **collapse clause**, i.e. a rule of the form $x\rightarrow r\Leftarrow F$, and a **term rewriting system** is an equational program, where for each clause $l\rightarrow r\Leftarrow F$ we find that each variable occurring in F and r occurs also in l. The set of function symbols is divided by a term rewriting system into two disjoint subsets, the set of **defined function symbols** and the set of **constructors**, where f is said to be a defined function symbol iff the term rewriting system contains a rule for f.

Theorem 3: *Let EP be a ground confluent equational program and R be a computation rule. If θ is a correct answer substitution for EP and ⇐F, then there exists an R-computed answer substitution σ obtained by a refutation of EP∪{⇐F} wrt {→ip,→r} such that σ is more general than θ.*

Theorem 4: (Strong completeness of $\{\rightarrow_n,\rightarrow_r\}$)
Let EP be a collapse-free and ground confluent term rewriting system and R be a computation rule. If θ is a normalized correct answer substitution for EP and ⇐F, then there exists an R-computed answer substitution σ obtained by a refutation of EP∪{⇐F} wrt {→n,→r} such that σ is more general than θ.

As a consequence narrowing and reflection is complete for canonical conditional term rewriting systems and rewriting can be applied as a simplification rule, where a goal bind a variable in G. Moreover, we may apply other simplifications rules such as removal of trivial equations, decomposition of decomposible equations [26], and elimination of variables if the goal clause contains an equation of the form x=t and no defined function symbol occurs in t.

4 THE TRANSFORMATIONS

As we have mentioned in the introduction the transformation rules are an extension of the rules invented by Herbrand [15] and Martelli & Montanari [30] to compute the most general unifier of two expressions. Therefore, we will briefly repeat these rules:

The **term decomposition** ($\rightarrow_d$) rule decomposes an equation of the form $f(s_1,\ldots,s_n)=f(t_1,\ldots,t_n)$ into the set of corresponding arguments, i.e.

$$\Leftarrow F\cup\{f(s_1,\ldots,s_n)=f(t_1,\ldots,t_n)\} \rightarrow_d \Leftarrow F\cup\{s_i=t_i \mid 1\le i\le n\}.$$

The **variable elimination** ($\rightarrow_v$) rule applied to a goal clause $\Leftarrow F\cup\{x=t\}$ elimates the variable x by replacing each occurrence of x by t if x does not occur in t, i.e.

$$\Leftarrow F\cup\{x=t\} \rightarrow_{v(\{x\leftarrow t\})} \Leftarrow\{x\leftarrow t\}F.$$

The rule **removal of trivial equations** ($\rightarrow_t$) removes a trivial equation, i.e.

$$\Leftarrow F\cup\{t=t\} \rightarrow_t \Leftarrow F.$$

A reflection step can be modelled by a sequence of $\rightarrow_t$, $\rightarrow_v$, and $\rightarrow_d$ steps [30]. These transformations achieve syntactic unification, whereas the following three rules are only applicable wrt an equational program:

The **lazy narrowing** ($\rightarrow_{ln}$) rule applied to an equation of the form $f(s_1,\ldots,s_n)=s_{n+1}$ and using an equational clause $f(t_1,\ldots,t_n)\rightarrow t_{n+1}\Leftarrow F^*$ forces the comparison of corresponding arguments and right hand sides, i.e.

$$\Leftarrow F\cup\{f(s_1,\ldots,s_n)=s_{n+1}\} \rightarrow_{ln} \Leftarrow F\cup F^*\cup\{s_i=t_i \mid 1\le i\le n+1\}.$$

We have called this rule lazy, since the corresponding arguments are not immediately solved but added to the new goal clause and, hence, can be handled according to the overall strategy encoded in a computation rule.

The rule **paramodulation upon variables** ($\rightarrow_{pv}$) applied to an equation of the form x=s and using an equational clause $f(t_1,\ldots,t_n)\rightarrow r\Leftarrow F^*$ instantiates x to $f(x_1,\ldots,x_n)$ and, then, forces the comparison of corresponding arguments and right hand sides, i.e.

$$\Leftarrow F\cup\{x=s\} \rightarrow_{pv(\sigma)} \Leftarrow \sigma F\cup F^*\cup\{x_i=t_i \mid 1\le i\le n\}\cup\{\sigma s=r\},$$

where $\sigma=\{x\leftarrow f(x_1,\ldots,x_n)\}$, $x_1,\ldots,x_n$ are new variables, and s is a non-variable term.

So far we cannot use collapse clauses. The rule **application of a collapse clause** ($\rightarrow_{cc}$) applied to an equation s=t and using a collapse clause $x\rightarrow r\Leftarrow F^*$ forces the comparison of corresponding left (resp. right) hand sides, i.e.

$$\Leftarrow F\cup\{s=t\} \rightarrow_{cc} \Leftarrow F\cup F^*\cup\{s=x,r=t\}.$$

As we will learn from the proof of the completeness of the transformation rules, neither $\rightarrow_{ln}$, nor $\rightarrow_{pv}$, nor $\rightarrow_{cc}$ need to be applied upon s=x anymore.

