Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany

Artificial
Intelligence
Laboratories

SEKI-REPORT



# Topics in Completion Theorem Proving

Jürgen Müller & Rolf Socher-Ambrosius

SEKI-Report SR-88-13

# Topics in Completion Theorem Proving

*Jürgen Müller*
*Rolf Socher-Ambrosius*
*Fachbereich Informatik, Universität Kaiserslautern*
*Postfach 3049, 6750 Kaiserslautern*

**Abstract:** Completion Theorem Proving is based on the observation that proving a formula is equivalent to solving a system of equations over a boolean polynomial ring. The latter can be accomplished by means of the completion of a set of rewrite rules obtained from the equational system. This report deals with three problems concerning Completion Theorem Provers (CTPs):
- Are multiple overlaps necessary for the completeness of CTPs?
- How can simplification be interpreted in terms of resolution inference rules ?
- How can polynomials efficiently be represented?

The answer to the first question is "no". Even more it is shown in this report that the removal of multiple overlap steps does not increase the length of completion refutations. This amounts to an extension of Dietrich's (1986) result concerning the correspondence of completion proofs without simplification for Horn Logic to resolution proofs.
Concerning the second question we show that simplification in general cannot be translated into resolution reduction rules. However, two special cases of reduction can be interpreted as subsumption deletion and replacement resolution, respectively. Changing the point of view, we can also identify resolution reduction rules, such as tautology elimination, merging and subsumption deletion, as part of CTPs without being explicitly defined as inference rules.
The last section deals with the technical question of choosing a datastructure well suited for an efficient implementation of CTPs.

## 1. Introduction

The basic idea of completion theorem proving, as it was first proposed by Hsiang (1982), is the equivalence of the validity of a first order formula and the solvability of a system of equations over a boolean polynomial ring. Several methods to solve such systems have been developed, the one used in connection with completion theorem proving is the completion procedure of Knuth & Bendix (1970).
The completion method proceeds in the following way: Given a first order formula $\mathcal{F}$ which is to be proved valid, transform $\mathcal{F}$ into a system $E_{\mathcal{F}}=\{p_1=0,...,p_n=0\}$ of equations over first order polynomials. The polynomials represent the formula $\mathcal{F}$ in terms of the connectives "XOR" and

the logical "AND", which are the + and the * of the boolean ring, respectively. Hsiang (1985) presents a canonical rewrite system, which accomplishes the transformation of a first order formula into a set of polynomials. The fomula $\mathcal{F}$ is valid, iff the system $E_{\mathcal{F}}$ is unsolvable. In order to prove the unsolvability of such an equational system, it is subjected to the Knuth-Bendix completion procedure. If the equality 1=0 can be derived from the system $E_{\mathcal{F}}$ then $E_{\mathcal{F}}$ is unsolvable.

The completion process is founded on two mechanisms: The formation of critical pairs to generate new rules and the reduction (simplification) of existing rules. In some sense the respective analogues in resolution theorem proving are the resolution rule and the well-known reduction rules like subsumption deletion.

In analogy to resolution theorem proving, the greatest obstacle on the way to find a completion refutation is the extent of the search space for generating critical pairs. Several strategies to restrict this search space have been proposed and the various completion provers (for instance the systems of Hsiang (1985), Kapur & Narendran (1985), Bachmair & Dershowitz (1987) and Müller (1987) (1988)) differ mainly in the choice of the strategy. In this report we consider mainly the N-strategy as it is described in Hsiang (1985). This strategy requires one partner of a superposition step to be an N-rule (i.e. a rule $m \rightarrow 0$, where m is a product of atoms), which cuts down the number of possible superpositions.

The objective of the first part of this paper is to show that this number can be further decreased significantly without any prolongation of the whole refutation. The following example illustrates this reduction of the search space:

1.1 Example:

Let $r_1$ = Pxa*Qby*Rxy $\rightarrow$0 and

$r_2$ = Paz*Qbz*Rab + Paz*Qbz + Paz*Rab + Paz $\rightarrow$ 0.

If we let $m_1,..,m_4$ denote the monomials of $r_2$ and m the monomial of $r_1$, we have to compute all possible overlaps between m and the $m_i$. For instance, the monomials Pxa*Qby and Paz*Qbz are unifiable with unifier $\theta$ = {x$\leftarrow$a, z$\leftarrow$a, y$\leftarrow$a}. This yields an overlap Paa*Qba*Raa*Rab between m and $m_1$ resulting in the critical pair $\langle$ Paa*Qba*Raa + Paa*Rab*Raa + Paa*Raa, 0 $\rangle$. We call this overlap a multiple overlap, since it was accomplished by the unification of monomials that are not single atoms. In contrast to the multiple overlap, the overlap Paa*Qby*Ray between m and $m_4$ for instance is a single overlap, resulting from unification of Pxa and Paz.

We will show that it is sufficient to take into account only the single overlaps, provided the completion refutation follows the N-strategy. This yields a drastical reduction in the number of possible critical pairs and we will show that the removed critical pairs represent "blind alleys" of the whole refutation. This shows that the restriction to single overlaps improves the search for a proof by cutting down a lot of redundant steps.

This result implies a generalization of Dietrich's (1986) result on Hsiang's system: He showed

that the superposition step of two *Horn clause* rules can be translated into a resolution step between C and D. We show that a *single overlap* superposition of two *arbitrary* clause rules corresponds to a resolution step of the two corresponding clauses. Together with our result on the unnecessity of multiple overlaps this yields a translation from N-refutations without reductions into resolution refutations.

This result holds for systems without simplification. Being the most important inference rule for any Knuth-Bendix like procedure, simplification also plays a significant part in CTPs. In the first place, simplifying rules with other rules yields a kind of minimal representation of the underlying problem. Second, rewriting alone can be sufficient to solve problems (see example 4.8) This arises the question, whether simplification can be interpreted in terms of resolution reduction rules. It will turn out that in general such a translation does not exist. However, we will show that a clausal rule, which can be removed by a simplification step, corresponds to a subsumed clause; that is, simplification steps removing rules correspond to the deletion of subsumed clauses. We also obtain the following result: If the result of reducing a clausal rule is again a clausal rule, then this reduction corresponds to a special resolution step, called "replacement resolution" by Markgraf (1984).

Changing the point of view, in section 5 we consider the translation of inference and reduction rules from resolution based theorem provers into the CTP paradigm. This extends the work of Kapur and Narendran (1985) who proved, that resolution and factoring can be simulated by the completion method, provided that no simplification takes place. We show that tautology elimination, merging, and subsumption deletion are automatically performed within CTPs.

The last section is reserved for a technical contribution. One problem with CTPs is the representation of the - usually very large - polynomials. So arised the question for a datastructure representing efficently polynomials and minimizing the time complexity of the search for possible inference and reduction steps. The solution presented here consists of a structure sharing approach and a graph concept similar to the one used by Kowalski (1975) in his Connection Graph Procedure.

Before starting a detailed discussion, we will give a short introduction to the completion method.

## 2. The Completion Method

This section gives a short overview on the completion procedure as it is proposed by Hsiang. We present a slightly modified version: First, superpositions are made "parallel" on one polynomial, which abbreviates a sequence of inference steps. Second, as superposition alone does not guarantee completeness, factoring is considered as an additional inference rule (cf. Müller (1988), see also appendix).

The structure $(\mathbb{A}, \wedge, \vee, \neg, 1, 0)$, where $\mathbb{A}$ is the set of first order atoms and $\wedge$, $\vee$, $\neg$ are the logical connectives is a boolean algebra. Then the structure $(\mathbb{A},+,*,0,1)$ defined by

$$x + y = (x \wedge \neg y) \vee (\neg x \wedge y) \qquad x*y = x \wedge y$$

is a boolean ring. Let $\mathbb{T}_{BR}$ be the set of all terms over $(\mathbb{A},+,*,0,1)$. Boolean rings have the property that each $t \in \mathbb{T}_{BR}$ can be written as a sum of products of different atoms. This form is unique up to commutativity of + and *. We call this form the normal form of t, $nf_{BR}(t)$.
The mapping $\varphi$: $(A,\wedge,\vee,\neg,0,1) \rightarrow (A,+,*,0,1)$ transforming terms of a boolean algebra into terms of a boolean ring, is defined by

$$a\varphi = a$$
$$(s \wedge t)\varphi = s\varphi\, t\varphi$$
$$(s \vee t)\varphi = s\varphi + t\varphi + s\varphi\, t\varphi$$
$$(\neg s)\varphi = 1 + s\varphi$$

for $a \in \mathbb{A}$ and boolean algebra terms s and t.
A predicate logic formula $\mathcal{F}$ in clausal form, which is essentially a boolean algebra term, is represented by the equality $nf_{BR}(\mathcal{F}\varphi) = 1$ or equivalently $nf_{BR}(\neg\mathcal{F}\varphi) = 0$ . In order to prove the unsolvability of a set of boolean equalities, each equality of the form t=0 is transformed into a rewrite rule $t \rightarrow 0$. Then a special Knuth-Bendix completion procedure is applied to the resulting set of rules. The procedure is special in the sense that not all possible critical pairs are generated, only overlaps where one partner is a so called N-term are considered.

