A Model Elimination Calculus for Generalized Clauses

Toni Bollinger
IBM Deiitschland GmbH - Wisseiischaftlichcs Zentrum
Institut fur Wissensbasierte Systeme
SchloBstr.70
D-7000 Stuttgart 1, Germany
e-mail: bollinRe@fls01iloR.bit.net

Abstract

Generalized clauses differ from (ordinary) clauses by allowing conjunctions of literals in the role of (ordinary) literals, i.e. they are disjunctions of conjunctions of simple literals. An advantage of this clausal form is that implications with conjunctive conclusions or disjunctive premises are not split into multiple clauses. An extension of Lovelands model elimination calculus [Loveland, 1969a, Loveland, 1978] is presented able to deal with such generalized clauses. Furthermore we describe a method for generating lemmas that correspond to valid instances of conjunctive conclusions. Using these lemmas it is possible to avoid multiple proofs of the premises of implications with conjunctive conclusions.

1 Introduction

Several extensions of the resolution rule and its refinements have been proposed able to deal with quantifier free first order logic formulas (e.g. see [Muray, 1982] for an extension of the resolution rule and [Stiekel, 1982] for a modification of the connection-graph procedure). These proof procedures have the advantage that it is not necessary to build a clausal normal from, i.e., despite from establishing a prenex normal form and skolemizing existentially quantified variables no normalization operations arc necessary. This enhances the readability and understand ability of proofs and can also lead to shorter deductions.

However, these modifications do not necessarily exploit the control structure implicit in logical formulas. In particular, although an implication *a* —> *b* A *c* is not broken into two clauses navf» and -a V c, it is not excluded that *a* has to be proven twice in a proof of *b* A *c.* This happens when, for instance the proof search performs a kind of backward chaining.

In this paper we present an extension of the model elimination calculus [Loveland, 1969a, Loveland, 1978] which overcomes this drawback. It works on so called *generalized clauses* which are disjunctions of *complex literals* being conjunctions of (simple) literals. For example, the implication (aVb) -> *(cAd)* is equivalent to the generalized clause (-a A *->b)* V *(r* A *d)* where *->a* A *-ib* and *r* A *d* are complex literals.

This restriction to generalized clauses is mainly motivated by pragmatic considerations, as we use the extended model elimination calculus for processing the predicate logic part of the knowledge representation language LLILOG [Pletat and v. Luck, 1990, Bollinger and Pletat, 1991) in the LEU/2[1] text understanding and question answering system [Geurts, 199()]. Items of a knowledge base need to be easily comprehensible. This is the case for Prolog like rules and facts, but not for arbitrary logical formulas. Even if they are quite simple they can be misinterpreted, e.g., for seeing that the formulae *a* --» *(b* —> *r)* and *(a* —* *b)* —* *c* are not equivalent one needs at least some basic knowledge in logic. On the other hand by using definite or indefinite clauses, knowledge cannot be expressed in a compact way. Generalized clauses are therefore a good compromise. Rules with a conjuntive conclusion or a disjunctive premise are not torn apart. Equivalences between conjunctions of literals can be represented by two generalized clauses.

In the next section we explain how Lovelands weak mode] elimination calculus can be adapted to generalized clauses. Compared to Lovelands calculus, deductions in this calculus require in general less inference steps as well as less computations. We then describe a method for generating lemmas. Using these lemmas, we show that redundant proofs of the premises of rules with a conjunctive conclusion can be avoided. We conclude with a discussion of the calculus.

2 Extending Model Elimination to Generalized Clauses

2.1 Model Elimination for Ordinary Clauses

Model Elimination[2] can be considered as a refinement of linear resolution where resolution steps are restricted to input clauses. Resolution steps with predecessor clauses in the proof tree are simulated by the reduction rule. For this it is necessary to mark resolved literals instead of deleting them.

'LEU is the acronym for "LILOG Experinienticnimgrbung" - LILOG experimental environment.

[2]To be more precise it is Loveland's *weak* model elimination method [Loveland, 1978] that we will present in the following.