Due to the lazy nature of the transformation rules introduced so far, lazy narrowing can only be applied to the elements of an equation but not to proper subterms of these elements. This would lead to an incompleteness of our transformation rules: Suppose the only program clause is $f(x)\rightarrow a\Leftarrow$ and considerter the goal clause $\Leftarrow y=c(f(y))$. In a refutation wrt narrowing and reflection, narrowing would be applied upon f(y) yielding $\Leftarrow y=c(a)$, which can be solved by binding c(a) to y. However, this refutation cannot be modelled by the transformation rules introduced so far.

The **imitation** ($\rightarrow_{im}$) rule applied to an equation of the form $x=f(t_1,\ldots,t_n)$ instantiates x to $f(x_1,\ldots,x_n)$ and then forces the comparison of corresponding arguments, i.e.

$$\Leftarrow F\cup\{x=f(t_1,\ldots,t_n)\} \rightarrow_{im(\sigma)} \Leftarrow \sigma F\cup\{x_i=\sigma t_i \mid 1\le i\le n\},$$

where $\sigma = \{x\leftarrow f(x_1,\ldots,x_n)\}$ and $x_1,\ldots,x_n$ are new variables. In our example,

$$\begin{array}{lll} \Leftarrow y=c(f(y)) & \rightarrow_{im(\{y\leftarrow c(z)\}} & \Leftarrow z=f(c(z)) \\ & \rightarrow_{ln} & \Leftarrow z=a,\ c(z)=x \\ & \rightarrow_{v(\{z\leftarrow a\})} & \Leftarrow c(a)=x \\ & \rightarrow_{v} & \square. \end{array}$$

Notation: In the sequel let **TRANS** = $\{\rightarrow_d,\rightarrow_v,\rightarrow_t,\rightarrow_{ln},\rightarrow_{pv},\rightarrow_{cc},\rightarrow_{im}\}$.

The transformation rules can be divided into three classes: The **unification rules** term decomposition, variable elimination, and removal of trivial equations, the **lazy paramodulation rules** lazy narrowing, paramodulation upon variables, and application of a collapse clause, and the **imitation rule**. It should be noted that we have no transformation which corresponds to the instantiation rule.

Our transformations can be regarded as an extension of the rules BT given by Gallier & Snyder [12] to conditional equational theories. They differ if the selected equation is of the form x=t: If t is a variable, then Gallier & Snyder provide an additional transformation, which instantiates the goal clause by $\{x\leftarrow f(x_1,\ldots,x_n)\}$. If t is not a variable, then Gallier & Snyder apply lazy paramodulation rules only upon t. As an example consider the term rewriting system $INF = \{f\rightarrow c(f)\Leftarrow\}$. Then,

$$\Leftarrow x=c(x) \rightarrow_{ln} \Leftarrow x=f, x=f \rightarrow_{v(\{x\leftarrow f\})} \Leftarrow f=f \rightarrow_t \square,$$

and in the lazy narrowing step the equational clause $c(f)\rightarrow f\Leftarrow \in INF^{-1}$ has been used. This is the only way Gallier & Snyder can solve the initial goal clause, since paramodulation at variables cannot be applied due to the chosen restriction, variable elimination cannot be applied since the occur check fails, and an imitation yields a variant of x=c(x). The strange observation about Gallier & Snyder's restricted use of lazy paramodulation rules is that even if the equational program is ground confluent (like INF) they have to use equational clauses in both directions, whereas without the restriction we find

$$\Leftarrow x=c(x) \rightarrow_{pv(\{x\leftarrow f\})} \Leftarrow c(f)=c(f) \rightarrow_t \square.$$

It is easy to see that the transformations are sound: Each derivation step wrt TRANS can be modelled by a sequence of resolution steps using the axioms of equality. Therefore, we will concentrate on the completeness proof.

5 COMPLETENESS OF THE TRANSFORMATIONS

To obtain the completeness of our transformations we will show that each refutation wrt reflection, instantiation and paramodulation can be modelled by our rules. Before we can turn to the proof itself we need some definitions and technical propositions.

Suppose $\Leftarrow F'$ has been obtained from $\Leftarrow F$ by reflection or instantiation and paramodulation using substitution σ. If $E \in F$ was not the selected equation, then $\sigma E \in F'$ is the **immediate descendant** of E. If E was the selected equation and $\rightarrow_{ip}$ has been applied transforming E into E', then $E' \in F'$ is the **immediate descendant** of E. E' is a **descendant** of E iff E' is in the transitive and reflexive closure of the "immediate descendant relation".

