Splitting through new Proposition Symbols

Hans de Nivelle

Max-Planck Institut für Informatik

Saarbrücken, Germany

December 4th 2001

Overview of this talk

- What is the splitting rule, what is it good for?
- Problems with implementation of the splitting rule.
- A partial solution, and why is it partial.
- A complete solution.
- Conclusions, Future Work.

Proof Search with Resolution

Definition: A clause is a finite disjunction ($\vee$) of literals. A literal is an atom $p(t_1, \ldots, t_n)$, or the negation of an atom $\neg p(t_1, \ldots, t_n)$.

If, for two clauses $A_1 \vee R_1,\ \neg A_2 \vee R_2$ there is substitution Θ such that $A_1\Theta = A_2\Theta$, then $R_1\Theta \vee R_2\Theta$ is a logical consequence of $A_1 \vee R_1$ and $\neg A_2 \vee R_2$. We call this clause a resolvent.

If, for one clause $A_1 \vee A_2 \vee R$, there is a substitution Θ such that $A_1\Theta = A_2\Theta$, then $A_1\Theta \vee R\Theta$ is a logical consequence of $A_1 \vee A_2 \vee R$. We call this clause a factor.

If, for two clauses $A_1,\ A_2$, there is a substitution Θ, such that $A_1\Theta \subseteq A_2$, then A_2 is a logical consequence of A_1. We say that A_1 subsumes A_2.

Proof Search with Resolution (2)

We want to know if some set of clauses S is consistent. The resolution prover continues doing one of the following three things:

- Construct a factor of a clause in S, and add the factor to S.
- Construct a resolvent of two clauses in S, and add the resolvent to S.
- Find a clause in S, that is subsumed by another clause in S. If such clauses can be found, then the subsumed clause is deleted.

Theorem This is a complete proof search strategy, i.e. if the initial set of clauses is inconsistent, then eventually the empty clause will be derived.

The Splitting Rule

The splitting rule is the $\vee$-rule from semantic tableaux, applied in the resolution context:

$$C_1 \vee C_2$$

$$C_1 \qquad\qquad C_2$$

- There must be no shared variables between C_1 and C_2.
- In each of the branches, $C_1 \vee C_2$ is deleted. (because it is subsumed)

Advantages of the Splitting Rule

- The components of a split clause often can be used for backward subsumption, or backward simplification.
- There are resolution decision procedures that depend on the splitting rule. (For example the Gödel class, or the E^+-class)
- Experiments (Spass) show that the splitting rule is useful. More proofs and saturations can be found in shorter time.

Problems with the Splitting Rule

The splitting rule is hard to implement in a resolution prover:

One needs to introduce choice points in the resolution prover. All clause creations, simplifications and deletions need to be logged, and they need to be reversible.

The bookkeeping reduces the efficiency of the basic operations.

Examples:

Splitting is possible on the following clauses

$$p(X) \vee q(Y)$$

$$p(X, Y) \vee X \approx Y \vee q(Z) \vee r(Z)$$

Splitting is not possible on the following clauses

$$p(X, Y) \vee p(Z, T) \vee p(Y, Z)$$

$$\neg p(X) \vee \neg q(X) \vee X \approx Y \vee r(Y).$$

Examples (2)

On the following unsatisfiable clause set, splitting allows backward simplification:

$$p(0)$$

$$\neg p(s^{16}(0))$$

$$s(s(X)) \approx X \lor s(Y) \approx Y$$

both $s(s(X)) \approx X$ and $s(Y) \approx Y$ simplify $\neg p(s^{16}(0))$ into $\neg p(0)$.

A Partial Solution

A well-known solution is to simulate splitting by the intruduction of new proposition symbols.

The clause $C_1 \vee C_2$ can be replaced by two clauses $C_1 \vee \alpha$ and $\neg\alpha \vee C_2$.

Here α is a fresh propositional symbol.

α must be least preferred in C_1 and all clauses that can be derived from C_1.

$\neg\alpha$ must be most preferred in C_2.

In this way, C_2 is switched off until C_1 has been refuted.

This is splitting through new proposition symbols.

Why this is only a Partial Solution

Splitting through new symbols does indeed simulate splitting through backtracking, as long as only derivations are considered.

However, if one also considers simplification and subsumption, then it does not.

Example (simplification is lost!)

Clauses:

$$p(0),$$

$$\neg p(s^{16}(0)),$$

$$s(s(X)) \approx X \vee s(Y) \approx Y.$$

The last clause is split into

$$s(s(X)) \approx X \vee \alpha,$$

$$\neg\alpha \vee s(Y) \approx Y.$$

No backward simplifications are possible, due to α.