2.1 Definition:
A boolean ring term t is called an **N-term** (sometimes also called a **monomial**), if it is a product of atoms. A rewrite rule $t \rightarrow 0$ where t is a boolean term in normal form is called a **boolean rule**. A boolean rule $t \rightarrow 0$ is called an **N-rule**, if t is an N-term.

2.2 Definition: (Reduction relation)
Let $t,t' \in \mathbb{T}_{BR}$ and let $\mathfrak{R}$ be a set of rewrite rules. We say t **reduces** to t' with respect to $\mathfrak{R}$ and $\mu$, iff there is a rewrite rule $l \rightarrow r$ in $\mathfrak{R}$ and a substitution $\mu$ such that $t=t[l\mu]$ and $t'=nf_{BR}(t[r\mu])$. This is denoted by $t \mapsto_{\mathfrak{R},\mu} t'$. The reflexive, transitive closure of $\mapsto$ is denoted by $\mapsto^*$.

2.3 Definition:
Let $m_1$ and $m_2$ be N-terms.

(i) $m_1$ and $m_2$ are **BN-unifiable** under $\sigma$ iff $\sigma$ is a most general unifier (see e.g. Herold 1983) for $m_1$ and $m_2$ under the boolean ring theory of *.

(ii) Let $m_1 = u_1 s_1$ and $m_2 = u_2 s_2$. Then $(s_1 m_2)\sigma$ is an **overlap** of $m_1$ and $m_2$ iff $u_1$ and $u_2$ are BN-unifiable under $\sigma$. We say that the overlap is on the pair $(u_1,u_2)$ in this case. If $\sigma$ is the identity, then $u_1=u_2$ and we say the overlap is on $u_1$. (This is of interest in connection with propositional logic.) The overlap is called a **multiple** overlap, iff the monomial $u_1$ (and hence also $u_2$) is a product of more than one atom. Otherwise it is called a **single** overlap.

(iii) Let $s \rightarrow 0$ be an N-rule, $s=s_1s_2$, and $m_1+m_2+...+m_n \rightarrow 0$ a boolean rule. $\langle (s_2m_2+...+s_2m_n)\sigma,0\rangle$ is called an **N-critical pair** of the two rules iff s has an overlap $(s_2m_1)\sigma$ with $m_1$. It is called **divergent**, if $nf_{BR}((s_2m_2+...+s_2m_n)\sigma) \neq 0$.

Note that we may, without loss of generality, always choose the first element of the sum $m_1+m_2+...+m_n$ for overlapping with s, due to the commutativity of "+".
Now we are ready to define the basic inference rule of the completion algorithm.

2.4 Definition:
Let $s \rightarrow 0$ be an N-rule and $r \rightarrow 0$ a boolean rule. We say that the rule $t \rightarrow 0$ results from **superposition** of the two rules iff these two rules determine a divergent N-critical pair $\langle t',0\rangle$, and $t = nf_{BR}(t')$.

The above proof strategy is called N-strategy by Hsiang. It is shown to be sound and complete (see remarks in the appendix):

2.5 Theorem (Hsiang):

*Given a set of clauses S in first-order predicate calculus, S is inconsistent if and only if $1 \rightarrow 0$ can be produced using the N-strategy.* ■

The transformation of first order formulae into boolean rules can be done in two different ways: According to the first method the formulae first are transformed in clausal normal form in the same way as it is done for classical resolution proofs. Then all clauses separately are transformed into rewrite rules. This method is called the clausal strategy. According to the other method, the nonclausal strategy, the formulae are directly transformed into boolean rules. The N-strategy, however, cannot be applied with the nonclausal strategy, because the existence of N-terms is only guaranteed, if the set of boolean rules corresponds to an unsatisfiable set of clauses. For this reason we only consider the clausal strategy.
As Hsiang's original procedure is very rough, we adopt two ideas from the THEOPOGLES system (cf. Müller (1987) and (1988)), namely the ideas of "parallel superpositions" and the incorporation of "factorization".
Suppose that we have two rules $pt_1+...+pt_n+r_1+...+r_m \rightarrow 0$ and $qs \rightarrow 0$ where the monomial p does not occur in the $t_i$ and $r_j$ and $\sigma$ is a unifier of p and q. Then $\langle (pt_2s+...+pt_ns+r_1s+...+r_ms)\sigma, 0\rangle$ is an N-critical pair of the two rules. But the left hand term can be reduced to $(r_1s+...+r_ms)\sigma$ with the rule $qs \rightarrow 0$. Since Hsiang's strategy requires critical pairs to be reduced with respect to the rules of boolean rings and the current rewrite system, we can compute the rule resulting from superposition also in the following "parallel" way (cf. Müller 1987)): transform the rule $pt_1+...+pt_n+r_1+...+r_m \rightarrow 0$ into $p(t_1+...+t_n)+r_1+...+r_m \rightarrow 0$ and then compute the critical pair. Especially overlaps between a rule $t \rightarrow 0$ and an N-rule $m \rightarrow 0$ on a monomial $\mu$ that divides t (i.e. that occurs in each monomial of t) do not yield divergent

critical pairs, since this is the case where $r_1+...+r_m=0$.
Furthermore we introduce the notion of I-critical pairs, which can be seen as a kind of explicit "factoring" of monomials.

2.6 Definition:
Let r be the N-rule $m\rightarrow 0$ with $m=pqm'$, where p and q are unifiable atoms. If $\theta\in mgu(p,q)$, then $\langle(pm')\theta,0\rangle$ is called an **I-critical pair** of r.

We just resume the sligthly modified N-strategy the next section is based on:
The formula $\mathcal{F}$ which is to be proved valid first is transformed into a set $\mathfrak{R}$ of rewrite rules. Then (N- and I-)critical pairs of rules of $\mathfrak{R}$ are formed with parallel superposition and the rules resulting from divergent critical pairs are added to $\mathfrak{R}$ until the rule $1\rightarrow 0$ is deduced. There are no reduction steps w.r.t. the current rewrite rule system $\mathfrak{R}$.

## 3. Removing Multiple Overlaps From Completion Proofs

Clauses can be transformed into boolean rules in the following way:
If C is a clause of the form $\neg p_1\vee...\vee\neg p_n \vee q_1\vee...\vee q_m$ then $\neg C = p_1 \wedge...\wedge p_n\wedge\neg q_1\wedge...\wedge q_m$.
Therefore C can be transformed (using the mapping $\varphi$) into the boolean rule $p_1...p_n(1+q_1)...(1+q_m)\rightarrow 0$. We say that the rule $r\rightarrow 0$ is a **clause rule**, if it can be represented in this way. Especially N-rules are clause rules. Furthermore we say that the $p_i$ occur **positively** in $p_1...p_n(1+q_1)...(1+q_m)$, and the $q_j$ occur **negatively** in $p_1...p_n(1+q_1)...(1+q_m)$.

3.1 Definition:
Let t be a term and let M be a set of monomials.

(i) We write $t \Vdash_M s$ iff there is some $m\in M$, such that $s\rightarrow 0$ is the result of a superposition of $t\rightarrow 0$ and $m\rightarrow 0$.

(ii) We write $t \vdash_M s$ iff $t \Vdash_M s$ and the superposition results from a single overlap.

(iii) Let $P \subseteq \mathbb{A}$ and let $m=p_1...p_n$ be a monomial such that $p_i\in P$ for each $i\in\{1,..,n\}$. Then we call m a **monomial in P**. We call m **P-free**, iff m is a monomial in $\mathbb{A}\backslash P$. The P-part (P-free part) of a monomial m is the maximal monomial in P (P-free monomial) dividing m.

If $M=\{m\}$, we write $t \vdash_m s$ and analogously for the other relations.

The proof of our main result will be accomplished in a way analogous to completeness proofs: First it is proved for propositional logic and then the lifting lemma is used to transfer the result to full first order logic. Therefore in the following (lemmata 3.2 - 3.6) all terms are ground terms.

3.2 Lemma:

*Let $t \to 0$ be a boolean rule and $m \to 0$ be an $\mathcal{N}$-rule and let the rule $s \to 0$ result from superposition of $t \to 0$ and $m \to 0$. Then each divisor of t is a divisor of s.*

Proof:

Let $nf_{BR}(t) = m_1t_1+..+m_1t_n+r_1+..+r_k$ and $m=m_1m_2$, such that $s=r_1m_2+..+ r_km_2$. A divisor of t divides all $r_i$, hence also s.

