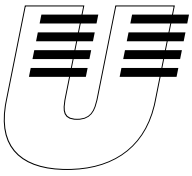


UNIVERSITÄT
KOBLENZ · LANDAU

A Confluent Connection Calculus

Peter Baumgartner, Norbert Eisinger,
Ulrich Furbach

23/98

Fachberichte
INFORMATIK

Universität Koblenz-Landau
Institut für Informatik, Rheinau 1, D-56075 Koblenz
E-mail: researchreports@infko.uni-koblenz.de,
WWW: http://www.uni-koblenz.de/fb4/

A Confluent Connection Calculus*

Peter Baumgartner
Universität Koblenz
peter@uni-koblenz.de

Norbert Eisinger
Universität München
Norbert.Eisinger@informatik.uni-muenchen.de

Ulrich Furbach
Universität Koblenz
uli@uni-koblenz.de

November 18, 1998

ABSTRACT

This work is concerned with basic issues of the design of calculi and proof procedures for first-order connection methods and tableaux calculi. Proof procedures for these type of calculi developed so far suffer from not exploiting proof confluence, and very often unnecessarily rely on a heavily backtrack oriented control regime.

As a new result, we present a variant of a connection calculus and prove its *strong* completeness. This enables the design of backtrack-free control regimes. To demonstrate that the underlying fairness condition is reasonably implementable we define an effective search strategy. We show that with the new approach the search space can be exponentially smaller than those of current, backtracking-oriented proof procedures based on *weak* completeness results.

1 INTRODUCTION

This work is concerned with basic issues of the design of calculi and proof procedures for first-order connection methods and tableaux calculi. Calculi we have in mind include connection calculi [Bibel, 1987; Eder, 1992], first-order clausal tableaux with rigid variables [Fitting, 1990], more recent developments like A-ordered tableaux [Klingenbeck and Hähnle, 1994; Hähnle and Klingenbeck, 1996], tableaux with selection function [Hähnle and Pape, 1997]. Let us refer to all these calculi by the term "rigid variable methods". Recently complexity issues for those kinds of calculi have been considered in [Voronkov, 1998; Voronkov, 1997].

We emphasise that in this paper we do *not* consider model elimination (ME) [Loveland, 1968]. Although ME can be presented as a tableau calculus [Letz *et al.*, 1994; Baumgartner and Furbach, 1993], it is *not* proof confluent. The same holds for Restart ME [Baumgartner *et al.*, 1997] and related methods. These calculi are not even proof-confluent at the propositional level and cannot be treated by the methods presented here. Nevertheless they are the basis of some high performance theorem provers like SETHEO [Letz *et al.*, 1992] or Protein [Baumgartner and Furbach, 1994].

*This article will be published in the book: Steffen Hölldobler (editor). *Intellectics and Computational Logic – Papers in Honor of Wolfgang Bibel*, Kluwer, 1999.

In this paper we propose a new technique for the design of proof procedures for rigid variable methods. The new technique is motivated by the desire to get more efficient proof procedures and implementations thereof than those which have been developed so far.

The proposed technique should also be applicable to calculi which *avoid* rigid variables in the first place, like SATCHMO [Manthey and Bry, 1988], MGTP [Fujita and Hasegawa, 1991], hyper tableaux [Baumgartner *et al.*, 1996] and ordered semantic hyper linking [Plaisted and Zhu, 1997]. Usually the price for getting around rigid variables in these approaches is that they involve some uninformed ground instantiation in special cases. These special cases may be irrelevant for most typical applications, but nevertheless these calculi are likely to profit from techniques enabling them to handle rigid variables as well.

Our new approach is based on the observation that current proof procedures for rigid variable methods follow the following weak completeness theorem for rigid variable methods:

> *Weak completeness:* a clause set S is unsatisfiable if and only if *there is* a derivation from S which is also a refutation.

The search space thus is the space of *derivations*; it requires a tentative control regime such as backtracking, which explores *all* possible derivations. Proof confluent calculi like the ones mentioned above, however, should admit a *strong* completeness theorem of the form:

> *Strong completeness:* a clause set S is unsatisfiable if and only if *every* (fair) derivation from S is a refutation.

Consequently, proof procedures following this theorem can do with an irrevocable control regime that needs to develop only *one single* derivation and may safely ignore alternatives as it proceeds. They can thus *reuse* information which would be lost in a backtracking intensive approach. Typically they enumerate *models* but not *derivations* (the hyper tableaux calculus [Baumgartner *et al.*, 1996] is an example that enumerates models, model elimination [Loveland, 1968] is an example for the enumeration of derivations).

Put abstractly, the source to gain efficiency is that there are usually many derivations for the *same* model, and all but one derivation can be avoided. Table 1 summarises the issues addressed so far.

In this paper, we will develop a strong completeness result for a modified connection calculus (the CCC calculus, Section 3) together with first steps towards a respective proof procedure.

This result closes a strange gap in Table 1: the connection calculus in [Bibel, 1987] is proof confluent on the propositional level, but its first-order version is not. This is the only calculus we are aware of having this property. Although other free-variable methods, such as the first-order tableaux calculus in Fitting's book [Fitting, 1990] *are* proof-confluent, they are *implemented* as if they were non-confluent. This yields unnecessary inefficiencies, and the main motivation for the work presented in this paper is to find a cure for them.

Calculus	*Completeness*	*Proof procedure*	*Enumeration of*
CCC Bibel's prop. CC Tableaux	Strong	No Backtracking	models
Bibel's first-order CC ME	Weak	Backtracking	derivations

Table 1: Properties of calculi. CCC is the new *confluent connection calculus* of Section 3. For Bibel's connection calculi (CC) we refer to [Bibel, 1987].

The rest of this paper is structured as follows: in the next section we will briefly summarise the idea behind connection and tableaux calculi. The problems with them will be illustrated using an example. Then, after having introduced some preliminaries, we define our new calculus CCC (confluent connection calculus), suggest a rather general search strategy and prove its fairness. The subsequent section contains the completeness proof. Finally we conclude the paper by indicating some future work.

2 CURRENT APPROACHES

Let us first briefly recall the basic idea of current proof procedures for rigid variable methods and identify the tackled source of inefficiency.

"Usual" proof procedures like the one used in the 3TAP prover [Hähnle *et al.*, 1994], the LeanTAP prover [Beckert and Posegga, 1995]), the connection procedure proposed in [Bibel, 1987], and the one in Fitting's Book [Fitting, 1990] follow an idea suggested by Prawitz [Prawitz, 1960] and can be seen as more or less direct translation of the following formulation of the Herbrand-Skolem-Gödel theorem (we can restrict our attention to clause logic here):

> A clause set S is unsatisfiable if and only if there is a finite set S' of variants of clauses of S and there is a substitution δ such that $S'\delta$ is unsatisfiable, where $S'\delta$ is viewed as a propositional clause set.

Now, in accordance with the theorem, proof procedures for rigid variable methods typically realise the following scheme to prove that given clause set $S = \{C_1, \ldots, C_n\}$ is unsatisfiable. Following Voronkov [Voronkov, 1998; Voronkov, 1997], we call it *The Procedure*:

PROCEDURE 2.1 (*The Procedure*)

(i) Let $\mu = 1$ called *multiplicity*.

(ii) Let $S^\mu = \{ C_1^1, \ldots, C_1^\mu$
$\vdots$
$C_n^1, \ldots, C_n^\mu \}$

be a set of pairwise variable-disjoint clauses, such that C_i^j is a variant of C_i for $1 \leq i \leq n$, $1 \leq j \leq \mu$. It is usual to call S^μ an *amplification* of S.

(iii) Check if there is a substitution δ such that $S^\mu\delta$ is propositionally unsatisfiable.

(iv) If such a δ exists, then stop and return "unsatisfiable"; otherwise let $\mu = \mu + 1$ and go to step (ii).

□

Completeness of *The Procedure* is achieved by, first, a fairness condition in the generation of the amplifications S^μ in step (ii), namely by uniformly taking $1, 2, 3, \ldots$ variants of every clause in S in round $1, 2, 3, \ldots$, and, second, by exhaustively searching for substitutions in step (iii).

2.1 CONNECTION METHODS

How is step (iii) in *The Procedure* realised? Our primary interest is in connection methods (also called matrix methods), hence we will briefly recall the idea: define a matrix to be any set of quantifier free formulae. For our purposes it suffices to consider clause sets only. Notice that S^μ is a matrix. A *path* through a matrix is obtained by taking exactly one literal from every clause. A connection is a pair of literals, which can be made complementary by application of a substitution; with each connection we associate a most general unifier σ achieving this. In order to realise step 3, proof procedures for connection calculi search for a substitution δ such that every path through $S^\mu\delta$ contains a pair of complementary literals (we say that δ *closes* S^μ). If such a δ exists S must be unsatisfiable: take some arbitrary ground instance of $S^\mu\delta$ and observe that this set is propositionally unsatisfiable, because any possible way to satisfy a conjunction of disjunctions is excluded by virtue of the complementary literals.

For δ, it is sufficient to search through the *finite* space of most general unifiers making literals along paths (or branches) complementary (this guarantees the termination of step (iii). See e.g. [Bibel, 1987] for a concrete procedure to decide if a δ exists which renders all paths complementary. For our purposes it suffices to rephrase the underlying idea: let $p_1, p_2, \ldots, p_n$ be any enumeration of all paths through the current amplification. It can be shown by usual lifting techniques that there is δ which simultaneously renders all paths as complementary if and only if there is a sequence $p_1\delta_1,\ p_2\delta_1\delta_2, \ldots, p_n\delta_1\cdots\delta_n$, where δ_i is a most general unifier associated with a connection in $p_i\delta_1\cdots\delta_{i-1}$. In other words, by defining $\delta := \delta_1\cdots\delta_n$ one recognises that δ can be computed incrementally. Notice, however, that the "there is" quantification is to be translated in a backtracking-oriented procedure.