More precisely, instead of clauses, we have *chains* which are ordered sequences of ordinary literals, called *O-literals*, and resolved literals, *R-literals*. R-literals are enclosed in brackets ([L]) for distinguishing them from O-literals. Chains containing only O-literals are *elementary* chains. *Unit* chains consist of only one literal. We call the universal closure of the disjunction of its O-literals the *clausal interpretation* of a chain. A chain is called *admissible* if the right-most literal is an O-literal[3]. The *contraction-operation* transforms a chain into an admissible one by removing all R-literals to the right of the right-most O-literal.

The model elimination method consists of two inference operations:

- the *extension rule* corresponding to the resolution rule and
- the *reduction rule*.

For defining them we use the notion of *complementarity of literals*:

Definition 1 (Complementarity of literals) *Two literals L_1 and L_2 can be made complementary with a most general unifier Θ iff L_1 and the complement of L_2 are unifiable with a most general unifier Θ.*

The two model elimination rules are defined as follows:

Definition 2 (ME extension) *Let $c_1 = c_1' L_1$ and $c_2 = c_2' L_2 c_2''$ be two admissible chains, L_1 and L_2 be O-literals and α be a renaming of the variables in c_2 such that c_1 and $c_2\alpha$ have no variables in common. If L_1 and $L_2\alpha$ can be made complementary with a most general unifier Θ, then the contraction of $(c_1'[L_1]c_2'\alpha c_2''\alpha)\Theta$ is called the* ME extension *of c_1 with c_2.*

Definition 3 (ME reduction) *Let $c = c'[L_1]c''L_2$ be an admissible chain. If L_1 and L_2 can be made complementary with a most general unifier Θ then the contraction of $(c'[L_1]c'')\Theta$ is a result of the* ME reduction *of c.*

The rationale behind the reduction rule is that in a chain $c'[L]c''$ the subchain $c'L$ is always an instance of a chain used for an extension step. Hence the reduction of a chain corresponds to an extension step eventually followed by a factorization.

For completing the presentation of model elimination we have to introduce the notions of model elimination (ME) deduction and model elimination (ME) refutation:

Definition 4 (ME deduction) *Given a set S of elementary chains, an* ME deduction *of a chain c from S is a finite sequence of chains $c_1, \ldots, c_n$ with*

- $c_1 \in S$,
- *for $i=2,\ldots,n$ c_i is obtained by ME extension of c_{i-1} with an (auxiliary) chain from S or by ME reduction of c_{i-1},*
- $c = c_n$.

[3]Loveland's definition in [Loveland, 1978] contains 3 supplementary conditions for the admissibility of chains. But it is easy to see and has already been stated in [Casanova *et al.*, 1989] that their removal neither affects soundness nor completeness of (weak) model elimination.

Definition 5 (ME refutation) *A refutation of a set S of elementary chains is a deduction from S of the empty chain $\square$.*

Example 1 *We will use the following example throughout this paper for illustrating our modifications of the standard model elimination calculus.*
Given the following axioms:
ax_1: $\forall x G(x) \vee Y(x) \rightarrow C(x) \wedge S(x)$,
ax_2: $G(p) \vee Y(p)$,
where $G(x)$ may stand for "x is a gourmet", $Y(x)$ for "x is a yuppy", $C(x)$ for "x likes to drink champaign", $S(x)$ for "x likes to eat snails" and p can be interpreted as "Peter", one likes to prove the goal
g: $\exists x C(x) \wedge S(x)$.
After having built a clausal normal form of the axioms and the negated goal we get the following refutation of the resulting set of chains:

(a_{11})	$\neg G(x)C(x)$	
(a_{12})	$\neg G(x)S(x)$	
(a_{13})	$\neg Y(x)C(x)$	
(a_{14})	$\neg Y(x)S(x)$	
(a_2)	$G(p)Y(p)$	
(1)	$\neg C(x)\neg S(x)$	*negated goal*
(2)	$\neg C(x)[\neg S(x)]\neg G(x)$	*ext. with a_{12}*
(3)	$\neg C(p)[\neg S(p)][\neg G(p)]Y(p)$	*ext. with a_2*
(4)	$\neg C(p)[\neg S(p)][\neg G(p)][Y(p)]S(p)$	*ext. with a_{14}*
(5)	$\neg C(p)$	*reduction*
(6)	$[\neg C(p)]\neg G(p)$	*ext. with a_{11}*
(7)	$[\neg C(p)][\neg G(p)]Y(p)$	*ext. with a_2*
(8)	$[\neg C(p)][\neg G(p)][Y(p)]C(p)$	*ext. with a_{13}*
(9)	$\square$	*reduction*

