J. Symbolic Computation (1994) **11**, 1–000

Consolution as a Framework for Comparing Calculi

PETER BAUMGARTNER and ULRICH FURBACH

University of Koblenz, Rheinau 1,

56075 Koblenz, Germany

{peter,uli}@informatik.uni-koblenz.de

(Received 20 May 1994)

In this paper, stepwise and nearly stepwise simulation results for a number of first-order proof calculi are presented and an overview is given that illustrates the relations between these calculi. For this purpose, we modify the *consolution* calculus in such a way that it can be instantiated to *resolution*, *tableaux model elimination*, a *connection method* and Loveland's *model elimination*.

1. Introduction

In (Eder, 1991) the consolution method is introduced as a bridge between the idea of matrix methods (see (Bibel, 1987)) and resolution. Eder defines a consolution calculus together with a construction which yields a consolution refutation for every resolution refutation. This gives both a demonstration that resolution can be seen as a special strategy within consolution and a completeness proof for consolution. However, Eder did not investigate a formal treatment of matrix methods with consolution; he only gave an informal description of matrix methods as a starting point for consolution.

It appeared fascinating to us to see how easily the two ideas of a matrix method and a resolution calculus can be combined in the consolution method. We tried to follow this line to use consolution as a framework to relate matrix methods and resolution by defining special consolution calculi. It turned out that this is *not* as straightforward as we expected. However we can reach this goal by making numerous changes to the definition of consolution. The changes are harmless in the sense that the inference rules of the modified calculus subsume that of the original calculus. The documentation of these changes is one purpose of this paper.

Another purpose is to demonstrate that consolution is well suited for comparing calculi. This is a topic of significant importance for anyone trying to understand the essence of deduction or to implement a deduction system.

In particular we show that when defined appropriately, consolution can simulate step by step a certain model elimination calculus (model elimination as defined by Loveland), a connection method calculus and a tableau model elimination calculus on which proof procedures like PTTP (Stickel, 1989; Stickel, 1988) and SETHEO (Letz et al., 1992) are based. The result for resolution and consolution was shown in (Eder, 1991). Altogether we will prove the diagram from figure 1, where the $A \rightarrow B$ means that calculus A can be simulated stepwise by calculus B; dashed arrows indicate a "weaker" simulatability relation after certain calculus restrictions.

0747–7171/90/000000 + 00 $03.00/0

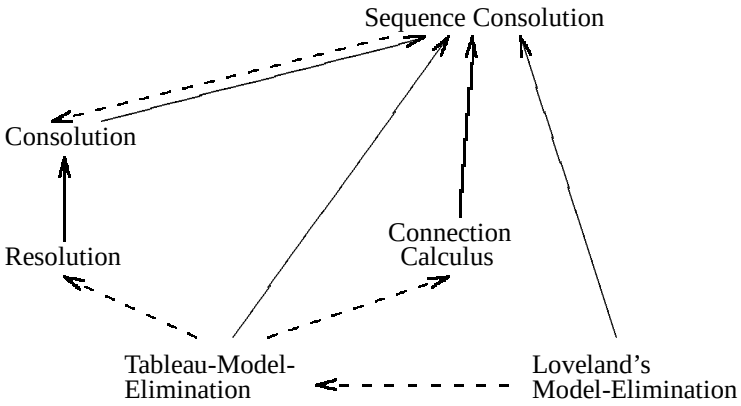


Figure 1. Relations between calculi.

Furthermore all these calculi are defined in a uniform and consolution-like language, making it easy to relate them among each other. This is a significant difference when compared with other work — we use one formal framework for the definition of the various calculi, which allows rigorous comparisons together with proofs.

In the next section we briefly review the idea and the consolution calculus as it is given in (Eder, 1991). In section 3 we define a model elimination calculus in a consolution-like language, which we call "tableau model elimination", and show how to modify consolution in order to simulate tableau model elimination. As a result we come up with a consolution method called "sequence consolution" which loses part of its elegance and simplicity compared with the original, but which makes some of its parameters explicit. Section 4 relates consolution to sequence consolution in a formal way. Sequence consolution then serves as a framework for the definition and comparison of other calculi. This is done in section 5 for Loveland's original model elimination calculus and in section 6 for a connection calculus. Section 7 relates the tableau model elimination calculus to resolution. Finally section 8 contains the discussion.

2. The Idea of Consolution

Consolution can be seen as a procedure for converting a formula given in one normal form into another normal form: assume we are given a (for simplicity: ground) formula in disjunctive normal form (DNF) and want to prove its validity. This can be done by converting it in a first step into conjunctive normal form (CNF). The second step then uses the fact that a formula in CNF is valid iff every conjunct contains complementary literals. Thus, a simple test for complementary literals in every conjunct suffices to decide the validity of the CNF and also the DNF. Now, with some additional optimizations this is just how consolution works. Consider for example the DNF-formula

$$\underbrace{(P \wedge Q) \vee (\neg P \wedge Q)} \vee \neg Q$$

Conversion to CNF can be begun by applying the law of distributivity to the underbraced part, yielding

$$((P \vee \neg P) \wedge (P \vee Q) \wedge (Q \vee \neg P) \wedge (Q \vee Q)) \vee \neg Q$$

This operation is also carried out as a first step in an consolution inference. The subsequent steps deal with the above-mentioned optimizations: first, disjuncts such as $P \vee \neg P$ which contain complementary literals are tautological and thus can be removed. Second, disjuncts may be shortened; for example, $(Q \vee \neg P)$ may be replaced by Q. This corresponds in some sense to the "weakening" rule in Gentzen's sequent calculus (see e.g. (Gallier, 1987) for the sequent calculus). The shortening step is sound since it preserves non-tautologyhood. However, it may cause incompleteness by throwing away the "wrong" literal, i.e. the literal that contributes to a complementary pair in a later stage. Third, $Q \vee Q$ can be replaced by Q. This rule corresponds to the "contraction" rule in the sequent calculus. It is implicitly present in consolution by means of the set data structure, which collapses multiple occurences of literals into a single one. Similarly, identical conjuncts such as Q in $Q \wedge Q$ can be contracted to a single one. Carrying out these suggested operations results in the formula

$$((P \vee Q) \wedge Q) \vee \neg Q$$

This formula is the result of an application of the consolution inference rule. Applying the rule again yields by the law of distributivity the formula

$$(P \vee Q \vee \neg Q) \wedge (Q \vee \neg Q)$$

This formula can be simplified as explained above towards the "empty" conjunct, which is a proof for the validity of the given formula.

Consolution is slightly more general than just explained: instead of logical formulas in DNF, consolution works on sets of clauses, where a clause is a set of literals. The semantics of clause sets is then obtained by interpreting the outer commas by "$\vee$" and the inner commas by "$\wedge$". The clause set data structure is more general, since the interpretation of the outer comma and inner comma can be exchanged. In other words, one starts with a CNF instead of a DNF. A derivation of the empty clause can then be interpreted as proof of the unsatisfiablity of the DNF-formula, instead of a proof of the validity of a (logically different) CNF-formula. This duality is not specific to consolution but applies to every calculus with clause sets as data structure. It gives us the freedom to directly relate derivations in e.g. model elimination (which is usually formulated in the refutational setting) and consolution (which was formulated in Eder's theorem in the affirmative setting).

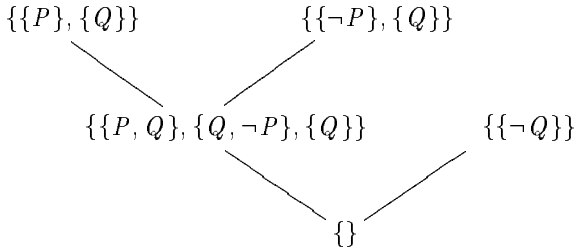
Consolution can also be explained from the background of the connection method (cf. (Bibel, 1987)). Here, sets of clauses are called *matrices*, and the method is concerned with proving that every path through this matrix contains two complementary literals, called connections in this framework (a *path* through a matrix is built by selecting exactly one literal from every clause in the matrix).

To illustrate consolution we use an example from (Eder, 1991):

The three clauses $\{P, Q\}$, $\{\neg P, Q\}$ and $\neg Q$ are represented in the connection method as a matrix M:

$$\begin{array}{lll} P & \neg P & \neg Q \\ Q & Q & \end{array}$$

Thus the possible paths through this matrix are $\{P, \neg P, \neg Q\}$, $\{P, Q, \neg Q\}$, $\{Q, \neg P, \neg Q\}$ and $\{Q, \neg Q\}$. Consolution shares with the connection method the idea of showing that every path contains a connection. Consolution does so by combining partial paths through a matrix to even longer partial paths and thereby ruling out paths containing a connection. The following tree is a proof tree in consolution. The nodes are marked with path sets, e.g. $\{\{P, Q\}, \{Q, \neg P\}, \{Q\}\}$ is a set with three partial paths through the two leftmost clauses in the matrix M. Now, in an inference the cross product of the elements of the parent nodes is built, and paths containing connections are deleted.



The root of this tree is the empty path set, which proves that all paths through M are complementary.

To introduce consolution formally we need only the following definitions.

A *connection* in a set of clauses or a matrix is a pair (K, L) of literals which can be made complementary by application of a substitution.

DEFINITION 2.1. **(Eder's Consolution (Eder, 1991))** A *path* is a finite set of literals.

If p and q are paths and $\mathcal{P}$ and $\mathcal{Q}$ are path sets, then $pq := p \cup q$ and $\mathcal{P} \cdot \mathcal{Q} := \{pq \mid p \in \mathcal{P} \text{ and } q \in \mathcal{Q}\}$. For ease of notation we write $p \cdot \mathcal{P}$ as an abbreviation for $\{p\} \cdot \mathcal{P}$. If C is a clause then its path set is $\mathcal{P}_C := \{\{L\} \mid L \in C\}$.

A path set $\mathcal{Q}$ is obtained from a path set $\mathcal{P}$ by *elimination of complementary paths* if there is a set of connections in $\mathcal{P}$ and a most general unifier σ of this set of connections such that $\mathcal{Q}$ is the set of non-complementary elements in $\mathcal{P}\sigma$. A path set $\mathcal{Q}$ is obtained from a path set $\mathcal{P}$ by *shortening of paths* if there is a surjective mapping $f : \mathcal{P} \to \mathcal{Q}$ such that $f(p) \subseteq p$ holds for all $p \in \mathcal{P}$.

A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *simplification (Version 0)*† if there exists a path set $\mathcal{Q}$ such that $\mathcal{Q}$ is obtained from $\mathcal{P}$ by elimination of complementary paths and such that $\mathcal{R}$ is obtained from $\mathcal{Q}$ by shortening of paths.

The inference rule *consolution* is defined as follows

$$\frac{\mathcal{P} \qquad \mathcal{Q}}{\mathcal{R}}$$

† Later on we will introduce different versions of simplification.

if there exists a variant $\mathcal{Q}'$ of $\mathcal{Q}$ which does not have variables in common with $\mathcal{P}$ such that $\mathcal{R}$ is obtained from $\mathcal{P} \cdot \mathcal{Q}'$ by simplification. $\mathcal{R}$ is called a *consolvent* of $\mathcal{P}$ and $\mathcal{Q}$. □

To recognize the soundness of the consolution inference rule consider again the explanation at the beginning of the previous section. In particular, multiplication of paths corresponds to the application of the law of distributivity, which is an equivalence transformation. Furthermore, if path sets are interpreted as conjunctions of disjunctions then shortening of paths preserves non-tautologyhood; or dually, if path sets are interpreted as disjunctions of conjunctions then shortening of paths preserves satisfiability.

DEFINITION 2.2. **(Derivation)** A *derivation* of a matrix M is a finite sequence $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ of path sets, where $n \geq 1$ and for all $k = 1, \ldots, n$ the set $\mathcal{P}_k$ equals $\mathcal{P}_C$ for some $C \in M$, or $\mathcal{P}_k$ is a consolvent of $\mathcal{P}_i$ and $\mathcal{P}_j$ for some $i, j < k$. A derivation ending in a path set $\mathcal{P}_n = \emptyset$ is also called a refutation.† □

The reader is invited to figure out in detail the derivation which led to the consolution proof tree given above.

THEOREM 2.1. *(Eder, 1991) A formula in disjunctive normal form is valid if and only if there is a refutation of its matrix by consolution.*

Eder gave a proof of this theorem which used the completeness of resolution. The proof is by constructing a consolution derivation for every resolution refutation, which gives the completeness of consolution (see theorem 8.1 in section 8).

3. Model Elimination and Consolution

In this section we will informally introduce a model elimination calculus. Then a formalization in a consolution-like language follows. Building on this, we define a more refined consolution method, which can then be shown to simulate stepwise the model elimination calculus.

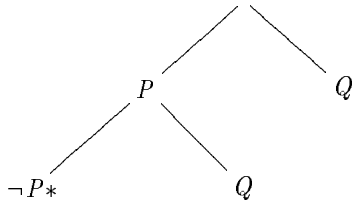
Model elimination was originally introduced by Loveland (cf. (Loveland, 1968; Loveland, 1969; Loveland, 1978)). It can be seen as a restricted form of linear resolution (see (Loveland, 1978) and section 7). However, model elimination and its linear proof format was invented *before* linear resolution, which came up in 1970. More recently it became obvious that model elimination can be understood very naturally as a matrix method; here we will follow the lines from (Letz et al., 1992) and define the inference rules as tree-transforming operators. Then the calculus is much in the spirit of semantic tableaux with unification for clauses (see (Fitting, 1990)), but with an important restriction. This restriction will be explained below and justifies using the new name — "tableau model elimination" — instead of qualifying it as "analytical tableaux for clauses with unification".