■

3.3 Lemma:

*Let $t_0= (1+p_1)...(1+p_r)$. Assume there is a set $G=\{g_1 \to 0,..., g_n \to 0\}$ of $\mathcal{N}$-rules, such that*

$$t_0 \Vdash_{g_1} t_1 \Vdash_{g_2} \ldots \Vdash_{g_n} t_n = 1$$

*(i) Each $g_i$ is a monomial in $\{p_1,..,p_r\}$*

*(ii) Each superposition $t_i \to 0$, $g_i \to 0$ is made on $g_i$.*

*(iii) If m is a monomial of t, then all divisors of m are monomials of t, too.*

Proof:

(i) Let k be the smallest number, such that $g_k$ is not in $\{p_1,..,p_r\}$. Then there is some atom $q \neq 1$ with $q \notin \{p_1,..,p_r\}$ such that $g_k=qg'$. We have $t_{k-1} \Vdash_{gk} t_k$. Since q does not occur in $t_0$ nor in any of the $g_1,..,g_{k-1}$, it cannot occur in $t_{k-1}$, too. Hence the overlap of $t_{k-1}$ and $g_k$ must be on a divisor g'' of g'. Let $g_k=g''\gamma$, which implies that q is a divisor of $\gamma$ and let $t_{k-1}= g''t_1 + t_2$, where g'' does not occur in $t_1$ and $t_2$. The resulting critical pair is $\langle \gamma t_2,0\rangle$ and $t_k=\gamma t_2$. Now q divides $\gamma$ and hence also $t_k$. Then according to 3.2 q must divide all $t_i$ with $i \geq k$, and especially $t_n$. But $t_n=1$, which implies $q=1$, a contradiction.

(ii) Suppose the superposition of $t_{k-1} \to 0$, $g_k \to 0$ is made on a divisor $\gamma$ of $g_k$. Let $g_k=\gamma g'$. Then it is easy to see that g' divides $t_k$ and with the same argument as in (i) we obtain $g'=1$.

(iii) Suppose $\mu$ is a divisor of m and $\mu$ is no monomial of t. Then $\mu$ must have been deleted from $t_0$ in a superposition step. Since according to (ii) superposition steps have the effect of removing a monomial together with all its multiples, m must have been deleted, too, which is a contradiction.

In the special situation of lemma 3.3 each superposition step $t_{i-1} \Vdash_{g_i} t_i$ is a step of removing in $t_{i-1}$ those monomials that have $g_i$ as a divisor. As a consequence, a deduction as in 3.3 must contain all superposition steps with the minimal divisors $p_i$ and these steps "subsume" all other superposition steps.

3.4 Lemma:

*Let $t_0 = (1+p_1)...(1+p_r)$. Assume there is a set $G=\{g_1 \to 0,..., g_n \to 0\}$ of $\mathcal{N}$-rules, such that*

$$t_0 \Vdash_{g_1} t_1 \Vdash_{g_2} \dots \Vdash_{g_n} t_n = 1$$

*then there is a subset $H=\{h_1 \to 0,...,h_n \to 0\}$ of $G$ such that*

$$t_0 \vdash_{h_1} t_1' \vdash_{h_2} \dots \vdash_{h_\nu} t_\nu' = 1$$

Proof:

The (boolean ring) normal form of $t_0$ is $t_0 = (1+ \sum_i p_i + \sum_{i<j} p_i p_j + ...+ \prod_i p_i )$.

Let $k \in \{1,..,n\}$ and suppose that $g_k$ is a product of more than one atom.

Then $t_{k+1}$ is the result of deleting in $t_k$ all monomials $\mu$ that have $g_k$ as a divisor. Let q be a proper divisor of $g_k$. Then, according to lemma 3.3(iii), q is a monomial of t. Since q cannot be deleted in the step $t_k \vdash_{gk} t_{k+1}$, it is a monomial of $t_{k+1}$, too.

In order to reduce $t_{k+1}$ to 1, there must be a subsequent superposition step deleting q. Obviously, this superposition results from a single overlap. But then the step $t_k \vdash_{gk} t_{k+1}$ can be canceled from the whole deduction, since the subsequent step, which deletes q, in fact removes all monomials that have q as a divisor and especially those monomials having $g_k$ as a divisor, i.e. those monomials that are removed by $t_k \vdash_{gk} t_{k+1}$. It remains a deduction with the desired properties.

■

3.5 Lemma:

*Let t be a ground term and suppose $t = m_1 f$, where f is a term in $\mathcal{P}$ and $m_1$ is $\mathcal{P}$-free monomial. Furthermore let m be a monomial with $\mathcal{P}$-free part $\mu$. Let $\langle s,0 \rangle$ be a nontrivial critical pair of the two rules $t \to 0$ and $m \to 0$.*

*Then the $\mathcal{P}$-free part of s is $m_1\mu$.*

Proof:

Let $t=m_1f_1+...+m_1f_n$ and $m=k_1k_2$ and let the overlap be on $k_1$. From 3.2 follows that $m_1$ divides s.

Let q be an atom dividing $\mu$, which implies that q is P-free. If q divides $k_1$, then q must divide a monomial of t, say $m_1f_1$. Since $f_1$ is a monomial in P, we obtain that q divides $m_1$ and hence q divides s. If q divides $k_2$, then q divides s, since $k_2$ is a divisor of s. Hence $m_1\mu$ is P-free and divides s. It is easy to see, that each P-free divisor of s either divides $m_1$ or $\mu$.

■

3.6 Lemma:

*Let $t_0$ be a clause term. If there is a set $G=\{g_1 \to 0,\ldots, g_n \to 0\}$ of $\mathcal{N}$-rules, such that*

$$t_0 \Vdash_{g_1} t_1 \Vdash_{g_2} \ldots \Vdash_{g_n} t_n$$

*and $t_n$ is a monomial, then there are $t_1',..,t_n'$ and a subset $H=\{h_1 \to 0,\ldots, h_n \to 0\}$ of $G$ such that*

$$t_0 \vdash_{h_1} t_1' \vdash_{h_2} \ldots \vdash_{h_v} t_v'$$

*and $t_n'$ is a monomial subsuming $t_n$.*

Proof: We have $t_0 = q_1...q_u(1+p_1)...(1+p_r)$. Let $P:=\{p_1,..,p_r\}$.
First we transform the deduction

$$t_0 \Vdash_{g_1} t_1 \Vdash_{g_2} \ldots \Vdash_{g_n} t_n$$

into a deduction

$$s_0 = (1+p_1) \ldots (1+p_r) \Vdash_{\gamma_1} s_1 \Vdash_{\gamma_2} \ldots \Vdash_{\gamma_n} s_n = 1$$

Each $t_k$ can be written in the form $\mu_k f_k$ with a P-free monomial $\mu_k$ and a polynomial $f_k$ in P. In the same way we can decompose $g_k$ in a p-part $\pi_k$ and a rest $g_k'$, such that $g_k = \pi_k g_k'$. According to lemma 3.5 we have $\mu_{k+1} = \mu_k g_{k+1}'$ for each $k \in \{1,..,n-1\}$. Furthermore $\mu_0 = q_1...q_u$. Hence $\mu_n = q_1...q_u g_1'...g_n'$.
Now let $s_k := f_k$ and $\gamma_k = \pi_k$. Hence we have the following situation:

$$\begin{array}{ccc} t_k = \mu_k f_k & \xrightarrow{\pi_k g_k'} & t_{k+1} = \mu_{k+1} f_{k+1} \\ \downarrow & & \downarrow \\ s_k = f_k & \xrightarrow{\pi_k} & s_{k+1} = f_{k+1} \end{array}$$

It is obvious, that $s_0 = (1+p_1)...(1+p_r)$. We only have to show, that $s_n = 1$ holds: The construction above shows, that $s_k$ is a term of the form $1+\sum_i m_i$, where each $m_i$ is a monomial in $\{p_1,.., p_1\}$. Furthermore $s_k \mu_k = t_k$. Since $\mu_n$ and $t_n$ are monomials, $s_n=1$ must hold. Now we have a deduction

$$s_0 = (1+p_1) \ldots (1+p_r) \Vdash_{\gamma_1} s_1 \Vdash_{\gamma_2} \ldots \Vdash_{\gamma_n} s_n = 1$$

According to 3.4 there is a subset $\{\eta_1,...,\eta_v\}$ of $\{\gamma_1,..,\gamma_n\}$ such that

$$s_0 = (1+p_1) \ldots (1+p_r) \vdash_{\eta_1} s_1' \vdash_{\eta_2} \ldots \vdash_{\eta_v} s_v' = 1$$

This deduction is transformed back in the following way:
Let $t_0' = q_1...q_u\ (1+p_1)...(1+p_r)$ and if $\gamma_k = \pi_j$, then let $h_k = \pi_j m_j'$. Now it is easy to see, that there must be elements $g_1'',..,g_v''$ of $\{g_1',..,g_n'\}$ such that $t_v' = q_1..q_u g_1''..g_v''$. Now $\mu_n = q_1...q_u g_1'...g_n'$ is a divisor of $t_n$, which implies that $t_v'$ is also a divisor of $t_n$.
■

The following example shows, that the generation of I-critical pairs, i.e. the factorization of N-terms, is an esssential condition for the generalization of lemma 3.6 to full first order logic.