Extension and reduction are sound inference rules. Further they form a (refutationally) complete calculus. The following two theorems state this in a more formal way:

Theorem 1 (Soundness of ME) *If a chain c is the result of an ME deduction from a set S of elementary chains, then the clausal interpretation of c is a logical consequence from S.*

Theorem 2 (Completeness of ME) *If a set S of elementary chains if minimally unsatisfiahle, i.e., if every proper subset of S is satisfiable, then for any chain r e S there is a ME refutation of S starting wtth c.*

2.2 Resolution for Generalized Clauses

First we formally define the notion of complex literal and generalized clause.

Definition 6 (Complex literal) *A* complex literal *ts a conjunction of (simple) literals.*

Definition 7 (Generalized clause)
A generalized *clause is a disjunction of complex literals.*

When resolving two arbitrary quantifier free formulas (see [Muray, 1982]) first the polarities of all atomic subformulas have to be determined. If they contain two unifiable atomic subformulas with opposite polarity, the unifying substitution is applied to the two formulas. Then all occurences of the atom with positive polarity are replaced by F, representing *false,* and all occurrences of the negative atom in the other formula by T, standing for *true.* The disjunction of these two modified formulas represents the resolvent, which is further simplified by exploiting properties of logical connectives like o A F < - > F o r a V F ^ a.

Adapting this resolution rule to generalized clauses is straightforward. In generalized clauses the sign of the simple literals indicates the polarity of their atomic formulas. During the resolution operation the positive literal is replaced by F, such that the complex literals it appears in can be reduced to *F* (complex literals are conjunctions). Hence they can be removed from the resolvent. The same happens with the complex literals where the negative (simple) literal occurrs in, as its atomic formula is replaced by *T* which is equivalent to replacing the literal by *F.* Therefore resolving two generalized clauses CG_1 and CG_2 consists of unifying a simple literal *L* of CG_1 with the complement of a simple literal *L2* from CG2,APPLYING the unifying substitution 0 to CG_1 V CG_2 and discarding from the resulting generalized clause all complex literals with occurrences of L10 or L_2 0.

2.3 Model Elimination for Generalized Clauses

The notions introduced in Section 2.1 can easily be adapted to generalized clauses. Now we have *complex 0- and R-literals* and *generalized chains* consisting of complex O- and R-literals. Complex literals are noted as conjunctions of simple literals. If a complex O-literal consists of more than one simple literal it is enclosed in (round) parentheses. As before, R-literals are marked by (square) brackets. The chain *P(a)[Q(b)][P(b)* A *R(a)](Q(c)* A *P(c)),* for instance, consists of two complex O- and R-literals.

The definition of admissibility and the contraction operation have to be modified accordingly.

When resolving two generalized clauses only simple literals are compared. This also holds when two complex literals are made complementary. We can therefore define the *complementarity of complex literals* in the following way:

Definition 8 (Complementarity of compl. literals) *Let* L_1 = L_{11} A ... A L_{1m} *and L2* - -^21 A ... A Z/2n *be two complex literals.* L_1 *and* L_2 *can be made* complementary, *with a* most general unifier 0, *iff a simple literal* L_{1i} *from* L_1 *and the complement of a simple literal* L_{2J} *from* L_2 are *unifiable with most general unifier* 0.

There may be more than one possible most general unifier for making two complex literals complementary; e.g. for *P(a)AP(b)* and *-*P(x)* we have two most general unifiers: 0j = *{a/x)* and 02 = {fr/*}- But all the most general unifiers can be computed since the number of all simple literal pairs from two complex literals is finite.

When resolving two generalized clauses, all complex literals containing the resolved simple literals are discarded. Deleting or bracketing these literals during an extension step may lead to complications, for complex literals standing left of an R-literal may be deleted or transformed to an R-literal. It is therefore possible that the left subchain relative to an R-literal is no longer an instance of a predecessor chain in a deduction. But this is a condition for the correctness of the reduction rule.