Solution

Use propositional symbols that cannot be matched for 'switching off' clauses.

Examples (formal definition comes later)

Subsumption Assume that C_1 subsumes C_2. Furthermore assume that, instead of C_1, there is the clause $C_1 \vee \alpha$ in the database. Then C_2 is replaced by $\neg\alpha \vee C_2$.

Simplification Assume that D_1 can simplify D_2 into D_3. Furthermore assume that, instead of D_1, there is the clause $D_1 \vee \alpha$ in the database.
Then D_2 is replaced by $D_3 \vee \alpha$ and $\neg\alpha \vee D_2$.

Examples:

Assume that there are clauses $p(X) \vee \alpha$ and $p(0) \vee q(0)$. The last clause can be replaced by $\neg\alpha \vee p(0) \vee q(0)$.

The clause $\neg\beta \vee p(0) \vee q(0)$ can be replaced by $\neg\alpha \vee \neg\beta \vee p(0) \vee q(0)$.

Assume that there are clauses $p(X) \vee \alpha \vee \beta$ and $p(0) \vee q(0)$. The last clause can be replaced by $(\neg\alpha \wedge \neg\beta) \vee p(0) \vee q(0)$.

Alternatively, one could replace it by

$$\neg\ \gamma \vee p(0) \vee q(0), \quad \neg\alpha \vee \gamma, \quad \neg\beta \vee \gamma,$$

with γ a new symbol.

Completeness Proof

- Proof by standard redundancy not possible, because the multiset condition is not fullfilled. (Replacement of C by $C \vee \alpha$)
- Proof by simulating real splitting is not possible, because the fresh-literal method is stronger.
- We do not want to prove completeness separately for each decision procedure/refinement of resolution.

Solution: We prove a 'meta-completeness' theorem:

If some calculs $\mathcal{C}$ is complete with some set of derivation rules and some set of simplification/ subsumption-rules, then this calculus remains complete if one adds splitting by fresh literals.

An abstract View of a Calculus

The plan is to extend a resolution-based calculus into a meta calculus (with adjoined propositional symbols)

Definition:

We define a calculus $\mathcal{C}$ by its derivation rules and its redundancy rules. A *calculus* is an ordered quadruple (A, D, R, e), where

- A is the set of clauses of $\mathcal{C}$.
- $D \subseteq A^* \times A$ is the set of derivation rules of $\mathcal{C}$.
- $R \subseteq A^* \times A$ is the set of redundancy rules of $\mathcal{C}$. It must be the case that R is reflexive and transitive.
- $e \in A$ is the empty clause of $\mathcal{C}$.

Transitivity/Reflexivity

- Relation R is reflexive if for all $a \in A$,

$$R(a, a).$$

- Relation R is transitive if

$$R(a_1, \ldots, a_n, a)$$

  and

$$R(b_1, \ldots, b_{i-1}, a, b_{i+1}, \ldots, b_m, b)$$

  imply

$$R(b_1, \ldots, b_{i-1}, a_1, \ldots, a_n, b_{i+1}, \ldots, b_m, b).$$

An abstract View of a Calculus (2)

Definition:

A saturated set of $\mathcal{C}$ is a set $M \subseteq A$, such that for each set of clauses $a_1, \ldots, a_n \in M$ $(n \geq 0)$ and clause $a \in A$, for which $D(a_1, \ldots, a_n, a)$, there are clauses $b_1, \ldots, b_m \in M$ $(m \geq 0)$, such that $R(b_1, \ldots, b_m, a)$.

M is a *saturation of* some set of initial clauses I if in addition, for each $a \in I$, there are clauses $a_1, \ldots, a_m \in M$, such that $R(a_1, \ldots, a_m, a)$.

Examples of calculi are the superposition calculus, all of resolution decision procedures.

Extension of Calculus $\mathcal{C}$ with splitting symbols

Definition:

Let $\mathcal{C} = (A, D, R, e)$ be a calculus. Let $(\Sigma, \prec)$ be a well-ordered set of propositional atoms, non-overlapping with any object in A. The *extended* calculus $\mathcal{C}^\Sigma$ is defined as follows:

- A^Σ consists of clauses of form

$$(\neg\sigma_1 \vee \cdots \vee \neg\sigma_p) \vee a \vee \tau.$$

  Each σ_i is a disjunction of atoms of Σ.