3.7 Example:

Let $\Re$ be the system $\{t\to 0, m_1\to 0,...,m_4\to 0\}$ of rules, where

$t = 1+Pax+Pyb+Qxy+PaxPyb+PaxQxy+PybQxy+ PaxPybQxy$

$m_1$=PaxPyb, $m_2$= PybQxy,

$m_3$=PaxQba, $m_4$= Qxy.

We have the following deduction:

$$t \Vdash_{m_1} t_1 \Vdash_{m_2} t_2 \Vdash_{m_3} t_3 \Vdash_{m_4} t_4 = 1$$

with $t_1 = 1+Pax+Pyb+Qxy+PaxQxy+PybQxy$

$t_2 = 1+Pax+Pyb+Qxy+PaxQxy$

$t_3 = 1+Qba$

The multiple overlap of $t_2 \Vdash_{m3} t_3$ cannot be removed, since the most general unifier $\{x\leftarrow a, y\leftarrow b\}$of this step is also responsible for the factorization of $t_2$.

With factorization of monomials the following deduction with single overlaps is possible:

$m_1 \vdash_{m1} m_1'$ (with an I-critical pair)

$$t \vdash_{m_1'} t_1' \vdash_{m_1'} t_2' \vdash_{m_4} t_4 = 1$$

with $m_1' = Pab$, $t_1' = 1+Pyb+Qxy+PybQxy$, $t_2' = 1+Qxy$

In order to generalize lemma 3.6 to full first order logic, we have to prove the following

3.8 Lemma: (Lifting lemma for N-superpositions)

*Let $x', y'$ be clause terms and let $m'$ be a monomial such that $x' \Vdash_{m'} y'$ holds. Suppose that $x'$ and $m'$ are instances of clause terms $x$ and $m$, respectively. Then there is a clause term $y$ such that $y'$ is an instance of $y$ and $x \Vdash_m y$ holds.*

Proof:

Analogous to the proof of the lifting lemma for resolution, cf. e.g. Chang & Lee (1973).

3.9 Lemma:

*Let $t_0$ be a first order clause term. If there is a set $G=\{g_1\to 0,..., g_n\to 0\}$ of $\mathcal{N}$-rules, such that*

$$t_0 \Vdash_{g_1} t_1 \Vdash_{g_2} \dots \Vdash_{g_n} t_n$$

*and $t_n$ is a monomial, then there are $t_1',...,t_n'$ and a subset $\mathcal{H}=\{h_1\to 0,..., h_n\to 0\}$ of $G$ such that*

$$t_0 \vdash_{h_1} t_1' \vdash_{h_2} \dots \vdash_{h_v} t_v'$$

*and $t_n'$ is a monomial subsuming $t_n$.*

Proof:

The lifting lemma is used to generalize the proposition of lemma 3.6 to arbitrary terms. ■

3.10 Theorem:

*Let $C$ be a set of clause rules and $\mathcal{D}$ be an $\mathcal{N}$-completion refutation, i.e. a completion deduction of the equation 1=0 following the $\mathcal{N}$-strategy. Then there is an $\mathcal{N}$-refutation $\mathcal{D}'$ such that each $\mathcal{D}'$-step is a $\mathcal{D}$-step and all superpositions result from single overlaps.*

Proof:

We show that a N-completion refutation can be decomposed completely into chains that satisfy the assumptions of 3.9.

$\mathcal{D}$ is an N-refutation, hence one partner of each superposition is a monomial. If t is a non N-term occurring in the deduction, then t can only overlap with N-terms. Let

$$t \Vdash t_1 \Vdash \dots \Vdash t_n$$

be the chain of all superpositions starting with t occurring in $\mathcal{D}$. If $t_n \neq 1$, then the whole chain is redundant. Hence assume $t_n = 1$. Let $t_k$ be the first monomial in this chain, which exists, since $t_n$ is a monomial. Then, obviously, the chain $t \Vdash t_1 \Vdash \dots \Vdash t_k$ satisfies the assumptions of lemma 3.9 and we can delete all multiple-atom superpositions from this chain with resulting term $t_k'$ subsuming $t_k$. Since $\mathcal{D}$ is completely composed of these chains, we obtain an N-refutation $\mathcal{D}'$. ■

3.11 Lemma:

*Any rule obtained by superposing two rules belonging to clauses $C$ and $\mathcal{D}$, respectively, by means of a single overlap is a clause rule and the clause belonging to this rule is a resolvent of $C$ and $\mathcal{D}$.*

Proof:

Let $m \to 0$ be an N-rule and $t \to 0$ a clause rule, such that there is a single overlap of m and some monomial of t.

The N-rule is of the form $q_1 \dots q_n \to 0$ corresponding to the clause $C = \neg Q_1 \vee \dots \vee \neg Q_n$ and the other rule can be written as $(1+r_1)\dots(1+r_m)s_1\dots s_k \to 0$ corresponding to the clause $D = R_1 \vee \dots \vee R_m \vee \neg S_1 \dots \vee \neg S_k$.

a) If the overlap is on a pair $(q_i, s_j)$ then no divergent critical pair is possible.

b) Now suppose the overlap is on some pair $(q_i, r_j)$ with mgu $\theta$. W.l.o.g let i=j=1. Then the rule

$(1+r_1)\dots (1+r_m)s_1\dots s_k \to 0$ can be written in the form $r_1 f + f \to 0$ where $f = (1+r_2)\dots (1+r_m)s_1\dots s_k$. The only resulting N-critical pair is

$$\langle (f q_2 \dots q_n)\theta, 0 \rangle,$$

which results in the rule

$$((1+r_2)\dots (1+r_m)s_1\dots s_k q_2 \dots q_n)\theta \to 0.$$

This rule corresponds to the clause

$$(R_2 \vee \ldots \vee R_m \vee \neg S_1 \ldots \vee \neg S_k \vee \neg Q_2 \vee \ldots \vee \neg Q_n)\theta,$$

with $\theta \in mgu(Q_1, R_1)$, and this clause is a resolvent of C and D. ■

3.12 Corollary:

*Suppose $\mathcal{D}$ satisfies the conditions of 3.10. Then the $\mathcal{N}$-refutation $\mathcal{D}$ corresponds to a resolution refutation by removing multiple overlaps.*

■

## 4. Reduction in Terms of Resolution Inferences

In the previous section we proved that each N-completion refutation can be simulated by a resolution refutation. So we analysed the superposition process of the N-strategy (and THEOPOGLES of course) in terms of binary resolution without considering simplification. But the process of rewriting (simplification, normalization) is the most important inference rule of any Knuth-Bendix like algorithm. While the completion process adds new rules to the system, normalization simplifies rules and removes redundant rules. So simplification guarantees a minimal representation of the problem.

This section concerns the question how the rewriting mechanism can be translated into rules known from resolution based theorem proving. We first provide the basic definitions, and then we analyse the reduction steps locally.

In 2.1 and 2.2 we gave a rough definition of rewrite rules and the reduction relation. Now a refined definition is given.

4.1 Definition:

a) Let $p=hd(p)+tl(p)=0$ be a polynomial equation. $hd(p) \rightarrow tl(p)$ is a **rewrite rule w.r.t. $p=0$**, iff $hd(p)$ is a monomial and each atom occurring in $tl(p)$ also occurs in $hd(p)$.

b) Let $p,p'$ be polynomials and $\mathfrak{R}$ be a set of rewrite rules. The rewrite relation $\Rightarrow_{\mathfrak{R}}$ is defined by: $p \Rightarrow_{\mathfrak{R}} p'$ iff there is a rewrite rule $hd \rightarrow tl$ in $\mathfrak{R}$ and a substitution $\mu$, such that $p \mapsto^*_{hd \rightarrow tl, \mu} p'$.

$\Rightarrow^*_{\mathfrak{R}}$ denotes the reflexive and transitive closure of $\Rightarrow_{\mathfrak{R}}$.

c) A polynomial p is called **irreducible** (w.r.t. $\mathfrak{R}$) , if there is no $p'$, s.th. $p \Rightarrow_{\mathfrak{R}} p'$.

$p'$ is called a **normalform** of p, iff $p \Rightarrow^*_{\mathfrak{R}} p'$ holds and $p'$ is irreducible.

Our notion of rewrite rules covers the rewrite rules defined by Hsiang ($m \rightarrow 0$ , $L \rightarrow 1$) as well as those of THEOPOGLES ($m \rightarrow 0$, $L \rightarrow 1$, $L*m \rightarrow m$ ). Kapur & Narendran (1985) additionally use an ordering on the set of atoms, which renders their system more complex. Our notion can be seen as a specialization of Kapur & Narendrans', using a simplification ordering. The notion of parallel reduction is taken from the THEOPOGLES system. The common definition

provides a rewriting of only one monomial in a given polynomial. Thus the reduction relation above can easily be seen as a multiple application of the usual one. Note also that this system allows equations that are not transformed into rules.

