Refinements for Restart Model Elimination

Peter Baumgartner · Ulrich Furbach
Universität Koblenz · Institut für Informatik
Rheinau 1 · D–56075 Koblenz · Germany
E-mail: {peter,uli}@informatik.uni-koblenz.de

We present various refinements for the restart model elimination (RME) calculus and discuss their interrelationship. As a new variant of the calculus we introduce RME with *early cancellation pruning* and investigate its compatibility with the other refinements.

1 Introduction

Restart Model Elimination (RME) has been introduced as a variant of model elimination in [1] as a calculus which avoids contrapositives and which introduces case analysis. In [4] a variant for computing answers to disjunctive logic programs was introduced. RME is implemented as part of the PROTEIN system [2].

One result of this paper is a table of completeness results with respect to the combination of the refinements *head selection function*, *strictness*, *regularity*, *independance of the goal clause* for RME (Figure 1 below). Another original result is completeness of “early cancellation pruning”[1].

In the following section we recall basic restart model elimination calculus, and in Section 2.2 we introduce refinements. The main results of this paper are then presented in Section 3.

2 Restart Model Elimination (RME)

A pair of literals (K,L) is a *connection with MGU* σ iff σ is a most general unifier for K and $\overline{L}$. A *clause* is considered as a multiset of literals, usually written as an implication $A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$ where the As and Bs are atoms. Clauses with $m \geq 1$ are called *program clauses* with *head literals* A_i and *body literals* B_i, if present. Negative clauses are written as $\leftarrow B_1 \wedge \ldots \wedge B_n$.

We assume our clause sets to be in *Goal normal form*, i.e. there exists only one negative clause which furthermore does not contain variables. Without loss of generality this can be achieved by introducing a new clause $\leftarrow Goal$ where $Goal$ is a new predicate symbol, and by replacing every negative clause $\leftarrow B_1 \wedge \ldots \wedge B_n$ by $Goal \leftarrow B_1 \wedge \cdots \wedge B_n$.

We consider literal trees (aka *tableaux*), i.e. finite, ordered trees, all nodes of which, except the root, are labeled with a literal. A branch of length n consisting of the nodes $N_0, N_1, \ldots, N_n$ with root N_0 and leaf N_n is usually written as $[L_1 \cdot \ldots \cdot L_n]$ where Li is the label of N_i. In general, we find it practical to confuse a node with its label. The letters p and q are branch-valued variables, and if $p = [L_1 \cdot \ldots \cdot L_{n-1}]$ then $[p \cdot L_n]$ is the branch $[L_1 \cdot \ldots \cdot L_{n-1} \cdot L_n]$. The functions *First* and *Leaf* return the first *labeled*, resp. last node of a branch. The *extension of* $[p]$ *with clause* C, written as $[p] \circ C$, is the branch set $\{[p \cdot L] \mid L \in C\}$. Equivalently, in tree view this operation extends the branch $[p]$ by $|C|$ new nodes which are labelled with the literals from C.

[1]Early cancellation pruning was introduced in [6] within the context of a nearHorn-Prolog variant, InH-Prolog.

A tableaux is represented by a set of branches; branch sets are denoted by the letters $\mathcal{P}, \mathcal{Q}$. We write $\mathcal{P}, \mathcal{Q}$ and mean $\mathcal{P} \cup \mathcal{Q}$. Similarly, $[p], \mathcal{Q}$ means $\{[p]\}, \mathcal{Q}$. We write $X \in [p]$ iff X occurs in $[p]$, where X is a node or a literal label. A substitution σ is applied to a branch set $\mathcal{P}$, written as $\mathcal{P}\sigma$, by applying σ to all labels of all nodes in $\mathcal{P}$.

Branches may be labelled with a "$\star$" as *closed*; branches which are not closed are *open*. A tableaux is *closed* if each of its branches is closed, otherwise it is *open*.

