Extending Semantic Resolution via Automated Model Building: applications

Ricardo Caferra
LIFIA-IMAG
46, Avenue Felix Viallet
38031 Grenoble Cedex FRANCE

Nicolas Peltier
L1F1A-IMAG
46, Avenue Felix Viallei
38031 Grenoble Cedex FRANCE

Ricardo.Caferra@imag.fr Nicolas.Peltier@imag.fr

Abstract

An extension of semantic resolution is proposed. It is also an extension of the set of support as it can be considered as a particular case of semantic resolution. It is proved sound and refutationally complete. The extension is based on our former method for model building. The approach uses constrained clauses (or c-clauses), i.e. couples *[clause : constraint].* Two important new features are introduced with respect to semantic resolution. Firstly, the method builds its own (finite or infinite) models to guide the search or to stop it if the initial set of clauses is satisfiable. Secondly, instead of evaluating a clause in an interpretation it *imposes conditions* (coded in its rules) to force a c-clause not to be evaluated to true in the interpretation it builds.

The extension is limited in this paper to binary resolution but generalizing it to nary-resolution should be straightforward. The prover implementing our method is an extension of OTTER and compares advantageously with it (if binary resolution is used in OTTER), both in the search space and in the proof lenght of a relevant non trivial example.

Last but not least, model building is *incremental.* This feature is specially useful when dealing with large set of clauses.

1 Introduction

Since the very beginning of automated theorem proving, researchers tried to put some semantics in theorem provers in order to improve their power and efficiency, (see for example [GELERNTER *et a*/., 1983]). This was a very natural idea: usually, human beings are strongly guided in their proofs by some interpretation of the conjecture they are trying to proof or to refute. But a problem immediately appeared: "how to *formalize* this idea ?" The first, fully automated formal approach taking semantics into account was the set of support strategy [WOS *et a*/., 1965] who was born almost simultaneously with resolution, and who is still considered as the most powerful restriction strategy available ([WOS, 1988], page 52). In 1967, Slagle [SLAGLE, 1967] proposed a very elegant and general restriction strategy for resolution, called *semantic resolution,* in which the application of the resolution rule is controlled using ad hoc ground models given by the user. Slagle was aware of the limits of his method:

"A disadvantage of a semantic strategy is that the program must at present be given a model along with the theorem to be proved. One might hope that someday the progreim could devise its own models" ([SLAGLE, 1967], page 695).

J. Slaney's SCOTT theorem prover [SLANEY, 1993] is an answer to this wish. SCOTT produces striking results using semantic resolution.

Slagle also emphasized the usefulness of models in improving the set of support strategy:

" With set of support resolution, resolvents are generally not maximal and the underlying model is unknown. If the models were known many resolutions and resolvents could be eliminated" ([SLAGLE, 1967], page 695).

The method we propose here overcomes the two drawbacks pointed out by Slagle (for semantic resolution and set of support). More precisely, we propose an extension of semantic resolution based on our former method for simultaneous search for refutation and model ([BOURELY *et a*/., 1994; CAFERRA and ZABEL, 1992]). It strongly relies on the use of constrained clauses, i.e. couples \clause : *constraint s*\, instead of standard clauses. The constraints allow in some sense to code meta-reasoning: they are logical formulas imposing some conditions concerning the inference rules.

Our approach has at least three important consequences: first it prunes the search space for refutation in a similar — but *stronger* — way than semantic resolution does. Second it allows to perform an *incremental* model construction. If a model of a formula *I1* is known, and if we want to get a model of *E/\F',* then our method computes the new model by *extending* the previous one, instead of re-computing it entirely. Last, but not least, it gives to all level of the support strategy a power similar to that of level 1.

The paper is divided into 5 sections. Section 2 recalls briefly the basic ideas of our method for model building and some of its key rules. In section 3 we propose an extension of semantic resolution and we prove its properties: soundness and refutational completeness. Section 4

is devoted to incremental model building: we state conditions assuring that a model can be automatically built. One running example illustrates this construction.

Section 5 shows, via an analysis of the search space of the halting problem proof how our method prunes the search space of OTTER (if binary resolution is used) and produces a shorter proof.

2 Simultaneous Search for refutations and models

With the aim of increasing the possibilities of theorem provers as (hopefully intelligent) research assistants, we have developed few years ago a (refutationally complete) method combining search for refutations and models for first order formulas. The method captures the "normal" attitude of a human being faced to a conjecture: he considers simultaneously two possibilities: to *prove* and to *disprove* the conjecture.

This method - called RAMC1 - uses *constrained clauses* (or c-clauses), i.e. couples *{clause* : *constraint].* Constraints code the conditions necessary either to the application or the impossibility of application of the inference rules and denote the range of the variables of the clauses. It is neither possible nor useful to paste here large parts of results published elsewhere. Instead we recall some inference and disinference rules used in the examples that give also a taste of the method (for technical details see [CAFERRA and ZABEL, 1992] and [BOUREL Y *etal,* 1994]).