Definition 4.2:

a) Let $G=\{p_i=0\}$ be a set of polynomial equations.

The pair (R,E) of rules and equations is a **KB-system** w.r.t. G, if

$R=\{hd(p)\rightarrow tl(p) \mid hd(p)\rightarrow tl(p)$ is a rewrite rule w.r.t. $p=0$ in $G\}$ and

$E=\{p=0\in G \mid$ there is no rewrite rule w.r.t. $p=0\}$

b) A KB-sytem (R,E) is **interreduced**, if p is irreducible w.r.t. R for every equation $p=0\in E$ and hd+tl is irreducible w.r.t. $R\backslash\{hd\rightarrow tl\}$ for every rule $hd\rightarrow tl\in R$.

An interreduced KB-system can be obtained from a system (R,E) by applying successively the rules in R to the polynomials of the system. The interreduction steps together with the completion steps discussed in section 3 define a complete proof procedure (cf. Müller (1988)). Now we try to interpret reduction steps in terms of resolution inference steps. First an example is given showing that not each reduction of a clause polynomial by a clause rule results in a clause polynomial. Then the following lemma gives an idea what we can expect in reducing a clause polynomial by a clause polynomial rewrite rule. Afterwards an example is given, where a clause polynomial is transformed into a clause polynomial by a sequence of reduction steps, but there cannot be any direct simulation in the resolution calculus.

Sometimes the resulting polynomial corresponds to a resolvent of the two clause rules. In many cases the result is not a clause polynomial, but a polynomial representing important information, which cannot be achieved by any resolution steps.

4.3 Example:

Consider the clauses $C_1 = \neg P \vee Q$ and $C_2 = P \vee \neg Q$. Possible resolvents are $\neg P \vee P$ and $Q \vee \neg Q$, which are tautological and thus usually eliminated (see sec. 5). The corresponding rewrite rules are $P * Q \rightarrow P$ and $P * Q \rightarrow Q$, respectively, and here interreduction will delete $P * Q \rightarrow P$ and create the new equation $P + Q = 0$. Now $P + Q$ is neither a clause polynomial nor is it tautological. Moreover $P + Q = 0$ provides the explicit information, that P and Q must be equal under all possible interpretations. Since in the KN-method an ordering on the atoms is used, the rule $P \rightarrow Q$ is generated, which has the effect that all P's are replaced by Q's.

4.4 Definition:

Let p and q be clause polynomials corresponding to clauses C and D, respectively. We define the relation $\leq$ by

$p \leq q$, iff C is a subset of D.

4.5 Lemma:

*Let p,q be clause polynomials and hd→ tl be a rewrite rule w.r.t. q=0. If* $p \Rightarrow_{\{hd \to tl\}} p'$ *holds, then* $p' = BNF(p+ \mu(q)*r)$, *for some clause polynomial r with* $r \leq p$ *and some substitution* $\mu$.

Proof:

Let $\mu$ be the substitution used to rewrite p into p'. Let m be the monomial $\mu$(hd). Then all atoms of m occur in p. Let $p = P_1...P_n(1+Q_1)...(1+Q_m)$ and let w.l.o.g. $m= P_1..P_iQ_1..Q_j$. Then

$p = P_1...P_i(1+Q_1)...(1+Q_j)\ (P_{i+1}...P_n(1+Q_{j+1})...(1+Q_m))$.

Let r be the clause polynomial $P_{i+1}...P_n(1+Q_{j+1})...(1+Q_m)$. Then we have

$p = (P_1..P_iQ_1..Q_j + s) * r = (m+s)*r = m*r+s*r$,

where m does not occur in s*r. Replacing m in all monomials of p by $\mu$(tl) yields

$p' = BNF(\mu(tl) *r+s*r )= BNF(p + m*r + \mu(tl) *r) = BNF(p + (\mu(hd)+\mu(tl))*r) =$
$BNF(p+ \mu(q)*r)$.

■

4.6 Lemma:

*Let p,q be clause polynomials with equal sets of atoms. If p+q is a clause polynomial, then there is an atom P and a clause polynomial s, such that p=Ps and q=(1+P)s.*

Proof:

Let $\mathbb{A}$ be the atom set of p. Let r=p+q. If each $P \in \mathbb{A}$ occurs in p with the same sign as in (the clause represented by) q, then p=q and r=p+q is no clause polynomial. Hence there must be $P \in \mathbb{A}$ occurring with different sign in p and q, say p=Ps and q=(1+P)t, where P does not occur in t nor in s. Then we have r = p+q= t+Ps+Pt. If P occurs positively in r, then Pu = r = t+Ps+Pt, hence t = Ps+Pt+Pu and P occurs in t, which is a contradiction. In the same way we obtain a contradiction, if P occurs negatively in r. Hence P does not occur in r, hence neither in r+t. But from r+t = Ps+Pt now follows that Ps+Pt=0. This implies P(s+t)=0, and since P does not occur in s nor in t we obtain s=t and p=Ps and q=(1+P)s.

■

4.7 Lemma :

*Let p,q be clause polynomials corresponding to the clause C, D, respectively and let hd → tl be a rewrite rule w.r.t. q=0.*

*If the relation* $p \Rightarrow_{\{hd \to tl\}} p'$ *holds, then*

*i) If p' = 0, then D subsumes C.*

*ii) If p' is a clause polynomial with clause RC, then RC is a resolvent of C and D and RC subsumes C.*

Proof:

Let $\mu$ be the substitution used to rewrite p into p' and let m be the monomial $\mu$(hd). Then all atoms of m occur in p.

i) According to 4.5 we have p' = p+$\mu$(q)*r = 0 with a clause polynomial r, hence p = $\mu$(q)*r. From this follows that $\mu$(D) is a subset of C, i.e. D subsumes C.

ii) According to 4.5 we have p' = p+$\mu$(q)*r, where the atom sets of p and $\mu$(q)*r are equal. Hence the conditions of lemma 4.6 are satisfied for p and $\mu$(q)*r. From 4.6 now follows that there is exactly one atom P occurring with different signs in p and $\mu$(q)*r, say p = P*s and $\mu$(q)*r = (1+P)*s. Furthermore 4.6 implies r≤p, hence (1+P) cannot occur in r. Thus $\mu$(q)=(1+P)s' with s'≤p. From p'=s now follows that the clause $\mathcal{RC}$ belonging to p' is the resolvent of C and D on the literal P, and s≤p implies that $\mathcal{RC}$ subsumes C. ■

Lemma 4.7 characterizes the rule p' resulting from interreducing a clause rule p with another clause rule: Either p' is a clause rule, and in this case it represents a resolvent of the two clauses, which subsumes one parent clause. This combined resolution and subsumption deletion step is called "replacement resolution" (K. Markgraf 1984). If r is not clausal, then either r=0, which corresponds to a subsumption deletion step or otherwise there is no correspondence on the "resolution side".

However, as the following example points out, interreduction behaves very differently from N-superpositions: In section 3 it was shown, that sequences of N-superpositions can be translated to resolution steps. This is in general not possible for interreduction, which allows rules other than N-rules.

4.8 Example

Let S = {P ∨ ¬Q ∨ R, ¬P ∨ Q ∨ ¬R, ¬P ∨ ¬Q, Q ∨ R, P ∨ ¬R}.

This clause set corresponds to the following set of rewrite rules

E = {PQR → PQ + RQ + Q, PQR → PR, PQ → 0, QR → Q + R + 1, PR → R}.

With interreduction steps only we obtain a deduction of 1:

$$\begin{aligned} PQR + PQ + RQ + Q &\Rightarrow_{\{PQR \to PR\}} PR + PQ + RQ + Q \\ &\Rightarrow_{\{PQ \to 0\}} PR + RQ + Q \\ &\Rightarrow_{\{PR \to R\}} R + RQ + Q \\ &\Rightarrow_{\{QR \to Q + R + 1\}} 1 \end{aligned}$$

However, S does not admit any resolution refutation consisting of four steps only. Hence, this deduction cannot be translated into a resolution deduction.

## 5. Resolution Inferences in CTPs

Kapur and Narendran (1985) showed that each resolvent of two clauses $C_1$ and $C_2$ can be generated as a clause polynomial from the corresponding clause polynomials $p_{C1}$ , $p_{C2}$ of $C_1$, $C_2$ by a completion step in the KN-method. The same is true with factors. If C is a clause and FC is a (binary) factor of C, then the corresponding factorpolynomial can be generated with an I-critical pair of $p_C$ and $x*x \rightarrow x$. Their results can easily be stated in our framework, provided that the generated resolvents are not tautologies nor mergeclauses. The reason is that merging and tautology elimination are performed in all CTPs implicitely. So we first consider the computation with the axioms of the underlying boolean algebra, i.e. the computation of the BNF of polynomials. Then we will give the results analogous to the KN-method. Finally we show that the subsumption rule can be simulated by interreduction.