Fortunately, it suffices to bracket resp. delete only the complex literals that are made complementary. We can therefore define *generalized model elimination (MGE) extension* and *reduction* rule in the following way:

Definition 9 (GME extension) *Let* $C_1 = C_1'\mathcal{L}_1$ *and* $C_2 = C_2'\mathcal{L}_2C_2''$ *be two admissible generalized chains,* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *be complex O-literals and* α *be a renaming of the variables in* C_2 *such that* C_1 *and* $C_2\alpha$ *have no variables in common. If* $\mathcal{L}_1$ *and* $\mathcal{L}_2\alpha$ *can be made complementary with a most general unifier* Θ*, then the contraction of* $(C_1'[\mathcal{L}_1]C_2'\alpha C_2''\alpha)\Theta$ *is called the* GME extension *of* C_1 *with* C_2.

Definition 10 (GME reduction) *Let* $C = C'[\mathcal{L}_1]C''\mathcal{L}_2$ *be an admissible generalized chain. If the complex literals* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *can be made complementary with a most general unifier* Θ *then the contraction of* $(C'[\mathcal{L}_1]C'')\Theta$ *is a result of the* GME reduction *of* C.

GME deduction and GME refutation can be defined analogously to ME deduction and ME refutation. One has only to replace in Definition 4 and 5 "chain" by "generalized chain" and "ME" by "GME".

Example 2 *Having built generalized chains for the axioms and the negated goal of Example 1 we get the following GME-refutation:*

(a₁)	$(\neg G(x) \wedge \neg Y(x))(C(x) \wedge S(x))$	
(a₂)	$G(p)Y(p)$	
(1)	$\neg C(x)\neg S(x)$	*neg. goal*
(2)	$\neg C(x)[\neg S(x)](\neg G(x) \wedge \neg Y(x))$	*ext. (a₁)*
(3)	$\neg C(p)[\neg S(p)][\neg G(p) \wedge \neg Y(p)]G(p)$	*ext. (a₂)*
(4)	$\neg C(p)$	*reduction*
(5)	$[\neg C(p)](\neg G(p) \wedge \neg Y(p))$	*ext. (a₁)*
(6)	$[\neg C(p)][\neg G(p) \wedge \neg Y(p)]G(p)$	*ext. (a₂)*
(7)	$\square$	*reduction*

GME extension and GME reduction are sound inference rules. The soundness of GME extension follows from the soundness of resolution for quantifier free formulas. A GME reduction can be simulated by an extension step with a predecessor clause in the deduction such that the clausal interpretations of the resulting chains are equivalent. Hence the GME reduction rule is sound too.

We get therefore the following theorem that can be proven by induction on the length of the concerning deduction:

Theorem 3 (Soundness of GME) *If a generalized chain* C *is the result of a GME deduction from a set* S *of elementary generalized chains, then the clausal interpretation of* C *is a logical consequence of* S.

The proof of the completeness of generalized model elimination is based on the following lemma, where the function *coc* produces a kind of conjunctive normal form of a generalized chain.

Lemma 1 *Let $\mathcal{S}$ be a set of generalized elementary chains and let* coc *("contained ordinary chains") be a mapping from generalized chains to sets of ordinary chains that is defined as follows:*
$coc(\mathcal{C}) = \{L_{1i_1}L_{2i_2}...L_{ni_n} \| j_i = 1,...,m_j, j = 1,...,n\}$ *with* $\mathcal{C} = \mathcal{L}_1\mathcal{L}_2...\mathcal{L}_n$, *where* $\mathcal{L}_j = L_{j1} \wedge ... \wedge L_{jm_j}$ *is either a complex O- or an R-literal and L_{ji} inherites the literal type from $\mathcal{L}_j$. Furthermore let $S = \bigcup_{\mathcal{C}\in\mathcal{S}} coc(\mathcal{C})$. If there is a ME-deduction of a chain c from S then there is also a GME-deduction of a generalized chain $\mathcal{C}$ from $\mathcal{S}$ with $c \in coc(\mathcal{C})$.*

Given an ME-deduction of a chain c from S, the proof of this lemma consists essentially of constructing from $\mathcal{S}$ an isomorphic GME-deduction of a generalized chain $\mathcal{C}$ containing c. Being technical it is omitted here and can be found in an extended version of this paper [Bollinger, 1991].