2.1 The Basic Calculus

Definition 2.1 (Restart Model Elimination) A *computation rule* is a total function c which maps a tableau to one of its open branches[2].Let S be a clause set in *Goal* normal form. The inference rules of RME are defined as follows:

Extension Step:

$$\frac{[p], \mathcal{P} \qquad A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n}{([p \cdot A_i]\star, [p] \circ (A_1 \vee \ldots \vee A_{i-1} \vee A_{i+1} \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n), \mathcal{P})\sigma}$$

if (1) $A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n$ (with $m \geq 1$, $n \geq 0$ and $i \in \{1, \ldots, m\}$) is a new variant (called *extending clause*) of a clause in S, and
(2) $(Leaf([p]), A_i)$ is a connection with MGU σ. A_i is called the *extension literal*.

Reduction Step:

$$\frac{[p], \mathcal{P}}{([p]\star, \mathcal{P})\sigma}$$

if $(L, Leaf([p]))$ is a connection with MGU σ, for some $L \in [p]$.

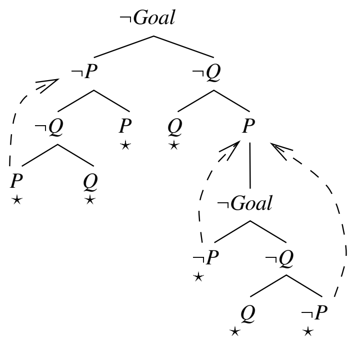
Restart Step:

$$\frac{[p], \mathcal{P}}{[p] \circ First([p]), \mathcal{P}}$$

if $Leaf([p])$ is a positive literal.

A *restart model elimination derivation* (RME derivation) of $\mathcal{P}_n$ is a sequence $(([\neg Goal] \equiv \mathcal{P}_0), \mathcal{P}_1, \ldots, \mathcal{P}_n)$ of tableaux where $\mathcal{P}_i$ is obtained from $\mathcal{P}_{i-1}$ by one single application of one of the above inference rules $(1 \leq i \leq n)$. Any branch set which is derivable by some RME derivation is also called a *RME tableau*. Finally, a *RME refutation* is an RME derivation such that $\mathcal{P}_n$ is closed. The term "RME" is dropped if context allows. □

Note that in extension steps we can connect only with the head literals of input clauses. Since in general this restriction is too strong, because it destroys completeness, we have to "restart" the computation with a fresh copy of a negative clause. This is achieved by the restart rule, because refutations of clause sets in *Goal* normal form always start with $First([p]) \equiv \neg Goal$, and thus only extension steps are possible to $\neg Goal$, which in turn introduce a new copy of a negative clause. The figure on the right conatins a closed RME tableaux (positive *Goal* nodes are not displayed, dashed lines indicate reduction steps).



2.2 Refinements of RME

Refinement: Head Selection Function. Unlike model elimination (ME), RME restricts in extension steps the connections into the extending clause to the head literals. We now go one step further, by introducing a *head selection function*. This concept is also present in Plaisted's Problem Reduction Formats [7], but not in the nearHorn Prolog family [5].

[2]It is required that c is *stable under lifting*, which means that for any substitution σ, whenever $c(\mathcal{Q}\sigma) = [q]\sigma$ then $c(\mathcal{Q}) = [q]$.

Definition 2.2 (Head Selection Function, [1]) A *head selection function* f is a function that maps a clause $A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ with $n \geq 1$ to an atom $L \in \{A_1, \ldots, A_n\}$. L is called the *selected literal* of that clause by f[3]. A RME derivation is called a *derivation with selection function* f if in every extension step the extension literal A_i is selected in $A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$ by f. $\square$

For example, to derive the RME tableaux in Section 2.1 requires a head selection function which selects Q in $Q \vee P \leftarrow$. The head selection function is a severe restriction of the calculus, but it can be applied and combined with some other refinements (but not all) to still yield a complete calculus.

Refinement: Strictness. In RME there are two possibilities to further derive from an open branch with positive leaf literal: either the branch can be closed by a reduction step or it can be extended in a restart step. In our sample refutation (Section 2.1) both possibilities are contained. In *strict* RME derivations we forbid reduction steps at positve leaf literals (i.e. in Definition 2.1, reduction step, $Leaf([p])$ must be a negative literal). See [1] for further discussion.