Compared to Loveland's model elimination, the calculus presented below is weaker, in that it lacks some efficiency improvements. Of course, these could be added, but for our purpose they are not essential and hence are omitted for the sake of simplicity.

As an example for model elimination in our format take the following tree, which is nothing more than a representation of the clause $\{P, Q\}$ of the matrix M in the consolution example.

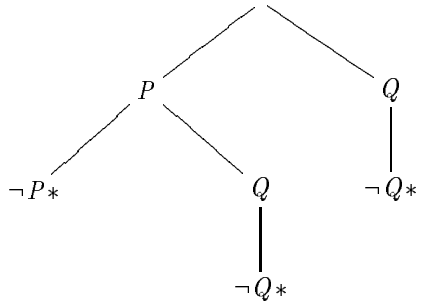
† Here we are not compatible with (Eder, 1991). What he calls "derivation" is called "refutation" by us; this was done to be compatible with subsequent calculi and to standard text books (e.g. (Chang and Lee, 1973)).

An *extension* step can be performed by selecting one branch of the tree, say the one leading to P, and an input clause that contains a literal which can be made complementary by unification with the leaf of that path; in the notation of matrix methods this is a connection. In our example the clause $\{\neg P, Q\}$ can be used to construct a new tree:



The selected path is extended with the literals of the clause as new leaves. As usual, the ground case for the connection is lifted to the general case by means of a most general unifier (MGU), and this unifier has to be applied to the entire tree. As a result of this operation, (at least) one extension of the selected path contains complementary literals and can be *closed*, i.e. it need not be considered further; this is marked with an asterisk. The other extensions remain unmarked. However, closing a branch can be achieved even without extending the tree with a new clause: whenever a leaf of a path has a connection with one of its ancestor literals, the path may be closed and the used MGU is applied to the entire tree. Such steps are called *reduction* steps. Unless the used MGU is empty there is a don't-know nondeterminism here wrt. the application of an extension step or a reduction step.

In our example, after two more extension steps we arrive at a tree which contains only closed paths, and thus serves as a refutation:



Let us use now the notation from consolution to describe tableau model elimination. Since consolution manipulates *sets of paths* our trees should be coded as consolution path sets. Then, neglecting already closed paths, for example, the second tree in the derivation above is naturally represented by the path set $\mathcal{P} = \{\{P, Q\}, \{Q\}\}$. For the extension step we have to select one path, say $\{P, Q\}$, and within that path we have to select the leaf as part of a connection. But which element of the set is the leaf? Due to *set* notation, the information about which literal is the leaf is lost. What we need are paths as *sequences* instead of sets:

DEFINITION 3.1. **(Path)** A path is a sequence of literals, written as $p = L_1 \circ \cdots \circ L_n$. L_n is called the *leaf* of p, which is also denoted by $leaf(p)$. By abuse of notation, '$\circ$' denotes also the append function for literal sequences.

The operations "elimination of complementary paths" and multiplication ("$\cdot$") are modified as expected.

A path q is obtained from a path $p = L_1 \circ \cdots \circ L_n$ by *immediate shortening of* p, $q \lhd p$, iff

$$\exists i (1 \leq i \leq n) : q = L_1 \circ \cdots \circ L_{i-1} \circ L_{i+1} \circ \cdots \circ L_n$$

For the transitive closure we say that q is obtained by *shortening of* p iff $q \lhd^+ p$.

The partial order $\leq$ on paths is defined as

$$p \leq q \quad \text{iff} \quad p = q \text{ or else } p \lhd^+ q$$

The definition of *shortening* of paths (cf. def. 2.1) is modified to take care of the new data structure: a path set $\mathcal{Q}$ is obtained from a path set $\mathcal{P}$ by *shortening of paths* if there is a surjective mapping $f : \mathcal{P} \rightarrow \mathcal{Q}$ such that $f(p) \leq p$ holds for all $p \in \mathcal{P}$. □

Now, when using sequences instead of sets we can identify a leaf in a path, i.e. in the first extension step of the above example we take in consolution the path P and multiply it with the path set $\{\neg P, Q\}$, which is derived from the clause $\{\neg P, Q\}$. This gives $\{P \circ \neg P, P \circ Q\}$; to get the closing effect from model elimination in consolution, we have to choose only connections we have just introduced during the multiplication. (In this example this is the only one). Then we have to apply the unifier for these connections to the path set and subsequently have to eliminate complementary paths. In our case the only complementary path we want to eliminate is the one which contained the connection chosen for this step.

Another property of tableau model elimination is that it only manipulates *one* path at a time, even if there are identical paths within a tableau. Thus, the paths of a tableau model elimination tree are technically a *multi*set. In consolution however we deal with *sets* of paths. In other words, multiple occurrences of identical paths in model elimination are collapsed into a single occurrence in consolution. As a consequence, a consolution inference may accidentally close several occurrences of identical paths in a single step — which is clearly not intended (at least for this simulation). As a solution we offer:

In the following *path sets* are *multisets of paths*.

Next we want to treat the mapping from tableau model elimination to consolution more rigorously. For this we first introduce tableau model elimination in a formal way. This is done in a consolution-like language. Then we describe the necessary changes to consolution.

DEFINITION 3.2. **(Tableau Model Elimination)** Given a matrix M.

The inference rule *extension* is defined as follows:

$$\frac{\mathcal{P} \cup \{p\} \qquad \mathcal{P}_C}{\mathcal{R}}$$

where

$\mathcal{P} \cup \{p\}$ is a path set, and C is a variable disjoint variant of a clause in M, and there is a literal $L \in C$ such that $(leaf(p), L)$ is a connection with MGU σ, and

$\mathcal{R} = (\mathcal{P} \cup (p \cdot \mathcal{P}_{C-\{L\}}))\sigma$

The inference rule *reduction* is defined as follows:

$$\frac{\mathcal{P} \cup \{p\}}{\mathcal{P}\sigma}$$

where

$\mathcal{P} \cup \{p\}$ is a path set, and
there is a literal L in p such that $(L, \mathit{leaf}(p))$ is a connection with MGU σ.

A sequence $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ is called a *model elimination derivation* iff

$\mathcal{P}_1$ is a path set $\mathcal{P}_C$, with C in M, and
$\mathcal{P}_{i+1}$ is obtained from $\mathcal{P}_i$ by means of an extension step with an appropriate $\mathcal{P}_C$, or
$\mathcal{P}_{i+1}$ is obtained from $\mathcal{P}_i$ by means of a reduction step.

The path p is called *selected path* in both inference rules. □

In order to simulate the closing of tableau model elimination of only one path at a time, in consolution we make explicit the step of fixing the set of connections:

DEFINITION 3.3. **(Spanning MGU)** A substitution σ is a *spanning MGU for a path set* $\mathcal{Q}$ iff there is a set of connections, one connection in each path of $\mathcal{Q}$, such that σ is a most general substitution unifying each of these connections, i.e., making the two literals of each of these connections syntactically complementary. □

Building on this, simplification is modified:

DEFINITION 3.4. **(Simplification, Version 1)** A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *simplification* iff

A) there exists a spanning MGU σ for some subset $Q \subseteq \mathcal{P}$, and
B) $\mathcal{P}^B$ is obtained from $\mathcal{P}\sigma$ by deleting zero or more paths containing complementary literals, and
C) $\mathcal{R}$ is obtained from $\mathcal{P}^B$ by shortening of zero or more paths.

□

The definition of consolution, Version 1 is literally the same as for consolution, with the only exception that the modified simplification, Version 1 is used.

However, this modification does not yet suffice to prove that consolution stepwisely simulates model elimination. The problem is that in tableau model elimination during an extension of $\mathcal{P} \cup \{p\}$ we must extend only p. In consolution, however, by multiplication all paths from $\mathcal{P} \cup \{p\}$ will be lengthened. As a consequence all paths from $\mathcal{P}$ will also be lengthened, and, in a subsequent simplification step they can be shortened again. This however leads to duplication of paths, and therefore these multiple occurrences must be eliminated again. This leads to an extra operation in simplification:

DEFINITION 3.5. **(Simplification, Version 2)** A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *simplification, Version 2* iff

A) there exists a spanning MGU σ for some subset $Q \subseteq P$, and
B) $\mathcal{P}^B$ is obtained from $\mathcal{P}\sigma$ by deleting zero or more paths containing complementary literals, and
C) $\mathcal{P}^C$ is obtained from $\mathcal{P}^B$ by shortening of zero or more paths, and
D) $\mathcal{R}$ is obtained from $\mathcal{P}^C$ in the following way: for every path $p \in \mathcal{P}^C$, zero or more, but not all, paths are deleted that are equal to p as a set of literals.

□

DEFINITION 3.6. **(Sequence Consolution)** The inference rule *sequence consolution* is defined as follows

$$\frac{\mathcal{P} \qquad \mathcal{Q}}{\mathcal{R}}$$

if there exists a variant $\mathcal{Q}'$ of $\mathcal{Q}$ which does not have variables in common with $\mathcal{P}$ such that $\mathcal{R}$ is obtained from $\mathcal{P} \cdot \mathcal{Q}'$ by simplification, Version 2. $\mathcal{R}$ is called a *sequence consolvent* of $\mathcal{P}$ and $\mathcal{Q}$.

The definition of a *sequence consolution derivation* is the same as for a consolution derivation, except that the sequence consolution inference rule is used. □

Deletion of duplicate paths (the term "equal as a set of literals" in item D in definition 3.5) is slightly more general than needed for the simulation of tableau model elimination. This will become clear in the next section.

The next theorem shows that sequence consolution is a framework to express tableau model elimination. Together with the completeness of tableau model elimination (Baumgartner, 1992) the theorem also yields completeness of sequence consolution.

THEOREM 3.1. **(Sequence Consolution Simulates Tableau Model Elimination)** *Let M be a matrix and $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ a tableau model elimination derivation. Then there exist variants of clauses $C_1, \ldots, C_n$ from M, such that $(\mathcal{P}_1, \mathcal{P}_{C_1} \ldots, \mathcal{P}_{n-1}, \mathcal{P}_{C_{n-1}}, \mathcal{P}_n)$ is a sequence consolution derivation of M.*

PROOF. Let $\mathcal{P}_{i+1}$ be derived by extension with $\mathcal{P}_i$ and $\mathcal{P}_{C_i}$; we construct the following sequence consolvent:
$\mathcal{P}_i$ has the form $\mathcal{P}_i' \cup \{p\}$ and C_i is a variant of a clause in M, hence $\mathcal{P}_i$ and $\mathcal{P}_{C_i}$ do not have variables in common. According to the sequence consolution inference rule we may then multiply these two to obtain $\mathcal{P}_i \cdot \mathcal{P}_{C_i} = (\mathcal{P}_i' \cup \{p\}) \cdot \mathcal{P}_{C_i} = (\mathcal{P}_i' \cdot \mathcal{P}_{C_i}) \cup (p \cdot \mathcal{P}_{C_i})$. Using simplification Version 2 from definition 3.5 we build:

A) σ is obtained by unifying the connection $(L, leaf(p))$, which was used in the extension step.
B) $\mathcal{P}^B$ is obtained from $(\mathcal{P}_i \cdot \mathcal{P}_{C_i})\sigma$ by deleting the path $(p \cdot \{L\})\sigma$ from $(p \cdot \mathcal{P}_{C_i})\sigma$; thus $\mathcal{P}^B = ((\mathcal{P}_i' \cdot \mathcal{P}_{C_i}) \cup (p \cdot \mathcal{P}_{C_i - \{L\}}))\sigma$
C,D) from $\mathcal{P}^B$ we shorten all paths from $(\mathcal{P}_i' \cdot \mathcal{P}_{C_i})\sigma$ and delete multiple occurrences to obtain $\mathcal{P}_i'\sigma$ which finally gives $\mathcal{R} = (\mathcal{P}_i' \cup (p \cdot \mathcal{P}_{C_i - \{L\}}))\sigma$

Clearly $\mathcal{R}$ is the result of the extension step as well, i.e. $\mathcal{R} = \mathcal{P}_{i+1}$. The case in which $\mathcal{P}_{i+1}$ is obtained by a reduction step is handled analogously. Here any variable disjoint variant C_i can be

used to perform a sequence consolution step; by appropriate shortening and deletion of multiple occurrences of paths, the undesired path extensions with C_i can be cancelled again. □

4. Consolution and Sequence Consolution

Sequence consolution is a true generalization of consolution with respect to data structures and inference rules. Thus, any consolution derivation can be reflected in a sequence consolution derivation. For the case of data structures, the *set* data structure of path sets in consolution can be simulated by deletion of duplicate paths (simplification, step D) in sequence consolution. Deletion of duplicate paths is defined without regard to the ordering of literals. This is important for the following reason: in sequence consolution a path set might contain the two different paths $P \circ Q$ and $Q \circ P$, while in consolution these are collapsed into the single path $\{P, Q\}$. But then in sequence consolution we can delete one of the two paths and will thus be compatible with consolution.

Similarly, appropriate shortening of paths (simplification, step C) in sequence consolution simulates the set data structure of paths in consolution. With that we obtain the following trivial theorem:

THEOREM 4.1. **(Sequence Consolution Simulates Consolution)** *Let $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a consolution derivation. Then there is a sequence consolution derivation $(\mathcal{P}'_1, \ldots, \mathcal{P}'_n)$ of the same matrix such that for every $i = 1 \ldots n$ there exists a bijective mapping $f_i : \mathcal{P}_i \mapsto \mathcal{P}'_i$ such that for every $p \in \mathcal{P}_i$, p and $f_i(p)$ are equal as multisets of literals.*

The converse of this theorem does, of course, not hold. This results from the following facts: firstly, due to set data structures, consolution collapses multiple occurrences into one occurrence; thus, in the course of a sequence consolution refutation more paths have to be eliminated. Secondly, the consolution inference deletes *all* complementary paths, whereas sequence consolution deletes only *some*; thus, in a sequence consolution refutation more inferences may have been employed. However a weaker notion of simulatability can be established (theorem 4.2 below). This will be done next. As a technical preliminary, we have to say how to relate data structures of consolution and sequence consolution: if a sequence appears where a multiset is required, the translation from sequences to multisets is done in the obvious way; similarly, if a multiset appears where a set is required, the translation is also done in the obvious way. For example, if $\mathcal{P}$ is a path set in sequence consolution, i.e. a multiset of sequences, and $\mathcal{P}'$ is a path set in consolution, i.e. a set of sets, then $\mathcal{P}' \subseteq \mathcal{P}$ means the subset relation where $\mathcal{P}$ is mapped in the above way to a set of sets.