EXAMPLE 2.2 (CONNECTION METHOD) Consider the following unsatisfiable clause set:

$$S = \{P(X) \vee Q(X),\ \neg P(c),\ \neg P(a) \vee \neg P(b),\ \neg Q(a),\ \neg Q(b)\}$$

We write a matrix as a vertical sequence of clauses. Hence S^1 and S^2 look like this[1]:

[1] In the literature, matrices are also written horizontally.

$$
\begin{array}{lcc}
 & & S^2\colon \quad P(X^1) \vee Q(X^1) \\
S^1\colon & P(X^1) \vee Q(X^1) & P(X^2) \vee Q(X^2) \\
 & \neg P(c) & \neg P(c) \\
 & \neg P(a) \vee \neg P(b) & \neg P(a) \vee \neg P(b) \\
 & \neg Q(a) & \neg Q(a) \\
 & \neg Q(b) & \neg Q(b)
\end{array}
$$

A path is obtained by traversing the matrix from top to bottom, picking up one literal from every clause.

The Procedure starts with $\mu = 1$, i.e. with S^1. By looking at the path through S^1 which passes through $P(X^1)$ one recognises that there are three candidate MGUs $\delta_1^1 = \{X^1/a\}$, $\delta_2^1 = \{X^1/b\}$ and $\delta_3^1 = \{X^1/c\}$. Since none of $S^1\delta_1^1$, $S^1\delta_2^1$ and $S^1\delta_3^1$ is propositionally unsatisfiable, *The Procedure* has to consider S^2. An incremental computation of a substitution δ which closes S^2 might proceed as follows: it starts by considering the connection $(P(X^1), \neg P(c))$ which results in $\delta_1^2 = \{X^1/c\}$. The next connection would be $(Q(X^2), \neg Q(a))$ with MGU $\delta_2^2 = \{X^2/a\}$. The combined substitution $\delta_1^2\delta_2^2 = \{X^1/c, X^2/a\}$ does not close the matrix, neither will $\delta_1^2\delta_3^2$, where $\delta_3^2 = \{X^2/b\}$. Hence, backtracking occurs until eventually the "right" substitution $\delta = \{X^1/a,\ X^2/b\}$ is computed, which closes S^2. □

2.2 PROOF SEARCH WITH TABLEAUX

Typically, the search for δ in step (iii) is organised as the construction of a free-variable tableau where a branch in a tableau stands for a path through the current amplification. According to Bibel [Bibel, 1987], any method which explores paths through a matrix in a systematical way qualifies as a connection calculi. Undoubtedly, this holds for tableaux calculi. Hence, tableaux calculi are connection calculi.

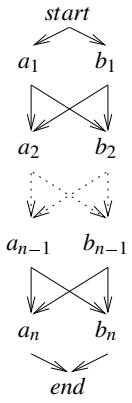
For space efficiency reasons it is prohibitive to explicitly represent in step (iii) all paths through S^μ in memory. For example, only 15 clauses consisting of 3 literals result in more than 14 million paths. A respective tableau thus has in the worst case the same number of branches. Hence, one possibility for step (iii) is to only keep in memory one path p (or branch in a tableau) at a time. A closing substitution δ_p is guessed, and if δ_p cannot be extended to a substitution δ which simultaneously closes all paths, then backtracking occurs and a different candidate δ_p is guessed. If all that fails, then a completely new tableau construction is started when entering step (iii) in the next round.

We emphasise that this is a common idea in all proof procedures for free-variable tableaux and connection calculi we are aware of. Let us refer to any such instance of *The Procedure* as a "tableau procedure". We are aware that there are numerous improvements; which, however will not be considered in the present paper, since we are interested in a basic weakness which is intrinsic to the tableau procedure.

2.3 DRAWBACK OF CURRENT APPROACHES

The tableau procedure from the previous section suffers from an unnecessarily large search space. For illustration, and in order to contrast it to our approach below, take the following example:

EXAMPLE 2.3 (INCESTUOUS GRAPH) Consider the graph in Figure 1. All edges shall be directed from top to bottom. The graph illustrates in an obvious way the clause set S to the right of it, which is written in the usual notation for Horn clauses. The R-clauses[2] denote the edges between successive layers, and the RT-clauses define the transitive closure of the R relation. Since the *end* node is reachable from the *start* node, the clause $RT(start, end)$ is a logical consequence of S, in other words, $S \cup \{\neg RT(start, end)\}$ is unsatisfiable. Notice that the length of any path through the graph from *start* to *end* is



Edges:

$$
\begin{array}{rrrr}
R(start, a_1) \leftarrow & R(start, b_1) \leftarrow & & \\
R(a_1, a_2) \leftarrow & R(a_1, b_2) \leftarrow & R(b_1, a_2) \leftarrow & R(b_1, b_2) \leftarrow \\
\vdots & \vdots & \vdots & \vdots \\
R(a_{n-1}, a_n) \leftarrow & R(a_{n-1}, b_n) \leftarrow & R(b_{n-1}, a_n) \leftarrow & R(b_{n-1}, b_n) \leftarrow \\
R(a_n, end) \leftarrow & R(b_n, end) \leftarrow & &
\end{array}
$$

Transitive closure:

$$
\begin{array}{ll}
RT(X, Y) \leftarrow R(X, Y) & \text{(RT-1)} \\
RT(X, Z) \leftarrow RT(X, Y) \wedge R(Y, Z) & \text{(RT-2)}
\end{array}
$$

Figure 1: Incestuous graphs.

$n+1$, but that there are 2^n, i.e. exponentially many, different paths from *start* to *end*. However, even a naive procedure which successively marks reachable nodes (in a greedy way) beginning with *start* would terminate after $O(n)$ steps with every node (and hence also *end*) as marked. Notably, bottom-up evaluation with hyper resolution [Robinson, 1965] or related calculi like hyper tableau [Baumgartner *et al.*, 1996] or PUHR tableaux [Bry and Yahya, 1996] would exactly achieve this when applied to the clause set[3]. □

The tableau procedure performs very poorly on this example. We will discuss why. In brief, the tableau procedure enumerates *proofs*, and there are exponentially many proofs that $RT(start, end)$ is a logical consequence of S, whereas a better approach here is to enumerate *models*, and there is only one model of S (which also contains $RT(start, end)$).

[2] By an X-clause we mean any clause whose predicate symbol of the head literal is X.

[3] It is well-known that in general the time complexity to compute reachability in graphs is $O(n^3)$ (Warshall-algorithm).

In order to derive $RT(start, end)$, the procedure needs at least $\mu_n \in O(n)$ variants of RT-clauses. This is because with k variants of RT-clauses not more than k nodes reachable in the graph from *start* are contained in that subset of the RT-relation which is a logical consequence of $S^k\delta$ (for any δ). In other words, $O(n)$ variants of RT-clauses are needed to derive $RT(start, end)$.

More explicitly, consider the following set (RT-2)^g of ground instances of (RT-2):

$$\left[\begin{array}{l} RT(start,end) \leftarrow RT(start,a_n) \wedge R(a_n,end) \\ RT(start,end) \leftarrow RT(start,b_n) \wedge R(b_n,end) \end{array}\right]$$

$$\left[\begin{array}{l} RT(start,a_n) \leftarrow RT(start,a_{n-1}) \wedge R(a_{n-1},a_n) \\ RT(start,a_n) \leftarrow RT(start,b_{n-1}) \wedge R(b_{n-1},a_n) \end{array}\right] \quad \left[\begin{array}{l} RT(start,b_n) \leftarrow RT(start,a_{n-1}) \wedge R(a_{n-1},b_n) \\ RT(start,b_n) \leftarrow RT(start,b_{n-1}) \wedge R(b_{n-1},b_n) \end{array}\right]$$

$$\vdots \qquad\qquad \vdots$$

$$\left[\begin{array}{l} RT(start,a_2) \leftarrow RT(start,a_1) \wedge R(a_1,a_2) \\ RT(start,a_2) \leftarrow RT(start,b_1) \wedge R(b_1,a_2) \end{array}\right] \quad \left[\begin{array}{l} RT(start,b_2) \leftarrow RT(start,a_1) \wedge R(a_1,b_2) \\ RT(start,b_2) \leftarrow RT(start,b_1) \wedge R(b_1,b_2) \end{array}\right]$$

We want to identify a *minimal* subset X of (RT-2)^g such that X together with (sufficiently many) ground instances of (RT-1) plus the other facts is unsatisfiable. To do so, proceed as follows: choose one of the two clauses in the topmost group (a "group" is denoted by enclosing its clauses in brackets). Depending on the RT-literal in the body, walk down to the respective left or right group below it. Choose any of the two clauses in that group. Proceed as in the first step, and go on until a bottommost group is reached.

This selection determines a minimal set X as claimed: obviously, we have to make all n selections, because otherwise some RT-literal in the body of some clause would remain unprovable. But these n selections suffice, because the RT-literal in the body of the clause in the bottommost group can be reduced by an instance of the (RT-1) clause to an R-literal. This leaves us only with R-literals, which all are given as facts.

Of course, (RT-2)^g is not the full set of ground instances of (RT-2). All other ground instances of (RT-2) are similar to (RT-2)^g and they express the relations $RT(a_1, end)$, $RT(b_1, end)$, $RT(a_2, end)$, ... instead of $RT(start, end)$. It is easy to figure out that this full set of ground instances of (RT-2) comprises $O(n^3)$ clauses.

The just sketched procedure allows to identify $O(2^n)$ different sets X because in each group 2 selections are possible. Hence, for *every* path from *start* to *end* there is a corresponding amplification S^μ of S and there is a ground substitution γ such that the path from *start* to *end* is represented in $S^\mu\gamma$ by respective instances of the RT-clauses. That is, there are exponentially many ways to instantiate S^μ so that $RT(start, end)$ is a logical consequence of the instantiated set.