Theorem 4 (Completeness of GME) *If a set $\mathcal{S}$ of generalized elementary chains is minimally unsatisfiable, i.e., if every proper subset of $\mathcal{S}$ is satisfiable, then for any chain $\mathcal{C} \in \mathcal{S}$ there is a refutation of $\mathcal{S}$ starting with $\mathcal{C}$.*
Proof: *As $\mathcal{S}$ is unsatisfiable, $S = \bigcup_{\mathcal{C}\in\mathcal{S}} coc(\mathcal{C})$ is unsatisfiable too. Let $S' \subseteq S$ be a minimally unsatisfiable subset of S. For any $\mathcal{C} \in \mathcal{S}$ we have $coc(\mathcal{C}) \cap S' \neq \emptyset$. If that would not be the case, $\mathcal{S}\backslash\{\mathcal{C}\}$ would be unsatisfiable (due to $coc(\mathcal{S}\backslash\{\mathcal{C}\}) \supseteq S'$), which contradicts the assumption that $\mathcal{S}$ is minimally unsatisfiable.*
Let $c \in coc(\mathcal{C}) \cap S'$. Due to the completeness of model elimination there is a ME deduction of $\Box$ starting with c. As $c \in coc(\mathcal{C})$ Lemma 1 can be applied and we obtain that there is also a GME deduction of $\Box$ from $\mathcal{S}$ starting with $\mathcal{C}$. $\Box$

3 Generating Lemmas

Although GME-refutations in general are shorter than the corresponding ME-refutations, we still have the problem of redundant proofs of premises, whose rules have conjunctive conclusions. In Example 2 we saw that the chain a_1 was applied twice, whereas one application should suffice. In chain (2) the "premise" literal $\neg G(x) \wedge \neg Y(x)$ of a_1 stands to the right of the resolved literal. If this literal is proven one can deduce the conclusion. This is done by the two subsequent inference steps, i.e. at that moment $C(p) \wedge S(p)$ should have been deduced as a lemma. How can we achieve this?

We introduce a third type of literals called *lemma candidates* or *L-literals*. When extending a chain $\mathcal{C}$ with an auxiliary chain $\mathcal{C}_a$ the resolved literal in $\mathcal{C}_a$ is declared as a lemma candidate and is put beside the bracketed literal, the resolved (complex) literal of $\mathcal{C}$ in the resulting chain. Lemma candidates are put into braces, such that in Example 2 above, the extension step of chain (1) with chain a_1 yields:

(2') $\neg C(x)[\neg S(x)]\{C(x) \wedge S(x)\}(\neg G(x) \wedge \neg Y(x))$

If it is possible to deduce the empty chain from the subchain to the right of an L-literal, the corresponding instantiation of the L-literal becomes a valid lemma.

More formally we have:
If $\mathcal{C}_a^1 \neq \Box$ or $\mathcal{C}_a^2 \neq \Box$, an extension step of $\mathcal{C}_1 = \mathcal{C}_1^1\mathcal{L}_1$ with $\mathcal{C}_a = \mathcal{C}_a^1\mathcal{L}_2\mathcal{C}_a^2$ yields,
$\mathcal{C}_2 = (\mathcal{C}_1^1[\mathcal{L}_1]\{\mathcal{L}_2\}\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$.
If there is a deduction of $\Box$ starting with $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$ and if Φ is the composition of the most general unifiers in that deduction, then we also have a deduction of $\mathcal{L}_2\Theta\Phi$ starting with $(\mathcal{L}_2\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$.

The refutation of $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$ has to be performed when trying to refute $\mathcal{C}_2$. Since chains represent disjunctions of literals, it cannot be taken for granted that a refutation of $\mathcal{C}_2$ contains also a refutation of $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$. For this no literal of $\mathcal{C}_1^1[\mathcal{L}_1]$ has to be involved in any extension or reduction step when, during the refutation of $\mathcal{C}_2$, $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$ is processed. This is certainly the case if the literals in $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$ are eliminated by extension steps. The situation is different for reduction steps. A reduction performed with an R-literal from $\mathcal{C}_1^1[\mathcal{L}_1]$ during the refutation of $\mathcal{C}_2$ can't be transferred to a refutation of $(\mathcal{C}_a^1\mathcal{C}_a^2)\Theta$. For that the R-literal has to stand to the right of $[\mathcal{L}_1]$ in $\mathcal{C}_2$ or a chain deduced from $\mathcal{C}_2$.