- The rule of *binary c-resolution* (abbreviated *bc-resolution)* on c_1 and c_2 upon $l^c(\bar{t})$ and $l(\bar{s})$ defined as follows:

$$\frac{[\![l^c(\bar{t}) \vee c'_1 \ : \mathcal{X}]\!] \qquad [\![l(\bar{s}) \vee c'_2 \ : \mathcal{Y}]\!]}{[\![c'_1 \vee c'_2 \ : \mathcal{X} \wedge \mathcal{Y} \wedge \bar{t} = \bar{s}]\!]}$$

- The rule of *distautology generation* imposes constraints on a c-clause in order to prevent it from being a tautology.

$$\frac{[\![l(\bar{t}) \vee l^c(\bar{s}) \vee c' \ : \mathcal{X}]\!]}{[\![l(t) \vee l^c(\bar{s}) \vee c' \ : \mathcal{X} \wedge \bar{s} \neq \bar{t}]\!]}$$

- The *unit bc-dissubsumption rule* imposes constraints preventing a c-clause from being subsumed by a unit c-clause. It allows the elimination of the c-clauses that are logical consequences of other c-clauses of *S.* It is defined as follows (where $\bar{x} = var(\mathcal{X}) \cup var(l(\bar{s}))$)

$$\frac{[\![l(\bar{t}) \vee c' \ : \mathcal{Y}]\!] \qquad [\![l(\bar{s}) \ : \mathcal{X}]\!]}{[\![l(t) \vee c' \ : \mathcal{Y} \wedge \forall \bar{x}.[\neg \mathcal{X} \vee \bar{s} \neq \bar{t}]]\!]}$$

The *binary-c-disresolution* rule is a natural extension of the unit disresolution, defined in [CAFERRA and ZABEL, 1992].

$$\frac{[\![P(\bar{t1}) \vee a \ : \mathcal{X}]\!] \qquad [\![\neg P(\bar{t2}) \vee b \ : \mathcal{Y}]\!]}{[\![P(\bar{t1}) \vee a \ : \mathcal{X} \wedge (\forall \bar{x}.\neg \mathcal{Y} \vee \bar{t1} \neq \bar{t2})]\!] \quad [\![P(\bar{t1}) \vee a \ : \mathcal{X} \wedge (\exists \bar{x}.\mathcal{Y} \wedge \bar{t1} = \bar{t2})]\!]}$$

where $\bar{x} = var(\bar{t2}) \cup var(\mathcal{Y})$

'standing for Refutation And Model Construction

- The *GPL-rule,* ("Generating Pure Literal" rule). Let *S* be a finite set of c-clauses. Let *c* be a c-clause in *S* and / be a literal in *c.* The GPL rule computes constraints for *c* in order to prevent application of bc-resolution upon / and *lc* (i.e. the complementary of /) between the c-clause *c* and any of the c-clauses in *S.*

Formally, let *S* be a set of c-clauses and *c* : $[\![l(\bar{t}) \vee c' \ : \mathcal{X}]\!]$ be a c-clause in 5. The *GPL-rule* is defined as follows:

$$\frac{[\![l(\bar{t}) \vee c' \ : \mathcal{X}]\!] \qquad S}{[\![l(\bar{t}) \ : \mathcal{X}_{pure}]\!]}$$

where $\mathcal{X}_{pure} = \bigwedge\{\forall \bar{y}.[\neg \mathcal{Y} \vee \bar{s} \neq \bar{t}] : [\![k : \mathcal{Y}]\!] \in S \text{ and } l^c(\bar{s}) \in k\} \wedge \mathcal{X}$ where $\bar{y}$ are the variables in $var(\mathcal{Y}) \cup var(k)$.

In [BOURELY *et* a/., 1994] the GPL rule is extended in order to deal with self-resolvent clauses, (it is easy to see that the result of the previous rule applied on self-resolvent clauses, is in most cases useless).

The reader can easily imagine the meaning of other rules of the method (based exactly on the same idea).

Our approach is not limited to Herbrand models: in [BOURELY *et* a/., 1994], the method is extended to sets of equational clauses. We defined the c-*paramodulation* and c-*dis-paramodulation* rules (similar to the c-resolution and c-disresolution rule) and we give two termination criteria identifying classes of satisfiable equational unit clauses for which a model can be surely built. They allow to compute the domain *D* (a subset of the Herbrand universe) and to interpret function symbols on the domain.

If RAMC stops, and if the set of c-clauses that we obtain contains only unit c-clauses, it gives a model of the initial formula. The partial interpretation of the predicates is given by the unit c-clauses produced by the method: each n-ary predicate *P* is mapped to two subsets $\mathcal{I}(P)^+$ and $\mathcal{I}(P)^-$ of D^n corresponding to the sets of n-tuples of ground terms for which *P* is respectively evaluated to True and to False. The interpretation is *partial* b e $\mathcal{I}(P)^+ \cup \mathcal{I}(P)^- \subseteq D^n$. If $\mathcal{I}(P)^+ \cup \mathcal{I}(P)^- \quad Dn$, we have a total interpretation of predicate P^2. Obviously total interpretations are particular cases of partial interpretations. The sets $\mathcal{I}(P)^+$ and $\mathcal{I}(P)^-$ are presently expressed by equational problems in an equational theory. Equational problems are a decidable class of logical formulas quantified in a particular way and using only =,not= , ^, A, V. We shall soon incorporate in the method constraints with more expressive power [PELTIER, 1995]. Interpretations (models) expressible by equational problems are called equational (or eq-) interpretations (models).