5.1 Lemma:

*If C is a tautological clause then the corresponding polynomial $p_c$ in BNF is 0.*

Proof:

Let $C = L_1 \vee L_2 \vee ... \vee L_n$.

Since C is tautological, there are complementary literals in C, i.e. $L_i = L$ and $L_j = \neg L$.

W.l.o.g. let i = 1, j = 2.

Then $(C)\varphi = (L)\varphi * (\neg L)\varphi * (L_3 \vee ... \vee L_n)\varphi$

$= (L + 1) * L * (L_3 \vee ... \vee L_n)\varphi$

$= (L*L + L) * (L_3 \vee ... \vee L_n)\varphi$

$= (L + L) * (L_3 \vee ... \vee L_n)\varphi$ by $x * x = x$

$= 0 * (L_3 \vee ... \vee L_n)\varphi$ by $x + x = 0$

$= 0$ by $x * 0 = 0$

■

According to the definition of $\Rightarrow_R$ and the critical pairs, the derived polynomials are in BNF. Thus, whenever a polynomial p is generated, which corresponds to a tautological clause it has to be 0. So the inference rule of tautology elimination in resolution ATP's is part of CTP.

If clauses are not taken as sets in a resolution system, there may be multiple occurrences of literals in the generated resolvents. Those clauses are called merge clauses and are replaced by the clauses where multiple literals are deleted. This inference rule is called merging and is also part of CTP in a natural way.

5.2 Lemma:

*Let $C_1$ be a merge clause and $C_2$ the clause derived from $C_1$ by deletion of multiple occurrences of literals. Then the corresponding polynomial $p_c$ w.r.t. $C_1$ is the same as for $C_2$.*

Proof.

Let $C_1 = L_1 \vee L_2 \vee ... \vee L_n$.

Since $C_1$ is a merge clause there must be multiple occurrences of literals. W.l.o.g. let $L_1 = L_2$ be the only one. Then $C_2 = L_1 \vee L_3 \vee ... \vee L_n$.

$$\begin{aligned}(C_1)\varphi &= (L_1)\varphi * (L_2)\varphi * (L_3 \vee ... \vee L_n)\varphi \\ &= (L_1)\varphi * (L_3 \vee ... \vee L_n)\varphi \quad \text{by } x*x = x \text{ and } (L_1)\varphi = (L_2)\varphi \\ &= (C_2)\varphi\end{aligned}$$

■

Since derived polynomials are in BNF, no multiple occurrences of literals are possible and hence the merging inference rule is implicitly in the CTP.

5.3 Corollary:

*Let $C_1$, $C_2$ be two clauses, such that $C_1$ only contains negative literals, and $m \rightarrow 0$ and $hd \rightarrow tl$ be the corresponding rewrite rules for $C_1$, $C_2$ respectively. If $\mathcal{RC}$ is a binary resolvent for $C_1$, $C_2$ and $\mathcal{RC}$ is not a tautology nor a merge clause, then the rewrite rule $hd(p_{\mathcal{RC}}) \rightarrow tl(p_{\mathcal{RC}})$ for $\mathcal{RC}$ results from a superposition from $m \rightarrow 0$ and $hd \rightarrow tl$.*

5.4 Corollary:

*Let $C$ be a clause containing only nagative literals and let $m \rightarrow 0$ be the corresponding rewrite rule. If $\mathcal{FC}$ is a binary factor of $C$ and $\mathcal{FC}$ is not a tautologie nor a merge clause, then the rewrite rule $m_{\mathcal{FC}} \rightarrow 0$ for $\mathcal{FC}$ results from a I-critical pair of $m \rightarrow 0$.*

Whereas tautology elimination and merging ignore the context of other clauses, the subsumption rule is context sensitive. It says, that a clause C can be deleted from a set of clauses S, if there is another clause D in S and a substitution $\mu$, s.t. $\mu(D) \subseteq C$.

We now show, that deletion of subsumed clauses is also accomplished by the interreduction rule.

5.5 Lemma:

*Let $C_1$, $C_2$ be two clauses, neither tautological nor merge clauses, s.t. $C_1$ subsumes $C_2$.*

*If $p_1$, $p_2$ are the corresponding clause polynomials w.r.t. $C_1$,$C_2$ then the rule $hd(p_2) \rightarrow tl(p_2)$ is eliminated by interreduction.*

Proof:

Let NC1 = $\{L_1,...,L_i\}$, NC2a = $\{L_1',...,L_i'\}$, NC2b = $\{L'_{n+1},...,L'_{n+j}\}$ and NC2 = NC2a $\cup$ NC2b be the sets of negative literals in $C_1$,$C_2$, respectively.

Further, let PC1 = $\{L_{i+1},...,L_n\}$, PC2a = $\{L'_{i+1},...,L'_n\}$, PC2b = $\{L'_{n+j+1},...,L'_m\}$ and PC2 = PC2a $\cup$ PC2b be the sets of positive literals in $C_2$ and

$\mu$(NC1 $\cup$ PC1) = (NC2a $\cup$ PC2a) for some substitution $\mu$, which implies $\mu(C_1) \subseteq C_2$.

The corresponding clause polynomials are:

$$p_1 = \prod_{x \varepsilon NC1} x * \sum_{\alpha \subseteq PC1} \prod_{y \varepsilon \alpha} y$$

$$= \prod_{x \varepsilon NC1} x * \prod_{z \varepsilon PC1} z + \prod_{x \varepsilon NC1} x * \sum_{\alpha \subset PC1} \prod_{y \varepsilon \alpha} y$$

Then $hd(p_1) \rightarrow tl(p_1)$ is:

$$\prod_{x \varepsilon NC1 \cup PC1} x \rightarrow \prod_{x \varepsilon NC1} x * \sum_{\alpha \subset PC1} \prod_{y \varepsilon \alpha} y$$

And

$$p_2 = \prod_{x \varepsilon NC2} x * \sum_{\alpha \subseteq PC2} \prod_{y \varepsilon \alpha} y$$

$$= \prod_{x \varepsilon NC2a} x * \prod_{y \varepsilon NC2b} y * \sum_{\alpha \subseteq PC2a z \varepsilon \alpha} \prod z * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u$$

$$= \prod_{x \varepsilon NC2a \cup PC2a} x * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u +$$

$$\prod_{x \varepsilon NC2a} x * \sum_{\alpha \subset PC2a} \prod_{z \varepsilon \alpha} z * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u$$

Since $\mu(NC1 \cup PC1) = (NC2a \cup PC2a)$, we have $p_2 \overset{*}{\Longrightarrow} \{hd(p_1) \rightarrow tl(p_1)\ p_2'$, where

$$p_2' = \mu\left(\prod_{x \varepsilon NC1} x * \sum_{\alpha \subset PC1} \prod_{y \varepsilon \alpha} y\right) * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u +$$

$$\prod_{x \varepsilon NC2a} x * \sum_{\alpha \subset PC2a} \prod_{z \varepsilon \alpha} z * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u$$

$$= \prod_{x \varepsilon NC2a} x * \sum_{\alpha \subset PC2a} \prod_{z \varepsilon \alpha} z * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u +$$

$$\prod_{x \varepsilon NC2a} x * \sum_{\alpha \subset PC2a} \prod_{z \varepsilon \alpha} z * \prod_{y \varepsilon NC2b} y * \sum_{\beta \subseteq PC2b} \prod_{u \varepsilon \beta} u$$

by $\mu(NC1)=NC2a$ and $\mu(PC1)=PC2a$

$=0$ by $x + x = 0$

So $p_2'$ is 0 and according to the interreduction rule, the rule $hd(p_2) \rightarrow tl(p_2)$ will be deleted.

■

Since interreduction is performed on the set of rewrite rules until all rewrite rules are "irreducible", a CTP will eliminate all those clause polynomials whose corresponding clauses are deleted due to subsumption in a resolution ATP.

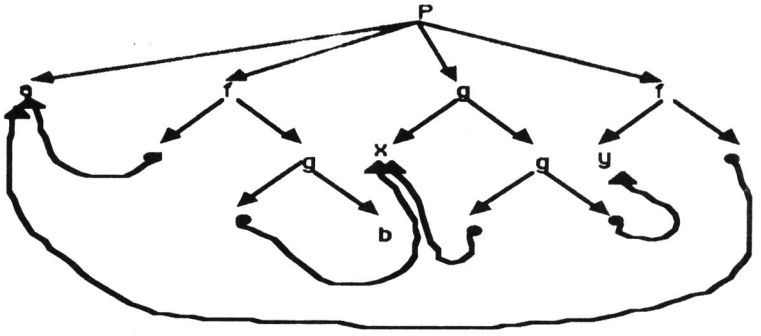
## 6. A well-suited Datastructure for Implementing CTPs

In implementing CTP's various problems concerning efficency in time and space arrive. One problem is to find a good representation for the polynomials and the second point is to minimize the time complexity for the search of possible reduction and completion steps. Further, it should be easy to incorporate modifications of the inference mechanisms and to adopt heuristics and control strategies. The solutions presented here consist of a structure sharing approach and a graph concept in analogy of Kowalski's Connection Graph Procedure (Kowalski 75).