THEOREM 4.2. **(Consolution Simulates Sequence Consolution)** *Let M be a matrix and $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a sequence consolution derivation of M. Then there is an integer $m \leq n$ and there is a consolution derivation $(\mathcal{P}'_1, \ldots, \mathcal{P}'_m)$ of M and there is a monotonic mapping ϕ such that for every $k = 1 \ldots n$ there is a substitution δ_k and $\mathcal{P}'_{\phi(k)}\delta_k \subseteq \mathcal{P}_k$.*

The consolution derivation may be shorter than the sequence consolution derivation, since in sequence consolution some complementary paths may persist through simplification, whereas they are definitively removed in consolution. These paths in the sequence consolution derivation may furthermore be used in inference steps that do not have a counterpart in the consolution derivation. The consolution derivation pauses meanwhile; however, these steps further instantiate the path set, giving the need for the substitution δ_k.

PROOF. (Theorem 4.2) We show that a consolution derivation of the *same* length exists, for which the theorem holds when ϕ is the identity function. This derivation may contain consecutive identical elements $\mathcal{P}'_i, \mathcal{P}'_{i+1}, \ldots, \mathcal{P}'_{i+j}$. Clearly, we may drop $\mathcal{P}'_{i+1}$ to $\mathcal{P}'_{i+j}$ and redefine ϕ so that $\phi(i+1) = \phi(i+2) = \cdots \phi(i+j) = i$, and the theorem will hold for the resulting derivation. The proof is by induction on the length n of the sequence consolution derivation.

Base case: If $n = 1$ then the sequence consolution derivation $(\mathcal{P}_1)$ consists of a single path set $\mathcal{P}_C$ (a multiset of sequences) for some $C \in M$. Let the consolution derivation be $(\mathcal{P}'_1)$ which consists of a single path set $\mathcal{P}_C$ (a set of sets). When read as a set of sets, it evidently holds that $\mathcal{P}_1$ equals $\mathcal{P}'_1$. Hence with $\delta_1 := \epsilon$ the claim holds.

Induction step: Suppose the result to hold for all sequence consolution derivations $(\mathcal{P}_1, \ldots, \mathcal{P}_{n-1})$ (for some $n > 1$). Thus there exists a consolution derivation (of the same length) $(\mathcal{P}'_1, \ldots, \mathcal{P}'_{n-1})$ for M and for $k = 1 \ldots n-1$ there is a substitution δ_k and $\mathcal{P}'_k\delta_k \subseteq \mathcal{P}_k$.

We have to show that the result holds for $\mathcal{P}'_n$ as well, i.e. that a substitution δ_n such that $\mathcal{P}'_n\delta_n \subseteq \mathcal{P}_n$ exists. According to the definition of derivation, we distinguish two cases:

Subcase 1: $\mathcal{P}_n$ is the path set $\mathcal{P}_C$ of a clause $C \in M$. In this case the same argumentation applies as for the base case.

Subcase 2: $\mathcal{P}_n$ is obtained as a sequence consolvent of some $\mathcal{P}_i$ and $\mathcal{P}_j$ with $i, j < n$. Without loss of generality assume that $\mathcal{P}_j$ is already variable disjoint from $\mathcal{P}_i$. We will trace through the simplification of the product $\mathcal{P} := \mathcal{P}_i \cdot \mathcal{P}_j$ and construct an appropriate consolvent of $\mathcal{P}'_i$ and $\mathcal{P}'_j$. Assume also that $\mathcal{P}'_i$ and $\mathcal{P}'_j$ are variable disjoint and hence can be used for the multiplication $\mathcal{P}'_i \cdot \mathcal{P}'_j$ without renaming.

In the simplification of $\mathcal{P}$ in the first step (Step A) a spanning MGU σ for some subset of $\mathcal{P}$ is determined and applied to $\mathcal{P}$. Hence let $\mathcal{P}^A := \mathcal{P}\sigma$.

Now for the intended consolution step we build first the product $\mathcal{P}' := \mathcal{P}'_i \cdot \mathcal{P}'_j$. Since $\mathcal{P}'_i$ and $\mathcal{P}'_j$ are variable disjoint, the domains of δ_i and δ_j can be supposed to be disjoint. Hence $\mathcal{P}'_i\delta_i = \mathcal{P}'_i\delta_i\delta_j$ and $\mathcal{P}'_j\delta_j = \mathcal{P}'_j\delta_i\delta_j$. It follows

$$\begin{aligned} \mathcal{P}'\delta_i\delta_j &= (\mathcal{P}'_i \cdot \mathcal{P}'_j)\delta_i\delta_j \\ \text{(Consequence of def. of product)} \quad &= \mathcal{P}'_i\delta_i\delta_j \cdot \mathcal{P}'_j\delta_i\delta_j \\ &= \mathcal{P}'_i\delta_i \cdot \mathcal{P}'_j\delta_j \end{aligned} \tag{4.1}$$

By the induction hypothesis we learn that substitutions δ_i and δ_j exist such that $\mathcal{P}'_i\delta_i \subseteq \mathcal{P}_i$ and $\mathcal{P}'_j\delta_j \subseteq \mathcal{P}_j$. From this and (4.1) it follows

$$\mathcal{P} = \mathcal{P}_i \cdot \mathcal{P}_j \supseteq \mathcal{P}'_i\delta_i \cdot \mathcal{P}'_j\delta_j \overset{(4.1)}{=} \mathcal{P}'\delta_i\delta_j \tag{4.2}$$

Applying σ to both sides yields

$$\mathcal{P}^A = \mathcal{P}\sigma \supseteq \mathcal{P}'\delta_i\delta_j\sigma \tag{4.3}$$

Next, in step B of simplification in sequence consolution, zero or more paths containing complementary literals are deleted from $\mathcal{P}^A$, which yields $\mathcal{P}^B$. We distinguish two disjoint cases now in order to determine the operation "elimination of complementary paths" for the consolution step:

Case 1: Only paths which are not contained in $\mathcal{P}'\delta_i\delta_j\sigma$ are deleted from $\mathcal{P}^A$, i.e. it holds $\mathcal{P}^B \supseteq \mathcal{P}'\delta_i\delta_j\sigma$. In this case consider the empty set of connections for "elimination of complementary paths" in $\mathcal{P}'$. Clearly $\sigma' := \epsilon$ is an MGU for this set. Let $\mathcal{P}'^B$ be the path set obtained from $\mathcal{P}' = \mathcal{P}'\sigma'$ by removing all the paths containing complementary literals. Finally define $\delta_n := \delta_i\delta_j\sigma$. Using this definition $\mathcal{P}'^B\delta_n \subseteq \mathcal{P}^B$ holds

In case 2, the negation of case 1, some paths are deleted from $\mathcal{P}^A$ that are also contained in $\mathcal{P}'\delta_i\delta_j\sigma$. Define the path set $\mathcal{Q}' \subseteq \mathcal{P}'$, which corresponds to the deleted paths in $\mathcal{P}^A \setminus \mathcal{P}^B$ ($\mathcal{Q}'$

will be subject to step A in simplification):

$$\mathcal{Q}' := \{q \in \mathcal{P}' \mid q\delta_i\delta_j\sigma \in \mathcal{P}^A \setminus \mathcal{P}^B\} \tag{4.4}$$

Every deleted path in $\mathcal{P}^A \setminus \mathcal{P}^B$ contains complementary literals. By definition of $\mathcal{Q}'$ every path in $\mathcal{Q}'\delta_i\delta_j\sigma$ is contained in $\mathcal{P}^A \setminus \mathcal{P}^B$ and hence also contains complementary literals. Thus an MGU σ' and a substitution δ_n exist such that $\delta_i\delta_j\sigma = \sigma'\delta_n$. Since, by definition, $\mathcal{Q}' \subseteq \mathcal{P}'$ we can continue the simplification of the consolution step with $\mathcal{P}'$ by selecting $\mathcal{Q}'$ for elimination of complementary paths, where the MGU used is σ'. Let $\mathcal{P}'^B$ be the path set obtained from $\mathcal{P}'\sigma'$ by removing all the paths containing complementary literals, i.e. at least the elements of $\mathcal{Q}'\sigma'$ are removed. Using these definitions it follows from the definition of $\mathcal{Q}'$ and with (4.3) that $\mathcal{P}'^B\delta_n \subseteq \mathcal{P}^B$. This concludes case 2 (with the same result as case 1).

The next step in simplification is shortening of paths. Let $\mathcal{P}^C$ be obtained from $\mathcal{P}^B$ by shortening of the paths in $\mathcal{Q} \subseteq \mathcal{P}^B$. Define a corresponding subset $\mathcal{Q}'' \subseteq \mathcal{P}'^B$ as

$$\mathcal{Q}'' := \{q \in \mathcal{P}'^B \mid q\delta_n \in \mathcal{Q}\}$$

Since $\mathcal{P}'^B\delta_n \subseteq \mathcal{P}^B$ we can shorten the paths given by $\mathcal{Q}''$ in $\mathcal{P}'^B$ in such a way that for the resulting path set $\mathcal{R}'$ it holds $\mathcal{R}'\delta_n \subseteq \mathcal{P}^C$. Note that $\mathcal{R}'$ is the result of the consolution step. To obtain the result of the sequence consolution step, $\mathcal{P}_n$, zero or more, but not all duplicate paths are deleted from $\mathcal{P}^C$. But since for every path each of its duplicate paths when read as a set is collapsed into a single set of sets when compared to the paths in consolution, deletion of some, but not all duplicate paths preserves $\mathcal{R}'\delta_n \subseteq \mathcal{P}_n$. Thus we can set $\mathcal{P}'_n := \mathcal{R}'$ to complete the proof. □

The last two theorems have established the correspondence between our two versions of consolution, *consolution* and *sequence consolution*. The theorems might suggest that both versions are equivalent on a certain level of abstraction. But this rough equivalence does not help to solve our main goal: recall from the introduction that we are mainly interested in expressing (defining) other calculi, such as model elimination, in the consolution framework. Technically this means, again for model elimination, that we have to find a bijection between the traditional model elimination derivations and the model elimination expressed in the consolution framework. This approach, however, does not work for Eder's consolution (see the previous section about model elimination for a discussion of the reasons why). Sequence consolution offers a solution.

5. Model Elimination with Chains

In this section we will investigate an original chain-based model elimination calculus, as it is given in (Loveland, 1978). Among the several variants presented there we have chosen a version called "weak model elimination"; weak model elimination is the same as "model elimination" but lacks a factorization rule (which is not necessary for completeness).

In order to distinguish this calculus from the one given in the previous section we will prefix the inference rules with "ME".

The main data structure is a *chain*, i.e. a sequence of literals of distinguished type. Literals may be of type A or of type B. Chains are written by juxtaposing their literals, and A-literals are written in brackets. Example: the chain $D[E]F$ contains the B-literals D and F and the A-literal E.

A chain is called *admissible* iff

1 complementary B-literals are separated by an A-literal;

2 no B-literal is to the right of an identical A-literal;
3 no A-literal is identical or complementary to another A-literal; and
4 the rightmost literal is a B-literal.

To simulate those chains in the sequence consolution calculus we need a further refinement of our principal data structure:

In the following we have to use *sequences of paths* instead of multisets.

Admissible chains can be understood as trees, where the A-literals are inner nodes and B-literals are leaves:

DEFINITION 5.1. Let $\mathcal{P} = (p_1, \ldots, p_m)$ and $\mathcal{Q} = (q_1, \ldots, q_n)$ be sequences of paths. Then $\mathcal{P} \bullet \mathcal{Q}$ denotes their concatenation $(p_1, \ldots, p_m, q_1, \ldots, q_n)$, and the dot product (cf. also the corresponding definition 2.1 for consolution) is the path sequence

$$\mathcal{P} \cdot \mathcal{Q} := (p_1 \circ q_1, \ldots, p_1 \circ q_n, \ldots, p_m \circ q_1, \ldots, p_m \circ q_n)$$

Let $a_1 b_1 \ldots a_n b_n$ be an admissible chain, where a_i (resp. b_i) is a sequence of A-(resp. B-)literals. The transformation $\mathcal{M}$ transforms an admissible chain into a path set which represents a tree (depicted in figure 2):

$$\begin{array}{lcl} \mathcal{M}(()) & = & () \\ \mathcal{M}(a_i K) & = & (a_i) \cdot \mathcal{M}(K) \\ \mathcal{M}(b_i K) & = & ((b_i^1), \ldots, (b_i^{k_i})) \bullet \mathcal{M}(K), \\ & & \text{where } b_i = (b_i^1, \ldots, b_i^{k_i}) \end{array}$$

□

Such degenerated, linear shaped trees are called *ferns* in (Baumgartner et al., 1992). More formally, ferns are trees, where every inner node lies on one single path from the root to a leaf. The following lemma can be shown after rigorous formalization by induction on the length of chains:

LEMMA 5.1. *Let c be an admissible chain. Then $\mathcal{M}(c)$ is a fern.*

Take as an example the chain $Q[P]Q$, where the A-literal is in brackets. This chain is admissible and

$$\mathcal{M}(Q[P]Q) = (Q, P \circ Q)$$

is the corresponding path sequence. This is obviously one of the trees we depicted on page 6 with the closed path being omitted (and right and left interchanged).