3 Extending semantic resolution

The principle of semantic resolution [SLAGLE, 1967] is the following: it uses an interpretation *1 (given by the user)* and prevents the application of resolution between two c-clauses that are true in J. This restriction is refutationally complete. The main idea of the method we propose is a very simple one: since we are able to build

2 Notice that not to be false in a partial model in not equivalent to be true in it

models of sets of clauses, why not use them in order to guide research in similar way as semantic resolution (or the set of support strategy) does ?

Immediately, we can go further: instead of evaluating clauses in a (in general partial) model, why not impose conditions — coded in the constraints — forcing a clause to be evaluated to false or not to be evaluated to false in the model ?

These simple ideas have important consequences: search space can be restricted with respect to that of semantic resolution and it gives to the method at all levels a power similar to that of set of support at level 1[3]. The model building capabilities of our approach are also improved, with respect to our previous work: we will show that the use of an interpretation *1* that is a model of some of the c-clauses makes the model construction for the whole set easier.

3.1 Semantic c-rules

We translate these ideas into the inference (and disinference) rules of our method. Let *1* an eq-interpretation, represented by a finite set of unit c-clauses:

$$\mathcal{I} = \{[\![L_1(s_1) \ : \mathcal{X}_1]\!], \ldots, [\![L_n(s_n) \ : \mathcal{X}_n]\!]\}$$

The variables of each c-clause $[\![L_i(s_i) \ : \mathcal{X}_i]\!]$ are denoted by y_1.

If $C : [\![\bigvee P_i(\overline{t_i}) \ : \mathcal{X}]\!]$ then we note $\mathcal{P}_{\mathcal{I}}(C)$ the following equational problem:

$$\mathcal{P}_{\mathcal{I}}(C) = \bigvee_{P_j = L_i} (\exists \overline{y_i}. \mathcal{X}_i \wedge s_i = t_j)$$

It is easy to verify:

$$\mathcal{X} \wedge \mathcal{P}_{\mathcal{I}}(C) \equiv \top \Rightarrow \mathcal{I} \models C$$

(i.e. there is a literal in *C* evaluated to true in 2).

We introduce a new inference rule and a new disinference rule.

binary I-c-resolution :

$$\frac{c_1 : [\![L(\overline{t_1}) \vee R_1 \ : \mathcal{X}]\!] \qquad c_2 : [\![L^c(\overline{t_2}) \vee R_2 \ : \mathcal{Y}]\!]}{r_1 : [\![R_1 \vee R_2 \ : \mathcal{X} \wedge \mathcal{Y} \wedge \overline{t_1} = \overline{t_2} \wedge (\neg \mathcal{P}_{\mathcal{I}}(c_1) \vee \neg \mathcal{P}_{\mathcal{I}}(c_2))]\!]}$$

r_1 is a *$\mathcal{I}$-c-resolvent* of c_1 and c_2.

Remark: $\neg \mathcal{P}_{\mathcal{I}}(c_1) \vee \neg \mathcal{P}_{\mathcal{I}}(c_2)$ ensures that at least one of the c_1, c_2 will not be evaluated to **true** in $\mathcal{I}$.

binary $\mathcal{I}$-c-disresolution :

$$\frac{c_1 : [\![L(\overline{t_1}) \vee R_1 \ : \mathcal{X}]\!] \qquad c_2 : [\![L^c(\overline{t_2}) \vee R_2 \ : \mathcal{Y}]\!]}{\begin{array}{l} r_1 : [\![L(\overline{t_1}) \vee R_1 \ : \mathcal{X} \wedge \mathcal{Y} \wedge \overline{t1} = \overline{t2} \wedge (\neg \mathcal{P}_{\mathcal{I}}(c_1) \vee \neg \mathcal{P}_{\mathcal{I}}(c_2))]\!] \\ r_2 : [\![L^c(\overline{t_2}) \vee R_2 \ : \mathcal{X} \wedge (\forall \overline{y}. \neg \mathcal{Y} \vee \overline{t1} \neq \overline{t2} \vee (\mathcal{P}_{\mathcal{I}}(c_1) \wedge \mathcal{P}_{\mathcal{I}}(c_2)))]\!] \end{array}}$$

where $\overline{y}$ denote the variables of c_2.

r_1 and r_2 are two *$\mathcal{I}$-c-disresolvents* of c_1 and c_2.

[3] This corresponds to Wos's wish in the first of his 33 basic research problem [WOS, 1988]

We shall call RAMCS the method RAMC augmented by the two rules above.

Remark: RAMC is a particular case of RAMCS, with $\mathcal{I} = \emptyset$ (i.e. $\mathcal{I}(P)^+ = \mathcal{I}(P)^- = \emptyset$, **for all** P).

Notice that the use of equational problems allows to restrict the application of the resolution rule in a stronger way than semantic resolution, as defined by [SLAGLE, 1967] does, as showed by considering the following set of clauses:

$$c_1 : P(x) \vee S(x) \qquad c_2 : \neg P(x) \qquad c_3 : \neg S(a) \vee R(x)$$

and the Her brand interpretation:

$$I = \{S(a)\}$$

The following is a valid semantic derivation, according to Slagle's definition.