Refinement: Independence of the Goal Clause. In order to arrive at a really *goal*-oriented calculus, one wants to restrict the starting branch set to be derived from a negative clause (negative clause *before* the transformation to *Goal* normal form). Since we assume *Goal* normal form let us call any clause $Goal \leftarrow B_1 \wedge \cdots \wedge B_n \in S$ a *goal clause*. What we really want is to be independant of this goal clause:

Definition 2.3 A (refinement of the) RME caculus is called *independent of goal clause* if every derivation $(([\neg Goal] \equiv \mathcal{P}_0), \mathcal{P}_1)$ where $\mathcal{P}_1$ is obtained by extension with a goal clause from a minimal unsatisfiable subset of the clause set S can be extended to a refutation, if a refutation exists. $\square$

Refinement: Regularity. This is a well-know "loop-check" refinement: a branch is regular iff no literal occurs more than once on it; a tableaux is regular if each of its branches is regular. Regularity is easy to implement[4] and it is one of the most effective restrictions for model elimination procedures. Unfortunately, the regularity check is *not* compatible with RME.

Definition 2.4 (Blockwise Regularity, [4]) However, what can be achieved is *blockwise regularity*: A branch $[p] = [\neg B^1_1 \cdots \neg B^1_{k_1} \cdot A^1 \cdot \neg B^2_1 \cdots \neg B^2_{k_2} \cdot A^2 \cdots A^{n-1} \cdot \neg B^n_1 \cdots \neg B^n_{k_n}]$ (where the As and Bs are atoms) is called *blockwise regular* iff

1. $A^i \neq A^j$ for $1 \leq i,j \leq n-1$, $i \neq j$ *(Regularity wrt. positive literals)*, and
2. $B^l_i \neq B^l_j$ for $1 \leq l \leq n$, $1 \leq i,j \leq k_l$, $i \neq j$ *(Regularity inside blocks)*.

A tableaux is called *blockwise regular* iff every branch in it is blockwise regular. For example, the RME tableaux in Section 2.1 is blockwise regular. $\square$

[3]The head selection function f is required to be *stable under lifting* which means that if f selects $L\gamma$ in the instance of the clause $(A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m)\gamma$ (for some substitution γ) then f selects L in $A_1 \vee \ldots \vee A_n \leftarrow B_1 \wedge \ldots \wedge B_m$.

[4]For every branch one states pairwise syntactical inequality constraints.

Refinement: Early Cancellation Pruning. This refinement is essentially due to [6] and was called *strong early cancellation pruning rule* in Inh-Prolog.

Definition 2.5 (RME with Early Cancellation Pruning) We allow to label positive nodes in RME tableaux by the symbol "*r*" (meaning: used for *r*eduction steps). If node L is labeled in this way we will write L^r. The calculus *RME with early cancellation pruning (RMEP)* consists of the inference rule "extension step" of Def. 2.1 and the following inference rules:

Labeling Reduction Step: $$\frac{[p \cdot L \cdot q],\ \mathcal{P}}{([p \cdot L^r \cdot q]\star,\ \mathcal{P})\sigma}$$ if $(L, \mathit{Leaf}([p]))$ is a connection with MGU σ.

Restricted Restart Step: $$\frac{[p],\ \mathcal{P}}{[p \cdot \mathit{First}([p])],\ \mathcal{P}}$$ if $\mathit{Leaf}([p])$ is a positive literal, and the leafmost positive inner node of $[p]$, if it exists, is labeled with r.

The notion of *derivation* is taken from Def. 2.1. A RMEP refutation is a derivation of closed RME tableau where every inner positive node is labeled with r[5]. □

The idea of the early cancellation pruning is to achieve a relevance check: a new "case" by means of a restart step applied to $[\cdots L' \cdots L]$ may only be examined if the previous case L' turned out to be "relevant" for the derivation of the new case L. Here, "relevant" means that L' is the target for a reduction step *before* (hence "early") the restart step at L is attempted. For example, in Figure 2, restart at leaf D is not possible because ancestor B is not labelled with r.