This indicates the weakness with the tableau procedure; the general problem with the tableau procedure is that too much information is lost during computation and so is computed over and over again. There are two phenomena:

LOSS IN INNER LOOP: Suppose S^μ as given, for some μ, and we are in step (iii) of *The Procedure*. Recall from above that the tableau procedure searches through the space of candidate substitutions δ such that $S^\mu\delta$ is propositionally unsatisfiable. As concluded above, there are $O(2^n)$ different paths from *start* to *end* and each of these is represented by $S^n\gamma$ for some γ. If e.g. $\mu = n - 1$ there still are $O(2^n)$

different paths, and, again, each of these is represented by a ground instance of S^μ. Further, each of these ground instances is obtainable as some instance of $S^\mu\delta$ which is generated in step (iii) as a solution candidate. None of these, however, will lead to a proof, because μ is not sufficiently large. Thus, there is exponential search space within step (iii).

LOSS IN OUTER LOOP: When returning from step (iv) to step (ii), the set of paths through S^μ in round μ is a strict subset of the set of paths through $S^{\mu+1}$. Since all substitutions δ searched for S^μ are lost on backtracking, each of these will be computed again for $S^{\mu+1}$.

We note that keeping all the substitutions δ for S^μ when exploring $S^{\mu+1}$ would be no solution, as there are exponentially many.

For further illustration of this drawback think of a respective control regime within a level-saturation resolution proof procedure: after unsuccessful termination of level n and on entering level $n+1$ one would *delete* all clauses derived in level n and start with the plain input set again! Probably no-one would seriously consider this.

On the other hand, the outer loop on μ in *The Procedure* is a form of iterative deepening, and it is well-known that, in the limit, iterative deepening has the same search space as a procedure which would instantly guess the "right" μ. Consequently, the true problem lies in the loss of information in the inner loop.

To sum up, there are possibly exponentially many ways to compute the same information. In our example, there are $O(2^k)$ ways to prove that $RT(start, a_k)$ holds, and all of these will be computed. In other words, there is a large potential for *reuse* which is not exploited.

We thus now turn to our confluent connection calculus which avoids the problem. Like the mentioned bottom-up evaluation with hyper resolution, it will solve the incestuous graph problem in polynomially many inference steps (see the remarks in the conclulsions).

3 A CONFLUENT CONNECTION CALCULUS

In this section we indroduce the confluent version of a connection calculus. For this we briefly have to set up the usual prerequisites. After indroducing the calculus together with an abstract notion of fairness, we show how this fairness can be realised and finally we prove strong completeness.

3.1 PRELIMINARIES

For a literal L we denote by $|L|$ the atom of L, i.e. $|A| = A$ and $|\neg A| = A$ for any atom A. Literals K and L are *complementary* iff K and L have different sign and $|L| = |K|$.

By var(*object*) we denote the set of variables occurring in *object*, where *object* is a term, a literal, a clause or a set of one of these. Our notion of a *substitution* is the

usual one [Baader and Schulz, 1998], and, as usual, we confuse a substitution with its homomorphic extension to terms, literals, clauses and clause sets. For a substitution σ, we denote by $\mathrm{dom}(\sigma)$ the (finite) set $\{X \mid X\sigma \neq X\}$, by $\mathrm{cod}(\sigma)$ the set $\{X\sigma \mid X \in \mathrm{dom}(\sigma)\}$ and by $\mathrm{vcod}(\sigma)$ the set $\mathrm{var}(\mathrm{cod}(\sigma))$. Substitution γ is a *ground* substitution iff $\mathrm{vcod}(\gamma) = \emptyset$; it is a *ground substitution for* X iff additionally $\mathrm{var}(X) \subseteq \mathrm{dom}(\gamma)$. A substitution σ is idempotent, if $X\sigma\sigma = X\sigma$ for each variable X. This is the case iff $\mathrm{dom}(\sigma) \cap \mathrm{vcod}(\sigma) = \emptyset$.

A clause is a finite disjunction of literals. In the following, S is the given finite input clause set, and M is a matrix for S, i.e. a set of clauses, each of which is an instance of a clause from S. It is worth emphasizing that in a matrix we consider clauses *not* to be individually universally quantified. That is, all variables are free, and thus applying a substitution, say, $\{Y/a\}$ to the matrix $\{\neg P(Y), P(Y) \vee Q(Y)\}$ would affect all occurrences of Y.

By $\mathsf{new}(S)$ we denote any matrix $\{D_1, \ldots, D_n\}$ such that D_i is a variant of C_i ($1 \leq i \leq n$), and each D_i is variable disjoint with all other D_j and with all clauses in the matrix given by the context.

A *connection with substitution* σ is a pair of literals (L, K) such that $L\sigma$ and $K\sigma$ are complementary. A *connection* is a pair of literals that is a connection with some substitution. In these definitions, we replace "substitution σ" by "MGU σ" only if additionally $\sigma = \mathrm{unify}(\{|K|, |L|\})$, where unify returns any most general unifier of its argument. Below we make use of the following assumption:

ASSUMPTION 3.1 *If a set Q of atoms is unifiable, then* $\mathrm{unify}(Q)$ *returns an idempotent most general unifier* σ *with (i)* $\mathrm{dom}(\sigma) \subseteq \mathrm{var}(Q)$ *and (ii)* $\mathrm{vcod}(\sigma) \subseteq \mathrm{var}(Q)$.

Notice that this is a very mild assumption: (i) says that σ operates on the variables of M only, and (ii) says that σ must not introduce new variables. Clearly, this is satisfied by any "standard" unification procedure.

A *path* through a matrix M is a set of literals, obtained by taking exactly one literal from each clause of M. A path is *closed* iff it contains a pair of complementary literals, otherwise it is *open*. A matrix is *open* iff there is a (at least one) path through it that is open, otherwise it is *closed*.

Notice that there is exactly one path through the "empty" matrix $\{\}$, which is the empty set $\{\}$ of literals; notice further that this path is open. On the other hand, if a matrix M contains the empty clause, then there is no path through M, in particular no open path, and the matrix is closed.

3.2 THE CALCULUS

We are going to define the calculus *CCC* (Confluent Connection Calculus). The constituents are the inference rules, the notion of a derivation, and a fairness condition for derivations.

DEFINITION 3.2 (CCC INFERENCE RULES AND DERIVATION)
The inference rule *variant step on M* is defined as follows:

$$\frac{M}{M \cup \mathsf{new}(S)}$$

The inference rule *connection step on* (K,L) *in* M is defined as follows:

$$\frac{M}{M \cup M\sigma}$$

if

1. there are clauses $C \in M$ and $D \in M$ such that $C = K \vee R_K$, $D = L \vee R_L$, for some clauses R_K and R_L, and
2. $\{K,L\} \subseteq p$ for some open path p through M, and
3. (K,L) is a connection with MGU σ.

The set $(M \cup M\sigma) \setminus M$ is called the set of *new clauses* (of this inference step). If conditions 1 to 3 above hold for M and (K,L), we say that a connection step is *applicable* to (K,L) in M or that (K,L) is a *candidate* for a connection step in M.

We say that a connection step on (K,L) in M is *progressive* if $M \cup M\sigma \neq M$ (i.e. at least one clause in the conclusion is new). If $M \cup M\sigma = M$ we say that it is *non-progressive*.

Note that a connection step on a connection (K,L) with $|K| = |L|$ is impossible and therefore neither progressive nor non-progressive – any path containing $\{K,L\}$ would be closed, contradicting condition 2. above.

Any sequence

$$D \quad = \quad (\{\} = M_0), M_1, \ldots, M_n, \ldots$$

is called a *derivation from S*, provided that M_{i+1} is obtained from M_i, which is open, by a single application of one of the inference rules (for $i \geq 0$). A *refutation* is a derivation that contains (i.e. ends in) a closed matrix. □

Notice that we do not have inference rules that delete clauses. Thus, every derivation has the following *chain property*:

$$(\{\} = M_0) \subseteq M_1 \subseteq \cdots \subseteq M_n \subseteq \cdots$$

The inclusions are not strict, because non-progressive steps are allowed as well (though they are not needed at all).

3.3 EXAMPLE

Suppose the given input clause set is $S = \{\neg P(X),\ P(Y) \vee Q(Y),\ \neg Q(Z)\}$. Clearly, S is unsatisfiable. We develop a refutation.

We have to start with the empty matrix M_0. Only a variant step can be applied to M_0, hence M_1 is just a copy of S. For each matrix M_i we discuss the possibilities for one open path through M_i, which is indicated by underlining.