$C_4 : S(x)$ (resolution between c_1 and C_2)
$C_5 : R(x)$ (resolution between C_4 and C_3)

While RAMCS gives the following c-clauses:

$$c_4 : [\![S(x) \ : x \neq a]\!] \ (\mathcal{I}\text{-c-res}, c_1, c_2)$$
$$c_5 : [\![R(x) \ : x \neq a \wedge x = a]\!] \ (\mathcal{I}\text{-c-res}, c_4, c_3)$$

The constraints of C_5 have no solution and thus C_5 *can be removed* from the set of c-clauses.

More precisely, it is possible to show that all the c-refutations generated by our method do correspond to a ground semantic refutation (see theorem 1 below). This is not the case with semantic resolution: in previous example, the first derivation did not correspond to any ground semantic derivation (for more details about this problem, see for example [McCUNE and HENSCHEN, 1985]).

In [STANDFORD, 1980], a method was proposed in order to avoid inferences that do not correspond to any ground inference: it attaches to each clause a list of literals whose descendants must be falsified (it was called the *FSL list)*. This method appears to be not efficient [McCUNE and HENSCHEN, 1985].

"The FSL method was implemented but during preliminary experiments we found that it required too much processing time. During deep searches the FSL lists became very long and the improvement was small or nonexistent." ([McCUNE and HENSCHEN, 1985], page 256)

Our approach seems to be more adapted to solve-this problem in a more natural and efficient way.

In theorems and lemmas below "interpretation" means peq-interpretation.

Theorem 1 *Let 1 be an interpretation, S a set of c-clauses and d be a derivation from S using the I-c-resolution rule. Let d' be a corresponding ground derivation (i.e. $d' = \sigma_{gr}(d)$). Then d' is a ground semantic derivation from ground instances of c-clauses in S.*

Proof 1 *By induction on the length n of d:*

- *If $n = 0$, the theorem obviously holds*
- *Assume that the theorem holds for n. Let d be a derivation of length $n+1$. The first step of d is of the form $S \rightarrow S \cup \{R\}$, where R is of the form:*

$$[\![R_1 \vee R_2 \ : \mathcal{X} \wedge \mathcal{Y} \wedge \overline{l_1} = \overline{l_2} \wedge (\neg \mathcal{P}_{\mathcal{I}}(c_1) \vee \neg \mathcal{P}_{\mathcal{I}}(c_2))]\!]$$

and there exists $c_1, c_2 \in S$ such that:

$$c_1 = [\![L(\overline{l_1}) \vee R_1 \ : \mathcal{X}]\!]$$
$$c_2 = [\![L^c(\overline{l_2}) \vee R_2 \ : \mathcal{Y}]\!]$$

The first step of d' is of the form:

$$S \rightarrow S \cup \{\sigma(R)\}$$

*where $\sigma \in Sol(\mathcal{X} \wedge \mathcal{Y} \wedge \overline{l_1} = \overline{l_2} \wedge (\neg \mathcal{P}_{\mathcal{I}}(c_1) \vee \neg \mathcal{P}_{\mathcal{I}}(c_2)))$. $\sigma \notin Sol(\mathcal{P}_{\mathcal{I}}(c_1))$ or $\sigma \notin Sol(\mathcal{P}_{\mathcal{I}}(c_2))$. hence: $\mathcal{I} \not\models \sigma(L(\overline{l_1}) \vee R_1)$ or $\mathcal{I} \not\models \sigma(\neg L^c(\overline{l_2}) \vee R_2)$. At least one of the two parent clauses is **false** in $\mathcal{I}$. The first step of d' is therefore a ground semantic inference. Using induction hypothesis, we deduce that d' is a ground semantic derivation.*

3.2 Soundness and Refutational Completeness of the method

Theorem 2 (Soundness) *Let c_1 and c_2 be two c-clauses, let $\mathcal{I}$ be a interpretation, r a binary $\mathcal{I}$-c-resolvent of c_1 and c_2, r_1 and r_2 the binary $\mathcal{I}$-c-disresolvents of c_1 and c_2. Then:*

- *r is a logical consequence of r_1 and r_2.*
- *$c_1 \wedge c_2 \equiv r_1 \wedge r_2 \wedge c_2$ i.e. r_1 can be deleted from the set of c-clauses and replaced by r_1 and r_2.*

Proof 2 *The proof is similar to those of lemmas 3.1 and 3.6 in [CAFERRA and ZABEL, 1992].*

The proof of refutational completeness needs the following lemma:

Lemma 1 *Let $\mathcal{I}$ be a partial interpretation $\mathcal{I}$. Then there exists a total interpretation $\mathcal{I}'$ such that, for all ground clause C:*

$$\mathcal{I} \models C \Rightarrow \mathcal{I}' \models C$$

Proof 3 *Let $\mathcal{I}$ be a partial interpretation. Let $\mathcal{I}'$ be the interpretation defined by, for all predicate P (D is the domain of $\mathcal{I}$):*

$$\mathcal{I}(P)^+ = \mathcal{I}(P)^+$$
$$\mathcal{I}(P)^- = D^n \setminus \mathcal{I}(P)^+$$

Let $L(\overline{t})$ a ground literal. If L is positive we have: $\mathcal{I} \models L(\overline{t})$ iff $\overline{t} \in \mathcal{I}(P)^+$ i.e. $\overline{t} \in \mathcal{I}'(P)^+$, hence $\mathcal{I}' \models L(\overline{t})$. If L is negative, $\mathcal{I} \models L(\overline{t})$ implies $\mathcal{I} \not\models L^c(\overline{t})$, hence $\mathcal{I}' \not\models L^c(\overline{t})$ (since L^c is positive) and $\mathcal{I}' \models L(\overline{t})$ (since $\mathcal{I}'$ is total).

