Appeared in the Proceedings of the 11th International Conference on Automated Deduction 1992

Cycle Unification

Wolfgang Bibel[*], Steffen Hölldobler[*], Jörg Würtz[†]

Abstract Two-literal clauses of the form $L \leftarrow R$ occur quite frequently in logic programs, deductive databases, and – disguised as an equation – in term rewriting systems. These clauses define a cycle if the atoms L and R are weakly unifiable, ie. if L unifies with a new variant of R. The obvious problem with cycles is to control the number of iterations through the cycle. In this paper we consider the cycle unification problem of unifying two literals G and F modulo a cycle. We review the state of the art of cycle unification and give some new results for a special type of cycles called matching cycles, ie. cycles $L \leftarrow R$ for which there exists a substitution σ such that $\sigma L = R$ or $L = \sigma R$. Altogether, these results show how the deductive process can be efficiently controlled for special classes of cycles without losing completeness.

1 Introduction

It is the foremost goal of the research in the field of automated deduction to develop general *and* adequate proof methods and techniques for the logics under consideration. It is comparatively easy to invent a general proof method, but it is much more difficult to develop a general *and* adequate proof technique. For example, the resolution principle [15] and the connection method [1] are general proof methods for first-order logic. But are they adequate? What is the meaning of adequateness in the first place? Roughly speaking, we will consider a technique as being adequate if it solves simpler problems faster than more difficult ones. We illustrate the notion of adequateness by a problem, where the known general proof techniques face difficulties whereas trained humans seem to be able to solve it quite reasonably.

For this purpose, consider the following set of clauses in Prolog-like notation which is taken from [14] and was originally studied by Łucasiewicz.

$$\begin{array}{lcll} & \leftarrow & Pi(iab, i(ibc, iac)). & \text{G} \\ Pw & \leftarrow & Pv,\ Pivw. & \text{MP} \\ Pi(i(ixy, z), i(izx, iux)). & & & \text{A} \end{array}$$

[*]Fachgruppe Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt, Alexanderstraße 10, 6100 Darmstadt, Germany

[†]Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany

The terms represent implicational formulas, ie. iab encodes $a \to b$ and P asserts the derivability of its argument. Thus, the second clause represents modus ponens. It contains several *cycles* [2] defined by the connections between the atom Pw and the atoms $Pivw$ and Pv. The clause MP can be applied to itself and this may lead to an exponential growth of the search space. The obvious problem is to control the self-applicability of MP while retaining completeness. Łukasiewicz has found a 29 step proof. He must have exercised a good control over MP! Quintus PROLOG on a Sun SPARC station 2 did not find a proof in several days. Nearly all existing automatic theorem provers cannot solve this problem as well since they are not able to exercise a good control over MP. E. Lusk[1] reports that the parallel version of Otter at Argonne is able to obtain a hyperresolution proof with about 150 proof-steps while generating 6.5 million clauses in about half an hour during the search for it. Their prover does not have a good control over MP as well. It solves the problem by sheer power.

In [3] it was conjectured that a problem like the Łucasiewicz-formula could be solved in less than a second by way of a technique called *cycle unification*. At present this conjecture remains a challenge since the Łucasiewicz-formula is a particularly difficult instance of a class of formulas which could eventually be treated by cycle unification. In this paper we make a first step towards this goal by restricting our attention to the special case of formulas with exactly one cycle. In fact, we even focus our analysis on the simple class of two-literal clauses of the form $Pl_1 \ldots l_n \leftarrow Pr_1 \ldots r_n$ which consists of nothing but a single cycle. This additional restriction simplifies the discussion without loss of generality of the method.

Such a two-literal clause is usually embedded in the context of some larger formula, or set of clauses. Again for simplicity of the discussion and without loss of generality, we restrict the treatment to the case of two additional clauses, namely a goal clause – referred to as *(calling) goal* – of the form $\leftarrow Ps_1 \ldots s_n$, which calls the cycle, and a fact – called *(terminating) fact* – of the form $Pt_1 \ldots t_n$, which terminates the cycle. In our restricted case a *cycle unification problem* is then the following one:

> Is there a substitution σ such that $\sigma Ps_1 \ldots s_n$ is a logical consequence of $Pl_1 \ldots l_n \leftarrow Pr_1 \ldots r_n$ and $Pt_1 \ldots t_n$?

If such a substitution σ exists, then σ is said to be a *solution* for the cycle unification problem. For more general cases, cycle unification can be defined in an analogue way.

In order to be able to control a cycle we have to answer the following questions. Is cycle unification decidable? How many independent most general solutions has a cycle unification problem? Does there exist a unification algorithm which enumerates a minimal and complete set of solutions for a cycle unification problem? Answers to these questions may help to increase the power of automated theorem provers significantly. For example, if a cycle is embedded in a

[1]private communications

larger formula and it can be determined that the corresponding cycle unification problem is unsolvable, then the clauses defining the cycle can be eliminated from the formula. If a minimal and complete set Σ of solutions for a cycle unification problem exists and can be enumerated, then any other solution is subsumed by a solution in Σ and need not to be considered. If Σ is finite, then this may prune a potentially infinite search space to a finite one. But theorem proving is not the only task which may benefit from cycle unification. There are a variety of applications for cycle unification such as intelligent backtracking, deductive databases, program transformation, and termination proofs for logic programs, to mention just a few.

Although cycle unification is of significant importance for the field of automated deduction, it has received surprisingly little attention in the literature. Function-free cycle unification problems, ie. cycle unification problems defined over variables and constants only, occur mainly in deductive databases and it can be shown that under certain conditions these problems do not give rise to infinite computations (cf. [10]). In [13] the number of iterations through a cycle can be limited via a user-defined parameter. In [20] certain cycle unification problems are solved by generalization and subsumption. There, after several iterations through a cycle, subterms occurring in a goal are replaced by variables. Subsumption techniques may now be applied to terminate otherwise infinite derivations. The technique is shown to be complete. Unfortunately, answers to the generalized goal need not to be answers to the initial goal. M. Schmidt–Schauß [16] has shown that cycle unification is decidable provided that the goal and the fact are ground, ie. they do not contain variable occurrences. Independently, P. Devienne [7] has given a more general result for cycle unification problems with linear goals and facts, ie. each variable occurs at most once in the goal and the fact. He uses essentially the same ideas as Schmidt–Schauß, but a very special technique based on directed weighted graphs. Devienne's results were used by De Schreye et al. [17] to decide whether cycles admit nonterminating queries to deductive systems. Another approach has been taken by H.J. Ohlbach [11] who represented sets of terms by so-called abstraction trees which may compress the search space. Moreover, abstraction trees can be used to compile two-literal clauses and in certain cases a finite abstraction tree can represent infinitely many solutions of a cycle unification problem [12].