The **depth** of a variable is 1 and the **depth** of a term of the form $f(t_1,\ldots,t_n)$ is $1+\max\{\text{depth}(t_i) \mid 1\le i\le n\}$. For each substitution θ the complexity $\mathbf{D}(\theta)$ is defined to be the multiset $\{\text{depth}(t) \mid x\leftarrow t\in\theta\}$. As an example consider the substitutions $\theta = \{x\leftarrow c(a)\}$ and $\sigma = \{y\leftarrow a,\ z\leftarrow b\}$, then $D(\theta) = \{2\}$ and $D(\sigma) = \{1,1\}$. It should be noted that $D(\theta) = D(\sigma)$ whenever σ and θ are variants.

Dershowitz & Manna [5] have shown that a well-founded ordering $<$ over a set S induces a well-founded ordering $<<$ over multisets whose elements are taken from S as follows: Let M and M' be two finite multisets over S. $M' << M$ iff M' can be obtained from M by replacing one or more elements in M by any finite number of elements taken from S, each of which is smaller than one of the replaced elements. For example, $D(\{x_i\leftarrow t_i \mid 1\le i\le n\}) << D(\{x\leftarrow f(t_1,\ldots,t_n)\})$.

We can now assign a complexity to refutations wrt reflection, instantiation and paramodulation: The **complexity** of a refutation $EP\cup\{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$ and computed answer substitution θ is $<\#p,D(\theta),\#s,\#e>$, where #p is the number of applications of paramodulation in the refutation, #s is the number of symbols occurring in F, and #e is the number of equations in F. The ordering « is defined to be the lexicographic combination of: the $<$ ordering on natural numbers, the $<<$ ordering on multisets of natural numbers, the $<$ ordering on natural numbers, and again the $<$ ordering on natural numbers. Obviously, « is well-founded.

Proposition 5: *Let $E = f(s_1,\ldots,s_m)=f(t_1,\ldots,t_m)$. If there exists a refutation of $EP\cup\{\Leftarrow F\cup\{E\}\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ and complexity $M = <\#p,D(\theta),\#s,\#e>$, where paramodulation has never been applied upon an element of a descendant of E, then there exists a refutation of $EP\cup\{\Leftarrow F\cup\{s_i=t_i \mid 1\le i\le n\}\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ and complexity $<\#p,D(\theta),\#s-2,\#e+m-1> « M$.*

Proposition 6: *Suppose there exists a refutation of* $EP\cup\{\Leftarrow F\cup\{s=t\}\}$ *wrt* $\{\rightarrow_{ip},\rightarrow_r\}$, *computed answer substitution* θ *and complexity* $M = \langle\#p,D(\theta),\#s,\#e\rangle$, *where paramodulation has been applied upon an element, say s', of a descendant* $s'=t'$ *of* $s=t$. *Let* $P = l\rightarrow r\Leftarrow F^*$ *be the program clause used in the first of these applications. Then there exists a refutation of* $EP\cup\{\Leftarrow F\cup F^*\cup\{s=l,r=t\}\}$ *wrt* $\{\rightarrow_{ip},\rightarrow_r\}$, *computed answer substitution* θ' *and complexity* $\langle\#p\text{-}1,D(\theta'),\#s',\#e'\rangle \ll M$ *such that* $\theta'|_{Var(F\cup\{s=t\})} = \theta$.

The interested reader may verify that in the refutation of $EP\cup\{\Leftarrow F\cup F^*\cup\{s=l,r=t\}\}$ paramodulation need not be applied upon an element of a descendant of s=l.

Proposition 7: *Suppose there exists a refutation of* $EP\cup\{\Leftarrow F\}$ *wrt* $\{\rightarrow_{ip},\rightarrow_r\}$, *computed answer substitution* θ *and complexity M. Suppose* $x\leftarrow f(t_1,\ldots,t_n) \in \theta$ *and let* x_i, $1\le i\le n$, *be new variables and* $\gamma = \{x\leftarrow f(x_1,\ldots,x_n)\}$. *Then there exists a refutation of* $EP\cup\{\Leftarrow\gamma F\}$ *wrt* $\{\rightarrow_{ip},\rightarrow_r\}$, *computed answer substitution* θ' *and complexity M' such that* $\theta'\gamma|_{Var(F)}$ *is more general than* θ *and* $M' \ll M$.

As an example consider the program clause $f(x)\rightarrow a\Leftarrow$ and the refutation

$$\Leftarrow y=c(f(y)) \rightarrow_{p(\{x\leftarrow y\})} \Leftarrow y=c(a) \rightarrow_{v\{y\leftarrow c(a)\}} \square.$$

$\{y\leftarrow c(a)\}$ is the computed answer substitution and $\langle 1,\{2\},5,1\rangle$ the complexity of this refutation. Now, let $\gamma = \{y\leftarrow c(z)\}$. Then

$$\begin{array}{lll} \Leftarrow\gamma(y=c(f(y))) = & & \Leftarrow c(z)=c(f(c(z))) \\ & \rightarrow_{p(\{x\leftarrow c(z)\})} & \Leftarrow c(z)=c(a) \\ & \rightarrow_{r(\{z\leftarrow a\})} & \square \end{array}$$

with computed answer substitution $\{z\leftarrow a\}$ and complexity $\langle 1,\{1\},7,1\rangle \ll \langle 1,\{2\},5,1\rangle$.