Then for each ground clause $C = \bigvee_{i=1}^{n} L_i(\overline{t_i})$, we have:

$\mathcal{I} \models C$ iff:

- *$\exists i. \mathcal{I} \models L_i(\overline{t_i})$, then $\mathcal{I}' \models L_i(\overline{t_i})$ and therefore $\mathcal{I}' \models C$*
- *or $\exists i, j. L_i^c(\overline{t_i}) = L_j(\overline{t_j})$, then $\mathcal{I}' \models C$ (C is a tautology).*

Lemma 2 (Lifting lemma)
Let $[\![C_1 \ : \mathcal{X}_1]\!]$ and $[\![C_2 \ : \mathcal{X}_2]\!]$ be two c-clauses, σ_1 and σ_2 solutions of $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively, $\mathcal{I}$ a interpretation, $\mathcal{I}'$ the total interpretation as in lemma 1, and R_g a $\mathcal{I}'$-resolvent of $\sigma_1(C_1)$ and $\sigma_2(C_2)$. Then there exists a $\mathcal{I}$-c-resolvent $[\![C \ : \mathcal{Y}]\!]$ of $[\![C_1 \ : \mathcal{X}_1]\!]$ and $[\![C_2 \ : \mathcal{X}_2]\!]$ and a solution σ of $\mathcal{Y}$ such that $\sigma(C) = R_g$.

Proof 4 *Let $c_1 : [\![L(\overline{t_1}) \vee R_1 \ : \mathcal{X}_1]\!]$ and $c_2 : [\![L^c(\overline{t_2}) \vee R_2 \ : \mathcal{X}_2]\!]$ be two c-clauses, $\sigma_1 \in Sol(\mathcal{X}_1)$, $\sigma_2 \in Sol(\mathcal{X}_2)$, $\mathcal{I}$ a interpretation.*

Let R_g a $\mathcal{I}$-resolvent of $\sigma_1(L(\overline{t_1}) \vee R_1)$ and $\sigma_2(L^c(\overline{t_2}) \vee R_2)$.

Let $R : [\![C \ : \mathcal{Y}]\!]$ be the $\mathcal{I}'$-c-resolvent of c_1 and c_2. Let $\sigma = \sigma_1 \cup \sigma_2$. We have $\sigma \in Sol(\mathcal{X}_1 \wedge \mathcal{X}_2)$. Moreover $\sigma_1(t_1) = \sigma_2(t_2)$, hence $\sigma \in Sol(\mathcal{X}_1 \wedge \mathcal{X}_2 \wedge t_1 = t_2)$.

R_g is a $\mathcal{I}'$-resolvent of $\sigma_1(L(\overline{t_1}) \vee R_1)$ and $\sigma_2(L^c(\overline{t_2}) \vee R_2)$, therefore we have either $\mathcal{I}' \not\models \sigma_1(L(\overline{t_1}) \vee R_1)$ or $\mathcal{I}' \not\models \sigma_2(L^c(\overline{t_2}) \vee R_2)$ By lemma 1 we have: $\mathcal{I} \not\models \sigma_1(L(\overline{t_1}) \vee R_1)$ or $\mathcal{I} \not\models \sigma_2(L^c(\overline{t_2}) \vee R_2)$ i.e. $\sigma_1 \in Sol(P_{\mathcal{I}}(c_1))$ or $\sigma_2 \in Sol(P_{\mathcal{I}}c_2)$.

Hence $\sigma \in Sol(\mathcal{Y})$.

Theorem 3 (Refutational Completeness)
For all interpretation $\mathcal{I}$ and for all unsatisfiable set of c-clause S there exists a refutation of S using only $\mathcal{I}$-c-rules.

Proof 5 *By refutational completeness of semantic resolution [SLAGLE, 1967] there exists a ground $\mathcal{I}'$-refutation of S. By lemma 2 it can be lifted to non ground c-clauses, to obtain a $\mathcal{I}$-c-refutation of S.*

4 Incremental Model building

One of the most interesting features of our method is that it allows building models *incrementally*. This characteristic is particularly interesting when dealing with large sets of clauses (databases,...). The following theorem gives conditions allowing to extract a model from a set of c-clauses obtained when RAMCS stops. This conditions are similar — but *weaker* — than those for RAMC (the former conditions imposed that all c-clauses had to be unit c-clauses). Therefore RAMCS is *stronger* than RAMC. Indeed, we take advantage of the interpretation $\mathcal{I}$ that is already a model of a some of the c-clauses, in order to compute the whole model as an *extension* of $\mathcal{I}$.