Most of our calculi variants allow for *arbitrary* computation rules. The notable exception are the variants which employ the early cancellation pruning. In these cases we have to restrict to the following class of computation rules which prefers negative leafs over positive ones:

Definition 2.6 (Negative Preferrence Computation Rule) A *negative preferrence computation rule* is a computation rule c such that whenever a positive inner node L is contained in an open branch $[p]$ with positive leaf, and L is contained in an open branch $[p']$ with negative leaf, then c does not select $[p]$. □

3 Combining Refinements

Each refinement of the previous section was motivated in some way, so the question arises to what extent they can be combined. Unfortunately, some combinations cause incompleteness. Further, these incompatibilities are "inherent", in the sense that completeness cannot be recovered by relaxing other refinements, such as giving up "regularity" or "strictness". Figure 1 contains a summary of both, these negative results, and the positive results.

	Calculus	*Selection function*	*Regularity*	*Indep. of goal clause*	*Completeness*
(0)	ME	–	Full	Yes	Yes
(1)	RME	with	Blockwise	No	Yes
(2)		without	Blockwise	Yes	Yes
(3)	RMEP	with	–	–	No
(4)		without	Blockwise	Yes	Yes

Figure 1: Summary of properties of model elimination variants. "–" means "does not apply".

[5]We emphasize that in order to make the labeling meaningfull, the r-label is to be attached to the *node* (but not to the label), such that it is shared with other branches.

3.1 Negative Results

The negative results (the "no" entries in Figure 1) are shown by appropriate counterexamples to the assumption that the combination of the indicated features would yield a complete calculus.

"Head selection function" is incompatible to "early cancellation pruning". This addresses line (3) in Figure 1. Consider the clause set $M_1 = \{\ \leftarrow A,\quad \underline{A} \vee D \leftarrow,\quad A \leftarrow B \wedge D,\quad B \vee \underline{C} \leftarrow,\quad A \leftarrow C\}$. There is no RMEP refutation of the *Goal* normal form of M_1 with a head selection function which selects in M_1 the underlined atoms in the clause heads. Figure 2 shows an exhaustive case analysis: either the derivation contains a negative leaf $\neg D$ and gets stuck because the sole clause containing D in the head is $\underline{A} \vee D$, but D is not selected. Or, the other derivations are not weakly connected, so that a restart step at the positive branches is not permitted. There are some variations of these derivations by re-using clauses on a branch, but all of these run into the same problems. On the other side, there is a RMEP refutation of the *Goal* normal form of M_1 without selection function. Hence, in sum, the concept "head selection function" is not compatible to RMEP.

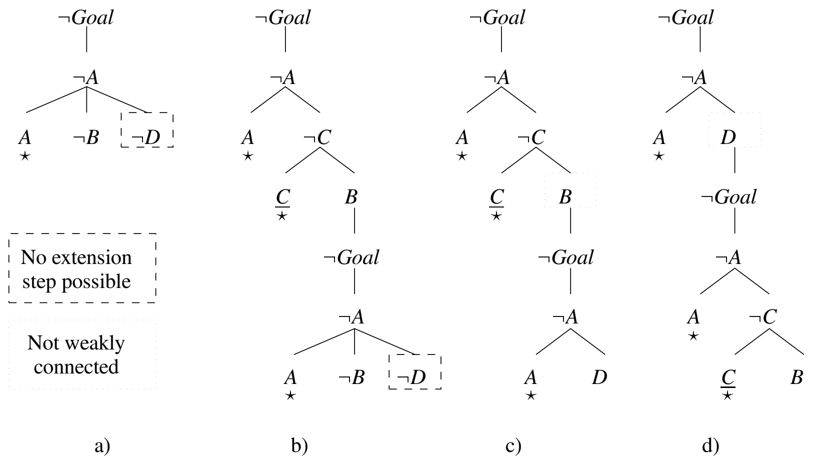


Figure 2: "Selection Function" is incompatible to "Early Pruning".