It is easy to see that, if in the refutation of $EP\cup\{\Leftarrow F\}$ paramodulation has never been applied upon an element of a descendant of $E \in F$, then paramodulation has never been applied upon an element of a desecendant of $\gamma E \in \gamma F$ in the refutation of $EP\cup\{\Leftarrow\gamma F\}$.

We can now prove that for each refutation wrt paramodulation, instantiation, and reflection there exists a corresponding refutation wrt TRANS yielding a more general answer substitution. Recall, in a refutation wrt reflection, instantiation and paramodulation we may apply a computation rule which never selects an equation of the form x=y if it has another choice. Furthermore, if the goal clause contains only equations of this form then it suffices to apply reflection.

Notation: Let R^+ be a computation rule that obeys this strategy.

Theorem 8: *If there exists a refutation of* $EP\cup\{\Leftarrow F\}$ *wrt* $\{\rightarrow_{ip},\rightarrow_r\}$ *and computed answer substitution* θ, *then there exists a refutation of* $EP\cup\{\Leftarrow F\}$ *wrt TRANS and via* R^+. *Furthermore, if* σ *is the computed answer substitution of the refutation wrt TRANS, then* σ *is more general than* θ.

Proof: The proof is by induction on the complexity M of the refutation of $EP\cup\{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$. The case $M = \langle 0,\emptyset,0,0\rangle$ being trivial we turn to the induction step and assume that the result holds for all $M \ll M'$. Suppose

$$\Leftarrow F'\cup\{s=t\} \rightarrow^* \square \qquad (1)$$

wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ', and complexity $M' = \langle\#p',D(\theta'),\#s',\#e'\rangle$. Let s=t be the first selected equation by R^+. By theorem 2 we may assume that s=t is the first selected equation in (1).

1 If s and t are variables, then F' contains only equations of the form x=y and we may assume that in (1) only reflection has been applied. Since reflection can be modelled by the rules $\rightarrow_t$, $\rightarrow_v$, and $\rightarrow_d$, the theorem follows immediately.

In the remaining cases we assume that s or t is a non-variable term.

2 Suppose that in (1) paramodulation is applied upon an element, say s', of a descendant $s'=t'$ of s=t. Let $P = l\rightarrow r\Leftarrow F^*$ be the program clause used in the first such application. By proposition 6 we find

$$\Leftarrow F'\cup\{s=l,r=t\}\cup F^* \rightarrow^* \square \qquad (2)$$

wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ^* and complexity M^* such that $\theta^*|_{Var(F'\cup\{s=t\})} = \theta'$ and $M^* \ll M'$. Recall, in (2) paramodulation need not be applied upon an element of a descendant of s=l.

2.1 Suppose P is a collapse clause. Let $F = F'\cup F^*\cup\{s=l,r=t\}$. Then,

$$\Leftarrow F'\cup\{s=t\} \rightarrow_{cc} \Leftarrow F$$

and (2) ensures that there exists a refutation of $EP\cup\{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution $\theta = \theta^*$, and complexity $M = M^* \ll M'$. The result follows by the induction hypothesis.

In the remaining two cases we assume that P is of the form $f(l_1,\ldots,l_m)\rightarrow r\Leftarrow F^*$.

2.2 Suppose s is of the form $f(s_1,\ldots,s_m)$. Let $F = F'\cup F^*\cup\{s_i=l_i \mid 1\le i\le m\}\cup\{r=t\}$. Then,

$$\Leftarrow F'\cup\{s=t\} \rightarrow_{in} \Leftarrow F.$$

By an application of proposition 6 upon (2) we find a refutation of $EP\cup\{\Leftarrow F\}$ wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution $\theta = \theta^*$ and complexity $M \ll M^* \ll M'$. The result follows by the induction hypothesis.