M_1:

$$\underline{\neg P(X)} \qquad (C_1)$$

$$\underline{P(Y)} \vee Q(Y) \qquad (C_2)$$

$$\underline{\neg Q(Z)} \qquad (C_3)$$

The underlined path is open, and we carry out a connection step on $(\neg P(X), P(Y))$. Suppose that unify returns $\sigma_1 = \{Y/X\}$. This step is progressive and results in the following matrix[4]:

M_2:

$$\underline{\neg P(X)} \qquad (C_1)$$

$$P(X) \vee \underline{Q(X)} \qquad (C_2\sigma_1)$$

$$\underline{\neg Q(Z)} \qquad (C_3)$$

$$P(Y) \vee \underline{Q(Y)} \qquad (C_2)$$

Now there are two connections in the underlined open path to which a connection step is applicable: $(Q(X), \neg Q(Z))$ and $(\neg Q(Z), Q(Y))$. By applying a connection step to the former we would obtain a closed matrix, thus ending the refutation. In order to make the example more illustrative, let us instead apply a connection step to $(\neg Q(Z), Q(Y))$ with MGU $\sigma_2 = \{Y/Z\}$. This gives us:

M_3:

$$\underline{\neg P(X)} \qquad (C_1)$$

$$P(X) \vee \underline{Q(X)} \qquad (C_2\sigma_1)$$

$$\underline{\neg Q(Z)} \qquad (C_3)$$

$$\underline{P(Z)} \vee Q(Z) \qquad (C_2\sigma_2)$$

$$P(Y) \vee \underline{Q(Y)} \qquad (C_2)$$

Here the underlined path offers three possibilities. First, the connection $(Q(X), \neg Q(Z))$ is still a candidate for a connection step, but we disregard it for the same reason as in the previous matrix. Second, the connection step on $(\neg Q(Z), Q(Y))$ with MGU $\{Y/Z\}$ would not be progressive and therefore not interesting (if unify returned $\{Z/Y\}$ instead,

[4] In writing down the matrices, we use the strategy to apply the substitution "in place" to the old matrix, and append the uninstantiated versions of the clauses affected by the substitution at the bottom. This suggests that a tableau procedure can be defined as a specific variant of CCC. We plan to investigate this in the future.

the step would be progressive, though). Third, a connection step on $(\neg P(X), P(Z))$ is progressive no matter which of the two MGUs unify returns. Suppose that this step is applied with $\sigma_3 = \{Z/X\}$:

$$
\begin{array}{lll}
M_4\text{:} & \neg P(X) & (C_1) \\
& P(X) \vee Q(X) & (C_2\sigma_1) \\
& \neg Q(X) & (C_3\sigma_3) \\
& P(X) \vee Q(X) & (C_2\sigma_2\sigma_3) \\
& P(Y) \vee Q(Y) & (C_2) \\
& \neg Q(Z) & (C_3) \\
& P(Z) \vee Q(Z) & (C_2\sigma_2)
\end{array}
$$

Note that $P(X) \vee Q(X)$ occurs twice in this listing. Since matrices are sets, deleting one of these occurrences would represent the same set.

M_4 is closed because either path through the first three clauses alone is closed. Hence we have found a refutation.

3.4 FAIRNESS

Control strategies for any kind of rule-based system usually depend on some notion of fairness, when several rules or rule instances are applicable and one of them has to be selected for the next step. In matrix M_2 above, two connections were candidates for progressive connection steps, and one of them was selected to derive M_3, while the other was disregarded. In M_3 the disregarded connection was again a candidate for a progressive connection step and was again disregarded. Fairness is a condition on the choices made by the strategy to prevent that an applicable step is disregarded forever.

In many cases, especially with control strategies for logical calculi, fairness can be defined very simply as *exhaustiveness*: every step that is applicable to any state will ultimately be performed. There are standard text book procedures to implement exhaustive strategies effectively; in particular, resolution calculi can be treated this way.

This simple approach is sufficient if the underlying rules are commutative in the following sense: whenever two steps are applicable in some state, each of them remains applicable in the successor state produced by the other. If, on the other hand, the rules are not commutative, then the application of one step might destroy the applicability of the other. Such a phenomenon makes exhaustiveness an inherent impossibility and turns fairness into a more difficult question.

Unfortunately, our system is of the non-commutative variety. As an example let

$$M_i = \{\underline{\neg P(a)} \vee R,\quad \underline{P(X)},\quad \underline{\neg Q(a)},\quad \underline{Q(Y)} \vee P(Y),\quad \ldots\}$$

where the underlined path through M_i is open. Both $(\neg P(a), P(X))$ and $(\neg Q(a), Q(Y))$ are candidates for progressive connection steps in M_i. Disregarding the former and

applying the latter yields

$$M_{i+1} = \{\neg P(a) \vee R, \quad P(X), \quad \neg Q(a), \quad Q(a) \vee P(a), \quad Q(Y) \vee P(Y), \quad \ldots\}$$

and now any path through M_{i+1} containing the disregarded connection $(\neg P(a), P(X))$ is closed. This is a consequence of the presence of the clauses $\neg Q(a)$ and $Q(a) \vee P(a)$ in M_{i+1}. Therefore a connection step on the first connection, which would introduce the new clause $P(a)$, is no longer applicable – the second condition in the definition of a connection step can no longer be satisfied for this connection. Note that other paths through M_{i+1} are still open, though.

By the way, had we selected the other applicable step in M_i, the next matrix would have been

$$M'_{i+1} = \{\neg P(a) \vee \underline{R}, \quad \underline{P(a)}, \quad \underline{P(X)}, \quad \underline{\neg Q(a)}, \quad \underline{Q(Y)} \vee P(Y), \quad \ldots\}$$

in which the disregarded connection $(\neg Q(a), Q(Y))$ is still a candidate for a progressive connection step as shown by the underlined open path through M'_{i+1}. Selecting this connection next we can achieve that the new clauses of both steps are present.

Should the presence of all of these clauses happen to be indispensable for closing the matrix, it might be that the sequence $M_0, \ldots, M_i, M_{i+1}$ cannot be extended to a refutation whereas the sequence $M_0, \ldots, M_i, M'_{i+1}$ can – this would be a typical case of non-confluence. Fortunately, it turns out that our calculus *is* confluent: in all such situations, either none or both sequences can be continued to a refutation. The example illustrates why this is not a trivial property.

Coming back to fairness, the simple exhaustiveness definition is not possible, and we need a more complex definition.

DEFINITION 3.3 (FAIRNESS)
A derivation $D = M_0, M_1, \ldots, M_i, \ldots$ is *fair* iff it is a refutation or else the following two conditions are satisfied:

1. For every $i \geq 0$ there is a $j \geq i$ such that M_{j+1} results from M_j by a variant step.
2. For every $i \geq 0$ and every connection (K, L) with MGU σ that is a candidate for a progressive connection step in M_i there is a $j \geq i$ such that one of the following is true:
   (a) $M_i\sigma \subseteq M_j$.
   (b) Every path p through M_j with $\{K, L\} \subseteq p$ is closed.

□

Condition 1 simply requires variant steps to be performed "every now and then".

Condition 2.(a) formalises that any progressive connection step that remains applicable sufficiently long must ultimately be performed. More precisely, the condition does not enforce that this very step be performed, but only that its effect be achieved at some point – regardless whether by this step or by some other.

In a nice case, if Condition 2.(a) holds, the connection (K,L) is irrelevant from M_j onward, because any connection step on (K,L) in some later matrix would result in clauses that have been introduced up to M_j anyway.

However, perhaps contrary to the intuition, this is not always so. The reason is that the MGU σ of the connection, say $\{X/Y\}$, may be applicable to clauses containing the variable X *that are derived only later in the derivation.* Hence, applying σ at a later time may well be progressive. Condition 2.(a) covers this possibility: let the later time be i', then there must again be a time j' at which the clauses that are new for $M_{i'}$ have been introduced.

Condition 2.(b) captures the case that a connection step loses its applicability because of other steps.

3.5 ACHIEVING FAIRNESS

Our notion of fairness – Definition 3.3 – is defined as an abstract mathematical property derivations may or may not enjoy. In order to implement a proof procedure, however, we need not only the property, but an effective strategy for the construction of a derivation that is guaranteed to have this property.

Fortunately, the existence of such a strategy can be demonstrated with a fairly simple approach: use an iteratively increasing bound T that serves both as a limit for the term nesting depth and as a trigger for the application of variant steps.

More precisely, we call a progressive connection step T*-progressive*, if at least one (but not necessarily all) of its new clauses has a term nesting depth not exceeding T. Building on this, we define an effective strategy as follows (in contrast to *The Procedure* in Section 2):

PROCEDURE 3.4 (*The CCC Procedure*)

(i) Initialize M with the empty matrix and T with the maximum term nesting depth of the clauses in S.

(ii) While M is open repeat:

  (a) Modify M by applying one variant step.

  (b) Increment T.

  (c) Modify M by applying a T-progressive connection step.
  Repeat this until saturation, i.e., until no such step is applicable any more.

□

The sequence of values of M over time corresponds to the constructed derivation. Let us denote these values by $M_0, M_1, \ldots, M_i, \ldots$

Note that this strategy is indeed irrevocable in the sense described in the introduction. It never backtracks to previous values of M or makes any other provision for reconsidering alternatives at a later point in time. A subtle difference to *The Procedure* given in Section 2 concerns the generated amplifications. One might think that each

iteration through (a) creates the next amplification S^μ of S. However, during the application of T-progressive connection steps in phase (c), the current M_i is extended by new clauses yielding M_{i+1}, whereas S^μ in *The Procedure* would only be instantiated, but not extended. In a sense our connection steps combine instantiation with (partial) amplification.

THEOREM 3.5 *Any derivation (from any finite input clause set) constructed by Procedure 3.4 is fair.*

PROOF. We first make sure that the saturation phase (c) always terminates. Each connection step has to be T-progressive, which means that it introduces at least one instance $C\sigma$ of some clause with $C \in M$ and $C\sigma \notin M$. The clause $C\sigma$ is either a variant or a proper instance of C. The total number of different variants that may be introduced in this way is finite, because Assumption 3.1 implies that $\mathrm{var}(C\sigma) \subseteq \mathrm{var}(M)$, which is finite (only variant steps introduce new variables, and $|\mathrm{var}(\mathsf{new}(S))|$ is finite). The total number of proper instances that may be introduced is also finite, because their term nesting depth is limited by T, which remains fixed during (c). Thus, after finitely many applications of connection steps all applications producing a clause within the term nesting limit will be used up, and (c) terminates.

Next, if the constructed derivation is a refutation, it is fair by definition. So suppose it is not a refutation. This means that the procedure above does not terminate.

Each of (a), (b), (c) terminates, therefore nontermination implies that (a) is performed periodically with finite intervals in between. Hence the derivation satisfies Condition 1 in the fairness definition.

For Condition 2 consider any point in time i where a progressive connection step is applicable to some connection (K,L) in M_i, and let $D_1,\ldots,D_n$ be the new clauses of this step (progressive means that there is at least one such clause).

CASE 1: M_i is the value of M at the beginning or during the saturation phase (c) for some value of T. Let m be the maximum term nesting depth of the clauses $D_1,\ldots,D_n$. If $m \leq T$, let $i' = i$. Otherwise let i' be the smallest point in time where $M_{i'}$ is the value of M at the beginning of phase (c) and where T has been incremented sufficiently often so that its value is greater or equal to m. (This may require several iterations of the body of the *while* loop, but only finitely many changes to M.)