This paper is a first step towards a theoretical foundation of cycle unification. After some preliminary notes on definitions and notations we formally define cycle unification in Section 3. In Section 4 we define various new classes of restricted cycle unification problems with increasing complexity and show that their unification problems are decidable, determine the unification types, and develop unification algorithms. The paper concludes with a summary of the results on cycle unification and an outline of future work.

2 Definitions and Notations

Our definitions and notations follow those suggested in [6]. Throughout this paper capital letters such as P, Q, ... denote *predicate symbols*, small letters such as a, b, ... denote *constants*, f, g, ... *function symbols*, and z, y, ... *variables*. A *term* is either a variable or of the form $f(t_1,\ldots,t_n)$, where t_1, ..., t_n are terms. s, t, ... denote terms. An *atom* is of the form $P(t_1,\ldots,t_n)$. Let X be an atom or a term. $\mathcal{V}ar(X)$ denotes the set of variables occurring in X. X is called *ground* iff X does not contain any variable. X is called *linear* iff every variable occurs at most once in X. By X^k we denote the syntactic object where each variable occurring in X has the index k attached to it.

A *substitution* is a mapping from the set of variables into the set of terms which is equal to the identity almost everywhere. Hence, it can be represented as a finite set of pairs $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, $x_i \neq t_i$, $1 \leq i \leq n$. Substitutions are denoted by small greek letters such as σ, θ, ... The identity substitution is called ε. $\sigma t = \sigma(t)$ if t is a variable and $\sigma t = f(\sigma t_1,\ldots,\sigma t_n)$ if $t = f(t_1,\ldots,t_n)$. $\mathcal{D}om(\sigma) = \{x \mid x \text{ is a variable and } \sigma x \neq x\}$ is the *domain* of σ. $\mathcal{VR}an(\sigma) = \bigcup_{x\in\mathcal{D}om(\sigma)} \mathcal{V}ar(\sigma x)$ is the *variable range* of σ.

The *composition* $\sigma\tau$ of two substitutions σ and τ is defined by $(\sigma\tau)x = \sigma(\tau x)$. The *restriction* of the substitution σ to the set V of variables is defined by $\sigma|_V\, x = \sigma x$ if $x \in V$ and $\sigma|_V\, x = x$ otherwise. A substitution σ is called *variable-pure* if $\{\sigma x \mid x \in \mathcal{D}om(\sigma)\}$ only consists of variables. A *renaming* is a variable-pure substitution σ such that $\sigma x = \sigma y$ implies $x = y$ for $x, y \in \mathcal{D}om(\sigma)$.

If W is a set of variables, then $\sigma = \tau\ [W]$ iff $\forall x \in W : \sigma x = \tau x$. A substitution σ is called *more general* than a substitution τ on W, $\sigma \lessdot \tau\ [W]$, iff there exists a substitution ρ such that $\rho\sigma = \tau\ [W]$. Two substitutions σ and τ are called *equivalent* (or *variants*) on W, $\sigma \sim \tau\ [W]$, iff $\sigma \lessdot \tau\ [W]$ and $\tau \lessdot \sigma\ [W]$. σ and τ are equal on W *modulo renaming* iff there exists a renaming ρ such that $\rho\sigma = \tau\ [W]$. Two substitutions σ and τ are called *independent* on W iff $\sigma \not\lessdot \tau\ [W]$ and $\tau \not\lessdot \sigma\ [W]$.

σ is called a *unifier* for t and t' iff $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(t) \cup \mathcal{V}ar(t')$ and $\sigma t = \sigma t'$. A unifier σ of t and t' is called *most general unifier* iff $\sigma \lessdot \tau\ [\mathcal{V}ar(t) \cup \mathcal{V}ar(t')]$ for all unifiers τ of t and t'. σ is called a *matcher* for t and t' iff $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(t')$ and $t = \sigma t'$ holds. A matcher σ of t and t' is called a *most general matcher* iff $\sigma \lessdot \tau\ [\mathcal{V}ar(t')]$ for all matchers τ of t and t'. The definitions above can be extended to atoms, equations, and sets of equations in the obvious way.

3 Cycle Unification

$C = \{L \leftarrow R\}$ is called a *cyclic theory*, or *cycle* for short, if the atoms L and R are weakly unifiable, ie. there exist two substitutions σ and σ' such that

$\sigma L = \sigma' R$ [8]. Let G and F be two atoms such that $\mathcal{V}ar(G) \cap \mathcal{V}ar(F) = \emptyset$. A *cycle unification problem* $\langle G \xrightarrow[C]{\circ} F\rangle$ (or $\langle G \xrightarrow{\circ} F\rangle_C$) is the problem whether there exists a substitution σ such that σG is a logical consequence of F and C. A substitution σ is a *solution* for the cycle unification problem if $\mathcal{D}om(\sigma) \subseteq \mathcal{V}ar(G)$ and σG is a logical consequence of F and C.[2]

Since solutions to cycle unification problems are substitutions, the notions of more general, independent, etc. substitutions can be extended to more general, independent, etc. solutions of cycle unification problems in the obvious way.

As a first example consider the problem

$$\langle Pa \xrightarrow{\circ} Pfffa\rangle_{\{Px \leftarrow Pfx\}}.$$

The empty substitution ε is the only most general solution for this problem. However, there may be more than one solution as the example

$$\langle Pxy \xrightarrow{\circ} Pab\rangle_{\{Pvw \leftarrow Pwv\}}$$

shows. This problem has the two independent most general solutions $\{x \mapsto a,\ y \mapsto b\}$ and $\{x \mapsto b,\ y \mapsto a\}$. But, there may be even infinitely many independent most general solutions. As an example consider

$$\langle Px \xrightarrow{\circ} Pa\rangle_{\{Pfy \leftarrow Py\}}.$$

This problem has the most general solutions $\{x \mapsto a\}$, $\{x \mapsto fa\}$, $\{x \mapsto ffa\}$, ...

For a cycle unification problem $\langle G \xrightarrow{\circ} F\rangle_{\{L \leftarrow R\}}$ to be solvable, the atoms F and G must be of the form $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$, respectively. Since L and R are weakly unifiable, their predicate symbols must also be identical, ie. L and R must be of the form $P'(l_1, \ldots, l_n)$ and $P'(r_1, \ldots, r_n)$, respectively. In the sequel we will only consider cycle unification problems of this form. Furthermore, as the case $P \neq P'$ is trivial, we assume $P = P'$.