We take into account this restriction in the following way: Let us suppose that a reduction step can be applied to $\mathcal{C} = \mathcal{C}^1[\mathcal{L}_2]\mathcal{C}^2\mathcal{L}_1$ by making complementary the literals $\mathcal{L}_1$ and $\mathcal{L}_2$. The lemma candidates in $\mathcal{C}^1$ are not affected by this as the involved O- and R-literal are on the right of $\mathcal{C}^1$. However, for the lemma candidates in $\mathcal{C}^2$ the R-literal $[\mathcal{L}_2]$ stands to the left. Therefore these lemma candidates become invalid and have to be eliminated.

The modifications of the contraction operation, of the GME extension and the GME reduction rule are clear now. The contraction operation also deletes rightmost L-literals, that become valid lemmas. The extension rule may introduce a new L-literal being the "resolved" literal of the auxiliary chain. L-literals standing between the R- and O-literal involved in a reduction operation are deleted.

Example 3 *The refutation of Example 2 is modified as follows if lemmas are generated:*
(a_1) $(\neg G(x) \wedge \neg Y(x))(C(x) \wedge S(x))$
(a_2) $G(p)Y(p)$
negated goal:
(1) $\neg C(x)\neg S(x)$
extension with (a_1):
(2) $\neg C(x)[\neg S(x)]\{C(x) \wedge S(x)\}(\neg G(x) \wedge \neg Y(x))$
extension with (a_2):
(3) $\neg C(p)[\neg S(p)]\{C(p) \wedge S(p)\}[\neg G(p) \wedge \neg Y(p)]G(p)$
reduction and generation of lemma (l) $C(p) \wedge S(p)$:
(4) $\neg C(p)$
extension with (l):
(5) $\Box$

Loveland also proposes a method for generating lemmas. In [Loveland, 1969b], he introduces the notion of *scope* associated with R-literals. It is initialized to 0 and updated when the R-literal is involved in a reduction step. In that case, it is set to the maximum of its actual scope and the number of R-literals standing to the right of it. If due to subsequent reduction and extension steps the scope exceeds the number N of R-literals to the right it is set to N. The subchain in the scope of an R-literal L_R with scope N is the subchain from L_R to the N-th R-literal to the right.

Lemmas can be formed when an R-literal L_R is removed by the contraction operation. The maximal sub-

chain in the scope of an R-literal is taken in which L_R occurs (as the last literal). The lemma is generated by leaving the O-literals unchanged and turning the negation of the R-literals into O-literals.

Example 4 *We show a part of a deduction where lemmas are generated. The indices of the R-literals indicate their scope.*

(i) $P(a)[Q(x)]_0[P(y)]_0\neg Q(a)[R(z)]_0\neg P(a)$

reduction without contraction:

(i+1) $P(a)[Q(x)]_0[P(a)]_1\neg Q(a)[R(z)]_0$

contraction and gen. of lemma $\neg P(a)\neg Q(a)\neg R(z)$*:*

(i+1') $P(a)[Q(x)]_0[P(a)]_1\neg Q(a)$

reduction without contraction:

(i+2) $P(a)[Q(a)]_1[P(a)]_0$

contraction and generation of lemma $\neg Q(a)\neg P(a)$*:*

(i+2') $P(a)$

chain in the scope of an R-literal is taken in which Lp occurs (as the last literal). The lemma is generated by leaving the O-litcrals unchanged and turning the negation of the R-literals into O-literals.

Example 4 We fhow a part of a deduction where lemma* are generated. The indices of the R-literals indicate their scope.