2.3 Suppose s is a variable. Hence, t must be a non-variable term. Let x_i, $1\le i\le m$, be new variables, $\gamma = \{s\leftarrow f(x_1,\ldots,x_m)\}$, and $F = \gamma(F'\cup F^*\cup\{x_i=l_i \mid 1\le i\le m\}\cup\{r=t\})$. Then,

$$\Leftarrow F'\cup\{s=t\} \rightarrow_{pv} \Leftarrow F.$$

By an application of proposition 7 upon (2) we find

$$\begin{array}{ll} & \Leftarrow\gamma(F'\cup F^*\cup\{s=l,r=t\}) \\ = & \gamma F'\cup F^*\cup\{f(x_1,\ldots,x_m)=f(l_1,\ldots,l_m),r=\gamma t\} \\ \rightarrow^* & \square \end{array} \qquad (3)$$

wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ^+, and complexity M^+ such that $\theta^+\gamma|_{Var(\gamma(F'\cup F^*\cup\{s=l,r=t\}))}$ is more general that θ^* and $M^+ \ll M^* \ll M'$. Note, in (3) paramodulation has never been applied upon an ele-

ment of a descendant of $f(x_1,\ldots,x_m)=f(l_1,\ldots,l_m)$. Hence, by an application of proposition 5 upon (3) we find a refutation of EP∪{⇐F} wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution $\theta = \theta^+$, and complexity $M \ll M^+ \ll M^* \ll M'$. The result follows by the induction hypothesis.

3 Suppose that in (1) paramodulation is never applied upon an element of a descendant of s=t.

3.1 If reflection has been applied in the first step of (1) then the result follows by the induction hypothesis, since each reflection step in (1) can be replaced by a sequence of $\rightarrow_t$, $\rightarrow_d$, and $\rightarrow_v$ steps and each application of $\rightarrow_t$, $\rightarrow_d$, and $\rightarrow_v$ decreases M' .

In the remaining two cases we assume that instantiation or paramodulation (using $P = l\rightarrow r\Leftarrow F^*$) has been applied in the first step of (1). Recall, s and t cannot both be variables.

3.2 Suppose s (resp. t) is of the form $f(s_1,\ldots,s_m)$ (resp. $g(t_1,\ldots,t_n)$). Since in (1) paramodulation is never applied upon an element of a descendant of s=t we find that f=g and n=m. Let $F = F'\cup\{s_i=t_i \mid 1\le i\le m\}$. Then,

$$\Leftarrow F'\cup\{s=t\} \rightarrow_d \Leftarrow F$$

and by an application of proposition 5 upon (1) we find a refutation of EP∪{⇐F} wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution $\theta = \theta'$, and complexity $M \ll M'$. The result follows by the induction hypothesis.

3.3 Finally, suppose that s is a variable and t is a term of the form $f(t_1,\ldots,t_m)$. Since in (1) paramodulation is never applied upon an element of a descendant of s=t we find a binding $s\leftarrow f(s_1,\ldots,s_m)$ in θ'. Now let x_i, $1\le i\le m$, be new variables, $\gamma = \{x\leftarrow f(x_1,\ldots,x_m)\}$, and $F = \gamma(F'\cup\{x_i=t_i \mid 1\le i\le m\})$. Then,

$$\Leftarrow F'\cup\{s=t\} \rightarrow_{im} \Leftarrow F.$$

By an application of proposition 7 upon (1) we find a refutation of EP∪{⇐F} wrt $\{\rightarrow_{ip},\rightarrow_r\}$, computed answer substitution θ, and complexity M such that $\theta\gamma|_{Var(F'\cup\{s=t\})}$ is more general than θ' and $M \ll M'$. The result follows by the induction hypothesis. qed

The proof of theorem 8 gives us a procedure that transforms refutations wrt reflection, instantiation and paramodulation into refutations wrt TRANS, e.g this procedure transforms the refutation in figure 1 into the refutation depicted in figure 2.

It should be noted that the empty clause has been derived in figure 2 by applying only lazy narrowing, term decomposition, variable elimination, and removal of trivial equations. This is remarkable, since the FUN-example has served to show that paramodulation and reflection is complete only if an instantiation rule is added. Since lazy narrowing has been applied upon f(x,x) using (f), the uninformed use of the instantiation rule in the refutation of FUN∪{⇐f(x,x)=d(x,x)} wrt $\{\rightarrow_{ip},\rightarrow_r\}$ to instantiate the variable x has been replaced by an informed application of term decomposition in the corresponding refutation of FUN∪{⇐f(x,x)=d(x,x)} wrt TRANS.

⇐f(x,x)=d(x,x)	
$\rightarrow_{ln(f)}$	⇐**x=c(g)**, x=c(a), d(x,x)=d(c(g),c(a))
$\rightarrow_{v(\{x\leftarrow c(g)\})}$	⇐**c(g)=c(a)**, d(c(g),c(g))=d(c(g),c(a))
$\rightarrow_d$	⇐g=a, **d(c(g),c(g))=d(c(g),c(a))**
$\rightarrow_d$	⇐g=a, **c(g)=c(g)**, c(g)=c(a)
$\rightarrow_t$	⇐g=a, **c(g)=c(a)**
$\rightarrow_d$	⇐g=a, g=a
$\rightarrow_{ln(g)}$	⇐**a=a**, g=a
$\rightarrow_t$	⇐**g=a**
$\rightarrow_{ln(g)}$	⇐**a=a**
$\rightarrow_t$	□

Figure 2

We can now show that the transformations are complete.