To solve a cycle unification problem $\langle G \xrightarrow[C]{\circ} F\rangle$ we have to find a substitution which either unifies G and F or unifies – viz. simultaneously unifies each equation in –

$$\mathcal{C}^k = \mathcal{N} \cup \mathcal{Y}^k \cup \mathcal{X}^k,$$

where

$\mathcal{N} = \{s_1 \doteq l_1^1,\ \ldots,\ s_n \doteq l_n^1\}$ is the set of *e*$\underline{n}$*try equations*,

$\mathcal{Y}^k = \{r_1^i \doteq l_1^{i+1},\ \ldots,\ r_n^i \doteq l_n^{i+1} \mid 1 \leq i \leq k\}$
is the set of *c*$\underline{y}$*cle equations* for k iterations through the cycle, and

$\mathcal{X}^k = \{r_1^{k+1} \doteq t_1,\ \ldots,\ r_n^{k+1} \doteq t_n\}$
is the set of *e*$\underline{x}$*it equations* after k iterations through the cycle.

[2]A cycle unification problem should not be confused with a theory unification problem $\langle G =_C F\rangle$, ie. the problem whether there exists a substitution σ such that $\sigma G =_C \sigma F$ [1, 19].

The following proposition is an immediate consequence of the completeness and soundness of the connection method [1] or SLD-resolution, eg. [9]. Due to lack of space we had to omit the proof of this proposition and all further theorems. They can be found in detail in [4].

Proposition 1 *σ is a solution for $\langle G \overset{\circ}{\underset{C}{\rightarrow}} F\rangle$ iff there exists a substitution θ such that θ unifies G and F and $\sigma = \theta|_{\mathcal{V}ar(G)}$ or there exists a natural number k such that θ unifies $\mathcal{C}^k$ and $\sigma = \theta|_{\mathcal{V}ar(G)}$.*

Throughout the paper τ will denote the most general unifier of G and F restricted to $\mathcal{V}ar(G)$, if it exist. Similarly, τ_k will denote the most general unifier of $\mathcal{C}^k$ restricted to $\mathcal{V}ar(G)$, if it exists.

Let $\mathcal{C} = \langle G \overset{\circ}{\underset{C}{\rightarrow}} F\rangle$ be a cycle unification problem. A set Σ of substitutions is a *complete* set of solutions for $\mathcal{C}$ iff each substitution in Σ is a solution for $\mathcal{C}$ and for each solution θ for $\mathcal{C}$ we find a substitution σ in Σ such that $\sigma \lessdot \theta[\mathcal{V}ar(G)]$. A complete set Σ of solutions for $\mathcal{C}$ is said to be *minimal* iff for all $\sigma,\ \theta \in \Sigma$ we find that $\sigma \lessdot \theta[\mathcal{V}ar(G)]$ implies $\sigma = \theta$.

In order to be able control a cycle, we are interested in the answer to three basic questions. Is cycle unification decidable? How many independent most general solutions has a cycle unification problem? Does there exist a unification algorithm which enumerates a minimal and complete set of solutions for a cycle unification problem?

Following [18], we define the type of a cycle unification problem as follows. A cycle unification problem is of type *unitary* iff there exists a single most general solution, *finitary* iff there exist finitely many most general solutions, and *infinitary* iff there exist infinitely many most general solutions.

4 Matching Cycles ($\mathcal{C}_m$)

A cycle $L \leftarrow R$ is called *left matching* and *right matching* iff there is a substitution σ such that $\sigma L = R$ and $L = \sigma R$, respectively. We assume σ to be the most general matcher. Throughout the paper we will consider only right matching cycles. The corresponding results for left matching cycles can be obtained analogously. The class of cycle unification problems with matching cycles is denoted by $\mathcal{C}_m$.

Matching cycles show some interesting properties. The substitution σ_i which is the most general unifier of R^i and L^{i+1} in the i-th iteration through the cycle can easily be obtained from the substitution σ which matches L against R as follows. Let

$$P_\sigma = \{(x, \sigma x) \mid x \in \mathcal{D}om(\sigma)\} \cup \{(x, x) \mid x \in \mathcal{V}ar(R) \setminus \mathcal{D}om(\sigma)\}.$$

Then,

$$\sigma_i = \{x^i \mapsto t^{i+1} \mid (x,t) \in P_\sigma\}.$$

For example, the cycle $\{Pgz,ga,z \leftarrow Pxyz\}$ is right matching with $\sigma = \{x \mapsto gz,\ y \mapsto ga\}$ and we obtain $P_\sigma = \{(x, gz),\ (y, ga),\ (z, z)\}$ and $\sigma_i = \{x^i \mapsto gz^{i+1},\ y^i \mapsto ga,\ z^i \mapsto z^{i+1}\}$. Furthermore, since $\mathcal{D}om(\sigma_i) \cap \mathcal{D}om(\sigma_{i+1}) = \emptyset$ the most general solution of the cycle equations $\mathcal{Y}^k$ is simply the composition $\rho = \sigma_k \cdots \sigma_1$. The solution τ_k for $\mathcal{C}^k$ can now be obtained by simultaneously unifying the atoms ρG, ρL^1 and ρR^{k+1}, ρF and restricting their most general unifier to $\mathcal{V}ar(G)$, if it exists. Since $\rho G = G$, $\rho R^{k+1} = R^{k+1}$, and $\rho F = F$, the interesting bindings in ρ are those for the variables occurring in L^1.

We will now consider two classes of matching cycle unification problems defined by the matching substitution σ. These are the classes of variable-pure matching cycles and non-recursive matching cycles.

4.1 Variable-pure Matching Cycles ($\mathcal{C}_{vp}$)

A variable-pure matching cycle $L \leftarrow R$ is a matching cycle, where the matching substitution σ is variable-pure. Hence, σ must be of the form

$$\{x_1 \mapsto y_1,\ \ldots,\ x_l \mapsto y_l\},$$

where $x_j \in \mathcal{V}ar(R),\ y_j \in \mathcal{V}ar(L),\ 1 \leq j \leq l$, and, thus, P_σ is a set of variable-pairs.