By construction we have in either case that $M_{i'}$ is the value of M during the saturation phase (c) when each of $D_1,\ldots,D_n$ has a term nesting depth not exceeding the current value of T. Now let j be the point in time right after termination of this saturation phase, when phase (a) would be next. We show that for this j Condition 2.(a) or Condition 2.(b) holds.

Assume to the contrary that neither holds, i.e., $D_k \notin M_j$ for one of the clauses $D_1,\ldots,D_n$ and some path p through M_j with $\{K,L\} \subseteq p$ is open. This means that a connection step is applicable to (K,L) in M_j and that it is T-progressive, because by construction the term nesting depth of D_k does not exceed T. Hence the saturation phase

(c) is not yet completed, contradicting the definition of the time point j. Therefore the assumption is wrong and Condition 2.(a) or Condition 2.(b) holds.

CASE 2: M_i is the value of M at the beginning of phase (a). Then M_{i+1} is obtained from M_i by a variant step. It is straightforward to see that in this case any progressive connection step on (K,L) in M_i is also a progressive connection step on (K,L) in M_{i+1}. Moreover, M_{i+1} is the value of M at the beginning of the saturation phase (c), thus we can construct $j \geq i+1$ as in the previous case, again satisfying Condition 2.

Altogether this proves all requirements of the fairness definition. ■

Of course there are certainly much better strategies. The procedure above is only meant as a proof that fair derivations can be effectively constructed.

What remains, now, is to show that the fairness of derivations, no matter how achieved, ensures confluence. This result is established in the next section.

4 COMPLETENESS

DEFINITION 4.1 (BOUNDED DOWNWARD CLOSED CLAUSE SET)
Let S be a finite clause set, and γ be a ground substitution for S. Define a set of sets $S \downarrow \gamma$ as follows:

$$S \downarrow \gamma = \{S\delta \mid S\delta\gamma = S\gamma \text{ for some substitution } \delta\}$$

□

That is, $S \downarrow \gamma$ is the set of instances of S, (including S, variants of S and $S\gamma$ itself) that can further be instantiated to $S\gamma$ by γ itself. Notice that if $S\delta' \in S \downarrow \gamma$ and $S\delta'\delta''\gamma = S\gamma$ for some δ'', then also $S\delta'\delta'' \in S \downarrow \gamma$ by simply taking $\delta = \delta'\delta''$. In this sense $S \downarrow \gamma$ is "downward closed".

LEMMA 4.2
$S \downarrow \gamma$ is a finite set of finite sets.

PROOF. We are given that S is finite. We show that $\bigcup(S \downarrow \gamma)$ is finite. Clearly, this suffices to show that $S \downarrow \gamma$ is a finite set of finite sets. In order for $\bigcup(S \downarrow \gamma)$ to be infinite, the clauses $C\delta$ in $\bigcup(S \downarrow \gamma)$ would have to grow (i) without bound wrt. term depth, or (ii) for some (at least one) clause $C\delta$ in $\bigcup(S \downarrow \gamma)$ there would have to be infinitely many variants of $C\delta$ in $\bigcup(S \downarrow \gamma)$. Now, (i) is impossible because the condition $C\delta\gamma = C\gamma$ can not be maintained if the term depth of $C\delta$ is higher than that of $C\gamma$. Hence, the term depth of $C\delta$ is limited. Similarly, (ii) cannot be the case, because having infinitely many variants $M = \{C\delta_1, \ldots, C\delta_n, \ldots\} \subseteq \bigcup(S \downarrow \gamma)$ of some $C\delta \in \bigcup(S \downarrow \gamma)$ implies that $\mathrm{var}(M)$ is infinite as well. However, γ is a substitution and hence, by definition of substitution, $\mathrm{dom}(\gamma)$ is finite. Thus, γ instantiates only a finite subset of $\mathrm{var}(M)$, and there must be some $C\delta_j \in M$ for which $C\delta_j\gamma$ is not ground. However, by construction $C\delta_j \in S\delta_j$ for some $S\delta_j \in S \downarrow \gamma$, which contradicts with $S\delta_j\gamma = S\gamma$ the given fact that $S\gamma$ is ground. Hence, in sum, $\bigcup(S \downarrow \gamma)$ must be a finite set, and so $S \downarrow \gamma$ is a finite set of finite sets. ■

LEMMA 4.3
Let γ be a ground substitution, let $V \subseteq \mathrm{dom}(\gamma)$ be a set of variables, and let σ be an idempotent substitution (i.e. $X\sigma = X\sigma\sigma$ for all variables X) with $\mathrm{dom}(\sigma) \subseteq V$ and $\mathrm{vcod}(\sigma) \subseteq V$. If there is a substitution δ such that $X\sigma\delta = X\gamma$ for every $X \in V$ then $X\sigma\gamma = X\gamma$ for every $X \in \mathrm{dom}(\gamma)$.

PROOF. If $X \notin \mathrm{dom}(\sigma)$ the lemma holds trivially. Hence suppose now that $X \in \mathrm{dom}(\sigma)$. Let δ be the given substitution such that

$$X\sigma\delta = X\gamma. \tag{1}$$

Now consider $\mathrm{var}(X\sigma)$. If this set is empty, $X\sigma$ is ground and $X\sigma\delta = X\sigma\gamma$ holds trivially, and with (1) the lemma follows immediately. Hence we consider the case that $\mathrm{var}(X\sigma)$ is nonempty. It suffices to show that

$$Y\delta = Y\gamma, \text{ for every } Y \in \mathrm{var}(X\sigma), \tag{2}$$

because this implies $X\sigma\delta = X\sigma\gamma$, and with (1) the lemma follows immediately.

Now, to prove (2), recall that $Y \in \mathrm{vcod}(\sigma) \subseteq V$ and $X\sigma\delta = X\gamma$ for every $X \in V$ (as given). Thus, it holds in particular that

$$Y\sigma\delta = Y\gamma \tag{3}$$

We are given that σ is idempotent. Hence $\mathrm{dom}(\sigma) \cap \mathrm{vcod}(\sigma) = \emptyset$. Thus $Y \notin \mathrm{dom}(\sigma)$. Hence $Y = Y\sigma$. But then with (3), (2) follows immediately, which remained to be shown. ■

The relevance of this lemma is to prove the following proposition, which serves a purpose analogous to that of the "lifting lemma" in resolution theory.

PROPOSITION 4.4
Let M be a matrix, p be a path through M, and γ be a ground substitution for M. Suppose that $\{K,L\} \subseteq p$ (for some literals K and L), that (K,L) is a connection with substitution γ, and that (K,L) is a connection with MGU σ. Then $M\gamma = M\sigma\gamma$.

PROOF. Let $V = \mathrm{var}(\{K,L\})$. By Assumption 3.1, σ is an idempotent substitution such that $\mathrm{dom}(\sigma) \subseteq V$ and $\mathrm{vcod}(\sigma) \subseteq V$. By the defining property of MGUs, there is a substitution δ such that $X\sigma\delta = X\gamma$ for every $X \in V$. But then Lemma 4.3 is applicable, and the result follows immediately. ■

DEFINITION 4.5 (ORDERING ON CLAUSE SETS)
Let M and N be finite clause sets. Define $M \succ N$ iff $\mathrm{var}(M) \supset \mathrm{var}(N)$. □

It is easy to see that $\succ$ is an irreflexive and transitive relation, hence a strict (partial) ordering. Obviously, since $\mathrm{var}(M)$ is always finite, it is well-founded as well.

LEMMA 4.6 (PATH EXTENSION)
Let $D = M_0, M_1, \ldots, M_n, \ldots$ be an infinite derivation from clause set S. Let M be a finite clause set, and suppose that $M \subseteq M_k$ for some k. Then there is a path p through M such that for every $i \geq k$ there is an open path p_i through M_i with $p \subseteq p_i$.

That is, we can find a path p through M such that p will be part of some open path as the derivation proceeds.

PROOF. Suppose, to the contrary, that for every path p through M there is a $i \geq k$ such that every path p_i through M_i with $p \subseteq p_i$ is closed. Let $\{i_1, \ldots, i_m\}$ be these time points, corresponding to the m paths p through M (matrices are finite).

Now consider $s = \max\{i_1, \ldots, i_m\}$ and let $j \in \{i_1, \ldots, i_m\}$ arbitrary.

Since $j \leq s$ the chain property gives us $M_j \subseteq M_s$. From $j \geq k$ and the chain property conclude that $M_k \subseteq M_j$. We are given that $M \subseteq M_k$. Altogether

$$M \subseteq M_k \subseteq M_j \subseteq M_s \text{ , for every } j \in \{i_1, \ldots, i_m\}. \tag{4}$$

Since the derivation is infinite, it contains no closed matrix, and through every open matrix there is, by definition, an open path. Hence let p_s be any open path through M_s. From 4 we conclude that there is an (open) path $p \subseteq p_s$ through M as well.

By the assumption made at the beginning of the proof, and using the subsequently defined naming of time points as $\{i_1, \ldots, i_m\}$, there is in particular for the just defined path p a time point $j \in \{i_1, \ldots, i_m\}$ such that every path p_j through M_j with $p \subseteq p_j$ is closed.

From the fact that p (resp. p_s) is a path through M (resp. M_s) and $p \subseteq p_s$ it follows with 4 that there is a path p' through M_j such that $p \subseteq p' \subseteq p_s$. Since p_s is an open path, p' is an open path as well (by $p' \subseteq p_s$). This, however, yields together with $p \subseteq p'$ a plain contradiction to the conclusion above, which stated that every path p_j through M_j with $p \subseteq p_j$ is closed. Hence, the assumption must have been wrong, and the lemma holds as claimed. ■

The main result of this paper is the following completeness theorem. Note that it assures completeness, whenever fairness is guaranteed; hence we have strong completeness and thus proof confluence.

