Model building in the cross-roads of consequence and non-consequence relations

Ricardo Caferra and Nicolas Peltier*
LEIBNIZ - IMAG

Keywords: theorem proving and constraints, model building.

1 Introduction

From an informal (but abstract) point of view we consider reasoning as the activity of extracting information or completing partial information (of course information is taken here as a basic concept). We see automated reasoning as an activity combining deduction with verification and producing consequences and non-consequences in an automated/interactive way. Of course the respective extent of each of this components varies in different situations.

The present work is an application of this unified view to the particular case of model building.

The possibility of systematic model building in first-order logic exists at least since the introduction of the tableaux method (Hintikka, Beth, Smullyan,...), approximately 40 years ago. Some striking results in interactive model building have been obtained less than 20 years ago. But it is only since less than 10 years that results on model building are regularly published.

To the best of our knowledge, McCarthy was the first to point out that deducing and verifying are two extremes of the general activity of *deducing-verifying*. *Consequence relations* have been extensively studied. A particular case of consequence relations deserves to be mentioned here, i.e. *anticonsequence relations*. These relations study the propositions that can be rejected on the ground of other rejected propositions.

In order to give a unified treatment of the search for refutations and models we have investigated relations between accepted formulae and formulae that surely are **not** consequences of them. We naturally suggest to call them *non-consequence relations*. Given a consequence relation a corresponding non-consequence relation is any relation in its complement. Due to the undecidability of first-order logic all non-consequence relation defined by means of rules can be extended. The ideas underlying our approach to model building can be summarized as follows. Let us consider a conjecture in the form of a finite set of first-order clauses (the restriction to clauses is not necessary but convenient for presentation). Possible Herbrand models of the conjecture can be captured by finitely expressing sets of ground consequences (using inference rules - consequence relations) or by recognizing conditions that will avoid inference rules to be applicable. The latter allow

*46, avenue Félix Viallet, 38031 Grenoble Cedex, FRANCE, Ricardo.Caferra@imag.fr, Nicolas.Peltier@imag.fr

to identify literals that surely **cannot** be deduced from the conjecture (disinference rules-non-consequence relations). These non-deductible literals are immediately available, so to speak, to conceive a partial model of the conjecture.

The present work is exclusively on model building and disregard search for refutations. It introduces an extension of a non-consequence relation in the form of a verification process of a proposed (and automatically modifiable) information. A 'strategy' (we use this term loosely, but this is not important here) necessary to preserve satisfiability is also defined.

There are two main technical results in the paper: one stating that the proposed rule and strategy strictly increase the power of our former method and the other characterizing the class of models that can be captured by the new method. Its capabilities with respect to other approaches are briefly mentioned. The method has been implemented and allowed us to build models for some set of clauses for which, as far as we know, no other method works.

The former method and its limits

Since 1990, we develop a method for building models of first-order formulae [2]. It is based on the use of *equational constraints* that are used to *represent* and *build Herbrand models* of first-order formulae. This method captures the standard attitude of a human being faced to a conjecture: trying simultaneously to **prove** or to **disprove** (by giving a counter-example) it. The method is called RAMC (**R**efutation **A**nd **M**odel **C**onstruction). In contrast to other methods, RAMC is *not restricted to a particular class of formulae* and, similarly to Fermüller and Leitsch's [3], can built *infinite* models as well as finite ones.

In our method, models are naturally represented by equational formulae interpreted in the empty theory. A partial interpretation (i.e. an interpretation not necessarily defined everywhere) is said to be an *eq-interpretation* if it can be represented by an equational formula. More precisely, the interpretation of a n-ary predicate P (i.e. the set of n-tuplets of ground terms such that $P(t_1, \ldots, t_n)$ is true) is expressed by an equational formula $\mathcal{X}_P$ with n free variables $x_1, \ldots, x_n$. We call this kind of interpretations (models) *eq-interpretations* (*eq-models*). This formalism is more expressive than, for example, the one in [3]. Moreover we can build eq-models for interesting formulae of first order logic.

In the following, we will identify satisfiable sets of unit c-clauses and partial Herbrand eq-interpretations (it is easy to justify this assertion).

The method RAMC uses *constrained clauses* i.e. couples $[\![clause : constraint]\!]$. A constrained clause denotes the set of its ground instances. 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. Roughly speaking the method associates to each inference rule its so called "disinference" counterpart and introduces some essentially new rules. The method appeared to be very general and to have many interesting applications in Automated Deduction, Artificial Intelligence and Computer Science (for building models of equational clauses, for extending semantic strategies, deciding sub-classes of first-order logic, extending the power of logic programs interpreters and correcting programs ... (see for ex. [1]).

From the theoretical point of view the original method is basically limited for two reasons. The first one is the limit of the expressive power of eq-models. We have proposed elsewhere to extend it by using tree automata and term schematization techniques (see [5]).

The second limitation of the method is that the inference and disinference rules cannot generate all eq-interpretations. Let us call S the following set of c-clauses.

$\{P(x,y) \lor \neg P(succ(x), succ(y)),\ \ \neg P(x,y) \lor P(succ(x), succ(y)),\ \ \neg P(0, succ(x)),$ $\neg P(succ(x), 0),\ P(0,0)\}$

It is easy to see that a model of S is obtained by interpreting P as the equality on the Herbrand universe. S has only one Herbrand model $\mathcal{M}$: $P(x,y)$ is true iff $x = y$. $\mathcal{M}$ is a eq-interpretation, however *no finite representation of $\mathcal{M}$ can be generated by using inference or disinference rules.*

In this paper we overcome this limitation by using RAMC to **guide** (by generating consequences or *non-consequences*, by detecting counter-models, simplifying clauses, ...) the enumeration of the set of eq-interpretations. We prove that by combining RAMC with a new-rule (mb-splitting) any eq-interpretation can eventually be reached.