Theorem 4 *If S is a set of c-clause stable by all the rules of RAMCS, and if all non unit c-clauses of S are **true** in $\mathcal{I}$, then a model of S can be automatically computed.*

Proof 6 *Let S be a stable set of c-clause, such that all non unit c-clause of S is **true** in S.*

The interpretation $\mathcal{I}'$ is defined as follows:
For all ground term $\overline{t}$:

*$\mathcal{I}'(L(\overline{t})) =$ **true***
if $\exists [\![L(\overline{s}) \ : \mathcal{X}]\!] \in S$ such that:
$\overline{t} = \overline{s} \wedge \mathcal{X} \not\equiv \bot$
$\mathcal{I}'(L(\overline{t})) = \mathcal{I}(L(\overline{t}))$, elsewhere

We prove that $\mathcal{I}' \models S$, by case analysis on the length of the c-clauses.

For all c-clause $c : [\![\bigvee_{i=1}^{n} L_i(\overline{t_i}) \ : \mathcal{X}]\!] \in S$ we have:

- *if $n = 1$ (C is a unit c-clause), then $\forall \sigma \in Sol(\mathcal{X}).\mathcal{I}' \models L_i(\sigma(\overline{t_i}))$ (by definition of $\mathcal{I}'$).*
- *if $n > 1$, then C is true in $\mathcal{I}'$. Let $\sigma \in Sol(\mathcal{X})$. Then, since C is stable by distautology, there exists i such that $\sigma(L_i(\overline{t_i}))$ is true in $\mathcal{I}$. Since S is stable by unit-c-$\mathcal{I}$-disresolution, for any unit c-clause $[\![L_i(\overline{s}) \ : \mathcal{Y}]\!]$, we have: $\mathcal{Y} \wedge \mathcal{X} \wedge \overline{s} = \overline{t_i} \equiv \bot$. Finally, by definition of $\mathcal{I}'$, we have: $\mathcal{I}' \models \sigma(L_i(\overline{t_i}))$, and*

$$\mathcal{I}' \models C$$

Remark: RAMCS allows therefore to perform an *incremental* model construction: the model $\mathcal{I}'$ in the proof above is built by *modifying* and *extending* the previous one. As already pointed out, this feature is especially useful when the set of c-clauses S is very large (for example in databases), and when the adding of the new c-clauses does not modify radically the intended meaning of S.

Example

The following example, taken from [FERMÜLLER *et al.*, 1993], illustrates the model building aspect of RAMCS.

```
1 [] [[ P(x) | Q(g(x,x)) : TRUE ]].
2 [] [[ -Q(x) | R(y,x) : TRUE ]].
3 [] [[ -R(a,a) | -R(f(b),a) : TRUE ]].
4 [] [[ R(f(x),y) : TRUE ]].
5 [] [[ P(x) | -P(f(x)) : TRUE ]].
6 [] [[ -P(a) : TRUE ]].
```

RAMCS builds automatically the following Herbrand model.

```
4 [] [[ R(f(x),y) : TRUE ]].
9 [gpl] [[ Q(g(x,x)) : TRUE ]].
12 [gpl] [[ -R(a,a) : TRUE ]].
10 [gpl] [[ -Q(a) : TRUE ]].
11 [gpl] [[ -P(f(x)) : TRUE ]].
6 [] [[ -P(a) : TRUE ]].
13 [gpl] [[ R(x,y) : ((a != y) | (a != x)) ]].
```

Then suppose we add the following c-clauses:

```
7 [] [[ -S(x) | S(g(x,x)) : TRUE ]].
8 [] [[ S(a) | S(f(x)) : TRUE ]].
9 [] [[ -S(x) | -R(b,g(x,x)) : TRUE ]].
```

RAMCS gives (with the previous model).

```
10 [resolution,9,7] [[ -S(x) | -R(b,g(x,x)) : TRUE ]].
11 [resolution,10,7] [[ -S(x) | -R(b,g(g(x,x),g(x,x)))
: TRUE ]].
12 [resolution,11,2] [[ -S(x) | -Q(g(g(x,x),g(x,x)))
: TRUE ]].
13 [resolution,12,1] [[ -S(x) | P(g(x,x)) : TRUE ]].
14 [gpl] [[ P(g(x,x)) : TRUE ]].
clause #13 is deleted by dissubsumption 14,13.
15 [dissubsumption,14,1] [[ P(x) | Q(g(x,x))
: (x != g(y,y)) ]].
16 [gpl] [[ -Q(g(g(x,x),g(x,x))) : TRUE ]].
clause #12 is deleted by dissubsumption 16,12.
17 [dissubsumption,16,2] [[ -Q(x) | R(y,x)
: (x != (g(g(z,z),g(z,z)))) ]].
18 [gpl] [[ -R(b,g(g(x,x),g(x,x))) : TRUE ]].
clause #11 is deleted by dissubsumption 18,11.
19 [dissubsumption,18,10] [[ -S(x) | -R(b,g(x,x))
: (x != g(y,y)) ]].
20 [gpl] [[ S(g(x,x)) : TRUE ]].
21 [gpl] [[ S(a) : TRUE ]].
```