THEOREM 4.7 (COMPLETENESS) *Let S be the given input clause set. If S is unsatisfiable, then every fair derivation from S is a refutation.*

PROOF. By Herbrand's theorem there is a finite set S^g of ground instances of clauses from S which is unsatisfiable. It can be presented as a finite set M of pairwise variable disjoint variants of clauses from S and a ground substitution γ such that $M\gamma = S^g$.

Now, let D be a fair derivation from S and assume contrary to the theorem that D is not a refutation.

Since D is fair, the variant rule must be applied infinitely often. Recall that we never delete clauses from matrices. Hence, at some time point, say l, it will thus be that $M \subseteq M_l$ (and hence also $M \subseteq \bigcup_{i \geq 0} M_i$). W.l.o.g. we can assume that each clause in M is

syntactically identical to one of the variants introduced up to M_l; otherwise rename M (and γ) appropriately.

Now let $N \in M \downarrow \gamma$ be a minimal set wrt. $\succ$ such that $N \subseteq \bigcup_{i\geq 0} M_i$. We have to check that such a set N exists: since $M \subseteq \bigcup_{i\geq 0} M_i$ and it trivially holds that $M \in M \downarrow \gamma$ it is clear that M itself might be a candidate to be taken as N. Now, since $\succ$ is a well-founded ordering on finite clause sets, and the elements of $M \downarrow \gamma$ are finite (cf. Lemma 4.2), any chain $(M_0 := M) \succ M_1 \succ \cdots \succ M_{n-1} \succ M_n$ with $M_i \in M \downarrow \gamma$ and $M_i \subseteq \bigcup_{i\geq 0} M_i$ must be finite, and we set $N := M_n$, provided that this chain cannot be extended to the right. Thus, N is minimal wrt. $\succ$.

The proof technique now is to construct an N' with $N' \in M \downarrow \gamma$ and $N' \subseteq \bigcup_{i\geq 0} M_i$ and $N \succ N'$, contradicting the minimality of N.

Since derivations have the chain property, the property $N \subseteq \bigcup_{i\geq 0} M_i$ implies by the fact that N is a finite set that

$$N \subseteq M_k \quad \text{, for some } k. \tag{5}$$

Therefore we can apply Lemma 4.6 and conclude that there is a path p through N such that for every $i \geq k$ there is an open path p_i through M_i with $p \subseteq p_i$. This result will be used further below – the immediate consequence that this p is open will be needed in a moment.

Clearly, $N\gamma$ is an unsatisfiable ground clause set, because $N \in M \downarrow \gamma$ means by definition that $N = M\delta$ for some δ such that $M\delta\gamma = M\gamma$, hence $N\gamma = M\gamma$, and $M\gamma$ was assumed to be ground and unsatisfiable above. Consequently, any path through $N\gamma$ contains a pair of complementary literals. In particular, $p\gamma$ contains a pair of complementary literals, say $K\gamma$ and $L\gamma$, where

$$\{K, L\} \subseteq p \quad . \tag{6}$$

In other words, (K, L) is a connection with substitution γ. But then (K, L) is a connection with MGU σ, where σ is computed by unify. Then Proposition 4.4 is applicable and we conclude that $N\sigma\gamma = N\gamma$. The element $N\sigma$ has the form $N\sigma = M(\delta\sigma)$ because $N = M\delta$. For this element we obtain by the previous equation and by $N\gamma = M\gamma$ the properties $M(\delta\sigma)\gamma = N\gamma = M\gamma$, which is just the definition for

$$N\sigma \in M \downarrow \gamma \quad . \tag{7}$$

The next subgoal is to show that $N\sigma$ is strictly smaller than N.

Since p is open and $\{K, L\} \subseteq p$ it trivially holds that $|K| \neq |L|$. On the other hand, σ is a most general unifier of $|K|$ and $|L|$. Hence there is at least one variable, say X, in $\{K, L\} \subseteq p$, for which $X\sigma \neq X$, that is, $X \in \mathrm{dom}(\sigma)$. Now p is a path through N, therefore $X \in \mathrm{var}(N)$.
Obviously, $\mathrm{var}(N\sigma) \subseteq \mathrm{var}(N) \cup \mathrm{vcod}(\sigma)$. By Assumption 3.1 $\mathrm{vcod}(\sigma) \subseteq \mathrm{var}(\{K, L\}) \subseteq \mathrm{var}(N)$, hence $\mathrm{var}(N\sigma) \subseteq \mathrm{var}(N)$.
The idempotence of σ implies $\mathrm{dom}(\sigma) \cap \mathrm{vcod}(\sigma) = \emptyset$, thus $X \notin \mathrm{vcod}(\sigma)$ and $X \notin \mathrm{var}(N\sigma)$. So we have $\mathrm{var}(N\sigma) \subset \mathrm{var}(N)$, which means nothing but

$$N \succ N\sigma \quad . \tag{8}$$

There remains to be shown that

$$N\sigma \subseteq \bigcup_{i\geq 0} M_i \ . \tag{9}$$

We distinguish two cases, each leading to the conclusion 9.

CASE 1: For every $i \geq k$, it is not the case that (K,L) is a candidate for a progressive connection step in M_i.

As concluded above by the application of Lemma 4.6, for every $i \geq k$ there is an open path p_i through M_i with $p \subseteq p_i$. With the fact that $\{K,L\} \subseteq p$ (obtained in 6 above) it follows that $\{K,L\}$ is a candidate for a connection step in M_i, for every $i \geq k$.

Together with the assumption of this Case 1 we obtain that for every $i \geq k$ a connection step on (K,L) in M_i exists, but this connection step is non-progressive. Hence, for every $i \geq k$, $M_i \cup M_i\sigma = M_i$. But then with $N \subseteq M_k$ as obtained in 5 we get the following chain:

$$N\sigma \ \subseteq \ M_k\sigma \ \subseteq \ \bigcup_{i\geq k}(M_i \cup M_i\sigma) \ = \ \bigcup_{i\geq k} M_i \ \subseteq \ \bigcup_{i\geq 0} M_i \ .$$

CASE 2: This is the complement of Case 1. Hence suppose that for some $i \geq k$ the connection (K,L) is a candidate for a progressive connection step in M_i. We are given that the given derivation D is fair (cf. Def. 3.3). In particular, Condition 2 in the definition of fairness holds wrt. the connection (K,L) and M_i. Let $j \geq i$ be the point in time claimed there.

Assume that Condition 2.(b) is satisfied (this will yield a contradiction). This means that every path p' through M_j with $\{K,L\} \subseteq p'$ is closed. But at the same time, however, observing that $j \geq k$ (since $j \geq i$ and $i \geq k$) we concluded further above by application of Lemma 4.6 that there is an open path p_j through M_j with $p \subseteq p_j$. Since $\{K,L\} \subseteq p$ (as by 6) and thus $\{K,L\} \subseteq p_j$, we arrive at a contradiction to the just assumed by setting $p' = p_j$.

Hence Condition 2.(b) cannot be satisfied, and consequently Condition 2.(a) must be satisfied. This means that $M_i\sigma \subseteq M_j$. Recall that derivations have the chain property. From $i \geq k$ we thus get $M_k \subseteq M_i$. Together with $N \subseteq M_k$ then $N\sigma \subseteq M_i\sigma$, and so $N\sigma \subseteq M_j$ follows immediately. Clearly, $N\sigma \subseteq \bigcup_{i\geq 0} M_i$ as well. This completes Case 2.

Notice that in both cases we concluded with 9. Altogether, 7, 8, and 9 contradict the minimality of N.

Hence, the assumption that D is not a refutation must be wrong, and the theorem holds. ■

5 CONCLUSIONS

In this paper we have defined CCC, a confluent connection calculus on the first-order level. We gave a proof of its strong completeness and hence, of its proof confluence.

We demonstrated the drawback of a naive connection method by means of an example and we briefly discussed tableaux oriented proof procedures. More precisely, in

Section 2.3 we identified an incestuous graph (IG) problem and argued that a prototypical proof procedure – *The Procedure* – exhibits an exponential search space. We did not rigorously argue that the proposed new proof procedure – *The CCC Procedure* – performs any better. However, it is indeed possible to solve the IG problem with *The CCC Procedure* in linear time (and space). To see this, observe first that the IG problem contains no function symbols. Further, it is possible to derive all required instances of the non-ground clauses (RT-1) and (RT-2) as descendant of *one single* variant of the (RT-1) and (RT-2) clauses, respectively. In other words, *The CCC procedure* will find the refutation during the *first* instance of the saturation phase (ii.c).

The CCC Procedure treats matrices, which are sets of clauses. Consequently, the order of clauses does not play any role for fairness or completeness. Hence, more refined proof procedures are free to represent sets as (duplicate-free) lists. A good strategy then would be to append new instances of clauses derived during the saturation phase or during variant steps at the end of the list, because then it is cheap to identify and represent those new closed paths that simply extend previously closed paths[5].

As a further ingredient, actual implementations of *The CCC Procedure* should include a "relevance test"[6]: consider, for example, a matrix $M \cup \{A \vee B\} \cup M'$, and suppose that p is some path through M, and that for every path p' through M' the path $p \cup \{A\} \cup p'$ is closed. Now, if $p \cup p'$ alone is closed (for every p'), then also $p \cup \{B\} \cup p'$ is closed (for every p'). In other words, if A is not relevant to obtain closed paths through M' in the context $p \cup \{A\}$, then no *search* for closed paths in M' in the context $p \cup \{B\}$ is needed.

Now, using the proposed list representation of matrices, and taking advantage of the relevance test, it can indeed be shown that *The CCC Procedure* solves the IG problem after $O(n)$ steps.

Our calculus achieves confluence by taking into account only derivations which obey a fairness condition. This condition is formulated as an abstract formal property of derivations. It allows to formulate a strong completeness theorem, stating that *any fair* derivation leads to a proof, provided that a proof exists at all.