Theorem 9: (Strong Completeness of TRANS)
For every correct answer substitution θ for EP and ⇐F there exists an R^+-computed answer substitution σ obtained by a refutation of EP∪EP^{-1}∪{⇐F} wrt TRANS such that σ is more general than θ.

Proof: If θ is a correct answer substitution for EP and ⇐F, then we find a computed answer substitution γ obtained by a refutation of EP∪EP^{-1}∪{⇐F} wrt $\{\rightarrow_{ip},\rightarrow_r\}$ such that γ is more general then θ (theorem 1). The result follows immediately by an application of theorem 8. qed

6 REFINING THE TRANSFORMATIONS

The proof of the completeness of the transformation rules suggest that the refinements of paramodulation can be carried over to refutations wrt TRANS. In theorem 8 we have shown that for each refutation of

EP∪{⇐F} wrt $\{\rightarrow_{ip},\rightarrow_r\}$ (*)

and computed answer substitution θ we can find a refutation of

EP∪{⇐F} wrt TRANS (**)

yielding a more general computed answer substitution. If we take a close look at the proof of this theorem we can make the following observations:

An equational clause P is used in an $\rightarrow_{ln}$, $\rightarrow_{pv}$, or $\rightarrow_{cc}$ step in (**) only if the same clause is used in a paramodulation step in (*). Moreover, in both refutations P is used in the same direction. Hence, by theorem 3 clauses from EP^{-1} are no longer needed if EP is ground confluent.

Corollary 10: *Let EP be a ground confluent equational program. For every computed answer substitution θ for EP and ⇐F there exists an R^+-computed answer substitution σ obtained by a refutation of EP∪{⇐F} wrt TRANS such that σ is more general than θ.*

If paramodulation is applied upon a variable in (**), then the computed answer substitution is not in normal form. As an example consider the equational clause $f(x)\rightarrow b\Leftarrow$ and the equation y=b. Then

$$\Leftarrow y=b \rightarrow_{pv(\{y\leftarrow f(z)\})} \Leftarrow z=x, b=b \rightarrow_t \Leftarrow z=x \rightarrow_{v(\{z\leftarrow y\})} \square$$

with computed answer substitution {y←f(x)}. Furthermore, a collapse clause is applied in (**) only if the same collapse clause is applied in (*). Hence, if EP is collapse free the rule $\rightarrow_{cc}$ is no longer needed.

Now, since narrowing and reflection is complete for collapse free and ground confluent term rewriting systems as long as we consider only normalized answer substitutions (theorem 4), we conclude that in this case $\rightarrow_{pv}$ and $\rightarrow_{cc}$ are unnecessary and that we have not to restrict our computation rule:

Corollary 11: *Let R be a computation rule and EP be a collapse free and ground confluent term rewriting system. For every normalized correct answer substitution* θ *for EP and* $\Leftarrow F$ *there exists an R-computed answer substitution* σ *obtained by a refutation of* $R\cup\{\Leftarrow F\}$ *wrt* $\{\rightarrow_t,\rightarrow_v,\rightarrow_d, \rightarrow_{ln},\rightarrow_{im}\}$ *such that* σ *is more general than* θ.

Finally, if the equational program is a canonical term rewriting system, then the rules removal of trivial equations, decomposition of decomposible equations, variable elimination applied to equations of the form x=t, where no defined function symbol occurs in t, and rewriting can be applied as simplification rules to refutations wrt narrowing and reflection. Furthermore, there does not exist a refutation of EP∪{⇐F} wrt TRANS if F contains an equation of the form $c(s_1,\ldots,s_n)=d(t_1,\ldots,t_m)$, where c and d are different constructors. Such a goal clause is often called a **failure**.

In analogy to [17] we define a function **simplify** which applies the above mentioned simplification rules upon a goal clause as long as possible and tests that it is not a failure. An **s-derivation** is a derivation where each goal clause is simplified.

Theorem 12: *Let R be a computation rule, EP be a canonical term rewriting system, and* θ *be a normalized correct answer substitution for EP and* $\Leftarrow F$. *Then there exists an R-computed answer substitution obtained by an s-refutation of* $EP\cup\{\Leftarrow F\}$ *wrt* $\{\rightarrow_d,\rightarrow_v,\rightarrow_{ln},\rightarrow_{im}\}$ *such that* σ *is more general than* θ.

Proof: The proof is in analogy to the proof of theorems 8 and 9 except that the first component #p of the complexity used in the proof of theorem 8 must be the maximum number of applications of rewriting steps in the refutation of $EP\cup\{\Leftarrow\theta F\}$ wrt rewriting and removal of trivial equations. qed

The following example shows that the imitation rule is needed to ensure theorem 12. Let {f(x)→a⇐} be the canonical term rewriting system. Then,

⇐y=c(f(y))	$\rightarrow_{im}$({y←c(z)})	⇐z=f(c(z))
	$\rightarrow_{ln}$	⇐z=a,c(z)=x
	$\rightarrow_{v}$({z←a})	⇐c(a)=x
	$\rightarrow_{v}$({x←c(a)})	□

with computed answer substitution {y←c(a)}. It should be noted that imitation is the only inference rule which is applicable to y=c(f(y)).