"Head selection function" is incompatible to "independence of the goal clause". This addresses line (1) in Figure 1. Consider the clause set $M_2 = \{\ \leftarrow P,\quad \leftarrow Q,\quad \underline{P} \vee Q \leftarrow\}$., which was also used in Section 2.2. It is easy to see that there is no RME refutation (and hence no RMEP refutation either) of the *Goal* normal form of M_2 with goal clause $Goal \leftarrow Q$ and a head selection function which selects P in $P \vee Q \leftarrow$. However, chosing $Goal \leftarrow P$ as the goal clause admits a RME refutation. In sum, the concepts "head selection function" and "independence of the goal clause" are incompatible for all considered RME variants.

3.2 Positive Results

For space reasons we can only cite the relevant results. Further, we will state the ground versions only. Lifting to the first-order case can be done along the lifting proof in [4]. We recall only that both the "computation rule" and the "head selection function" were defined to be stable under lifting (Defs. 2.1 and 2.2), which enables lifting them to the first-order level.

A standard strategy for completeness proofs of related calculi is to prove completeness of the *weakest* variants only, i.e. the variants, the refutations of which can stepwisely be simulated by the other variants. For instance, strict RME is weaker than non-strict RME. For the case of restart model elimination there are the following two weakest variants:

- RME with "head selection function", but without "independence of the goal clause. This addresses the "completeness" entry in line (1) in Figure 1. See [4] for a proof.
- RME without "selection function" but with "independence of the goal clause" and with "early cancellation pruning". This addresses the "completeness" entries in lines (2) and (4) in Figure 1. Since this result is new we state it here explicitly; see the full version [3] for a proof.

Theorem 3.1 (Ground completeness of Blockwise Regular Strict RMEP) *Let S be a minimal unsatisfiable ground clause set in Goal normal form, c be a negative preferrence computation rule. Then, for any clause $G = (Goal \leftarrow B_1 \wedge \cdots \wedge B_n) \in S$ there is a strict, blockwise regular RMEP refutation via c with top clause $\leftarrow Goal$ and goal clause G used in the first extension step.*

4 Conclusions

We studied various refinements for the restart model elimination calculus. One of our main concerns was compatibility among them. Further, we developed a new variant, RME with early cancellation pruning.

References

1. Peter Baumgartner and Ulrich Furbach. Model Elimination without Contrapositives and its Application to PTTP. *Journal of Automated Reasoning*, 13:339–359, 1994. Short version in: Proceedings of CADE-12, Springer LNAI 814, 1994, pp 87–101.
2. Peter Baumgartner and Ulrich Furbach. PROTEIN: A *PRO*ver with a *T*heory *E*xtension *I*nterface. In A. Bundy, editor, *Automated Deduction – CADE-12*, volume 814 of *Lecture Notes in Aritificial Intelligence*, pages 769–773. Springer, 1994. Available in the WWW, URL: `http://www.uni-koblenz.de/ag-ki/Systems/PROTEIN/`.
3. Peter Baumgartner and Ulrich Furbach. Refinements for Restart Model Elimination. Fachberichte Informatik 24–96, Universität Koblenz-Landau, Institut für Informatik, Rheinau 1, D-56075 Koblenz, 1996.
4. Peter Baumgartner, Ulrich Furbach, and Frieder Stolzenburg. Computing Answers with Model Elimination. *Artificial Intelligence*, 90(1–2):135–176, 1997.
5. D.W. Loveland. Near-Horn Prolog. In J.-L. Lassez, editor, *Proc. of the 4th Int. Conf. on Logic Programming*, pages 456–469. The MIT Press, 1987.
6. D.W. Loveland and D.W. Reed. A near-Horn Prolog for Compilation. In Jean-Luis Lassez and Gordon Plotkin, editors, *Computational Logic — Essays in Honor of Alan Robinson*, chapter III/16, pages 542–564. MIT Press, 1991.
7. D. Plaisted. Non-Horn Clause Logic Programming Without Contrapositives. *Journal of Automated Reasoning*, 4:287–325, 1988.