A set of variable-pairs P recursively defines a *sequence* of variables as follows. If $(x_1, x_2) \in P$, then P defines the sequence $\langle x_1, x_2\rangle$. If $\langle x_1, \ldots, x_l\rangle$ is a sequence defined by P and $(x_l, x_{l+1}) \in P$, then $\langle x_1, \ldots, x_l,\ x_{l+1}\rangle$ is a sequence defined by P. A sequence $\langle x_1, \ldots, x_l\rangle$ is called *linear* iff $x_i \neq x_j,\ 1 \leq i, j \leq l,\ i \neq j$. A sequence $\rho = \langle x_1, \ldots, x_l\rangle$ *contains* (or is a *subsequence* of) sequence π iff there exists an $i \geq 1$ and an $j \leq l$ such that $\pi = \langle x_i, \ldots, x_j\rangle$. A sequence π is called *maximal* iff there is no longer sequence containing π. A sequence $\pi = \langle x_1, \ldots, x_l\rangle$ is called a (*cyclic*) *permutation* iff $\langle x_1, \ldots, x_{l-1}\rangle$ is linear and $x_1 = x_l$. A sequence $\langle x_1, \ldots, x_l\rangle$ is called a (*cyclic*) *permutation with linear entry-sequence* iff $\langle x_1, \ldots, x_{l-1}\rangle$ is maximal and linear and there exists an j, $1 < j < l$, such that $x_l = x_j$. It is obvious that a permutation with linear entry-sequence can be divided into the linear entry-sequence and the permutation. Finally, a sequence $\langle x_1, \ldots, x_{l+1}\rangle$ has *length* l.

Let $C = \{L \leftarrow R\}$ be a variable-pure matching cycle with matching substitution σ. C is a *linear sequence* iff P_σ defines a single maximal and linear sequence. A cycle C is a *permutation* iff P_σ defines a permutation such that the variables occurring in it are equal to the variables occurring in P_σ. C is a *permutation with linear entry-sequence* iff P_σ defines precisely one permutation with linear entry-sequence.

In the following paragraphs we will formally investigate the properties of the classes of cycles defining a linear sequence ($\mathcal{C}_{ls}$), a permutation ($\mathcal{C}_p$), and a permutation with linear entry-sequence ($\mathcal{C}_{pls}$). As a combination we obtain results for the class of variable-pure matching cycles ($\mathcal{C}_{vp}$).

Linear Sequences ($\mathcal{C}_{ls}$). In this section we consider variable-pure matching cycles $\{L \leftarrow R\}$ with matching substitution σ where P_σ defines a single, maximal linear sequence. Such a cycle is contained in the unification problem

$$\langle Pw_1w_2w_3 \overset{\circ}{\longrightarrow} Pfa,bc\rangle_{\{Pfy,zv \leftarrow Pfx,yz\}},$$

where $\sigma = \{x \mapsto y,\ \ y \mapsto z,\ \ z \mapsto v\}$ is the matching substitution. P_σ defines the linear sequence $\langle x, y, z, v\rangle$ with length $l = 3$ and $\sigma_i = \{x^i \mapsto y^{i+1},\ y^i \mapsto z^{i+1},\ z^i \mapsto v^{i+1}\}$. As mentioned before, the solution for the cycle-equations $\mathcal{Y}^k$ is $\sigma_k \cdots \sigma_1$ and we are only interested in the restriction of $\sigma_k \cdots \sigma_1$ to $\mathcal{V}ar(L^1)$. Thus, we compute

$$\sigma_1|_{\{y^1,\ z^1,\ v^1\}} = \{y^1 \mapsto z^2,\ z^1 \mapsto v^2\}$$

and

$$\sigma_2\sigma_1|_{\{y^1,\ z^1,\ v^1\}} = \{y^1 \mapsto v^3,\ z^1 \mapsto v^2\}.$$

Since $\mathcal{D}om(\sigma_i)$, $i \geq 1$, does not contain any (superscribed) variable v, we find that for all $k > 2$

$$\sigma_k \cdots \sigma_1|_{\{y^1,\ z^1,\ v^1\}} = \sigma_2\sigma_1|_{\{y^1,\ z^1,\ v^1\}}$$

holds. One should observe that (i) a (superscribed) variable v is never in the domain of a most general solution for the exit equations and (ii) more and more variables in $\mathcal{V}ar(L^1)$ are mapped on a (superscribed) variable v until eventually all variables in $\mathcal{V}ar(L^1)$ are mapped on (superscribed) v. This is not only true in the example but holds for all solvable linear sequences. Intuitively, the more iterations through the cycle are considered the less is the influence of the exit equations on the solution of the cycle unification problem $\mathcal{C}^k$. In the example we obtain the solutions $\tau_0 = \{w_1 \mapsto fb,\ w_2 \mapsto c\}$ for $\mathcal{C}^0$, $\tau_1 = \{w_1 \mapsto fc\}$ for $\mathcal{C}^1$, $\tau_2 = \{w_1 \mapsto fv^3\}$ for $\mathcal{C}^2$, $\tau_3 = \tau_2$ for $\mathcal{C}^3$, etc. One should observe that

$$\tau_{l-1} \trianglelefteq \tau_{l-i} \quad \text{for } 0 < i \leq l \tag{1}$$

and

$$\tau_{l-1} = \tau_{l+i} \quad \text{for } 0 \leq i. \tag{2}$$

Thus, from (1) and (2) we conclude that cycle unification problems defining linear sequences with length l can be solved either by τ, ie. the restriction of the most general unifier of G and F to $\mathcal{V}ar(G)$, or by τ_{l-1}, ie. the restriction of the most general unifier of $\mathcal{C}^{l-1}$ to $\mathcal{V}ar(G)$, if they exist. In the example $\tau_{l-1} = \tau_2$ is even more general than the most general unifier $\tau = \{w_1 \mapsto fa,\ w_2 \mapsto b,\ w_3 \mapsto c\}$ of $Pw_1w_2w_3$ and Pfa,bc, but this is not the case in general. Conversely, if neither G and F are unifiable nor $\mathcal{C}^{l-1}$ is solvable, then the cycle unification problem is unsolvable.