7 DISCUSSION

We have generalized results obtained by Gallier & Snyder [11,12] and Martelli et al. [31] to hold for arbitrary equational programs (resp. conditional term rewriting systems). Moreover, we have refined their results: To ensure the completeness of their sets of transformations for canonical term rewriting system, Gallier & Snyder as well as Martelli et al. have modified the lazy narrowing rule to be applicable also to arbitrary proper subterms of an equation. This does not only violate the demand driven nature of the transformation rules but also expands the search space since, in general, there are several subterms of an equation whereupon their lazy narrowing rule can be applied. We ensure the completeness by repeated applications of the imitation rule as shown in the last example of section 6.

Gallier & Snyder [11,12] have pointed out that successive applications of the imitation rule upon an equation of the form x=t, where x occurs in t, will generate an instance of the equation and, thus, lead to a cycle. However, they have also shown that in case of unconditional equational theories these cycles can be avoided. We believe that this result holds also for Horn equaliy theories

It should be observed that, if variable elimination can be applied as a simplification rule, the transformation rules can be refined considerably: imitation and paramodulation upon variables need only to be applied upon x=t if x occurs in t. Similarly, lazy narrowing and application of a collapse clause need not to be applied upon x=t if x does not occur in t. Though many researches have suggested to use variable elimination as a simplification rule [11,16,31], none of them has been able to give a rigorous proof for it. Only recently Hsiang & Jouannaud [19] have announced such a proof for unconditional theories.

The transformations rules presented herein can be used as a computational method for equational logic programs as proposed by Jaffar et al. [23,24] or Goguen & Messequer [14] by adding a lazy resolution rule as suggested in [17]. This rule applied to a selected atom of the form $P(s_1,\ldots,s_n)$ and a program clause of the form $P(t_1,\ldots,t_n)\Leftarrow D^*$ forces the comparison of corresponding arguments, i.e.

$$\Leftarrow D\cup\{P(s_1,\ldots,s_n)\} \rightarrow \Leftarrow D\cup D^*\cup\{s_i=t_i|1\leq i\leq n\},$$

where D and D* are sets of atoms and equations.

There are, of course, other proposals to handle equational theories. We have already mentioned paramodulation and special forms of it like narrowing (e.g. [13,21,22,25,36,37]) or superposition [7,8]. Recently, Echahed [6] has shown that narrowing is independent of a computation rule which selects also a certain redex provided that the term rewriting system is completely defined and strictly non-subunifiable. It seems that the use of transformation rules cuts down the search space since there are less alternatives, the application is demand driven, and failures can be recognized earlier.

Another proposal is based on the idea to flatten goal and program clauses and then to apply SLD-resolution (e.g. [1,3,4]). The disadvantage of this technique is that rewriting

cannot be applied as a simplification rule anymore — in can only be simulated by a sequence of SLD-resolution steps using a complex computation rule. However, rewriting goal clauses may cut down the search space from an infinite to a finite one. Recently, Nutt et al. [33] have shown that narrowing and flattening can be combined in one system leaving it to the overall strategy whether goal clauses should be flattened or narrowing should be applied.

Unfortunately, there has been no thorough comparison between the various techniques so far. We only know for sure that each of them is superior to the others in certain aspects or for certain classes of equational theories.

REFERENCES

1 R. Barbuti, M. Bellia, G. Levi, M. Martelli: LEAF: A Language which Integrates Logic, Equations and Functions. In: Logic Programming (DeGroot, Lindstrom eds.), Prentice Hall: 1986

2 M. Bellia, G. Levi: The Relation Between Logic and Functional Languages: A Survey. J. Logic Programming, 217-236: 1986

3 P. G. Bosco, E. Giovanetti, C. Moiso: Refined Strategies for Semantic Unification. LNCS 250, 276-290: 1987

4 P. T. Cox, T. Pietrzykowski: Surface Deduction: A Uniform Mechanism for Logic Programming. Proc. SLP, 220-227: 1985

5 N. Dershowitz, Z. Manna: Proving Termination with Multiset Orderings. CACM, 465-475: 1979

6 R. Echahed: On Completeness of Narrowing Strategies. Proc. CAAP: 1988

7 L. Fribourg: Oriented Equational Clauses as a Programming Language. J. Logic Programming, 165-177: 1984

8 L. Fribourg: SLOG: A Logic Programming Language Interpreter Based on Clausal Superposition and Rewriting. Proc. SLP, 172-185: 1985

9 U. Furbach, S. Hölldobler, J. Schreiber: Horn Equality Theories and Paramodulation. To appear in: J. Automated Reasoning: 1988