Note that this interpretation of chains is based on the rightmost strategy used by the model elimination calculus of Loveland. More precisely, it is the transformation $\mathcal{M}$ which takes into account the way the chains are handled by Loveland's ME procedure.

From the above definition we immediately conclude that if C is a clause, then $\mathcal{P}_C = \mathcal{M}(K)$, where K is a chain containing only B-literals, which are exactly the literals from C.

Admissibility after the transformation $\mathcal{M}$ reads as follows:

PROPOSITION 5.1. *If K is an admissible chain, then for $\mathcal{M}(K) = (p_1, \ldots, p_n)$ the following holds:*

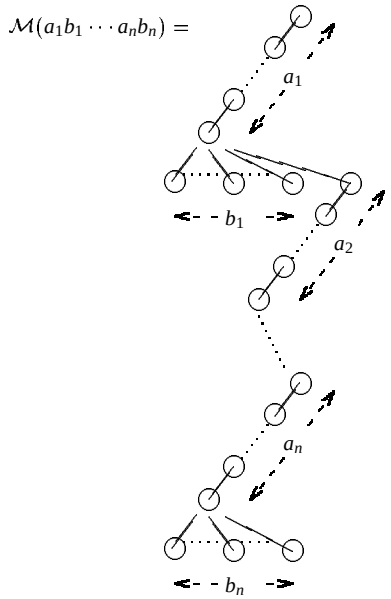


Figure 2. Result of the transformation $\mathcal{M}$ depicted as a tree.

1. *There is no i and j ($i \neq j$) such that p_i and p_j differ only in their leaf nodes, where those leaves are complementary;*
2. *there is no path, such that its leaf is identical to another element of the same path;*
3. *there is no path $p \circ L$, such that there are two identical or complementary literals in p.*

Another property we need below concerns the special nature of ferns:

LEMMA 5.2. *Let K be an admissible chain and $\mathcal{M}(K) = (p_1, \ldots, p_n)$ such that $leaf(p_i) = leaf(p_j)$ for some i, j with $i < j$. Then p_i can be obtained from p_j by shortening, i.e. $p_i \lhd^+ p_j$.*

PROOF. By lemma 5.1 $(p_1, \ldots, p_n)$ is a fern. Since every inner node of the fern $(p_1, \ldots, p_n)$ is an element of the same path, all elements of p_i except its leaf are elements of p_j. By assumption, $leaf(p_i)$ is an element of p_j, which proves the lemma. □

The inference rules of the model elimination calculus with chains are *ME-extension* and *ME-reduction*. They all make use of an *ME-accepting transformation T* on chains. This transformation performs reduction and factorizations for which no unifier is necessary:

DEFINITION 5.2. Let K be a chain, then T maps $\mathcal{M}(K)$ to a sequence of paths by the following modifications of $\mathcal{M}(K)$:

a) if $\mathcal{M}(K)$ contains two paths with identical leaves, then delete the rightmost ("ground factorization").
b) if $\mathcal{M}(K)$ contains a path with complementary literals, delete this path ("ground reduction").

□

In the original definition from (Loveland, 1978) there is an additional case, which allows the elimination of certain A-literals to the right of the right-most B-literal. Since we are dealing with an explicit representation of paths this is not necessary in our case. It is left as a simple exercise to the reader to show that the transformation T as defined above for sequences of paths is identical to the one given in Loveland's' calculus for chains.

The inference rules are similar to the ones of tableau model elimination, namely extension and reduction.

DEFINITION 5.3. Given a matrix M and an admissible chain K with $\mathcal{M}(K) = (p_1, \ldots, p_n)$.

The inference rule *ME-extension* is defined by:

$$\frac{(p_1, \ldots, p_n) \qquad \mathcal{P}_C}{T(\mathcal{R})}$$

where

C is a variable disjoint (from $p_1, \ldots, p_n$) variant of a clause from M
$(\mathit{leaf}(p_n), L)$ is a connection with MGU σ, where L which is the last (rightmost) literal of the sequence $\mathcal{P}_C$,
$\mathcal{R} = (p'_1, \ldots, p'_{m-1}, p'_m \cdot \mathit{rearrange}(T(\mathcal{P}_{C-\{L\}}\sigma)))$
with $(p'_1, \ldots, p'_m) = T(p_1\sigma, \ldots, p_n\sigma)$
and $(p'_1, \ldots, p'_m)$ is admissible
and $p'_m = p_n\sigma$ and $T(\mathcal{P}_C\sigma)$ still contains $L\sigma$ and is admissible.
The function *rearrange* sorts the paths of $T(\mathcal{P}_{C-\{L\}}\sigma)$, which all have length 1, according to a given order. This will be discussed below.

The inference rule *ME-reduction* is defined by:

$$\frac{(p_1, \ldots, p_n)}{T(p_1\sigma, \ldots, p_n\sigma)}$$

where

there is an L in p_n, such that $(L, \mathit{leaf}(p_n))$ is a connection with MGU σ.

□

We omitted an important feature of this sort of model elimination, namely the use of an *ordering rule*. It is explained in detail in (Loveland, 1978); for our purposes it is sufficient to note that it determines the order of these literals, which are added to a path in an extension step. The function *rearrange* in the ME-extension can be defined in a way which expresses the meaning of the ordering rule.

Analogously to the previous section, we use the notion of an *ME derivation* .

As an example take the Matrix $M = \{Pxqx, Px\neg Qx, \neg Px\}$ and the two sequences of (unit-)paths (Px, Qx) and $(Py, \neg Qy)$. The intermediate sequence of paths $\mathcal{R}$ is

$$\mathcal{R} = (Px, Qx \circ Px)$$

An application of the transformation T to this sequence deletes the rightmost path. Hence

we get $T(\mathcal{R}) = (Px)$ as the result of one *ME-extension* step. In the tableau model elimination calculus an extension step would give $\mathcal{R} = (Px,\ Qx \circ Px)$; the "ground factorization" performed by the T-operation, however, can only be simulated by an additional deduction, which eliminates the path $Qx \circ Px$.

THEOREM 5.1. **(Sequence Consolution Simulates ME)** *Let M be a Matrix and $(\mathcal{P}_1, \ldots, \mathcal{P}_s)$ an ME derivation. Then there are variants of clauses $C_1, \ldots, C_{s-1}$ from M, such that $(\mathcal{P}_1, \mathcal{P}_{C_1}, \ldots, \mathcal{P}_{s-1}, \mathcal{P}_{C_{s-1}}, \mathcal{P}_s)$ is a sequence consolution derivation of M.*

PROOF. In the first case let $\mathcal{P}_{i+1}$ be derived from $\mathcal{P}_i = (p_1, \ldots, p_n)$ by an ME-extension step with "auxiliary chain" $\mathcal{P}_{C_i}$. We construct a sequence consolvent by first multiplying path sequences:

$$\mathcal{P}_i \cdot \mathcal{P}_{C_i} = (p_1 \cdot \mathcal{P}_{C_i}, \ldots, p_n \cdot \mathcal{P}_{C_i})$$

We perform the following simplification:

A) The substitution σ is obtained by unifying the connection $(L, leaf(p_n))$, which was used in the extension step, where L is the rightmost literal of $\mathcal{P}_{C_i}$.

B) – From $(p_n \cdot \mathcal{P}_{C_i})\sigma$ we can delete the complementary path $(p_n \cdot (L))\sigma$ to obtain

$$(p_1 \cdot \mathcal{P}_{C_i}, \ldots, p_{n-1} \cdot \mathcal{P}_{C_i}, p_n \cdot \mathcal{P}_{C_i - \{L\}})\sigma =$$
$$(p_1\sigma \cdot \mathcal{P}_{C_i}\sigma, \ldots, p_{n-1}\sigma \cdot \mathcal{P}_{C_i}\sigma, p_n\sigma \cdot \mathcal{P}_{C_i - \{L\}}\sigma)$$

The result is depicted in figure 3.

– From this path sequence we delete those paths which contain a ground connection in the prefix $p_i\sigma$, $1 \leq i \leq n$ and get:

$$(p'_1 \cdot (\mathcal{P}_{C_i}\sigma), \ldots, p'_{r-1} \cdot (\mathcal{P}_{C_i}\sigma), p'_r \cdot (\mathcal{P}_{C_i - \{L\}}\sigma))$$

Note, that by definition of the ME-extension rule $p_n\sigma$ does not have a connection, hence we know $p'_r = p_n\sigma$.

C) Now we shorten

- paths to obtain

$$(p'_1, \ldots, p'_{r-1}, p'_r \cdot (\mathcal{P}_{C_i - \{L\}}\sigma))$$

- from this we shorten all p'_j for which a path p'_i exists with $i < j$ and $leaf(p'_j) = leaf(p'_i)$, such that $p'_j = p'_i$. This is possible because of lemma 5.2.

D) From the above results we delete rightmost multiple occurrences of paths, to obtain finally

$$T(\,T(p_1\sigma, \ldots, p_{n-1}\sigma) \circ (p_n\sigma \cdot\ T(\mathcal{P}_{C_i - \{L\}}\sigma)))$$

which is exactly the sequence $T(\mathcal{R})$ from the definition of the ME-extension rule. Note that by the outermost application of T only paths with identical leaves are deleted, paths containing complementary literals are already deleted by the inner applications. These paths are already eliminated by the second shortening in step C).

In the other case let $\mathcal{P}_{i+1}$ be derived from $\mathcal{P}_i = (p_1, \ldots, p_n)$ by an ME-reduction step. Let C_i be an arbitrary "auxiliary chain". There is a σ, such that $p_n\sigma$ is complementary. As before, we construct a consolvent by first multiplying path sequences:

$$\mathcal{P}_i \cdot \mathcal{P}_{C_i} = (p_1 \cdot \mathcal{P}_{C_i}, \ldots, p_n \cdot \mathcal{P}_{C_i})$$

From $(p_1 \cdot \mathcal{P}_{C_i}, \ldots, p_n \cdot \mathcal{P}_{C_i})\sigma$ we delete those complementary paths $(p_j \cdot \mathcal{P}_{C_i})\sigma$, which contain

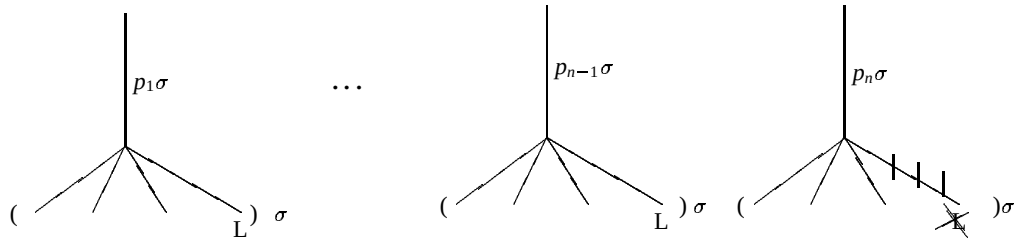


Figure 3. A sequence of paths from theorem 5.1.

their connection in the prefix $p_j\sigma$. Next we shorten paths and delete those multiple occurrences such that we finally obtain

$$T(p_1\sigma, \ldots p_n\sigma).$$

□

Let us conclude this section with remarks on a comparison of model elimination and tableau model elimination. In tableau model elimination extension and reduction steps are possible on any path from the tableau $\mathcal{P}$. ME-extension and ME-reduction steps, however, are only possible on the last path from $\mathcal{P}$, where $\mathcal{P}$ is understood as a sequence of paths. Another significant difference is that model elimination applies the T-operation, which corresponds to ground factorization and ground reduction at various opportunities within one application of an inference rule. In tableau model elimination these steps have to be simulated by additional inference steps. It is interesting to note that consolution has a direct counterpart to the ground-reduction part of the T-operation. After applying the MGU to a path set, all complementary paths have to be deleted, which is exactly the effect of a T-operation.

6. The Connection Method

As mentioned in the introduction, consolution is intended to bridge the gap between matrix methods, such as the connection method, and resolution. Hence, it should be possible to carry on in the spirit of the previous sections and show that consolution simulates the connection method. Such a result was not established in (Eder, 1991) and will be done in this section. As another interesting aspect we will discuss the relation of the connection method to tableau model elimination.

In order to do all this, one has to lay down the connection calculus. However, according to (Bibel, 1987) the connection method should not be understood as a single calculus, but as a methodology to design calculi; every calculus that proceeds by discovering a connection in every path through a given matrix (i.e. finds a "spanning set of connections") should be considered as a connection method. Clearly, for our purpose we have to commit ourselves to a single such "connection calculus". Here we will define a connection calculus that is very similar to the one in (Eder, 1992).

THE CONNECTION METHOD AND TABLEAU MODEL ELIMINATION

The example in section 3 can serve to illustrate the connection method in relation to tableau model elimination. The sequence of the trees depicted there can also be read as a connection method refutation. In general the following differences have to be obeyed:

1 In selecting the next branch to be extended, only a *longest* branch among the collection of open branches may be considered. This corresponds to the stack-like search organization in Eder's calculus. In other words, when deleting the branches already closed, every tree in a derivation then takes the form of a fern.
2 The clauses in Eder's calculus are sets. As a consequence, identically labeled branches have to be collapsed into a single occurrence.
3 Eder's calculus permits several of the new branches to be closed in one single extension step, i.e. several connections may be established in the step. This corresponds in tableau model elimination to one single extension step (with one single connection) followed by a sequence of reduction steps. This correspondence will be put more precisely below. A word of warning: the set of connections is allowed to be empty. Thus we may extend without closing any branch. Such steps have no counterpart in model elimination.
4 Most importantly, with regard to the search space, a connection need not be established between a literal of the extending clause and the *leaf* of the extended branch, but any other literal in the branch will do as well. Thus the search space in this connection method is much broader.