A Datastructure for First Order Polynomials

Datastructures for atoms are discussed by Boyer & Moore (1972), Paterson & Wegman (1978) and Corbin & Bidoit (1983). As unification is the basic operation for the generation of critical pairs, the datastructure for the atoms basically depends on the unification algorithm to be used. Full structure sharing with dags (directed acyclic graphs), as it is proposed by Paterson and Wegman, or partial structure sharing with dags (Corbin and Bidoit,) seem to be best. It is well known that the algorithm of Peterson and Wageman is of linear complexity but because of the big overhead only practical when large atoms are to be unified. On the other side Bidoit and Corbin's approach is quadratic in general but the overhead is acceptable. So we will use only partial structure sharing, that is atoms are represented as a dag, where each variable and each constant is only represented once.

*Example:* The atom P( a , f( a , g( x , b)) , g( x ,g( x , y) )) , f(y , a)) is represented as:

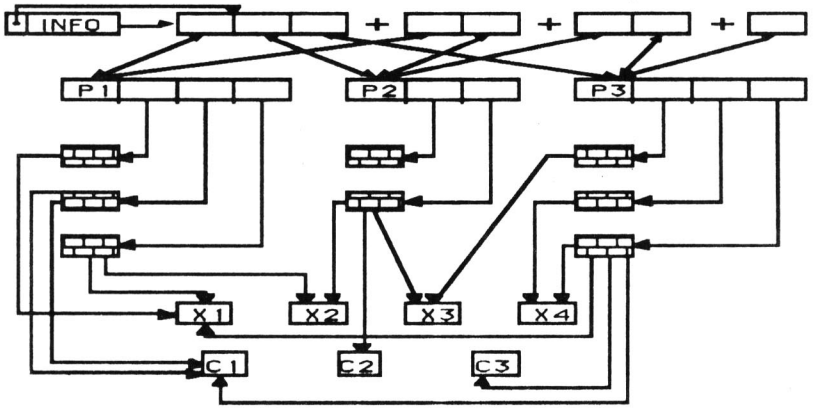



Constructing partial dags is straightforward, especially when LISP is used as implementation language, because then constants and literals are stored only once and the pointers are constructed by the interpreter.

Polynomials are defined as the sum of destinct monomials, but usually atoms have multiple occurences in a polynomial. For example in the polynomial

$P_1(x_1,f(c_1,c_1),f(x_1,x_2)) * P_2(a,g(x_2,c_2,x_3)) * P_3(x_3,x_4,g(f(x_4,x_1),c_1,c_3)) +$
$P_1(x_1,f(c_1,c_1),f(x_1,x_2)) * P_2(a,g(x_2,c_2,x_3)) +$
$P_2(a,g(x_2,c_2,x_3)) * P_3(x_3,x_4,g(f(x_4,x_1),c_1,c_3)) +$
$P_3(x_3,x_4,g(f(x_4,x_1),c_1,c_3))$

one atom occurs two times and the others occur three times. In using the structure sharing idea from above, we represent each atom only once and use pointers to the skeleton of the polynomial to mark their place. In detail we will represent the polynomial as in the following diagram:

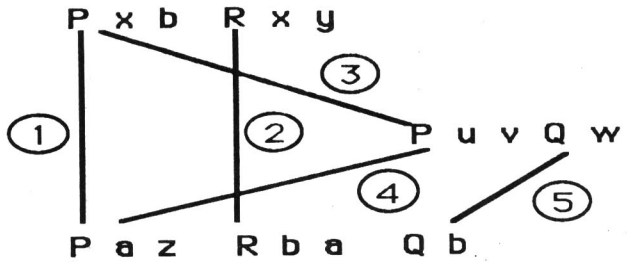



Each atom in the polynomial is represented in a atom list only once and there are pointers from each atom to the occurences in the polynomial and vice versa. The stoned squares stand for terms and the sharing of common variables and constants is extended, s.t. even different atoms may share the same elements. Since polynomials may be used as rewrite rules or as equations, there is a block with specific information (INFO) about the polynomials. There we may give a number to the polynomial, specify whether the polynomial is treated as an equation or as a rewrite rule, and if it is a rule, where the left hand side is. In the diagram above the first monomial is selected by the link from INFO to be the left hand side of the rewrite rule corresponding to the polynomial. The backward pointers from the skeleton to the atom list mark the atoms of the left hand side. Instances of these atoms have to be found to perform reduction steps and to generate critical pairs. How this problem is solved will be shown in following paragraph.

The Polynomial Graph

As in all Theorem Provers, the search for the objects to apply a given inference rule at is very time consuming in CTP's. In the Resolution context Kowalski (1975) proposed a clause graph where complementary literals are connected by an link and so all possible resolution steps were made explicit. When a resolvent is generated the link between the literals resolved upon is deleted and the resolvent is incorporated in the graph by inheriting the links of the father clauses.

In CTP's we would have to connect all atoms which are unifiable in all polynomials to get the desired informations. With the datastructure from above only the atoms in the atom lists of the polynomials have to be connected and since we want to avoid unnecessary computation of unifierers and matchers, all atoms with the same predicat symbol have to be connected by an (initial) link. Thus a graph as illustrated below is constructed.




The next step to improve the structure depends heavily on the special method to be used. To be more precise, applying a rule means, that only the left hand side atoms have to be matched against all other atoms. In performing superpositions only the left hand side atoms have to be unified (as singletons or as sets) with all other atoms or with only those which are also left hand side atoms. Thus we give the idea of the improvement which has to be adopted to the special procedure.

Links spreading from the list of left hand side atoms can be splitted into U-links (unification) and M-links (match) depending on whether the corresponding atom is also a left hand side atom or not. For example in the approach of Kapur and Narendran all overlaps of only left hand side atoms are generated. Hence the initial links between lhs atoms are splitted into U- and M-links, those between lhs atoms and others are changed into M-links and all others remain as initial links. In the case of Hsiang only overlaps between rules of the form m->0 (N-rules) and all other polynomials have to be considered, hence only U-links from the atoms of N-rules have to be created to all other atom lists.

The following sequence of diagrams will demonstrate how to work with the links. Figure a) shows the graph from above when the initial links are splitted into U-links (dark) and M-links (light). So each potential reduction and overlap situation is explicitely represented.

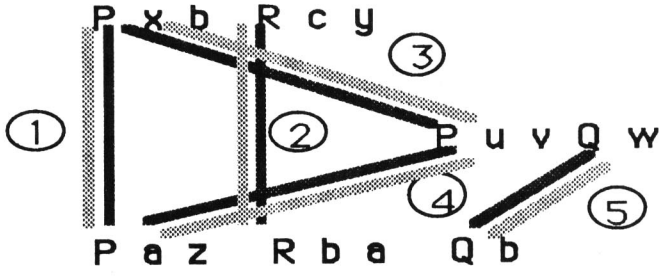



Figure a) Links after splitting

When U-link 2 is checked for unification it is noticed that there is no unifier of Rcy and Rba and thus the U-link is deleted. Further, since the atoms cannot be unified they cannot be matched either, and the M-link is also deleted. Checking M-link 1 for matching, results in deleting it because of unmatchability. The resulting graph after these two checks is figure b).

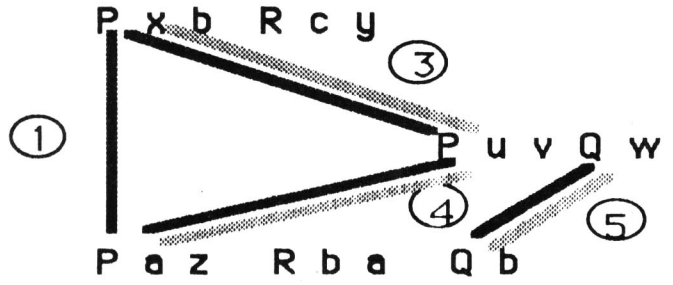



Figure b) Graph after deleting redundant links

Now considering U-link 3, there is the possibility for critical pair generation. Suppose that no atom with P as predicate symbol occurs in the generated polynomial (atoms are Rcw and Qb). U-link 3 is deleted, which means that the corresponding overlap is performed, and three new initial links have to be created. Link 6 and 7 are the links between the ancester atoms of the original atoms and their instances. Link 8 is the inherited from link 5.