Here we get a set of c-clauses satisfying the conditions of section 4, and the model is:

```
% modified previous literals
[[ R(f(x),y) : TRUE ]].
[[ Q(g(x,x)) : (x != g(y,y)) ]].
[[ -R(a,a) : TRUE ]].
[[ -Q(a) : TRUE ]].
[[ -P(f(x)) : TRUE ]].
[[ -P(a) : TRUE ]].
[[ R(x,y) : (all z ((a != y) | (a != x))
& ((x != b) | (y != g(g(z,z),g(z,z))))) ]].

% new literals
14 [gpl] [[ P(g(x,x)) : TRUE ]].
16 [gpl] [[ -Q(g(g(x,x),g(x,x))) : TRUE ]].
18 [gpl] [[ -R(b,g(g(x,x),g(x,x))) : TRUE ]].
20 [gpl] [[ S(g(x,x)) : TRUE ]].
21 [gpl] [[ S(a) : TRUE ]].
```

Remark 1: for the sake of clarity, we let the constraints of `R(x,y)` in unsolved form (i.e. in non canonical one). its solved form would be (after elimination of parameter z):

```
[[ R(x,y) : (((a != y) | (a != x))
& ((x != b) | (y = a) | (y = b) | (y = f(z)) |
((y = g(u,v)) & ((u != v) | ((u = a)
| (u = b) | (u = f(v1)) | ((u = g(v1,v2))
& (v1 != v2)))))))).
```

Remark 2: the model built by RAMCS is *infinite*.

The next section is devoted to the detailed analysis of a non trivial example in which, in addition to the pruning of the search space allowed by RAMCS, we show a nice side effect of its application: the possibility of finding *shorter proofs*.

5 Pruning the search space and shortening proofs: an example

To prove that a set of c-clauses S implies a conclusion C, we use — according to the ideas introduced in section 3 — the following procedure:

1. Compute a partial model $\mathcal{M}$ of S.
2. Use $\mathcal{M}$-c-rules on $S \cup \neg C$.

The first step may of course not stop (first order logic is undecidable), hence we have to fix a limit on the search for $\mathcal{M}$ (for example an upper bound on the number of c-clauses generated or on the search time...), in order to keep refutational completeness. This is similar to what is done in SCOTT [SLANEY, 1993]. Nevertheless, even if we do not get a model of S, we will get a model of a subset of S, that can be use in step 2 (in the worst case $\mathcal{M}$ will be empty).

It should be remarked that the use of partial models allows a *finer* and more *dynamic* control of the application of resolution rule because it can be enriched as the search for refutation goes along.

A problem we consider related to the use of semantic notions in theorem proving is the one of finding shorter proofs. This problem became to be an important one in automated deduction. Obviously, an exhaustive, breadth-first search (ensuring that the shortest proof will be found) is in general impossible. Sometimes the use of semantic notions as in our method allows to find short proofs in a reasonable time.

The halting problem

This fundamental problem in Computer Science is an interesting challenge for automated theorem provers, especially for those using resolution (for one of the usual formulations in the automated deduction literature there is no known fully automated resolution proof [DAFA, 1994]).

The formulation we consider in this section is the one of M. Bruschi [BRUSCHI, 1991][4].

With this formulation, M. Bruschi obtained a 26 steps proof of the theorem, using ENprover, the DSI implementation of the Hsiang's EN-strategy method. He used a breadth first strategy with set of support.

RAMCS builds the following model $\mathcal{M}$ of the hypothesis (without the denial of the conclusion):

```
1 [] [[ -algorithm_program_decides(A) : TRUE ]].
2 [] [[ -program(A) : TRUE ]] .
3 [] [[ -program_program_decides(A) : TRUE ]].
4 [] [[ -program_halts(A,B) : TRUE ]].
5 [] [[ -algorithm(A) : TRUE ]].
6 [] [[ -program_halts_halts2_outputs(A,B,C,D)
: TRUE ]].
7 [] [[ program_decides(A) : TRUE ]].

8 [] [[ -halts2_outputs(A,B,C,D) : TRUE ]].
9 [] [[ -program_not_halts(A,B) : TRUE ]].
10 [] [[ -program_not_halts_halts2_outputs(A,B,C,D)
: TRUE ]].
11 [] [[ -program_not_halts_halts2_outputs(A,B,C,D)
: TRUE ]].
12 [] [[ -program_not_halts_halts_outputs(A,B,C)
: TRUE ]].
13 [] [[ -halts_outputs(A,B,C)  : TRUE ]].
14 [] [[ -program_halts_halts_outputs(A,B,C)
: TRUE ]] .
15 [] [[ -outputs(A,B)  : TRUE ]].
16 [] [[ -program_not_halts_halts_outputs(A,B,C)
: TRUE ]].
```