The difficulty was to define the calculus in such a way that an *effective* fairness condition can be stated. Defining an effective fair strategy is much less straightforward than in resolution calculi (CCC is not commutative, unlike resolution).

That it is not trivial was observed already in [Bry and Eisinger, 1996]. There, a rigid-variable calculus is defined and a strong completeness result is proven. However, the question how to define an *effective* fair strategy had to be left open. Thus, our new approach can be seen to address open issues there.

We came up with a strategy, which is based on a term-depth bound and we proved that this strategy indeed results in fair derivations.

We are aware of the fact, that this effective strategy is only a first step towards the design of an efficient proof procedure based on the CCC-calculus. We expect improvements over "usual" tableaux based implementations of connection calculi, which do not

[5] This strategy resembles much a tableau procedure – one that takes advantage of *confluence*, however.

[6] Similar to *condensing* in [Oppacher and Suen, 1988] or *level cut* in [Baumgartner *et al.*, 1996]. Although so simple, this technique is very helpful in practice.

exploit confluence.

This article is a first step, a lot of work remains to be done. In particular the saturation step within our strategy for achieving fairness needs to be turned into a more algorithmic version. Another important topic is to avoid the generation of redundant clauses. To this end regularity, as it is implemented in clausal tableaux would be a first attempt. A further point would be to investigate under which conditions the variant inference rule can be dispensed with, or how to modify the calculus so that new variants of input clauses are introduced more sparsely.

ACKNOWLEDGMENTS

We are grateful to Donald Loveland for comments on an earlier version. Norbert Eisinger was a second reader of a first version of this paper; he is now an author.

REFERENCES

[Baader and Schulz, 1998] Franz Baader and Klaus U. Schulz. Unification Theory. In Wolfgang Bibel and Peter H. Schmitt, editors, *Automated Deduction. A Basis for Applications*. Kluwer Academic Publishers, 1998.

[Baumgartner and Furbach, 1993] Peter Baumgartner and Ulrich Furbach. Consolution as a Framework for Comparing Calculi. *Journal of Symbolic Computation*, 16(5):445–477, 1993.

[Baumgartner and Furbach, 1994] Peter Baumgartner and Ulrich Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. In A. Bundy, editor, *Automated Deduction – CADE-12*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 769–773. Springer, 1994.

[Baumgartner *et al.*, 1996] Peter Baumgartner, Ulrich Furbach, and Ilkka Niemelä. Hyper Tableaux. In *Proc. JELIA 96*, number 1126 in Lecture Notes in Artificial Intelligence. European Workshop on Logic in AI, Springer, 1996.

[Baumgartner *et al.*, 1997] Peter Baumgartner, Ulrich Furbach, and Frieder Stolzenburg. Computing Answers with Model Elimination. *Artificial Intelligence*, 90(1–2):135–176, 1997.

[Beckert and Posegga, 1995] Bernhard Beckert and Joachim Posegga. leanTAP: Lean tableau-based deduction. *Journal of Automated Reasoning*, 15(3):339–358, 1995.

[Bibel, 1987] W. Bibel. *Automated Theorem Proving*. Vieweg, 2nd edition, 1987.

[Bry and Eisinger, 1996] Francois Bry and Norbert Eisinger. Unit resolution tableaux. Research Report PMS-FB-1996-2, Institut für Informatik, LMU München, 1996.

[Bry and Yahya, 1996] François Bry and Adnan Yahya. Minimal Model Generation with Positive Unit Hyper-Resolution Tableaux. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, number 1071 in Lecture Notes in Artificial Intelligence, pages 143–159. Springer, 1996.

[Eder, 1992] E. Eder. *Relative Complexities of First Order Languages*. Vieweg, 1992.

[Fitting, 1990] M. Fitting. *First Order Logic and Automated Theorem Proving*. Texts and Monographs in Computer Science. Springer, 1990.

[Fujita and Hasegawa, 1991] H. Fujita and R. Hasegawa. A Model Generation Theorem Prover in KL1 using a Ramified-Stack Algorithm. In *Proc. of the Eigth International Conference on Logic Programming*, pages 535–548, Paris, France, 1991.

[Hähnle and Klingenbeck, 1996] Reiner Hähnle and Stefan Klingenbeck. A-Ordered Tableaux. *Journal of Logic and Computation*, 6(6):819–833, 1996.

[Hähnle and Pape, 1997] Reiner Hähnle and Christian Pape. Ordered tableaux: Extensions and applications. In Didier Galmiche, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, number 1227 in Lecture Notes in Artificial Intelligence, pages 173–187. Springer, 1997.

[Hähnle *et al.*, 1994] R. Hähnle, B. Beckert, and S. Gerberding. The Many-Valued Theorem Prover 3TAP. Interner Bericht 30/94, Universität Karlsruhe, 1994.

[Klingenbeck and Hähnle, 1994] Stefan Klingenbeck and Reiner Hähnle. Semantic tableaux with ordering restrictions. In Alan Bundy, editor, *Automated Deduction — CADE 12*, LNAI 814, pages 708–722, Nancy, France, June 1994. Springer-Verlag.

[Letz *et al.*, 1992] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 8(2), 1992.

[Letz *et al.*, 1994] R. Letz, K. Mayr, and C. Goller. Controlled Integrations of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13, 1994.

[Loveland, 1968] D. Loveland. Mechanical Theorem Proving by Model Elimination. *JACM*, 15(2), 1968.

[Manthey and Bry, 1988] Rainer Manthey and François Bry. SATCHMO: a theorem prover implemented in Prolog. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9th Conference on Automated Deduction, Argonne, Illinois, May 1988*, volume 310 of *Lecture Notes in Computer Science*, pages 415–434. Springer, 1988.

[Oppacher and Suen, 1988] F. Oppacher and E. Suen. HARP: A Tableau-Based Theorem Prover. *Journal of Automated Reasoning*, 4:69–100, 1988.

[Plaisted and Zhu, 1997] David A. Plaisted and Yunshan Zhu. Ordered Semantic Hyper Linking. In *Proceedings of Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 1997.

[Prawitz, 1960] D. Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[Robinson, 1965] J. A. Robinson. Automated deduction with hyper-resolution. *Internat. J. Comput. Math.*, 1:227–234, 1965.

[Voronkov, 1997] Andrej Voronkov. Strategies in rigid-variable methods. In *15th International Joint Conference on Artificial Intelligence (IJCAI 97)*, Nagoya, 1997. International Joint Conference on Artificial Intelligence.

[Voronkov, 1998] Andrej Voronkov. Herbrand's theorem, automated reasoning and semantic tableaux. In *IEEE Symposium on Logic in Computer Science*, 1998.

Available Research Reports (since 1995):

1998

23/98 *Peter Baumgartner, Norbert Eisinger, Ulrich Furbach.* A Confluent Connection Calculus.

22/98 *Bernt Kullbach, Andreas Winter.* Querying as an Enabling Technology in Software Reengineering.

21/98 *Jürgen Dix, V.S. Subrahmanian, George Pick.* Meta-Agent Programs.

20/98 *Jürgen Dix, Ulrich Furbach, Ilkka Niemelä .* Nonmonotonic Reasoning: Towards Efficient Calculi and Implementations.

19/98 *Jürgen Dix, Steffen Hölldobler.* Inference Mechanisms in Knowledge-Based Systems: Theory and Applications (Proceedings of WS at KI '98).

17/98 *Stefan Brass, Jürgen Dix, Teodor C. Przymusinski.* Super Logic Programs.

16/98 *Jürgen Dix.* The Logic Programming Paradigm.

15/98 *Stefan Brass, Jürgen Dix, Burkhard Freitag, Ulrich Zukowski.* Transformation-Based Bottom-Up Computation of the Well-Founded Model.

14/98 *Manfred Kamp.* GReQL – Eine Anfragesprache für das GUPRO–Repository – Sprachbeschreibung (Version 1.2).

12/98 *Peter Dahm, Jürgen Ebert, Angelika Franzke, Manfred Kamp, Andreas Winter.* TGraphen und EER-Schemata – formale Grundlagen.

11/98 *Peter Dahm, Friedbert Widmann.* Das Graphenlabor.

10/98 *Jörg Jooss, Thomas Marx.* Workflow Modeling according to WfMC.

9/98 *Dieter Zöbel.* Schedulability criteria for age constraint processes in hard real-time systems.

8/98 *Wenjin Lu, Ulrich Furbach.* Disjunctive logic program = Horn Program + Control program.

7/98 *Andreas Schmid.* Solution for the counting to infinity problem of distance vector routing.

6/98 *Ulrich Furbach, Michael Kühn, Frieder Stolzenburg.* Model-Guided Proof Debugging.

5/98 *Peter Baumgartner, Dorothea Schäfer.* Model Elimination with Simplification and its Application to Software Verification.

4/98 *Bernt Kullbach, Andreas Winter, Peter Dahm, Jürgen Ebert.* Program Comprehension in Multi-Language Systems.

3/98 *Jürgen Dix, Jorge Lobo.* Logic Programming and Nonmonotonic Reasoning.

2/98 *Hans-Michael Hanisch, Kurt Lautenbach, Carlo Simon, Jan Thieme.* Zeitstempelnetze in technischen Anwendungen.

1/98 *Manfred Kamp.* Managing a Multi-File, Multi-Language Software Repository for Program Comprehension Tools — A Generic Approach.

1997

32/97 *Peter Baumgartner.* Hyper Tableaux — The Next Generation.

31/97 *Jens Woch.* A component-based and abstractivistic Agent Architecture for the modelling of MAS in the Social Sciences.

30/97 *Marcel Bresink.* A Software Test-Bed for Global Illumination Research.

29/97 *Marcel Bresink.* Deutschsprachige Terminologie des Radiosity-Verfahrens.

28/97 *Jürgen Ebert, Bernt Kullbach, Andreas Panse.* The Extract-Transform-Rewrite Cycle - A Step towards MetaCARE.