0) . r(")[Q(*)]oir(v)}o-Q(a){R(z)}o-r(a)

reduction without contraction:

(i+l) P(a)[Q(x)]0[P(a)}^Q(a)[R(z)}0

contraction and gen. of lemma -iP(a)-*Q(a)-*R(z):

(i + r)_ P(a)[<?(*)]o[P(a)I,-.Q(a)

reduction without contraction:

(i+t) P(«)[Q(a)],[/>)]„

contraction and generation of lemma ~>Q(a)-»P(a):

Specializing our method for lemma generation to ordinary chains we see that it corresponds exactly to the case where Loveland's method is restricted to the generation of unit lemma chains. Lemma candidates are not explicitely recorded since for simple literals they are identical to the negation of the corresponding R-literal. For complex literals this is necessary, as we do not have £j0 =: -i£2^ in general when they are complementary.

The removal of lemma candidates after a reduction step has its counterpart in the fact that, according to Loveland's method, the generation of lemmas is based on the maximal subchain in the scope of an R-literal yielding up to the rightmost R-litcral. This has the effect that lemmas for R-literals within this subchain (except the leftmost one) are not created.

It is easy to generalize Lovcland's method to generalized chains. In this way it is possible to generate lemmas being the instance of disjunctive conclusions. However, from a practical point of view, non-unit lemmas are less useful in general.

4 Discussion

Generalized clauses and chains are interesting extensions of their standard versions. Any quantifier free logical formula can be represented by one such clause or chain. This facilitates especially the processing of disjunctive goals. In contrast to ordinary chains (and clauses) a negated goal can always be transformed to exactly one generalized chain. Moreover, having a set of ordinary chains representing the negated goal, a refutation may succeed with only one of them as the starting chain. Hence for ordinary chains we have the problem of choosing the right chain steming from the negated goal. This problem does not occur for generalized chains, as we have only one goal chain, and from Lemma 1 we know, that its refutation succeeds if it succeeds for one of the ordinary goal chains.

From an implementational viewpoint, generalized model elimination (GME) constitutes only a slight mod-

[illegible]

complex literals are made complementary. The only real difference to ME consists in the fact that several simple literals are deleted or bracketed during an extension and reduction operation. Of course, if lemmas should be generated, additional computations have to be performed concerning the L-literals. But compared with Loveland's method for lemma generation their complexity is not higher. The computational overhead of GME with respect to ME is therefore low.

However, as GME refutations are generally shorter than ME refutations, (cf. Example 1 compared to Example 2) the overall amount for computing a refutation is lower for GME. This effect is enhanced if lemmas are generated, since the really interesting lemmas are those generated for rules with conjunctive conclusions.

Looking at the search space, one realizes that the number of potential extension and reduction steps is greater for GME than for ME, since complex literals may consist of several simple literals. We know also from the proof of Lemma 1 that for every ME-deduction of a chain c there is an isomorphic GME-deduction of a generalized chain containing c. Hence the search space explored for finding GME-refutations contains more redundancies, i.e., paths leading to identical solutions. This is not critical, if one solution is looked for, as GME-refutations can be shorter. In such a case a depth oriented search strategy is appropriate. If all solutions are looked for, it is necessary to employ a strategy that avoids exploring redundant paths. The development of such strategies is one of the points that need to be investigated further.

GME may perform better than ME if the ressources for the search are restricted. E.g., if the search depth is limited, with GME more solutions can be found, as GME-refutations are shorter.

The generation of lemma should be restricted to the useful ones, since the addition of every new lemma increases the search space. Certainly, creating a lemma candidate when performing an extension step with a unit chain is superfluous, as the deduced lemma is an instance of the used auxiliary unit chain. Simple L-literals are in general less interesting than complex L-literals. Therefore it may be reasonable to generate lemma candidates only if the resolved literal in the auxiliary chain is a real complex literal.

However even the most useful lemmas introduce new redundancies into the search space, i.e., the same solution can be found by using a lemma and by performing the extension and reduction steps that led to the deduction of the lemma. These new redundancies can be neglected, when only one solution is looked for, since it suffices to apply the lemmas first, before trying extension steps with non-unit chains.

When looking for all solutions, we have again the problem of avoiding paths leading to identical solutions. The situation becomes even more complex since the knowledge base changes dynamically with the addition of lemmas. It may also happen that for finding all solutions, lemmas computed at another branch of the proof tree have to be used. Controling the generation and the use of lemmas is therefore another point that will be investigated further.