10 J. H. Gallier, S. Raatz: SLD-Resolution Methods for Horn Clauses with Equality Based on E-Unification. Proc. SLP, 168-179: 1986

11 J. H. Gallier, W. Snyder: A General Complete E-Unification Procedure. LNCS 256: 1987

12 J. H. Gallier, W. Snyder: Complete Sets of Transformations for General E-Unification. Univ. of Pennsylvania, Philadelphia: 1988

13 E. Giovannetti, C. Moiso: A Completeness Result for E-Unification Algorithms based on Conditional Narrowing. Proc. Workshop on Foundations of Logic and Functional Programming: 1987

14 J. A. Goguen, J. Meseguer: EQLOG: Equality, Types, and Generic Modules for Logic Programming. In: Logic Programming (DeGroot, Lindstrom eds.), 295-363: 1986

15 J. Herbrand: Sur la Théorie de la Démonstration. Logical Writings (Goldfarb ed.), Cambridge: 1971

16 S. Hölldobler: A Unification Algorithm for Confluent Theories. LNCS 267, 31-41: 1987

17 S. Hölldobler: Equational Logic Programming. Proc. SLP, 335-346: 1987

18 S. Hölldobler: From Paramodulation to Narrowing. Proc. ICLP/SLP, 327-342: 1988

19 J. Hsiang, J. P. Jouannaud: General E-Unification Revisited. 2nd. Int. Workshop on Unification, Val d'Ajol, France: 1988

20 G. Huet, D. C. Oppen: Equations and Rewrite Rules. In: Formal Languages: Perspectives and Open Problems (Book ed.), Academic Press, 1980

21 J. M. Hullot: Canonical Forms and Unification. Proc. CADE, 318-334: 1980

22 H. Hussmann: Unification in Conditional-Equational Theories. LNCS 204, 543-553: 1985

23 J. Jaffar, J-L. Lassez, M. J. Maher: A Theory of Complete Logic Programs with Equality. Proc. FGCS, 175-184: 1984

24 J. Jaffar, J-L. Lassez, M. J. Maher: A Logic Programming Language Scheme. In: Logic Programming (DeGroot, Lindstrom eds.), Prentice Hall, 1986

25 S. Kaplan: Fair Conditional Term Rewriting Systems: Unification, Termination, and Confluence. Recent Trends in Data Type Specification (Kreowski ed.), IFB 116, 136-155: 1986

26 C. Kirchner: A New Equational Unification Method: A Generalization of Martelli-Montanari's Algorithm. CADE 7: 1984

27 D. S. Lankford: Canonical Inference. Techn. Rep., Dept. of Mathematics, Southwestern Univ., Georgetown, Texas: 1975

28 J. W. Lloyd: Foundations of Logic Programming. Springer: 1984

29 B. Mahr, J. A. Makowsky: Characterizing Specification Languages which Admit Initial Semantics. LNCS 159, 300-316: 1983

30 A. Martelli, U. Montanari: An Efficient Unification Algorithm. ACM TOPLAS, 258-282: 1982

31 A. Martelli, C. Moiso, C. F. Rossi: An Algorithm for Unification in Equational Theories. Proc. SLP, 180-186: 1986

32 D. Miller, G. Nadathur: A Logic Programming Approach to Manipulating Formulas and Programs. Proc. SLP: 1987

33 W. Nutt, P. Réty, G. Smolka: Basic Narrowing Revisited. SEKI Report SR-87-07, Univ. Kaiserslautern: 1987

34 P. Padawitz: Foundations of Specification and Programming with Horn Clauses. Univ. Passau: 1987

35 G. D. Plotkin: Building-In Equational Theories. In: Machine Intelligence 7 (Meltzer, Michie eds.), 73-90: 1972

36 U. S. Reddy: Narrowing as the Operational Semantics of Functional Languages. Proc. SLP, 138-151: 1985

37 P. Réty, C. Kirchner, H. Kirchner, P. Lescanne: NARROWER: A New Algorithm for Unification and its Application to Logic Programming. LNCS 202, 141-155: 1985

38 G. A. Robinson, L. Wos: Paramodulation and Theorem Proving in First Order Theories with Equality. Machine Intelligence 4 (Meltzer, Mitchie, eds.): 1969

39 J. A. Robinson, E. E. Sibert: LOGLISP: An Alternative to PROLOG. Machine Intelligence 10 (Hayes, Mitchie, eds), 399-419: 1982

40 M. Sato, T. Sakurai: QUTE: A PROLOG/LISP Type Language for Logic Programming. Proc. 8th IJCAI, 507-513: 1983

41 E. E. Sibert: A Machine-Oriented Logic Incorporating the Equality Relation. Machine Intelligence 4 (Meltzer, Mitchie, eds.): 1969

42 J. R. Slagle: ATP with Built-In Theories Including Equality, Partial Ordering and Sets. J.ACM, 622-642: 1974