Once we have a model we can use RAMCS. The proof obtained by **RAMCS** using the model $\mathcal{M}$ is (input c-clauses are distinguished by the "[]" prefixing them):

```
11 [] [[ -program_halts(A,B) | halts(A,B) : TRUE ]].
17 [] [[ -program_not_halts(A,B) | -halts(A,B)
: TRUE ]].
22 [] [[ -program_halts_halts2_outputs(A,B,C,D) |
program_halts(B,C) : TRUE ]].
25 [] [[ -program_not_halts_halts2_outputs(A,B,C,D)
| program_not_halts(B,C) : TRUE ]].
34 [] [[ -algorithm_program_decides(A)
| program_program_decides(c1) : TRUE ]].
35 [] [[ -program_program_decides(A)
| program_halts_halts2_outputs(A,B,C,good)
: TRUE ]].
36 [] [[ -program_program_decides(A)
| program_not_halts_halts2_outputs(A,B,C,bad)
: TRUE ]].
43 [] [[ algorithm_program_decides(c4) : TRUE ]].
44 [resolution,43,34] [[ program_program_decides(c1)
: TRUE ]].
50 [resolution,44,36] [[ program_not_halts_halts2_
outputs(c1,A,B,bad) : TRUE ]].
52 [resolution,44,35] [[ program_halts_halts2_
outputs(c1,A,B,good) : TRUE ]].
79 [resolution,50,25] [[ program_not_halts(A,B)
: TRUE ]].
83 [resolution,52,22] [[ program_halts(A,B) : TRUE ]].
88 [resolution,83,11] [[ halts(A,B) : TRUE ]].
133 [resolution,79,17] [[ -halts(A,B) : TRUE ]].
134 [resolution,133,88] [[  : TRUE ]].
```

OTTER [McCUNE, 1990], using binary resolution and set of support strategy, produces a 12 steps proof, after generation of 90 clauses. The proof found by RAMCS is 8 steps long. The proof has therefore been shortened by 33%.

Discussion

Globally (i.e. for refutation and model building) our method generates 134 c-clauses, OTTER generates 90. In fact, when comparing the two methods, one should keep in mind than we apply *simultaneously* refutation and model building rules. So, for each clause generated by resolution, 2 or 3 c-clauses are generated by c-resolution and c-disresolution rules. If we consider only refutations rules RAMCS generates only about 50 c-clauses. The search space for refutation is therefore reduced by almost 50%. Computing time is roughly the same.

Of course, when looking only for refutation , the "model building rules" flag can be turned off and the search space is the refutation search space in the usual sense, which is reduced in this example by 45% with respect to that of OTTER.

Remark: the model building task performed by RAMCS can be consider as automated meta-reasoning done in the object level.

6 Conclusion and future work

We have extended semantic resolution and as side effect we have also improved the set of support strategy using a method for incremental automated construction of (finite or infinite) models.

Two points will be studied in the near future.

- The first one is a practical one, i.e. we shall extensively experiment our method on more complicated examples, particularly on open problems. For example, we shall look for a mechanical solution of the halting problem using the specification for which no fully automated resolution proof is known.
- We shall study the extension of our method to semantic paramodulation.

[4] taken from the TPTP library, problem COM003-2, see there the initial set of clauses

References

[BOURELY *et al,* 1994] Christophe BOURELY, Ricardo CAFERRA, and Nicolas PELTIER. A method for building models automatically. Experiments with an extension of OTTER. In *Proc. of CADE-12,* pages 72-86. Springer Verlag, 1994. LNA1 814.

[BRUSCHI, 1991] Massimo BRUSCHI. The halting problem. *AAR Newsletter,* 17, March 1991.

[CAFERRA and ZABEL, 1992] Ricardo CAFERRA and Nicolas ZABEL. A method for simultaneous search for refutations and models by equational constraint solving. *Journal of Symbolic Computation,* 13:613-641, 1992.

[DAFA, 1994] Li DAFA. The formulation of the halting problem is not suitable for describing the halting problem. *AAR Newsletter,* (27):I-7, 1994.

[FERMULLER *et al.,* 1993] C. FERMULLER, A. LEITSH, T. TAMMET, and N. ZAMOV. *Resolution Methods for the Decision Problem.* Lecture Notes in Artificial Intelligence. Springer-Verlag, 1993. LNAI 679.

[GELERNTER *et ai,* 1983] H.GELERNTER, J.R. HANSEN, and D.W. LOVELAND. Empirical explorations of the geometry theorem-proving machine. In Jorg Siekmann and Graham Wrightson, editors, *Automation of Reasoning, vol.* 7, pages 140-150. Springer-Verlag 1983, 1983. Originally published in 1960.

[McCUNE and HENSCHEN, 1985] W. McCUNE and L. HENSCHEN. Experiments with semantic paramodulation. *Journal of Automated Reasoning,* 1231 -261, 1985.

[McCUNE, 1990] WiUiam W. McCUNE. *OTTER 2.0 Users Guide.* Argonne National Laboratory, 1990.

[PELTIER, 1995] Nicolas PELTIER. Model building by constraint solving with terms with integer exponents. Submitted, 1995.

[SLAGLE, 1967] James R. SLAGLE. Automatic theorem proving with renamable and semantic resolution. *Journal of the ACM,* 14(4):687-697, October 1967.

[SLANEY, 1993] John SLANEY. Scott: A model-guided theorem prover. In *Proceedings of IJCAI93.* Morgan and Kauffman, 1993.

[STANDFORD, 1980] D. STANDFORD. *Using Sophisticated Models in Resolution Theorem Proving,* volume 90 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980. LNAI 679.

[WOS *et al,* 1966] L. WOS, G. ROBINSON, and D. CARSON. Efficiency and completeness of the set of support strategy in theorem proving. *Journal of the ACM,* 12:536-541, 1965.

[WOS, 1988] Larry WOS. *Automated Reasoning, 33 Basic Research Problems.* Prentice Hall, 1988.