These observations are reflected in the following definition.

DEFINITION 6.1. **(Connection Calculus)** Given a matrix M. A *path* is a sequence of literals. A *path set* is a set of paths. The inference rule *(connection) extension* is defined as follows:

$$\frac{\mathcal{P} \cup \{p\} \qquad \mathcal{P}_C}{\mathcal{R}}$$

where

$\mathcal{P} \cup \{p\}$ is a path set with longest branch p, and
C is a variable disjoint variant of a clause in M, and
let $D \subseteq C$ (D possibly empty) such that there is a set of connections between every literal in D and literals in p with MGU σ, and
$\mathcal{R} = (\mathcal{P} \cup (p \cdot \mathcal{P}_{C-D}))\sigma$

The path p is called the *selected path*. *Derivation* is defined with respect to the single inference rule "connection extension" as for model elimination (def. 3.2). □

Note the differences between *connection extension* and extension in tableau model elimination, which mirror the above informal consideration. In particular, D is a set of several literals and the paths resulting from the elements in D are all closed. Thus the connection calculus extension step can be viewed as a "macro" combination of a single tableau model elimination extension step, followed by some reduction steps with the remaining literals. This observation is the key for the stepwise simulation of tableau model elimination by the connection method. However, some precautions must be taken to achieve this goal: first, model elimination must be restricted

to work on a longest branch. This is motivated by the longest-branch restriction of extension in the connection method. The second point is more serious and concerns the macroscopic nature of connection extension: suppose we have executed a model elimination extension step E which results in the new open paths $p_1, \ldots, p_n$. In the subsequent refutation each of these paths will either be closed by a reduction step or will be extended to closed paths. Now, in order to enable the stepwise simulation by a connection extension step, tableau model elimination is restricted in the following way: if in the subsequent derivation there is an extension step on one of the paths $p_1, \ldots, p_n$, then every reduction step on (another) of the paths $p_1, \ldots, p_n$ must be executed prior to that extension step. Such derivations will be called compact in the next definition, which starts the formal treatment of this matter.

DEFINITION 6.2. **(Compact Model Elimination Derivation)** Let $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a model elimination derivation, and suppose the selected paths are denoted by $p_1, \ldots, p_{n-1}$ (in order). If the inference rule applied to some $\mathcal{P}_i$ ($i \in \{1 \ldots n-1\}$) is an extension step, then define the set of "new" paths brought in by that extension step as

$$new_i := (p_i \cdot \mathcal{P}_{C_i - \{L_i\}})\sigma_i$$

where C_i is the variant of the clause used, L_i is the literal to build the connection and σ_i is the MGU used (cf. def. 3.2).

A model elimination derivation is called *compact* iff whenever an extension step is applied to $\mathcal{P}_i$ ($i \in \{1 \ldots n-1\}$) the following holds:

> if some p_j ($n > j > i$) is the selected path in a reduction step, and p_j is an instance of an occurence of a path in new_i
> then every selected path p_k ($i < k < j$) is an instance of an occurence of a path in new_i and is subject to a reduction step.

□

The completeness of compact tableau model elimination follows from the completeness result in (Baumgartner, 1992) and its "independence of the computation rule", that allows for arbitrary re-ordering of inferences in a derivation.

Note that the connection method deals with *sets* of sequences, whereas tableau model elimination deals with *multisets* of sequences. Whenever required for relationships, multisets are mapped to sets in the obvious way (cf. also the similar discussion on data structures in section 4).

Building on this, we arrive at the following theorem:

THEOREM 6.1. **(The Connection Method Simulates Tableau Model Elimination)** *Let M be a matrix and let $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a compact tableau model elimination derivation from M, such that in every inference step the selected path is of maximal length. Then there exists a connection method derivation $(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$, for some $k \leq n$, from M and a substitution δ such that $\mathcal{Q}_k\delta \subseteq \mathcal{P}_n$.*

PROOF. Induction on the length n of the given derivation.

Base case: If $n = 1$ then the model elimination derivation $(\mathcal{P}_1)$ consists of a single path set $\mathcal{P}_C$ (a multiset of sequences) for some $C \in M$. Let the connection method derivation be $(\mathcal{Q}_1)$ which consists of a single path set $\mathcal{P}_C$ (a set of sequences). When read as a set of sets it evidently holds that $\mathcal{P}_1$ equals $\mathcal{Q}_1$. Hence with $\delta := \epsilon$ the claim holds.

Induction step: We observe that for derivations of length > 1 the given model elimination derivation has a very special structure: first, the sequences in $\mathcal{P}_1$ are all of length 1. Hence the first step is an extension step, since no reduction step is applicable to a sequence of length 1. Second, due to this, for every compact model elimination derivation the sequence of selected paths can evidently be written as

$$p_1^0, \underbrace{p_1^1, \ldots, p_1^{k_1}}_{new_1}, \ldots, p_m^0, \underbrace{p_m^1, \ldots, p_m^{k_m}}_{new_m}$$

where precisely the p_i^0 ($i = 1 \ldots m$) are selected for an extension step, and all other $p_i^{k_j}$ ($k_j \geq 0$) are selected for a reduction step with membership in new_i as indicated. We will make use of this structure below.

By the induction hypothesis suppose the result to hold for the model elimination derivations $(\mathcal{P}_1, \ldots, \mathcal{P}_m)$ (for all $m < n$). Hence, a connection method derivation $(\mathcal{Q}_1, \ldots, \mathcal{Q}_l)$exists with $l \leq m$ and a substitution δ such that $\mathcal{Q}_l\delta \subseteq \mathcal{P}_m$. We have to show that a connection method derivation $(\mathcal{Q}_1, \ldots, \mathcal{Q}_{l'})$ exists with $l' \leq n$ such that for some $\delta' : \mathcal{Q}_{l'}\delta' \subseteq \mathcal{P}_n$. In order to do so, we have to trace back to the most recent extension step, and assemble this step and all subsequent reduction steps to a single connection method extension step. More formally, according to the above consideration about the structure of compact model elimination derivations, the sequence of selected paths in the given model elimination derivation is

$$p_1^0, p_1^1, \ldots, p_1^{k_1}, \ldots, p_i^0, p_i^1, \ldots, p_i^{k_i}$$

where p_i^0 is the selected path in the most recent model elimination extension step, all subsequent steps are reduction steps, and $p_i^{k_i}$ is the selected path in $\mathcal{P}_{n-1}$. Note that if $k_i = 0$ then $\mathcal{P}_{n-1}$ is obtained by a model elimination extension step, otherwise by a model elimination reduction step.

Suppose C_i is the variant of the clause used in the model elimination extension step of $\mathcal{P}_i^0$, which can be partitioned as

$$\mathcal{P}_i^0 = \mathcal{P}_i'^0 \cup \{p_i^0\} \tag{6.1}$$

and σ_0 is the MGU used. Hence $new_i = (p_i^0 \cdot \mathcal{P}_{C_i - \{L_i\}})\sigma_0$ where L_i is the literal in C_i that builds the connection. Since the model elimination derivation is compact, all subsequent reduction steps remove instances of paths in new_i from the path set. More precisely, in the reduction step where p_i^j ($j \in \{1 \ldots k_i\}$) is the selected path and σ_j is the MGU used, a path in $new_i\sigma_0\sigma_1 \cdots \sigma_j$ is removed from $\mathcal{P}_i^j$. Now let

$$D_i' := \{L \in C_i \mid \exists j : p_i^0 \circ L\sigma_0\sigma_1 \cdots \sigma_j \in \ new_i\sigma_0\sigma_1 \cdots \sigma_j \text{ is removed in } \mathcal{P}_i^j\}$$

be those literals from C_i which extend the selected path p_i^0 to those paths being removed in the reduction steps. It somewhat eases argumentation to use $D_i := D_i' \cup \{L_i\}$ in the sequel. Then the extension step and the subsequent reduction steps each remove exactly one instance (by MGU $\sigma_0\sigma_1 \cdots \sigma_j$) of an element of $p_i^0 \cdot D_i$ from $\mathcal{P}_i^j$ (for $j \in \{0, \ldots k_i\}$). As the result of the final reduction step we can write with (6.1)

$$\mathcal{P}_n = (\mathcal{P}_i'^0 \cup p_i^0 \cdot \mathcal{P}_{C_i - D_i})\sigma_0\sigma_1 \cdots \sigma_{k_i} \tag{6.2}$$

Let $(\mathcal{P}_1^0, \mathcal{P}_1^1, \ldots, \mathcal{P}_1^{k_1}, \ldots, \mathcal{P}_i^0)$ be a prefix of the given model elimination derivation leading to the last extension step. By the induction hypothesis there exists a (not longer than the derivation ending in $\mathcal{P}_i^0$) connection method derivation $(\mathcal{Q}_1, \ldots, \mathcal{Q}_l)$ and a substitution δ with $\mathcal{Q}_l\delta \subseteq \mathcal{P}_i^0$.

We further distinguish two cases:

Case 1: $p_i^0 \notin \mathcal{Q}_l\delta$. In words: the model elimination derivation extends a path that is not present

in the corresponding connection method path set. Thus with $\mathcal{Q}_l\delta \subseteq \mathcal{P}_i'^0$ it follows

$$\begin{aligned} & \mathcal{Q}_l\delta\sigma_0\sigma_1\cdots\sigma_{k_i} \\ \subseteq\ & \mathcal{P}_i'^0\sigma_0\sigma_1\cdots\sigma_{k_i} \\ \subseteq\ & (\mathcal{P}_i'^0 \cup (p_i^0 \cdot \mathcal{P}_{C_i - D_i}))\sigma_0\sigma_1\cdots\sigma_{k_i} \\ =\ & \mathcal{P}_n \end{aligned}$$

Hence $\mathcal{Q}_l$ and the substitution $\delta\sigma_0\sigma_1\cdots\sigma_{k_i}$ satisfies the claim for $\mathcal{P}_n$.

Case 2 (¬ case 1): $p_i^0 \in \mathcal{Q}_l\delta$. We intend to simulate the model elimination extension step of $\mathcal{P}_i^0$ *and* the subsequent reduction steps (if any), including the final one which led to $\mathcal{P}_n$, by a single connection method extension step (figure 4). The one premise path set for the connection method extension step is $\mathcal{Q}_l = \mathcal{Q}_l' \cup \{q_l\}$, where the selected path q_l corresponds to p_i^0, i.e.

$$q_l\delta = p_i^0 \tag{6.3}$$

For the other path set recall from the definition that we have to find a variable disjoint variant C of an input clause from M and determine a set $D \subseteq C$ which forms a set of connections with the selected path q_l by an MGU. In this case we take for C the same clause as used in the last extension step in the model elimination derivation, i.e. $C := C_i$. For D we choose the literals used to build the deleted paths , i.e. $D := D_i$ in the connection method extension step. In order to carry out the inference it has to be shown that by some MGU τ every path in $(q_l \cdot \mathcal{P}_D)\tau$ is complementary (by this we mean that the leaf of a path is complementary to another element in the path). This can be shown as follows: every MGU $\sigma_0\sigma_1\cdots\sigma_j$ renders a path in $p_i^0 \cdot \mathcal{P}_{D_i}$ complementary ($j \in \{1, \ldots, k_i\}$). Hence also the more specific substitution $\sigma_0\sigma_1\cdots\sigma_{k_i}$ does. Thus every path in $(p_i^0 \cdot \mathcal{P}_{D_i})\sigma_0\sigma_1\cdots\sigma_{k_i}$ is also complementary. With (6.3) and the assumption that δ does not act upon the variables of C_i (without loss of generality C_i can also be assumed to be variable disjoint from $\mathcal{P}_i^0$) it follows

$$(p_i^0 \cdot \mathcal{P}_{C_i})\sigma_0\sigma_1\cdots\sigma_{k_i} = (q_l \cdot \mathcal{P}_{C_i})\delta\sigma_0\sigma_1\cdots\sigma_{k_i} \tag{6.4}$$

Since $\delta\sigma_0\sigma_1\cdots\sigma_{k_i}$ renders every path in $q_l \cdot \mathcal{P}_{D_i}$ complementary, there is an MGU τ and a substitution δ' such that†

$$\tau\delta'|_{dom(\delta\sigma_0\sigma_1\cdots\sigma_{k_i})} = \delta\sigma_0\sigma_1\cdots\sigma_{k_i} \tag{6.5}$$

Using τ the desired connection method extension step can be carried out, yielding the path set

$$\mathcal{Q}_{l+1} = (\mathcal{Q}_l' \cup (q_l \cdot \mathcal{P}_{C_i - D_i}))\tau \tag{6.6}$$

From the induction hypothesis and (6.3) we conclude

$$\mathcal{Q}_l'\delta \subseteq \mathcal{P}_i'^0 \tag{6.7}$$

Now things can be put together:

$$\begin{aligned} \mathcal{Q}_{l+1}\delta' &\overset{(6.6)}{=} (\mathcal{Q}_l' \cup (q_l \cdot \mathcal{P}_{C_i - D_i}))\tau\delta' \\ &= \mathcal{Q}_l'\tau\delta' \cup (q_l \cdot \mathcal{P}_{C_i - D_i})\tau\delta' \\ &\overset{(6.5)}{=} \mathcal{Q}_l'\delta\sigma_0\sigma_1\cdots\sigma_{k_i} \cup (q_l \cdot \mathcal{P}_{C_i - D_i})\tau\delta' \end{aligned}$$

† By an expression of the form $\sigma|_{dom(\delta)}$ we mean the restriction of σ to the domain of δ, i.e. from σ just those assignments to variables are selected which have also an assignment in δ.

$$\begin{aligned}
&\overset{(6.5,6.4)}{=} \mathcal{Q}'_l \delta \sigma_0 \sigma_1 \cdots \sigma_{k_i} \cup (p_i^0 \cdot \mathcal{P}_{C_i - D_i}) \sigma_0 \sigma_1 \cdots \sigma_{k_i} \\
&\overset{(6.7)}{\subseteq} (\mathcal{P}'^0_i \cup (p_i^0 \cdot \mathcal{P}_{C_i - D_i})) \sigma_0 \sigma_1 \cdots \sigma_{k_i} \\
&\overset{(6.2)}{=} \mathcal{P}_n
\end{aligned}$$

Hence $\mathcal{Q}_{l+1}$ and δ' satisfy the claim. □

The converse of the theorem does not hold. This is due to the restriction in tableau model elimination that in inferences the leaf of the selected path must be part of the connection.