A datastructure has been described that reduces the space for representing first-order polynomials drastically without increasing the overhead of management. The datastructure is further used to define a polynomial graph that makes searching for potential reductions and overlaps superfluous. By deleting and inheriting links the graph allways represents all possible occurences of inference rule applications in a CTP that have not been considered yet. The

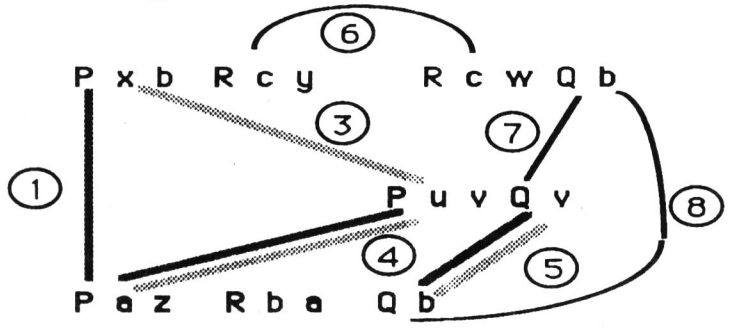



Figure c) Graph after completion step on U-link 3

proposed datastructure together with the polynomial graph is implemented in the CTP THEOPOGLES, described by Müller (1987) and is used there to realize various proof strategies by ordering the list of U-links in different ways.

## 6. Conclusion

We investigated several open problems in the framework of Completion Theorem Proving, i.e. the unnecessity of multiple overlaps, the relation of completion and simplification in terms of resolution inferences and vice versa and the tractability of polynomials in CTPs. It has been shown, that multiple overlaps are not necessary for the completeness of CTPs and moreover, that they may lead to longer refutation proofs. Simplification of rewrite rules by interreduction

was partially identified as simple resolution steps or subsumption deletion steps. However, as non-clausal rukles may be generated in the interreduction process, there is no one-to-one correspondence between interreduction and deduction (or reduction) rules known from resolution based theorem proving. But ignoring those intermediate nonclausal rules results in a transformation of CTP refutations to resolution refutations. The difficulties of managing polynomials in an implementation of CTPs were solved in introducing a well-suited datastructure that supports a fast application of the CTP inference rules.

## References


[BD87] Bachmair, L., Dershowitz N.: Inference Rules for Rewrite-Based First-Order Theorem Proving, *Proc. 2nd Annual Symp. on Logic and Computer Sience*, Ithaca, NY, 1987

[Bu85] Buchberger, B.: Basic Features and Development of the Critical-Pair/Completion Procedure, *1. Int. Conf. on Rewriting Techniques and Applications (RTA), Dijon*, May 1985, Springer LNCS 202 , pp.1-45.

[BM72] Boyer, R.S., Moore, J.S., The Sharing of Structure in Theorem Proving Program, in: *Machine ntelligence*, Vol.7, B. Meltzer, D. Michie (eds.), Edinburg University Press, Scotland, 1972, pp.101-116

[CB83] Corbin, J., Bidoit, M. : A Rehabilitation of Robinson's Unification Algorithm, in Information Processing 83, R.E.A. Mason (ed.), Elsevier Sc. Pub. (North Holland), pp.909-914

[CL73] Chang, C.L.& Lee, R.C.T.: Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York, 1973.

[D86] Dietrich, R. :Relating Resolution and Algebraic Completion for Horn Logic. In: J. Siekmann (Ed.): *Proc. 8th International Conference on Automated Deduction*, Oxford, England. Springer LNCS 230, 1986.

[H83] Herold, A. : Some Basic Notions of First-Order Unification Theory. MEMO-SEKI-VIII, Universität Karlsruhe 1983.

[Hs82] Hsiang, J.: Topics in Automated Theorem Proving and Program Generation Ph.D. Thesis, Dec. 1982, University of Illinois at Urbana-Champaign, UIUCDCS-R-82-1113 .

[Hs85a] Hsiang, J.: Two Results in Term Rewriting Theorem Proving, *Proc. RTA*, Dijon, May 1985, Springer LNCS 202, pp.301-324

[Hs85b] Hsiang, J.: Refutational Theorem Proving using Term-Rewriting Systems, *Artificial Intelligence* 25 (1985), pp.255-300.

[HD83] Hsiang, J. and Dershowitz, N.: Rewrite Methods for Clausal and Non-Clausal Theorem Proving, *Proc. of the 10. EATCS Int. Colloq. on Automata, Languages and Programming* (ICALP), Spain 1983.

[Hu81] Huet, G.: A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm, Journal of Computer and System Sciences 23 (1981), pp.11-21.

[KB70] Knuth, D.E. and Bendix, P.B.: Simple Word Problems in Universal Algebras in Computational Problems in Abstract Algebras (Ed. J. Leech), Pergamon Press, 1970, pp.263-297.

[KN85] Kapur, D. and Narendran, P.: An Equational Approach to Theorem Proving in First-Order Predicat Calculus, 84CRD322, General Electric Corporate Research and Development Report, Schenectady, N.Y., Sept.1985.

[Ko75] Kowalski, R.: A Proof Procedure Using Connection Graphs, *J. of the ACM* 22,4 (Oct.75), pp.572-595

[MK84] Markgraf, Karl: The Markgraf Karl Refutation Procedure. SEKI-Memo MK-84-01, University Kaiserslautern (1984).

[Mü87] Müller, J.: THEOPOGLES - A Theorem Prover based on First-Order Polinomials and a Special Knuth-Bendix Procedure, In: K. Morik (eds): *Proc.11th German Workshop on Artificial Intelligence*, Springer IFB152.

[Mü88] Müller, J.: Theorembeweisen mit Rewrite-Techniken. Dissertation Fachbereich Informatik, Universität Kaiserslautern.

[PW78] Paterson, M.S., Wegman, M.N.: Linear Unification, *J. of Computer and Systems Sciences* 16, 1978, pp.158-167

## Appendix:

This appendix contains some remarks concerning the completeness of Hsiang's N-strategy and especially the BN-unification (cf. [Hs 85b]). In [BD87] Bachmair remarks, that the completeness of the N-strategy is still an open problem. The reason is, that Hsiang's proof applies only to the propositional case (without reduction); he suggested that the first order case could easily be deduced with a lifting lemma based on BN-unification. However, the N-strategy using BN-unification is not complete as the following example shows.
Consider the clause set S={Pax Pxa, ¬Pay ¬Pya}. The corresponding set of rules is
R={Pax*Pxa + Pax + Pxa + 1 -> 0 , Pay*Pya->0}. Now the simplification of the first polynomial with the second rule gives R= {Pax + Pxa + 1 -> 0 , Pay*Pya->0}. BN-unification yields three different critical pairs, all of which are trivial. We have
⟨ Paa*Paa + Paa , 0 ⟩ from the unification of Pxa and Pay (Pax , Pya resp.)
⟨ Pay*Pay + Pay , 0 ⟩ from the unification of Pxa and Pya
⟨ Pya*Pya + Pya , 0 ⟩ from the unification of Pax and Pay.
According to the (normalizing) rules of BN the equation X*X+X=0 holds and thus the critical pairs are trivial (i.e. they reduce to ⟨ 0 , 0 ⟩ ) and the N-strategy stops with "consistent". But S (E resp.) are obviously inconsistent.
Well, one might argue, that simplification and normalization should not be done in such rigorous a way to preserve the completeness of the strategy. Yet, the problem is a deeper one. Hsiang defines BN-unification in the following way:

Two monomials m and m´ are BN-unifiable, iff there are monomials u,v and a substitution

$\mu$, s.th. $\mu(u*m) = \mu(v*m')$.

This problem is properly treated by his BN-unification algorithm. The point is, that we have to work with the *theory* introduced by the BN rule system and therefore we have to take "=" not as the syntactical equality, but as equality modulo the boolean ring theory.This gives:

Two monomials m and m´ are BN-unifiable, iff there are monomials u,v and a substitution $\mu$, s.th. $\mu(u*m) =_{BN} \mu(v*m')$.

Or equivalently:

Two monomials m and m´ are BN-unifiable, iff there are monomials u,v and a substitution $\mu$, s.th. $\mu(u*m) =^{*}>_{BN} w\ {}_{BN}<^{*} = \mu(v*m')$.

That is, either we may treat the atoms in m and m´ as sets and perform unification of atom sets, as it is defined in the early Robinson unification algorithm † or we have to perform an explicit factoring, that is unification of atoms within a monomial, as it is done later in Resolution theorem provers. The BN-unification algorithm only finds parts of all possible factoring substitutions. In Kapur and Narendran's approach the factoring process is captured by the overlap of monomials with the BN-rule x*x->x. In [Mü88] we show that in the final version of THEOPOGLES explicit factoring is also necessary and that factoring the N-rules is sufficient to guarantee completeness.

---

† J.A. Robinson, A Machine-Oriented Logic Based on the Resolution Principle, JACM 12 (1), Jan.1965, also in Automation of Reasoning Vol.1, (Eds.: J. Siekmann, G. Wrightson), Springer, Berlin 1983.

Note, that Robinsons first version of the paper was rejected in 1963 essentially for the same reasons discussed above.