Permutations ($\mathcal{C}_p$). We recall that a sequence $\langle x_1, \ldots, x_{l+1}\rangle$ is a permutation iff $\langle x_1, \ldots, x_l\rangle$ is linear and $x_1 = x_{l+1}$. The cycle unification problem

$$\langle Pxy \overset{\circ}{\longrightarrow} Pab\rangle_{\{Pvw \leftarrow Pwv\}}$$

defines the permutation $\langle w, v, w\rangle$ of length $l = 2$. If we solve

$$\mathcal{C}^0 = \{x \doteq v^1,\ y \doteq w^1,\ w^1 \doteq a,\ v^1 \doteq b\},$$

we obtain the solution $\tau_0 = \{x \mapsto b,\ y \mapsto a\}$. Considering one iteration through the cycle we have to solve

$$\mathcal{C}^1 = \{x \doteq v^1,\ y \doteq w^1,\ w^1 \doteq v^2,\ v^1 \doteq w^2,\ w^2 \doteq a,\ v^2 \doteq b\},$$

which results in $\tau_1 = \{x \mapsto a,\ y \mapsto b\}$. Two iterations through the cycle and solving

$$\mathcal{C}^2 = \{x \doteq v^1,\ y \doteq w^1,\ w^1 \doteq v^2,\ v^1 \doteq w^2,\ w^2 \doteq v^3,\ v^2 \doteq w^3,\ w^3 \doteq a,\ v^3 \doteq b\}$$

yields $\{x \mapsto b,\ y \mapsto a\} = \tau_0$. Thus, we periodically return to previously computed solutions. In general, we have only to consider the unifier of G and F , if it exists, and finitely many iterations through the cycle to obtain all possible most general solutions for a cycle unification problem in the class of permutations. Conversely, if neither G and F are unifiable nor any one of the sets $\mathcal{C}^i$, $0 \le i < l$, is solvable, then the cycle unification problem is unsolvable.

Permutations with Linear Entry-sequence ($\mathcal{C}_{pls}$). We recall that a permutation with linear entry-sequence has the form

$$\langle x_1, \ldots, x_l, x_{l+1}, \ldots, x_{l+m}, x_{l+1}\rangle$$

in which $\langle x_1, \ldots, x_{l+1}\rangle$ is a linear (entry-) sequence and $\langle x_{l+1}, \ldots, x_{l+m}, x_{l+1}\rangle$ is a permutation. Since the permutations with linear entry-sequence can be splitted into these two parts, their behaviour is determined as a combination of these parts. After $l-1$ iterations through the cycle the variables occurring in L^1 depend only on the (superscribed) variables $x_{l+1}, \ \ldots, \ x_{l+m}$. The values for $x_{l+1}, \ \ldots, \ x_{l+m}$ are solely determined by the permutation $\langle x_{l+1}, \ldots, x_{l+m}, x_{l+1}\rangle$ and the exit equations. As before, we have only to consider finitely many – viz. $m-1$ – further iterations to obtain all possible most general solutions for a cycle unification problem in the class $\mathcal{C}_{pls}$. Conversely, if neither G and F are unifiable nor any one of the sets $\mathcal{C}^i$, $0 \le i < l+m-1$, is solvable, then the cycle unification problem is unsolvable. We will give an example of $\mathcal{C}_{pls}$ at the end of Section 4.1.

Variable-pure Matching Cycles ($\mathcal{C}_{vp}$). Variable-pure matching cycles define combinations of permutations, permutations with linear entry-sequence, and maximal linear sequences. Let a variable-pure matching cycle define p permutations

$$\langle x_{1,i},\ \ldots,\ x_{m_i,i},\ x_{1,i}\rangle,\ 1 \leq i \leq p,$$

pl permutations with linear entry-sequence

$$\langle y_{1,i},\ \ldots,\ y_{l_i,i},\ y_{l_i+1,i},\ \ldots,\ y_{l_i+n_i,i},\ y_{l_i+1,i}\rangle,\ 1 \leq i \leq pl,$$

and l maximal linear sequences

$$\langle z_{1,i}, \ldots, z_{\tilde{l}_i+1,i}\rangle,\ 1 \leq i \leq l.$$

Let $M = \max(1, l_1, \ldots, l_{pl}, \tilde{l}_1, \ldots, \tilde{l}_l)$ and $N = \operatorname{lcm}(1, m_1, \ldots, m_p, n_1, \ldots, n_{pl})$, where lcm denotes the least common multiple.

As in the previous subsection we find that after $M{-}1$ iterations through the cycle the variables occurring in L^1 depend only on the (superscribed) variables occurring in the permutations – ie. in $\{x_{1,i},\ \ldots,\ x_{m_i,i} \mid 1 \leq i \leq p\}$.[3] – and that the values for the (superscribed) variables $x_{1,i},\ \ldots,\ x_{m_i,i}$, $1 \leq i \leq p$, are solely determined by the permutations $\langle x_{1,i}, \ldots, x_{m_i,i}, x_{1,i}\rangle$, $1 \leq i \leq p$, and the exit equations. As before, we have only to consider finitely many – viz. $N{-}1$ – further iterations to obtain all possible most general solutions for a cycle unification problem in the class $\mathcal{C}_{vp}$. More formally, we can show that for all $i \geq 0$ and $k > 0$ the claims

$$\mathcal{C}^{M-1+i} \text{ is solvable iff } \mathcal{C}^{M-1+i+k\cdot N} \text{ is solvable}$$

and

$$\tau_{M-1+i} \stackrel{\cdot}{\leq} \tau_{M-1+i+k\cdot N}$$

hold. One should observe that this result subsumes the result of linear sequences (where $N = 1$), of permutations (where $M = 1$) and of permutations with linear entry-sequence.

Let $\langle G \overset{\circ}{\longrightarrow} F\rangle_{\{L \leftarrow R\}}$ be a cycle unification problem. The steps in Figure 1 define a cycle unification algorithm for variable-pure matching cycles. Algorithms for $\mathcal{C}_{ls}$, $\mathcal{C}_p$, and $\mathcal{C}_{pls}$ are special cases. To illustrate the algorithm consider the cycle unification problem

$$\langle Pu_1u_2u_3u_4u_5 \overset{\circ}{\longrightarrow} Pabcde\rangle_{\{Pyzvzv \leftarrow Pxyzvw\}}.$$

We obtain the following steps.

1. $Pu_1u_2u_3u_4u_5$ and $Pabcde$ are unifiable by $\tau = \{u_1 \mapsto a,\ u_2 \mapsto b,\ u_3 \mapsto c,\ u_4 \mapsto d,\ u_5 \mapsto e\}$.

[3]Note, $\{\langle y_{l_i+1,i}, \ldots, y_{l_i+n_i,i}, y_{l_i+1,i}\rangle \mid 1 \leq i \leq pl\} \subseteq \{\langle x_{1,i}, \ldots, x_{m_i,i}, x_{1,i}\rangle \mid 1 \leq i \leq p\}$.