THE CONNECTION METHOD AND SEQUENCE CONSOLUTION

Let us now relate the connection method to consolution. Note that the observation we made during the discussion of tableau model elimination that led to the definition of sequence consolution can be made with the connection calculus as well. In the connection calculus one is *not* forced to delete all complementary paths, whereas by Eder's consolution one is forced to do this.

As a consequence we need sequence consolution to establish the following result:

THEOREM 6.2. **(Sequence Consolution Simulates the Connection Method)** *Let M be a matrix and $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ a connection calculus derivation. Then variants of clauses $C_1, \ldots, C_n$ from M exist, such that $(\mathcal{P}_1, \mathcal{P}_{C_1} \ldots, \mathcal{P}_{n-1}, \mathcal{P}_{C_{n-1}}, \mathcal{P}_n)$ is a sequence consolution derivation of M.*

PROOF. Let $\mathcal{P}_{i+1}$ be derived by extension with $\mathcal{P}_C$, then we construct the following sequence consolvent:
$\mathcal{P}_i$ has the form $\mathcal{P}'_i \cup \{p\}$ and C_i is a variant of a clause in M, hence $\mathcal{P}_i$ and $\mathcal{P}_{C_i}$ do not have variables in common. Let $D_i \subseteq C_i$ be the set of literals selected for closing. According to the sequence consolution inference rule we then have to multiply these two to obtain $\mathcal{P}_i \cdot \mathcal{P}_{C_i} = (\mathcal{P}'_i \cup \{p\}) \cdot \mathcal{P}_{C_i} = (\mathcal{P}'_i \cdot \mathcal{P}_{C_i}) \cup (p \cdot \mathcal{P}_{C_i})$, which further has to be modified with our simplification rule, Version 2 from definition 3.5:

A) σ is obtained by unifying the connection $(L, leaf(p))$, which was used in the extension step.
B) $\mathcal{P}^B$ is obtained from $(\mathcal{P}_i \cdot \mathcal{P}_{C_i})\sigma$ by deleting the paths $(p \cdot \{L\})\sigma$ for every $L \in D_i$ from $(p \cdot \mathcal{P}_{C_i})\sigma$; thus $\mathcal{P}^B = ((\mathcal{P}'_i \cdot \mathcal{P}_{C_i}) \cup (p \cdot \mathcal{P}_{C_i - D_i}))\sigma$
C,D) from $\mathcal{P}^B$ we shorten all paths from $(\mathcal{P}'_i \cdot \mathcal{P}_{C_i})\sigma$ and delete all duplicate occurrences to obtain $\mathcal{P}'_i\sigma$ which finally gives $\mathcal{R} = (\mathcal{P}'_i \cup (p \cdot \mathcal{P}_{C_i - D_i}))\sigma$

Clearly $\mathcal{R}$ is the result of the extension step as well. □

7. Resolution Simulates Tableau Model Elimination

Up to now we know the relation of several calculi to consolution; these include resolution, tableau model elimination, Loveland's model elimination and a connection calculus. In order to learn more about the relationships of these calculi, we show here the relation between resolution and tableau model elimination. More specifically, we will show that resolution stepwise simulates tableau model elimination.

To express the result, we define the *multiset of open leaves of a path set $\mathcal{P}$* as the multiset

$O(\mathcal{P}) := \{leaf(p) \mid p \in \mathcal{P}\}$. In order to compare multisets to sets, we map multisets X to sets in the obvious way by $X^{\downarrow} := \{x \mid x \text{ occurs at least once in } X\}$. Now we claim:

THEOREM 7.1. **(Resolution Simulates Tableau Model Elimination)** *Let M be a matrix and let $(\mathcal{P}_1, \ldots, \mathcal{P}_n)$ be a tableau model elimination derivation from M such that in every $\mathcal{P}_i$ ($i = 1 \ldots n-1$) a longest branch is selected. Then there exist a resolution derivation $(C_1, \ldots, C_k)$, for some $k \leq n$, from M and a substitution γ such that $C_k\gamma \subseteq O(\mathcal{P}_n)^{\downarrow}$.*

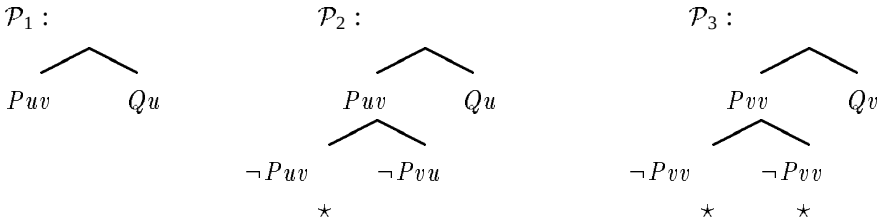
Let us call the elements of a tableau model elimination derivation *tableaux*.

In every tableau $\mathcal{P}_i$ in a derivation, $O(\mathcal{P}_i)$ (the set of leaves of open paths) can be seen as the goal set still to be proved. Furthermore, since the *leaf* of a branch selected for an inference has to be part of the connection (cf. definition 3.2), one could say that this goal set is consecutively processed in a linear way. It is thus quite natural to relate a tableau model elimination refutation to a linear resolution refutation by relating every $O(\mathcal{P}_i)$ in the model elimination refutation to the near parent in a linear resolution refutation (see e.g. (Chang and Lee, 1973) for linear resolution). This relationship is given in the theorem as the subset relation $C_k\gamma \subseteq O(\mathcal{P}_n)^{\downarrow}$. Thus the model elimination-tableau can be seen, modulo instantiation, as an "upper bound" for the corresponding goal clause in resolution. In other words, the resolution goal clause subsumes the tableau model elimination goal.

The *inference rules* of model elimination and resolution correspond as follows: an extension step corresponds to a resolution step of the goal clause with an input clause, and a reduction step corresponds to a resolution step of the goal clause with an ancestor clause. This will be put more precisely in the proof below.

The following example is typical for the simulation; it shows the mapping of data structures, the mapping of inference rules, and also where factoring comes into resolution, which is not needed in model elimination:

Model elimination derivation of $S = \{Puv \vee Qu,\ \neg Pyx \vee \neg Pxy\}$:



Corresponding resolution derivation:

$C_1 : \ Puv \vee Qu$ $\qquad$ $C_2 : \ \neg Pvu \vee Qu$ $\qquad$ $C_3 : \ Qu \vee Qv$

The set of open nodes $O(\mathcal{P}_1)^{\downarrow}$ of $\mathcal{P}_1$ is exactly the clause C_1; the same holds for $\mathcal{P}_2$ and C_2, where $\mathcal{P}_2$ is obtained from $\mathcal{P}_1$ by extension with $\neg Pyx \vee \neg Pxy$. The interesting case is the reduction step with MGU $\sigma = \{u \leftarrow v\}$ applied to obtain $\mathcal{P}_3$. In general, reduction steps with an ancestor literal L are mapped to linear resolution steps, where the far parent clause is the clause corresponding to a previously derived tableau that contains L as open leaf. In the example $L = Puv$, the previously derived tableau is $\mathcal{P}_1$, and the corresponding clause is C_1. However, resolution forces the parent clauses to be variable disjoint. Hence we have to use a variant of C_1,

e.g. $C_1' = Pu'v' \vee Qu'$. Using an appropriate unifier we arrive at the resolvent $Qu \vee Qv$ of C_2 and C_1'. Note that with $\gamma = \{v \leftarrow u\}$ we can find a substitution such that the theorem holds. Here we can see where factorization comes in: suppose that e.g. a unit clause $\neg Qa$ were given. $\neg Qa$ can be used to close $\mathcal{P}_3$, but if C_3 were not factorized prior to binary resolution to the clause Qu, then the resolvent would be Qv and the theorem would not hold.

It turns out that the simulation cannot be done precisely step by step. One could say that resolution is more "optimal" than model elimination in the following sense: in resolution due to the *set* data structure identical literals are collapsed into one single occurrence, whereas the corresponding data structure in tableau model elimination are *multisets* and thus permit multiple occurrences of identical literals. As a consequence for derivations, in tableau model elimination identical subgoals have to be proven independently from each other, whereas resolution treats them in one step. So it is not surprising that the resolution refutation may be shorter than the model elimination refutation. Informally, the resolution refutation has to "pause" while in the tableau model elimination refutation a subgoal is proved that is absent in the resolution clause due to the set data structure. Note that this property of resolution has nothing to do with the use of lemmas.

We will use the following standard definition of resolution (Chang and Lee, 1973):

DEFINITION 7.1. **(Resolution, (Chang and Lee, 1973))** Let C_1 and C_2 be clauses and ϕ_1 and ϕ_2 respectively be most general factorization substitutions for some subsets of C_1 and C_2, respectively. Let $L_1 \in C_1$ and $L_2 \in C_2$ such that $\overline{L_1}\phi_1$ and $L_2\phi_2$ are unifiable by MGU σ. Then the resolvent of C_1 and C_2 is the clause $(C_1\phi_1\sigma - \{L_1\}\phi_1\sigma) \cup (C_2\phi_2\sigma - \{L_2\}\phi_2\sigma)$ □

PROOF. (theorem 7.1) Induction on n, the length of the ME derivation.

Base case: If $n = 1$ then $\mathcal{P}_1$ is a path set $\mathcal{P}_C$ with $C \in M$. With $\gamma = \epsilon$ the result holds trivially.

Induction step: Suppose by the induction hypothesis that the result holds for derivations of length n, i.e. there is a resolution derivation for C_k from M and a γ such that

$$C_k\gamma \subseteq O(\mathcal{P}_n)^{\downarrow} \tag{7.1}$$

We show that the claim holds also for derivations of length $n+1$. This is done by case analyses with respect to the inference rule applied to $\mathcal{P}_n$.

Case 1: an extension step applied (cf figure 5). By definition of tableau model elimination 3.2, $\mathcal{P}_n = \mathcal{P} \cup \{p\}$, $leaf(p) = K$ and C is a variant of a clause in M such that for some $L \in C$, (L, K) is a connection. Let σ be the MGU used for that connection. Then

$$O(\mathcal{P}_{n+1}) = (O(\mathcal{P}_n) - \{K\})\sigma \cup (C - \{L\})\sigma. \tag{7.2}$$

We do a further case analysis. In the first (and trivial) case $K \notin C_k\gamma$. But then it follows from (7.1) and (7.2) $C_k\gamma\sigma \subseteq O(\mathcal{P}_{n+1})^{\downarrow}$. Setting $\gamma := \gamma\sigma$ proves the claim. In the second case $K \in C_k\gamma$. Here the extension step of $\mathcal{P}_n$ with C has to be simulated in a resolution step of C_k with C. As mentioned in the introductory example, a little care must be taken to factorize on "enough" literals of the parent clauses. For the *extension* step it suffices to factor only C_k. For this, C_k can be partitioned as $C_k = D \cup E$, where D and E are disjoint and D is a maximal set such that $D\gamma\sigma = \{K\sigma\}$. Let ϕ_1 be a most general factorization substitution for $D \subseteq C_k$ (by this we mean a most general substitution with domain and range as the variables of C_k such that D becomes a singleton). ϕ_1 is more general than $\gamma\sigma$. Hence there exists a substitution δ_1 such that

$$C_k\phi_1\delta_1 = C_k\gamma\sigma \tag{7.3}$$

Now consider C. Let δ_2 be the restriction of σ to the variables of C. Then

$$C\delta_2 = C\sigma \tag{7.4}$$

Since ϕ_1 does not introduce new variables to C_k it follows that δ_1, which is applied to $C_k\phi_1$, can be assumed to act on the variables of C_k only. Without loss of generality assume C to be a "new" variant. Hence δ_1 does not act upon the variables of C. Thus also $C = C\delta_1$. Applying δ_2 to both sides yields

$$C\delta_2 = C\delta_1\delta_2 \tag{7.5}$$

Since C is a new variant, δ_2 acts on the variables of C only. It follows

$$C_k\phi_1\delta_1 = C_k\phi_1\delta_1\delta_2 \tag{7.6}$$

Concerning D it follows from this, (7.3) and $D\gamma\sigma = \{K\sigma\}$

$$D\phi_1\delta_1\delta_2 = \{K\sigma\} \tag{7.7}$$

Since σ is a unifier for L and K (modulo sign) we have with (7.7), (7.5) and (7.4) that $\delta_1\delta_2$ is a unifier (modulo sign) for $D\phi_1$ and $\{L\}$. But then there is also an MGU σ' for $D\phi_1$ and $\{L\}$ and a substitution δ' such that

$$C_k\phi_1\sigma'\delta' = C_k\phi_1\delta_1\delta_2 \quad \text{and} \tag{7.8}$$

$$C\sigma'\delta' = C\delta_1\delta_2 \tag{7.9}$$

Hence we can resolve the factor $C_k\phi_1$ of C_k with selected literal $D\phi_1$ against C with selected literal L and MGU σ'. The resolvent C_{k+1} is the clause

$$\begin{aligned} C_{k+1} &= (C_k\phi_1\sigma' - D\phi_1\sigma') \cup (C\sigma' - \{L\}\sigma') \\ &= E\phi_1\sigma' \cup (C\sigma' - \{L\}\sigma') \end{aligned}$$

Here we have used the fact that D and E are disjoint, even after instantiation with $\phi_1\sigma'$. Next we will prove that C_{k+1} is the desired clause and δ' is the desired substitution, i.e. $C_{k+1}\delta' \subseteq O(\mathcal{P}_{n+1})^{\downarrow}$.

$$\begin{aligned} C_{k+1}\delta' &= (E\phi_1\sigma' \cup (C\sigma' - \{L\}\sigma'))\delta' \\ &= E\phi_1\sigma'\delta' \cup (C\sigma' - \{L\}\sigma')\delta' \\ &\overset{(7.8,7.6,7.3)}{=} E\gamma\sigma \cup (C\sigma' - \{L\}\sigma')\delta' \\ &\subseteq E\gamma\sigma \cup (C - \{L\})\sigma'\delta' \\ &\overset{(7.9,7.5,7.4)}{=} E\gamma\sigma \cup (C - \{L\})\sigma \\ &\overset{(*)}{=} (C_k\gamma\sigma - \{K\sigma\}) \cup (C - \{L\})\sigma \\ &\overset{(7.1)}{\subseteq} (O(\mathcal{P}_n)^{\downarrow} - \{K\})\sigma \cup (C - \{L\})\sigma \\ &\overset{(7.2)}{\subseteq} O(\mathcal{P}_{n+1})^{\downarrow} \end{aligned}$$

In $(*)$ we used the disjointness and maximality properties of D and E as defined above.