27/97 *Jose Arrazola, Jürgen Dix, Mauricio Osorio.* Confluent Rewriting Systems for Logic Programming Semantics.

26/97 *Lutz Priese.* A Note on Nondeterministic Reversible Computations.

25/97 *Stephan Philippi.* System modelling using Object-Oriented Pr/T-Nets.

24/97 *Lutz Priese, Yurii Rogojine, Maurice Margenstern.* Finite H-Systems with 3 Test Tubes are not Predictable.

23/97 *Peter Baumgartner (Hrsg.).* Jahrestreffen der GI-Fachgruppe 1.2.1 'Deduktionssysteme' — Kurzfassungen der Vorträge.

22/97 *Jens M. Felderhoff, Thomas Marx.* Erkennung semantischer Integritätsbedingungen in Datenbankanwendungen.

21/97 *Angelika Franzke.* Specifying Object Oriented Systems using GDMO, ZEST and SDL'92.

20/97 *Angelika Franzke.* Recommendations for an Improvement of GDMO.

19/97 *Jürgen Dix, Luís Moniz Pereira, Teodor Przymusinski.* Logic Programming and Knowledge Representation (LPKR '97) (Proceedings of the ILPS '97 Postconference Workshop).

18/97 *Lutz Priese, Harro Wimmel.* A Uniform Approach to True-Concurrency and Interleaving Semantics for Petri Nets.

17/97 *Ulrich Furbach (Ed.).* IJCAI-97 Workshop on Model Based Automated Reasoning.

16/97 *Jürgen Dix, Frieder Stolzenburg.* A Framework to Incorporate Non-Monotonic Reasoning into Constraint Logic Programming.

15/97 *Carlo Simon, Hanno Ridder, Thomas Marx.* The Petri Net Tools Neptun and Poseidon.

14/97 *Juha-Pekka Tolvanen, Andreas Winter (Eds.).* CAiSE'97 — 4th Doctoral Consortium on Advanced Information Systems Engineering, Barcelona, June 16-17, 1997, Proceedings.

13/97 *Jürgen Ebert, Roger Süttenbach.* An OMT Metamodel.

12/97 *Stefan Brass, Jürgen Dix, Teodor Przymusinski.* Super Logic Programs.

11/97 *Jürgen Dix, Mauricio Osorio.* Towards Well-Behaved Semantics Suitable for Aggregation.

10/97 *Chandrabose Aravindan, Peter Baumgartner.* A Rational and Efficient Algorithm for View Deletion in Databases.

9/97 *Wolfgang Albrecht, Dieter Zöbel.* Integrating Fixed Priority and Static Scheduling to Maintain External Consistency.

8/97 *Jürgen Ebert, Alexander Fronk.* Operational Semantics of Visual Notations.

7/97 *Thomas Marx.* APRIL - Visualisierung der Anforderungen.

6/97 *Jürgen Ebert, Manfred Kamp, Andreas Winter.* A Generic System to Support Multi-Level Understanding of Heterogeneous Software.

5/97 *Roger Süttenbach, Jürgen Ebert.* A Booch Metamodel.

4/97 *Jürgen Dix, Luis Pereira, Teodor Przymusinski.* Prolegomena to Logic Programming for Non-Monotonic Reasoning.

3/97 *Angelika Franzke.* GRAL 2.0: A Reference Manual.

2/97 *Ulrich Furbach.* A View to Automated Reasoning in Artificial Intelligence.

1/97 *Chandrabose Aravindan, Jürgen Dix, Ilkka Niemelä .* DisLoP: A Research Project on Disjunctive Logic Programming.

1996

28/96 *Wolfgang Albrecht.* Echtzeitplanung für Alters- oder Reaktionszeitanforderungen.

27/96 *Kurt Lautenbach.* Action Logical Correctness Proving.

26/96 *Frieder Stolzenburg, Stephan Höhne, Ulrich Koch, Martin Volk.* Constraint Logic Programming for Computational Linguistics.

25/96 *Kurt Lautenbach, Hanno Ridder.* Die Lineare Algebra der Verklemmungsvermeidung — Ein Petri-Netz-Ansatz.

24/96 *Peter Baumgartner, Ulrich Furbach.* Refinements for Restart Model Elimination.

23/96 *Peter Baumgartner, Peter Fröhlich, Ulrich Furbach, Wolfgang Nejdl.* Tableaux for Diagnosis Applications.

22/96 *Jürgen Ebert, Roger Süttenbach, Ingar Uhe.* Meta-CASE in Practice: a Case for KOGGE.

21/96 *Harro Wimmel, Lutz Priese.* Algebraic Characterization of Petri Net Pomset Semantics.

20/96 *Wenjin Lu.* Minimal Model Generation Based on E-Hyper Tableaux.

19/96 *Frieder Stolzenburg.* A Flexible System for Constraint Disjunctive Logic Programming.

18/96 *Ilkka Niemelä (Ed.).* Proceedings of the ECAI'96 Workshop on Integrating Nonmonotonicity into Automated Reasoning Systems.

17/96 *Jürgen Dix, Luis Moniz Pereira, Teodor Przymusinski.* Non-monotonic Extensions of Logic Programming: Theory, Implementation and Applications (Proceedings of the JICSLP '96 Postconference Workshop W1).

16/96 *Chandrabose Aravindan.* DisLoP: A Disjunctive Logic Programming System Based on PROTEIN Theorem Prover.

15/96 *Jürgen Dix, Gerhard Brewka.* Knowledge Representation with Logic Programs.

14/96 *Harro Wimmel, Lutz Priese.* An Application of Compositional Petri Net Semantics.

13/96 *Peter Baumgartner, Ulrich Furbach.* Calculi for Disjunctive Logic Programming.

12/96 *Klaus Zitzmann.* Physically Based Volume Rendering of Gaseous Objects.

11/96 *J. Ebert, A. Winter, P. Dahm, A. Franzke, R. Süttenbach.* Graph Based Modeling and Implementation with EER/GRAL.

10/96 *Angelika Franzke.* Querying Graph Structures with G^2QL.

9/96 *Chandrabose Aravindan.* An abductive framework for negation in disjunctive logic programming.

8/96 *Peter Baumgartner, Ulrich Furbach, Ilkka Niemelä .* Hyper Tableaux.

7/96 *Ilkka Niemelä, Patrik Simons.* Efficient Implementation of the Well-founded and Stable Model Semantics.

6/96 *Ilkka Niemelä .* Implementing Circumscription Using a Tableau Method.

5/96 *Ilkka Niemelä .* A Tableau Calculus for Minimal Model Reasoning.

4/96 *Stefan Brass, Jürgen Dix, Teodor. C. Przymusinski.* Characterizations and Implementation of Static Semantics of Disjunctive Programs.

3/96 *Jürgen Ebert, Manfred Kamp, Andreas Winter.* Generic Support for Understanding Heterogeneous Software.

2/96 *Stefan Brass, Jürgen Dix, Ilkka Niemelä, Teodor. C. Przymusinski.* A Comparison of STATIC Semantics with D-WFS.

1/96 *J. Ebert (Hrsg.).* Alternative Konzepte für Sprachen und Rechner, Bad Honnef 1995.

1995

21/95 *J. Dix and U. Furbach.* Logisches Programmieren mit Negation und Disjunktion.

20/95 *L. Priese, H. Wimmel.* On Some Compositional Petri Net Semantics.

19/95 *J. Ebert, G. Engels.* Specification of Object Life Cycle Definitions.

18/95 *J. Dix, D. Gottlob, V. Marek.* Reducing Disjunctive to Non-Disjunctive Semantics by Shift-Operations.

17/95 *P. Baumgartner, J. Dix, U. Furbach, D. Schäfer, F. Stolzenburg.* Deduktion und Logisches Programmieren.

16/95 *Doris Nolte, Lutz Priese.* Abstract Fairness and Semantics.

15/95 *Volker Rehrmann (Hrsg.).* 1. Workshop Farbbildverarbeitung.

14/95 *Frieder Stolzenburg, Bernd Thomas.* Analysing Rule Sets for the Calculation of Banking Fees by a Theorem Prover with Constraints.

13/95 *Frieder Stolzenburg.* Membership-Constraints and Complexity in Logic Programming with Sets.

12/95 *Stefan Brass, Jürgen Dix.* D-WFS: A Confluent Calculus and an Equivalent Characterization..

11/95 *Thomas Marx.* NetCASE — A Petri Net based Method for Database Application Design and Generation.

10/95 *Kurt Lautenbach, Hanno Ridder.* A Completion of the S-invariance Technique by means of Fixed Point Algorithms.

9/95 *Christian Fahrner, Thomas Marx, Stephan Philippi.* Integration of Integrity Constraints into Object-Oriented Database Schema according to ODMG-93.

8/95 *Christoph Steigner, Andreas Weihrauch.* Modelling Timeouts in Protocol Design..

7/95 *Jürgen Ebert, Gottfried Vossen.* I-Serializability: Generalized Correctness for Transaction-Based Environments.

6/95 *P. Baumgartner, S. Brüning.* A Disjunctive Positive Refinement of Model Elimination and its Application to Subsumption Deletion.

5/95 *P. Baumgartner, J. Schumann.* Implementing Restart Model Elimination and Theory Model Elimination on top of SETHEO.

4/95 *Lutz Priese, Jens Klieber, Raimund Lakmann, Volker Rehrmann, Rainer Schian.* Echtzeit-Verkehrszeichenerkennung mit dem Color Structure Code — Ein Projektbericht.

3/95 *Lutz Priese.* A Class of Fully Abstract Semantics for Petri-Nets.

2/95 *P. Baumgartner, R. Hähnle, J. Posegga (Hrsg.).* 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods — Poster Session and Short Papers.

1/95 *P. Baumgartner, U. Furbach, F. Stolzenburg.* Model Elimination, Logic Programming and Computing Answers.