1. If G and F are unifiable, then compute τ as the most general unifier for G and F restricted to the variables in G.
2. If $\langle G \overset{\circ}{\longrightarrow} F\rangle_{\{L\leftarrow R\}} \in \mathcal{C}_{vp}$, then compute the lengths $l_1,\ \ldots,\ l_i$ of all defined maximal linear sequences/linear entry-sequences and the lengths $m_1,\ \ldots,\ m_j$ of all defined permutations. Let $M = \max(1, l_1, \ldots, l_i)$ and $N = \mathrm{lcm}(1, m_1, \ldots, m_j)$.
3. If $\mathcal{C}^k$ is solvable, then compute τ_k as the most general unifier for $\mathcal{C}^k$, restricted to the variables occurring in G, $0 \leq k \leq M+N-2$.
4. Let Σ be the set of solutions obtained in steps (1) and (3). If $\Sigma = \emptyset$, the problem is unsolvable. Otherwise, iteratively eliminate a substitution α if the current set of solutions contains another substitution δ with $\delta \underline{\lessdot} \alpha\ [\mathcal{V}ar(G)]$. The obtained set is a minimal and complete set of solutions for the cycle unification problem $\langle G \overset{\circ}{\longrightarrow} F\rangle_{\{L\leftarrow R\}}$.

Figure 1: A Unification Algorithm for $\mathcal{C}_{vp}$.

2. $\sigma = \{x \mapsto y,\ y \mapsto z,\ z \mapsto v,\ v \mapsto z,\ w \mapsto v\}$ is a most general right matcher of *Pxyzvw* against *Pyzvzv*. P_σ defines two permutations with linear entry-sequence, viz. $\langle x, y, z, v, z\rangle$ and $\langle w, v, z, v\rangle$. Hence, $M = 2,\ N = 2$ and the problem is in $\mathcal{C}_{vp}$.
3. $\tau_0 = \{u_1 \mapsto b,\ u_2 \mapsto c,\ u_3 \mapsto d,\ u_4 \mapsto c,\ u_5 \mapsto d\}$, $\tau_1 = \{u_1 \mapsto c,\ u_2 \mapsto d,\ u_3 \mapsto c,\ u_4 \mapsto d,\ u_5 \mapsto c\}$, $\tau_2 = \{u_1 \mapsto d,\ u_2 \mapsto c,\ u_3 \mapsto d,\ u_4 \mapsto c,\ u_5 \mapsto d\}$ are the most general solutions obtained by solving $\mathcal{C}^0$, $\mathcal{C}^1$, and $\mathcal{C}^2$, restricted to $\{u_1,\ u_2,\ u_3,\ u_4,\ u_5\}$, respectively.
4. We obtain the set $\{\tau,\ \tau_0,\ \tau_1,\ \tau_2\}$ as a minimal and complete set of solutions.

Theorem 2 *Let* C *be a variable-pure matching cycle.*

(a) $\langle G \xrightarrow[C]{\circ} F\rangle$ *is decidable.*

(b) $\langle G \xrightarrow[C]{\circ} F\rangle$ *is finitary.*

(c) *There exists an algorithm computing a minimal and complete set of solutions for* $\langle G \xrightarrow[C]{\circ} F\rangle$ *.*

One should observe that this result holds for $\mathcal{C}_{ls}$, $\mathcal{C}_{p}$, and $\mathcal{C}_{pls}$ as well because they are subsets of $\mathcal{C}_{vp}$.

4.2 Non-recursive Matching Cycles ($\mathcal{C}_{nr}$)

In the preceding subsection we have considered only variable-pure matching cycles. We will now lift this restriction by considering matching cycles in which the matching substitution may bind variables to terms including function-symbols and constants. As an example consider the cycle unification problem $\langle Py \xrightarrow{\circ} Pa\rangle_{\{Pfx \leftarrow Px\}}$, which has an infinite, complete and minimal set $\{\{y \mapsto a\}, \{y \mapsto fa\}, \{y \mapsto ffa\}, \ldots\}$ of solutions. Since we do not yet know how to control such cycles, we restrict ourselves and exclude bindings like $x \mapsto fx$, as it is contained in the matcher of the example. A variable x is called *recursive* iff there exists an i, $i > 0$, such that $x \mapsto t \in \sigma^i$, $x \in \mathcal{V}ar(t)$, and $t \neq x$.[4] Because $t \neq x$, variables occurring in permutations are not recursive. However, the variable x in the example above is recursive. It is easy to verify that it is decidable whether σ contains recursive variables by considering σ^j, $1 \leq j \leq k$, where k is the number of bindings in σ . A cycle $\{L \leftarrow R\}$ is called *non-recursive matching* iff the domain of the matcher does not contain any recursive variable. The class of non-recursive matching cycle unification problems is denoted by $\mathcal{C}_{nr}$.

One should observe that if σ is a matching substitution for a cycle in $\mathcal{C}_{nr}$, then P_σ may contain pairs (x,t) , where t is not a variable. To be able to deal with those pairs, we have to extend the definition of sequences. A set of variable-term pairs P defines a *sequence* as follows. If $(x,t) \in P$ and t is ground, then P defines $\langle x,t\rangle$. If $(x,t) \in P$ and $\mathcal{V}ar(t) = \{x_1,\ldots,x_n\}$, then P defines $\langle x,x_1\rangle$, $\ldots$, $\langle x,x_n\rangle$. Let $\langle x_1,\ldots,x_n\rangle$ be a sequence defined by P and $(x_n,t) \in P$. If t is ground, then P defines $\langle x_1,\ldots,x_n,t\rangle$. If $\mathcal{V}ar(t) = \{y_1,\ldots,y_n\}$, then P defines $\langle x_1,\ldots,x_n,y_1\rangle$, $\ldots$, $\langle x_1,\ldots,x_n,y_n\rangle$. A sequence $\pi = \langle x_1,\ldots,x_l,t\rangle$ is called *linear* iff each variable occurs at most once in π . One should observe that since non-recursive cycles do not contain recursive variables, a (superscribed) variable x occurring in a permutation cannot be mapped on a term t such that $x \in \mathcal{V}ar(t)$ and $t \neq x$. Thus, permutations are constructed from pairs of variables in P_σ only.

[4]By σ^i we denote the i–fold composition of σ with itself, ie. $\sigma^1 = \sigma$ and $\sigma^i = \sigma(\sigma^{i-1})$.

It is easy to see, that we can allow linear sequences ending in a ground term without changing our previous results. Let a non-recursive matching cycle define p permutations, pl permutations with linear entry-sequence, and l maximal linear sequences possibly ending in a ground term. Furthermore, let M and N be defined as in the previous subsection. As before we obtain

$$\mathcal{C}^{M-1+i} \text{ is solvable iff } \mathcal{C}^{M-1+i+k\cdot N} \text{ is solvable}$$

and

$$\tau_{M-1+i} \overset{\bullet}{\leq} \tau_{M-1+i+k\cdot N}$$

for $i \geq 0$ and $k > 0$. Thus, the results for variable-pure cycle unification problems can be generalized to non-recursive matching cycle unification problems.