Thus the derivation of C_{k+1} and the substitution δ' proves the claim.

Case 2: a reduction step applied. By definition of reduction step (def. 3.2), $\mathcal{P}_n = \mathcal{P} \cup \{p\}$, $leaf(p) = K$ and p contains a literal L such that (L, K) is a connection. Let σ_{n+1} be the MGU used for that connection. As a result of the reduction step we obtain

$$O(\mathcal{P}_{n+1}) = (O(\mathcal{P}_n) - \{K\})\sigma_{n+1} \tag{7.10}$$

The literal L that is used in the reduction step must have been the leaf of a path q in a previously derived tableau $\mathcal{P}_m$ (for some $m < n$), i.e. $L \in O(\mathcal{P}_m)$. We may assume that $\mathcal{P}_m$ is the last tableau in the derivation that contains that occurrence of q. Thus $\mathcal{P}_{m+1}$ is obtained from $\mathcal{P}_m$ by extending the path q. Concerning the set of open leaves we have $O(\mathcal{P}_{m+1}) = ((O(\mathcal{P}_m) - \{L\}) \cup O_{m+1})\sigma_{m+1}$, where O_{m+1} is the (possibly empty) set of open leaves stemming from this extension step, and σ_{m+1} is the MGU used. By the given assumption in every inference step a longest path is selected. As a consequence of this and the fact that every inference removes exactly the selected path, the paths eliminated in the tableaux $\mathcal{P}_{m+1}, \ldots, \mathcal{P}_n$ are all (instances of) a longest extension of q. Thus no (instance of) a path already contained in $\mathcal{P}_m - \{q\}$ is removed. Hence $O(\mathcal{P}_m - \{q\}) = O(\mathcal{P}_m) - \{L\}$ persists; more formally we have:

$$\forall l = m+1 \ldots n+1: \quad (O(\mathcal{P}_m) - \{L\})\sigma_{m+1} \cdots \sigma_l \subseteq O(\mathcal{P}_l) \tag{7.11}$$

Since $m < n$ by the induction hypothesis there is a C_j and a γ_j such that

$$C_j\gamma_j \subseteq O(\mathcal{P}_m)^{\downarrow} \tag{7.12}$$

Also by the induction hypothesis there is a C_k and a γ_k such that

$$C_k\gamma_k \subseteq O(\mathcal{P}_n)^{\downarrow} \tag{7.13}$$

Let σ_i ($i = m+1 \ldots n$) be the MGU applied in the derivation of $\mathcal{P}_i$ from $\mathcal{P}_{i-1}$. Since the MGUs are applied to the whole path set, L is instantiated to $L\sigma_{m+1} \cdots \sigma_n$ in $\mathcal{P}_n$. Let σ_{n+1} be the MGU used in the reduction step. Thus

$$K\sigma_{n+1} = \overline{L}\sigma_{m+1} \cdots \sigma_n\sigma_{n+1} \tag{7.14}$$

In the sequel we will abbreviate $\sigma_{m+1} \cdots \sigma_n\sigma_{n+1}$ as τ.

Now we distinguish three cases. In the first (trivial) case, $K \notin C_k\gamma_k$. This is similar to the first trivial case in extension step, i.e. C_k and $\gamma := \gamma_k\sigma_{n+1}$ will do. In the second (also trivial) case $L \notin C_j\gamma_j$, i.e. the clause C_j corresponding to the previous tableau $\mathcal{P}_m$ does not contain the ancestor literal. Thus C_j and the substitution $\gamma := \gamma_j\tau$ will do.

In the non-trivial case $K \in C_k\gamma_k \wedge L \in C_j\gamma_j$. It is similar to the non-trivial case in the extension step, except that resolution of C_k with an input clause C is replaced by resolution of C_k with the ancestor clause C_j. Unlike in the extension step, C_j this time has to be factorized: take e.g. $C_j = P(x) \vee P(y)$ and q has the leaf $L = P(a)$. Suppose q is extended and contains the leaf $\neg P(a)$ which is subject to a reduction step with $P(a)$ now. Let $C_k = \neg P(u) \vee \neg P(v)$. After the reduction step the set of open leaves contains neither $P(a)$ nor $\neg P(a)$. Hence the corresponding literals from the corresponding clauses have to be resolved away. Note that this is possible only after factoring both $P(x) \vee P(y)$ as well as $\neg P(u) \vee \neg P(v)$ to a singleton.

C_k can be partitioned as $C_k = D \cup E$, where D and E are disjoint and D is a maximal set such that $D\gamma_k\sigma_{n+1} = \{K\sigma_{n+1}\}$. Let ϕ_1 be a most general factorization substitution for $D \subseteq C_k$. ϕ_1 is more general than $\gamma_k\sigma_{n+1}$. Hence there is a substitution δ_1 such that

$$C_k\phi_1\delta_1 = C_k\gamma_k\sigma_{n+1} \tag{7.15}$$

Now consider C_j. Similarly to C_k, C_j can be partitioned as $C_j = F \cup G$, where F and G are disjoint and F is a maximal set such that $F\gamma_j\tau = \{L\tau\}$. Let ϕ_2 be a most general factorization substitution for F. Since ϕ_2 is more general than $\gamma_j\tau$ there is a substitution δ_2 such that

$$C_j\phi_2\delta_2 = C_j\gamma_j\tau \tag{7.16}$$

Since ϕ_1 does not introduce new variables to C_k it follows that δ_1, which is applied to $C_k\phi_1$, can be assumed to act on the variables of C_k only. Without loss of generality assume C_j to be a

"new" variant, variable disjoint from C_k. Hence δ_1 does not act upon the variables of C_j. Thus also $C_j\phi_2 = C_j\phi_2\delta_1$. Applying δ_2 to both sides yields

$$C_j\phi_2\delta_2 = C_j\phi_2\delta_1\delta_2 \tag{7.17}$$

Concerning F it follows from this equation, (7.16) and $F\gamma_j\tau = \{L\tau\}$

$$F\phi_2\delta_1\delta_2 = \{L\tau\} \tag{7.18}$$

Since C_j is a new variant and ϕ_2 does not introduce new variables, δ_2 acts on the variables of C_j only. It follows

$$C_k\phi_1\delta_1 = C_k\phi_1\delta_1\delta_2 \tag{7.19}$$

Concerning D it follows from this equation, (7.15) and $D\gamma_k\sigma_{n+1} = \{K\sigma_{n+1}\}$

$$D\phi_1\delta_1\delta_2 = \{K\sigma_{n+1}\} \tag{7.20}$$

Since σ_{n+1} is a unifier for $L\sigma_{m+1}\cdots\sigma_n$ and K (modulo sign) we have with (7.18) and (7.20) that $\delta_1\delta_2$ is a unifier (modulo sign) for $D\phi_1$ and $F\phi_2$. But then there is also an MGU σ' for $D\phi_1$ and $F\phi_2$ and a substitution δ' such that

$$C_k\phi_1\sigma'\delta' = C_k\phi_1\delta_1\delta_2 \quad \text{and} \tag{7.21}$$

$$C_j\phi_2\sigma'\delta' = C_j\phi_2\delta_1\delta_2 \tag{7.22}$$

Hence we can resolve the factor $C_k\phi_1$ of C_k with selected literal $D\phi_1$ against the factor $C_j\phi_2$ of C_j with selected literal $F\phi_2$ and MGU σ'. The resolvent C_{k+1} is the clause

$$\begin{aligned} C_{k+1} &= (C_k\phi_1\sigma' - D\phi_1\sigma') \cup (C_j\phi_2\sigma' - F\phi_2\sigma') \\ &= E\phi_1\sigma' \cup G\phi_2\sigma' \end{aligned}$$

Here we have used the fact that D and E (or F and G respectively) are disjoint, even after instantiation with $\phi_1\sigma'$ (or $\phi_2\sigma'$). Next we will prove that C_{k+1} is the desired clause and δ' is the desired substitution, i.e. $C_{k+1}\delta' \subseteq O(\mathcal{P}_{n+1})^{\downarrow}$.

$$\begin{aligned} C_{k+1}\delta' &= (E\phi_1\sigma' \cup G\phi_2\sigma')\delta' \\ &= E\phi_1\sigma'\delta' \cup G\phi_2\sigma'\delta' \\ &\overset{(7.21,7.19,7.15)}{=} E\gamma_k\sigma_{n+1} \cup G\phi_2\sigma'\delta' \\ &\overset{(7.22,7.17,7.16)}{=} E\gamma_k\sigma_{n+1} \cup G\gamma_j\tau \\ &\overset{(*)}{=} (C_k\gamma_k\sigma_{n+1} - \{K\sigma_{n+1}\}) \cup G\gamma_j\tau \\ &\overset{(**)}{=} (C_k\gamma_k\sigma_{n+1} - \{K\sigma_{n+1}\}) \cup (C_j\gamma_j\tau - \{L\tau\}) \\ &\overset{(7.12,7.13)}{\subseteq} (O(\mathcal{P}_n)^{\downarrow} - \{K\})\sigma_{n+1} \cup (O(\mathcal{P}_m)^{\downarrow} - \{L\})\tau \\ &\overset{(7.10)}{\subseteq} (O(\mathcal{P}_{n+1}) \cup (O(\mathcal{P}_m) - \{L\})\tau)^{\downarrow} \\ &\overset{(7.11)}{=} O(\mathcal{P}_{n+1})^{\downarrow} \end{aligned}$$

In $(*)$ we used the disjointness and maximality properties of D and E as defined above. The same holds in $(**)$ for F and G.

Thus the derivation of C_{k+1} and the substitution δ' proves the claim. □

In (Eder, 1992) Eder gives a theorem that relates the connection calculus to resolution. This

theorem is also relevant for tableau model elimination, since tableau model elimination can be seen as a restricted variant of the connection method. In fact, the theorem can easily be adapted to tableau model elimination. Then it states, similarly to our theorem above, that for every tableau model elimination refutation there is a resolution refutation that is no longer. However there are some differences: firstly, his proof is informal, whereas we feel the need for a rigorous formal treatment. Using our notation, he states that either $O(\mathcal{P}_n)^{\downarrow}$ subsumes C_k or else an input clause subsumes C_k. Note that we do not need the input clause set and thus have a tighter coupling of the refutations. Furthermore note that we use the converse subsumption relation, i.e. we claim that C_k subsumes $O(\mathcal{P}_n)^{\downarrow}$. It might seem obvious then to conclude that C_k and $O(\mathcal{P}_n)^{\downarrow}$ are equivalent. However this is not true since Eder uses a different resolution calculus from the one we do: we use Chang and Lee's definition, where factorization is carried out on the parent clauses. Eder treats factorization by an extra inference rule and thus can even factor resolvents; but more important than the order of factoring is the fact that for Eder's proof the factorization inference rule has to be replaced by a subsumption inference rule. His subsumption rule however works in a highly non-standard way: it allows the derivation of a clause D from a clause C if C subsumes D. Note that this rule reverses the standard subsumption rule; in Eder's calculus the clause D is obtained from C by introducing arbitrary literals and by arbitrary instantiation. The rule is applied in place of factorization in the traditional calculus, i.e. it is applied to the parent clauses before the binary resolution inference. In our view, this is a counter-intuitive inference rule that also destroys the idea of simulating extension steps by resolution steps.
Using Eder's resolution it might be possible to prove the theorem the way we stated it (although we did not check it). However it is not possible to show Eder's subsumption relation with our calculus.

8. Discussion

RESUMÉE

One starting point for this work was Eder's consolution together with his investigation on its relation to resolution. We gave a variant of consolution, which is stronger than consolution, in the sense that every consolution derivation can be simulated stepwise by this variant. This *sequence consolution* was used as a framework for comparing other calculi. For this we presented the calculi under consideration in the language of consolution, i.e. using paths and sets or sequences of paths. For the sake of completeness we cite from (Eder, 1991) the following theorem, which relates resolution and consolution:

THEOREM 8.1. *(Eder, 1991)***(Consolution Simulates Resolution)** *Let M be a matrix and $(C_0, \ldots, C_n)$ be a resolution refutation of M. Then $(\mathcal{P}_{C_1}, \ldots, \mathcal{P}_{C_n})$ is a consolution refutation of M.*

Together with the results from this paper we arrive at the diagram in figure 6. There, $A \rightarrow B$ means that calculus A can be simulated stepwise by calculus B, and dashed arrows indicate a "weaker" simulatability relation after certain calculus restrictions.