  Component a consists of one clause from A, possibly e.

  The set τ is a finite disjunction of atoms from Σ, possibly empty. If $p = 0$, $a = e$, or τ is empty, we omit it.

Derivation Rules D^Σ of extended Calculus

CONTEXT: If $D(a_1, \ldots, a_n, a)$, $(n \geq 0)$, then

$$D^\Sigma(a_1 \vee \tau_1, \ldots, a_n \vee \tau_n, \;\; a \vee (\tau_1 \vee \cdots \vee \tau_n)\;).$$

RESTORE: Let $c = (\neg\sigma_1 \vee \cdots \vee \neg\sigma_p) \vee a \vee \tau$ be a $\mathcal{C}^\Sigma$-clause.

Let

$$c_1 = \alpha_1 \vee \tau_1, \;\ldots,\; c_n = \alpha_n \vee \tau_n$$

be clauses that contain only Σ-atoms, for which each α_i is maximal in c_i.

Then we put $D^\Sigma(c, c_1, \ldots, c_n, d)$ for

$$d = a \vee (\tau \vee \tau_1 \vee \cdots \vee \tau_n).$$

Redundancy Rule R^Σ of extended calculus

Assume that

$$R(a_{1,1}, \ldots, a_{1,m_1}, b_1),$$

$$\ldots$$

$$R(a_{k,1}, \ldots, a_{k,m_k}, b_k).$$

Furthermore assume that

$$\neg a_{1,1} \vee \cdots \vee \neg a_{1,m_1} \vee b_1, \ldots, \neg a_{k,1} \vee \cdots \vee \neg a_{k,m_k} \vee b_k,$$

$$c_1, \ldots, c_n \models c$$

in propositional logic, ignoring the internal structure of the $a_{i,j}$ and b_i.

Then

$$R^\Sigma(c_1, \ldots, c_n, c).$$

Examples to R^Σ

Splitting itself:

$$R^\Sigma(\alpha \vee C_1,\ \neg\alpha \vee C_2,\ C_1 \vee C_2),$$

because

$$\neg C_1 \vee [C_1 \vee C_2],\ \neg C_2 \vee [C_1 \vee C_2],\ \alpha \vee [C_1],\ \neg\alpha \vee [C_2] \models [C_1 \vee C_2].$$

Subsumption: If $R(C_1, C_2)$, then

$$R^\Sigma(C_1 \vee \alpha,\ \neg\alpha \vee C_2,\ C_2),$$

because

$$[C_1] \vee \alpha,\ \neg\alpha \vee [C_2],\ \neg[C_1] \vee [C_2]\ \models [C_2].$$

Examples to R^{Σ} (2)

Simplification: If $R(C_1, C_2, D)$, then

$$R^{\Sigma}(C_1 \vee \alpha,\ \neg\alpha \vee D,\ C_2 \vee \alpha,\ D),$$

because

$$[C_1] \vee \alpha,\ \neg\alpha \vee [D],\ [C_2] \vee \alpha,\ \neg[C_1] \vee \neg[C_2] \vee [D] \models [D].$$

Saturated Sets of $\mathcal{C}^\Sigma$.

A saturated set of $\mathcal{C}^\Sigma$ is defined as follows:

- For each clause c, for which there are $c_1, \ldots, c_n \in C^\Sigma$, such that $D^\Sigma(c_1, \ldots, c_n, c)$ by rule CONTEXT, there are clauses $d_1, \ldots, d_m \in C^\Sigma$, such that
$$R^\Sigma(d_1, \ldots, d_m, c).$$
- For each clause c, for which there are $c_1, \ldots, c_n \in C^\Sigma$, such that $D^\Sigma(c_1, \ldots, c_n, c)$ by rule RESTORE, there are clauses $d_1, \ldots, d_m \in C$, that do not contain negative Σ-literals, such that
$$R^\Sigma(d_1, \ldots, d_m, c).$$

It is necessary to restrict R^{Σ} in the definition of a saturated set, because the full combination of D^{Σ} and R^{Σ} would cause incompleteness.

Theorem: If $\mathcal{C}^{\Sigma}$ has a saturation that does not contain the empty clause, then $\mathcal{C}$ has a saturation not containing the empty clause.

Proof: By a fairly standard model construction.

Conclusions & Future Work.

We gave a way of simulating the splitting rule by introduction of fresh literals, which preserves the full power of subsumption/simplification, and which can be better handled by usual resolution implementation techniques.

A couple of choices can be made, when implementing this calculus. I do not know what are the right choices.