The unification algorithm for non-recursive matching cycles is analogous to the algorithm for variable-pure matching cycles but with the extended definition of sequences. As an example consider the cycle unification problem

$$\langle Pu_1u_2u_3u_4 \;\overset{\circ}{\longrightarrow}\; Pabcd\rangle_{\{Pfyz,vwz \leftarrow Pxyzw\}}.$$

An application of the algorithm yields the following results.

1. $Pu_1u_2u_3u_4$ and $Pabcd$ are unifiable by $\tau = \{u_1 \mapsto a,\ u_2 \mapsto b,\ u_3 \mapsto c,\ u_4 \mapsto d\}$.

2. $\sigma = \{x \mapsto fyz,\ z \mapsto w,\ w \mapsto z,\ y \mapsto v\}$ is a most general right matcher of $Pxyzw$ against $Pfyz,vwz$. P_σ defines the sequences $\langle x, y, v\rangle$ and $\langle x, z, w, z\rangle$ such that $M = 2,\ N = 2$ and the problem is in $\mathcal{C}_{nr}$ because the cycle's matcher does not contain any recursive variable.

3. $\tau_0 = \{u_1 \mapsto fbc,\ u_2 \mapsto v^1,\ u_3 \mapsto d,\ u_4 \mapsto c\}$, $\tau_1 = \{u_1 \mapsto fv^2d,\ u_3 \mapsto c,\ u_4 \mapsto d\}$, $\tau_2 = \{u_1 \mapsto fv^2c,\ u_3 \mapsto d,\ u_4 \mapsto c\}$ are the most general solutions obtained by solving $\mathcal{C}^0, \mathcal{C}^1$, $\mathcal{C}^2$, restricted to $\{u_1,\ u_2,\ u_3,\ u_4\}$, respectively.

4. We obtain the set $\{\tau,\ \tau_1,\ \tau_2\}$ as a minimal and complete set of solutions.

Theorem 3 *Statements (a), (b), and (c) of Theorem 2 hold also for non-recursive matching cycles.*

5 Summary and Future Work

In this paper we have formally defined cycle unification (for a restricted class of formulas). We have considered various classes of cycle unification problems with increasing complexity, have shown that they are decidable and finitary, and have specified a minimal and complete unification algorithm for these classes. Table 1 gives an overview of these results as well as of previous work. In each row

Class	Decidability	Type	Algorithm	References
$\mathcal{C}$	open	infinitary	open	
$\mathcal{C}_l$	decidable	infinitary	open	[7]
$\mathcal{C}_g$	decidable	unitary	yes	[16]
$\mathcal{C}_m$	open	infinitary	open	
$\mathcal{C}_{nr}$	decidable	finitary	yes	this paper
$\mathcal{C}_u$	decidable	finitary	yes	[22]

Table 1: Properties of cycle unification classes.

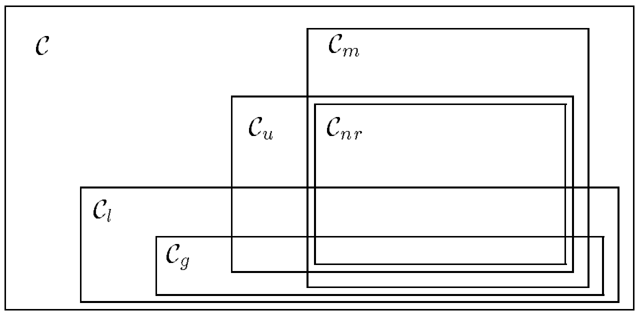

Figure 2: The relation between the classes $\mathcal{C}$, $\mathcal{C}_l$, $\mathcal{C}_g$, $\mathcal{C}_m$, $\mathcal{C}_{nr}$, and $\mathcal{C}_u$.

we state the decidability and the unification type for a particular class of cycle unification problems, indicate whether there exists an algorithm to compute a minimal and complete set of solutions, and provide the reference if there exists one. $\mathcal{C}$ denotes the class of unrestricted cycle unification problems. In $\mathcal{C}_l$ and $\mathcal{C}_g$ goals and facts are restricted to be linear and ground, respectively. $\mathcal{C}_u$ denotes the class of *unifying* cycles, ie. cycles $L \leftarrow R$, for which L and R are unifiable. This class has recently been investigated in [22]. For the definition of the more complicated classes $\mathcal{C}_m$ and $\mathcal{C}_{nr}$ the reader is referred to Section 4. The various classes are related as shown in Figure 2.

One might think that our results for the class $\mathcal{C}_{nr}$ may easily be extended for cycle unification problems $\langle G \xrightarrow{\circ} F\rangle_{\{L \leftarrow R\}}$ such that there exist two non-recursive substitutions σ and τ with $\sigma L = \tau R$. But this is not true. For the cycle $\{Px \leftarrow Pfx\}$ we find $\{x \mapsto fy\}Px = \{x \mapsto y\}Pfx$. Yet, as shown in Subsection 4.2 there are infinitely many independent solutions for this problem. In the case $\{Pfx{,}z \leftarrow Pyfx\}$ we find $\{z \mapsto fx\}Pfx{,}z = \{y \mapsto fx\}Pyfx$ and one might expect that only a single instance of the cycle is needed. However, as shown in [22], the latter example belongs to a class of *unifying* cycles, ie. cycles

$L \leftarrow R$ for which there exists a substitution σ with $\sigma L = \sigma R$, which need one cycle iteration, ie. two instances of the clause.

One of the major open problems in our restricted context is the question whether $\mathcal{C}$ is decidable. $\mathcal{C}_l$, $\mathcal{C}_u$, and $\mathcal{C}_{nr}$ are decidable. However, there are several results which point into the opposite direction for the case of $\mathcal{C}$. In [5] it is shown that the termination of a one rule term rewriting system, where rewriting may occur at proper subterms, is undecidable. Similarly, we know from [16] that the class of Horn clauses consisting of two clauses of the form $L \leftarrow R$ and two ground unit clauses is undecidable. It is, however, not obvious, how these result could be adapted to cycle unification problems.