Although our main concern was to demonstrate the advantages of sequence consolution for these investigations, we obtained several new results and formal proofs of folklore-like theorems.

RELATION TO OTHER WORK

In (Letz, 1993) the "clausal tableau calculus" is chosen as a starting point for a comparison of various calculi and proof procedures. It corresponds exactly to Beth's or Smullyan's analytic tableaux calculus (Beth, 1959; Smullyan, 1968) when restricted to propositional formulas which are given in clause form. This calculus can be further restricted, such that tableau expansions are allowed only when the clause used for the extension step introduces at least one complementary literal compared with the leaf choosen for extension. This calculus is exactly our tableau model elimination calculus; it is called "connection tableau calculus" in (Letz, 1993). However, this taxonomy is not so straightforward when first order calculi are compared. In this case, there is a significant difference, in that analytic tableaux use an instantiation rule, whereas the tableau model elimination calculus needs unification; this is the reason why, at least in the first order case, the tableau model elimination resembles very much the model elimination calculus introduced by Loveland in (Loveland, 1968).
Another interesting property discussed by Letz is proof-confluence. The tableau calculus for example is proof-confluent, because any tableau can be completed to a closed one if the input set is unsatisfiable, whereas tableau model elimination is not proof-confluent; i.e. a proof procedure based on the latter has to perform some kind of backtracking.

In (Wrightson, 1989) the relation of analytic tableaux to various calculi and strategies is discussed. Wrightson gives in his paper a transformation of some sample resolution and model elimination proofs into proofs using clausal tableaux. His work addresses very much the same problems as our work does; whereas Wrightson gives a number of example deductions together with transformations, our main concern was to present rigorous theorems, stating how some calculi can be related to each other. For example, Wrightson transforms one resolution refutation for a propositional formula into an analytic tableaux proof; the resolution proof is a linear input refutation and to us it is not clear if and how this approach works for non-linear resolution proofs as well.

Bibel gave a comparative study of several proof procedures in (Bibel, 1982). There, a number of proof procedures are compared with three versions of the connection method (which were called "matrix methods" at that time). Bibel gave a theorem which establishes a relation between the "pure model elimination" (PME) calculus and the connection method. This PME calculus corresponds to our tableau model elimination and hence our theorem 5.1 is a proof of Bibel's theorem by means of the consolution-framework. It is obvious that a similar one-to-one simulation for Loveland's model elimination is not possible. We are confident that our approach, using sequence consolution as a framework calculus, helps to understand the common ground and the differences between those calculi.

In (Loveland, 1972), the relation between ME and several other calculi is investigated. As a main result, which is also contained and proved in a similar form in (Loveland, 1978), it demonstrated that ME and a certain restriction of linear resolution simulate each other by steps *at the ground level*. On the non-ground level ME may require additional extension steps that do not have a counterpart in resolution. This is shown by means of an example, and this example can also be used to show the same negative result for tableau model elimination. Also in that paper the *linked conjunct procedure* and a *matrix reduction procedure* (Prawitz, 1970) are discussed, and it is demonstrated at the ground level that under certain circumstances there is a 1-to-1 relationship between matrix reduction and ME.

Plaisted (Plaisted, 1990) has obtained a mapping (on the ground level) from model elimination to a sequent calculus (see e.g. (Gallier, 1987)). Using this setting he shows that it suffices to apply reduction steps to leaves with one (say: negative) polarity only.

On Complexity

Let us finally make some remarks on complexity results. All results of this paper depicted in figure 6 state a stepwise or shorter – and hence p(olynomial) – simulatability relation between calculi. There are a lot of interesting problems when one tries to establish simulatability in directions opposite to those shown in figure 6 by giving up stepwise simulatability. In (Eder, 1992) it is shown that the connection calculus simulates resolution only exponentially with respect to proof length. With the same argumentation we can conclude that the tableau model elimination calculus exponentially simulates resolution.

Negative results with respect to stepwise simulatability are also contained in (Schreiber, 1974). He relates an early presentation of the connection method according to Bibel and two variants of the resolution calculus (binary resolution with factoring and linear resolution). By means of examples he shows several relationships between Bibel's method and the two resolution variants.

In (Letz, 1993) it is proven that the tableau model elimination calculus (in this paper called connection tableau as mentioned above) cannot polynomially simulate clausal tableaux. Since the connection calculus can be understood as a clausal tableau calculus with a special path-selection function, that result fits into our scenario. Letz further investigated the difference between regular versions of calculi, i.e. where multiple occurences of a literal within one path are forbidden. Among others, it then turns out that regular tableau model elimination cannot polynomially simulate (non-regular) tableau model elimination.

The importance of regularity and its detection in derivations is another argument for introducing sequence consolution. Recall that sequence consolution does not merge occurrences of the same literals along a path (as consolution does). Hence regularity, or the violation of regularity, can be detected easily.

Acknowledgements

Thanks to Elmar Eder and Don Loveland for commenting on previous versions of this work. Three anonymous referees gave many valuable hints for improvement.

References

Baumgartner, P. (1992). A Model Elimination Calculus with Built-in Theories. In Ohlbach, H.-J., editor, *Proceedings of the 16-th German AI-Conference (GWAI-92)*, pages 30–42. Springer. LNAI 671.

Baumgartner, P., Furbach, U., and Petermann, U. (1992). A Unified Approach to Theory Reasoning. Forschungsbericht 15/92, University of Koblenz. (submitted to Journal of Automated Reasoning).

Beth, E. (1959). *The Foundations of Mathematics*. Noth Holland.

Bibel, W. (1982). A Comparative Study of Several Proof Procedures. *Artificial Intelligence*, 18:269–293.

Bibel, W. (1987). *Automated Theorem Proving*. Vieweg, 2nd edition.

Chang, C. and Lee, R. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press.

Eder, E. (1991). Consolution and its Relation with Resolution. In *Proc. IJCAI '91*.

Eder, E. (1992). *Relative Complexities of First Order Languages*. Vieweg.

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

Gallier, J. (1987). *Logic for Computer Science: Foundations of Automatic Theorem Proving*. Wiley.

Letz, R. (1993). *First-Order Proof Calculi and Proof Procedures for Automated Deduction*. PhD thesis, Technische Hochschule Darmstadt.

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

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

Loveland, D. (1969). A Simplified Version for the Model Elimination Theorem Proving Procedure. *JACM*, 16(3).

Loveland, D. (1972). A Unifying View of Some Linear Herbrand Procedures. *JACM*, 19(2).

Loveland, D. (1978). *Automated Theorem Proving - A Logical Basis*. North Holland.

Plaisted, D. (1990). A Sequent-Style Model Elimination Strategy and a Positive Refinement. *Journal of Automated Reasoning*, 4(6):389–402.

Prawitz, D. (1970). A Proof Procedure with Matrix Reduction. In *Symposium on Automatic Demonstration*, pages 207–214. Springer. Lecture Notes in Mathematics 125.

Schreiber, J. (1974). Vergleichende qualitative and quantitative Untersuchungen von Beweisverfahren. Technical Report 7411, Techn. Universität München, Abteilung Mathematik.

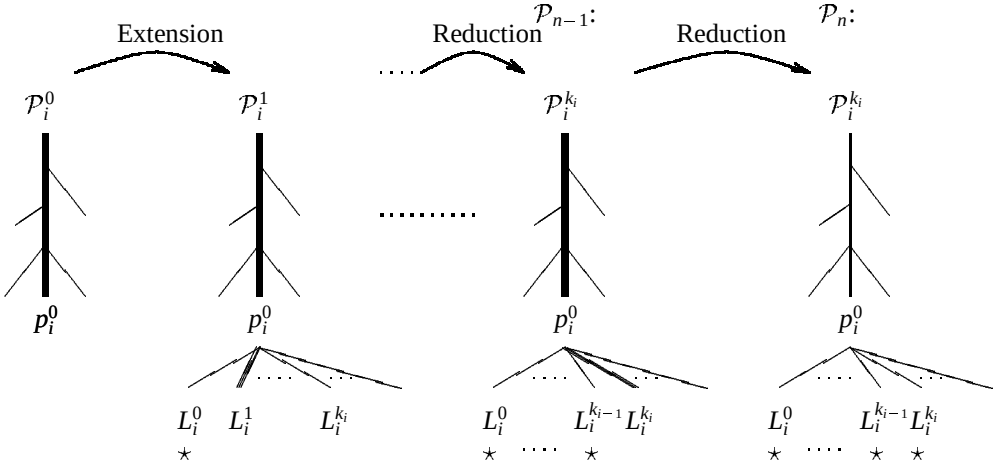
Smullyan, R. (1968). *First Order Logic*. Springer.

Stickel, M. (1988). A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning*, 4:353–380.

Stickel, M. (1989). A Prolog Technology Theorem Prover: A New Exposition and Implementation in Prolog. Technical note 464, SRI International.

Wrightson, G. (1989). Resolution and Analytical Tableaux. A Mapping. Technical Report UNARP–89–4, University of Newcastle, Australia.

Model Elimination:



Connection Method:

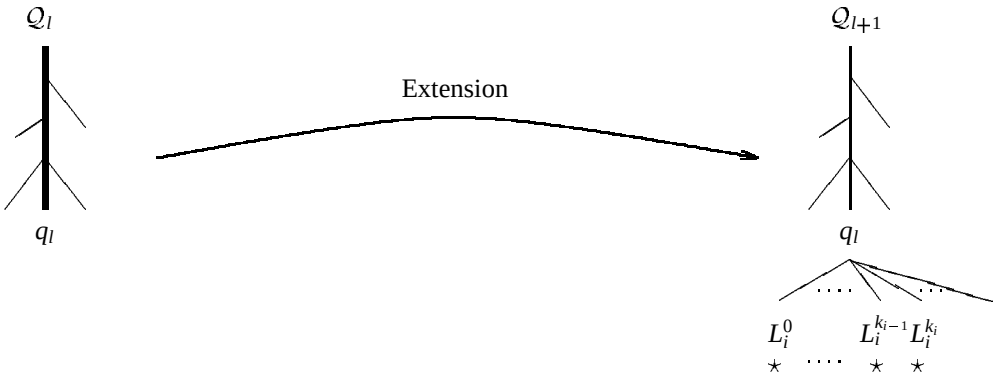


Figure 4. Simulation of a compact extension method subderivation by a single connection method extension step. The selected paths are drawn bold, and the involved substitutions are omitted.

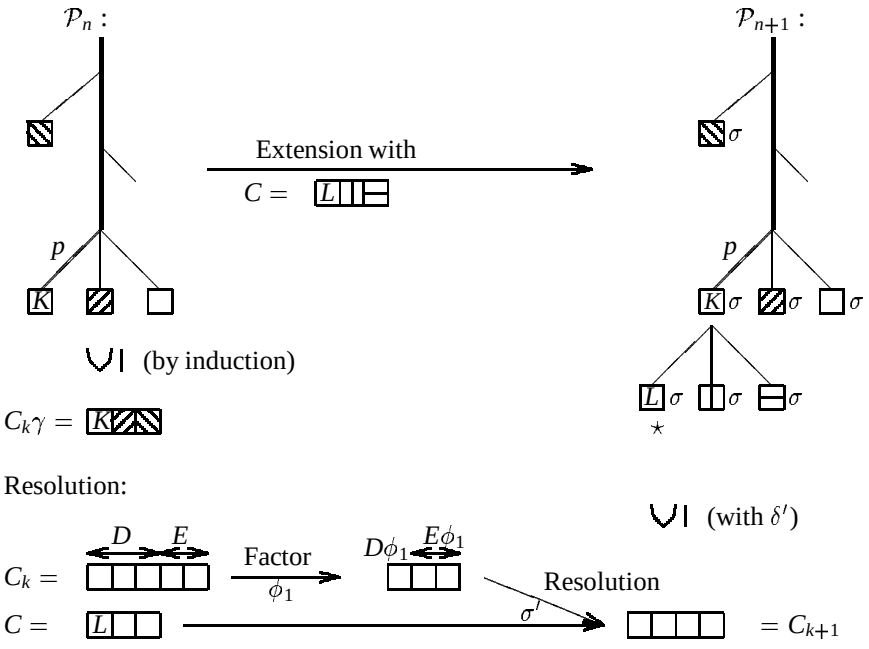


Figure 5. Simulating a model elimination extension step.

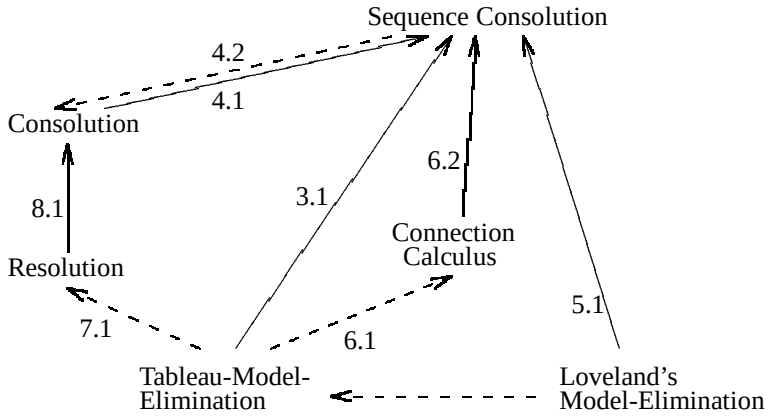


Figure 6. Theorems of this paper.