One pragmatical solution would be to leave it to the user to decide when to apply a rule that lead to the introduction of a lemma. Like with the *cut* in Prolog, the user would be able to prune a part of the search space. The user has then the responsability for taking care that no solutions are lost.

In the LILOG project the knowledge engineers asked for such a feature. It happens often that a conjunction of literals should be treated as one literal. Introducing a new predicate symbol was not possible due to constraints imposed by the text understanding application. In such a case the solutions found by the use of lemmas were sufficient.

5 Conclusion

GME is a useful extension of ME. We have established some of its advantages and have pointed out areas of further investigation. Since GME applied to ordinary chains behaves exactly like ME, one has the flexibility to r.witch between GME and ME. The normalization procedure has only to produce ordinary chains, and the computational overhead for processing ME with the generalized method is low.

An order sorted version of this calculus with lemma generation has been implemented in Quintus Prolog[4] for the LEU/2 text understanding system. GME has proven to be very useful, given the nature of the application domain, in which a majority of the rules written in the knowledge representation language LLILOG HAD conjunctive conclusions.

Control issues have to be investigated further. The major problem is how to avoid investigating paths in the search space that lead to redundant solutions. The generation of lemmas can be considered as a kind of rote learning. One may also think of storing certain lemmas permanently or of applying machine learning techniques like "explanation based generalization" for creating more general lemmas. However, for avoiding the classical machine learning problem: "the more you know the slower you go", good strategies need to be developed for controling the generation, the use and eventually the permanent storage of lemmas.

Acknowledgements

I would like to thank Zuzana Dobes, Karl Hans Blasius and Udo Pletat for helpful comments on this paper.

References

[Bollinger and Pletat, 199l] Toni Bollinger and Udo Pletat. Knowledge in operation. IWBS-Report, IBM Deutschland, Scientific Center, 1991.

[Bollinger, 199l] Toni Bollinger. A model elimination calculus for generalized clauses. IWBS-Report, IBM Deutschland, Scientific Center, 1991.

[4] Quintus Prolog is a trademark of Quintus Computer Sys terns, Inc.

[Casanova *et* al., 1989] M.A. Casanova, R. Guerreiro, and A. Silva. Logic programming with general clauses and defaults based on model elimination. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence,* pages 395 400, Detroit, MI, 1989.

[Geurts, 1990] Bart Geurts. Natural language understanding in LILOG: An intermediate overview. IWBS-Report 137, IBM Deutschland, Scientific Center, 1990.

[Loveland, 1969a] D. Loveland. A simplified format for the model elimination theorem-proving procedure. *Journal of the Association for Commuting Machinery,* 16(3):349 363, 1969. Also published in [Siekmann and Wright son, 1983, pages 233-248].

[Loveland, 1969b] D. Loveland. Theorem-provers combining model elimination and resolution. In B. Meltzer and D. Michie, editors, *Mas chine Intelligence 4,* pages 73 86. Edinburgh University Press, Edinburgh, 1969. Also published in [Siekmann and Wrightson, 1983, pages 249-263].

[Loveland, 1978] D. Loveland. *Automated Theorem Proving: A Logical Basis,* volume 6 of *Fundamental Studies in Computer Science.* North-Holland, New York, 1978.

[Muray, 1982] N.V. Muray. Completely non-clausal theorem proving. *Artificial Intelligence,* 18(1):67 85, 1982.

[Pletat and v. Luck, 1990] U. Pletat and K. v. Luck. Knowledge representation in LILOG. In K. H. Blasius, U. Hedtstuck, and C.-R. Rollinger, editors, *Sorts and Types for Artificial Intelligence,* volume 449 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin, Germany, 1990.

[Siekmann and Wrightson, 1983] Jorg Siekmann and Graham Wrightson, editors. *Automation of Reasoning 2.* Springer-Verlag, Berlin, Germany, 1983.

[Stickel, 1982]
Mark E. Stickel. A nonclausal connection-graph resolution theorem-proving program. In *Proceedings of the 2nd National Conference of the American Association for Artificial Intelligence,* pages 229 233, Pittsburgh, Pa., 1982.