In the future we intend to develop heuristics to control further classes of cycle unification problems. We are looking for a well–founded ordering based on a measure of complexity for the instances of the cycle in order to apply an idea similar to the one contained in [16]. Certain cycles $L \leftarrow R$ cause some of the terms occurring in L and R to grow or shrink monotonically at each iteration of the cycle. If there were an upper bound for these terms defined by G or F, then one would be able to decide the cycle unification problem $\langle G \xrightarrow{\circ} F\rangle_{\{L \leftarrow R\}}$. For illustration of this idea consider the cycle unification problem

$$\langle Pffx,x \xrightarrow{\circ} Pufu\rangle_{\{Pfffy,fz \leftarrow Pfy,z\}}.$$

The i-th instance of the right-hand side of the cycle $Pfffy,fz \leftarrow e\,Pfy,z$ is matched against the $i+1$-st instance of the left-hand side by $\sigma_i = \{y^i \doteq ffy^{i+1},\ z^i \doteq fz^{i+1}\}$. We observe that the depth of y and z decreases with each iteration through the cycle and the goal as well as the fact define upper bounds because the multiple occurrences of x and u correlate y and z via the entry- and exit equations. In [4] we have exploited this insight for the computation of the number k of iterations through the cycle to obtain a solution. For the example we obtain $k = 2$ and the solution $\tau_2 = \{x \mapsto f^5y^3\}$.

As shown in Section 3, there might be infinitely many independent solutions of a cycle unification problem. Hence, we need a compact representation of infinitely many terms for such cases and intend to use the one suggested in [11].

In order to solve the Łukasiewicz formula mentioned in the introduction as a challenge problem, the results obtained so far and in the future need to be generalized to the case of more than one interacting cycles. [3] contains first ideas how this might be achieved. Altogether there is quite a bit of work ahead of us until this challenge problem might be solved in less than a second as thought possible in [3].

As an example of an application of cycle unification in logic programming mentioned in the introduction consider the cycle $max(X,Y,Z) :- max(Y,X,Z)$ in Prolog notation. It is contained in a clause set representing SAM's lemma [21]. The cycle expresses the commutativity of the first two arguments of the maximum-predicate. This clause may be used in a Prolog program which computes the maximum of two numbers.

$$\begin{array}{lll} max(X,Y,Y) & :- & X =< Y. \\ max(X,Y,Z) & :- & max(Y,X,Z). \end{array}$$

If we ask the query $?- \; max(2,3,V)$, PROLOG yields the desired result and V is bound to 3. But if we ask for all solutions, the program does not terminate because the search space is infinite. Similarly, negative queries handled by negation-as-failure may not produce the expected results because of possibly infinite evaluation trees; for example consider the query $?- \; \sim max(2,3,2)$[5] leading to the subgoal $?- \; max(2,3,2)$.

With the tools of this paper it can be seen, however, that the cycle is a permutation of length 2. Therefore, it is sufficient to consider at most one self-application of the cycle to obtain all correct answers. As a result, the infinite search space collapses to a trivial one and the computation of the two program clauses becomes roughly equivalent with the following program.

$$\begin{array}{lll} max(X,Y,X) & :- & X > Y. \\ max(X,Y,Y) & :- & X =< Y. \end{array}$$

No query asked to this program will give rise to a non-terminating computation because all queries have a finite evaluation tree.

Acknowledgement: The third author was partially supported by the ESPRIT project MEDLAR and the Stadt Dreieich. We would like to thank Franz Baader and two anonymous referees. Their very valuable comments led to a significant improvement of the paper.

References

[1] W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, Braunschweig, 2 edition, 1987.

[2] W. Bibel. Advanced topics in automated deduction. In R. Nossum, editor, *Fundamentals of Artificial Intelligence II*, pages 41–59. Springer, LNCS *345*, 1988.

[3] W. Bibel. Perspectives on automated deduction. In R. S. Boyer, editor, *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 77–104. Kluwer Academic, Utrecht, 1991.

[4] W. Bibel, S. Hölldobler, and J. Würtz. Cycle unification. Technical Report AIDA-91-15, FG Intellektik, FB Informatik, TH Darmstadt, 1991.

[5] $\sim$ denotes the negation-as-failure.

[5] M. Dauchet. Simulation of a turing machine by a left-linear rewrite rule. In *Proceedings of the Conference on Rewriting Techniques and Applications*, pages 109–120. Springer, LNCS *355*, 1989.

[6] N. Dershowitz and J.-P. Jouannaud. Notations for rewriting. *EATCS Bulletin*, 43:162–172, 1991.

[7] P. Devienne. Weighted graphs: A tool for studying the halting problem and time complexity in term rewriting systems and logic programming. *Journal of Theoretical Computer Science*, 75:157–215, 1990.

[8] E. Eder. Properties of substitutions and unifications. *Journal of Symbolic Computation*, 1:31–46, 1985.

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

[10] J. Minker and J.-M. Nicolas. On recursive axioms in deductive databases. *Information Systems*, 8(1):1–13, 1983.

[11] H. J. Ohlbach. Abstraction tree indexing for terms. In L. C. Aiello, editor, *Proceedings of the European Conference on Artificial Intelligence*, pages 479–484, 1990.

[12] H. J. Ohlbach. Compilation of recursive two-literal clauses into unification algorithms. In P. Jorrand and V. Sgurev, editors, *Proceedings of the AIMSA*, pages 13–22, 1990.

[13] H. J. Ohlbach and G. Wrightson. Solving a problem in relevance logic with an automated theorem prover. In R. E. Shostak, editor, *Proceedings of the Conference on Automated Deduction*, pages 496–508. Springer, LNCS *170*, 1984.

[14] F. Pfenning. Single axioms in the implicational propositional calculus. In E. Lusk and R. Overbeek, editors, *Proceedings of the Conference on Automated Deduction*, pages 710–713. Springer, LNCS *310*, 1988.

[15] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, 1965.

[16] M. Schmidt-Schauß. Implication of clauses is undecidable. *Journal of Theoretical Computer Science*, 59:287–296, 1988.

[17] D. De Schreye, K. Verschaetse, and M. Bruynooghe. A practical technique for detecting non-terminating queries for a restricted class of horn clauses, using directed, weighted graphs. In *Proceedings of the International Conference on Logic Programming*, pages 649–663, 1990.

[18] J. H. Siekmann. Unification theory. *Journal of Symbolic Computation*, 7:207 – 274, 1989.

[19] M. E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasonsing*, 1:333–356, 1985.

[20] L. Vielle. Recursive query processing: The power of logic. Technical Report TR-KB-17, European Computer-Industry Research Center, 1987.

[21] L. Wos. The problem of finding a strategy to control binary paramodulation. *Journal of Automated Reasonsing*, pages 101–107, 1988.

[22] J. Würtz. Unifying cycles. Technical report, Deutsches Forschungszentrum für Künstliche Intelligenz, 1992. To appear.