2 The new method

Informal presentation

The method we propose borrows the approach and dis-inference rules from our former one called RAMC. It can also be seen as having similarities with the well known Davis and Putnam's procedure. The way the method works is very simple.

- Firstly, it simplifies the set of c-clauses at hand S by using particular cases of RAMC's rules (see [2]). More precisely by using *Simplify* (see below).
- Secondly, the problem of finding a model of a set of c-clauses S is divided into two other independent subproblems: to find models of $S \cup \{P\}$ and $S \cup \{\neg P\}$.

  This is not sound in general if P is not ground (since there can exists a model of S in which some instances of P are true and other are false). Hence we must control the application of this rule in order to preserve the soundness of our method

Simplification rules

We only recall the RAMC rule the most deeply related to the new rule (for the others see [2]).

The GPL[1] rule

A literal L is said to be *pure* in a set of c-clauses S iff its complementary (i.e. $\neg L$) does not appear in clauses of S. Our method exploits in one of its key rules the notion of pure literal in a set of clauses. It is clear that a pure literal in a set of clauses is a model of the clause to which it belongs (it can be evaluated to $\top$ independently of the rest of the interpretation). Therefore a natural idea in order to build a model of a set of clauses S is to try to *generate* for each clause a pure literal in it. The way our method realizes this idea is by setting conditions (coded in the constraints), restricting the domain of variables in the arguments of a literal $P(\overline{t})$ in order to avoid unification with the arguments of all literals $\neg P(\overline{s})$ in clauses of S. This generated literal is added to S. Obviously, if L is pure in S then $S \cup L$ is satisfiable iff S is satisfiable.

$$\frac{[\![P(\overline{t}) \lor R\ : \mathcal{X}]\!] \qquad S}{[\![P(\overline{t})\ : \mathcal{X}_{pure}]\!]}$$

where $\mathcal{X}_{pure} = \bigwedge_{[\![\neg P(\overline{s}) \lor r\ : \mathcal{Y}]\!] \in S} (\forall \overline{y}. \neg \mathcal{Y} \lor \overline{s} \neq \overline{t}) \land \mathcal{X}$ and where $\overline{y}$ are all the variables in $var([\![\neg P(\overline{s}) \lor r\ : \mathcal{Y}]\!])$.

[1] Standing for **G**enerating **P**ure **L**iteral.

We call **Simplify** the procedure applying the set of rules $\{unitc - resolution, unitc - disresolution, unitc - dissubsumption, GPL\}$ [2].

The mb-splitting rule

As mentioned in the Introduction a non-consequence relation can be always extended. The idea on which is based the new rule is trivial: "a set of ground literals will be evaluated to true or to false in a model of a given set of clauses" The way the mb-splitting (standing for **m**odel **b**uilding splitting) rule does it is by *proposing* c-literals (i.e. sets of ground literals) as candidates for a possible model and *verifying* afterwards that they are effectively useful in building a model for the given set of clauses. The problems are of course:

- **When** this rule must be used in the model building process ?
- **Which ground terms** must instantiate the predicates ?
- If the guess is not a good one, how to **modify** it ?

A possible partial answer to these questions is given by the strategy below.

The class of models that the method is able to build can be increased by using such a splitting rule. We add a rule substituting the set of c-clauses at hand by 2 extended sets S_1 and S_2. S_1 and S_2 are obtained from S by adding respectively two unit c-clauses of the form $[\![P(\bar{t}) \; : \mathcal{X}]\!]$ and $[\![\neg P(\bar{t}) \; : \mathcal{X}]\!]$. The **mb-splitting** rule can be depicted as follows.

$$\frac{S}{S \cup \{[\![P(\bar{t}) \; : \mathcal{X}]\!]\} \qquad S \cup \{[\![\neg P(\bar{t}) \; : \mathcal{X}]\!]\}}$$

where $P(\bar{t})$ is a atomic formula.

Obviously, the **mb-splitting** rule does not preserve the satisfiability of the sets of c-clauses. Indeed if $[\![P(\bar{t}) \; : \mathcal{X}]\!]$ is not ground there may exist models $\mathcal{M}$ of S such that neither $[\![P(\bar{t}) \; : \mathcal{X}]\!]$ nor $[\![\neg P(\bar{t}) \; : \mathcal{X}]\!]$ is true in $\mathcal{M}$. In this case, S can be satisfiable and $S \cup \{[\![P(\bar{t}) \; : \mathcal{X}]\!]\}, S \cup \{[\![\neg P(\bar{t}) \; : \mathcal{X}]\!]\}$ unsatisfiable. Therefore the application of the rule must be carefully controlled. The strategy guides the choice of the literals on which the rule is applied.

Remark 1 *The mb-splitting and* GPL *rules are deeply related in their aims and effects and strongly different in the principles they are based on.* GPL*'s goal is to generate new literals in the model by* computing *the conditions ensuring that a literal is pure in a set of (constrained) clauses, hence that the adding of this literals will preserve the satisfiability of the whole set of c-clauses. The mb-splitting rule* asserts *that either a literal or its complementary will be evaluated to true in a model.* **The point is that what is asserted by mb-splitting is not necessarily computable by GPL (neither by the other rules of** RAMC**).** *This short description loosely explain why the splitting rule increases the model building capabilities of* RAMC*.*

The strategy

The proposed strategy is based on the following principle: choose a *finite partition* of the Herbrand base and use the **mb-splitting** rule for enumerating eq-interpretations that are *compatible* with this partition (i.e. sets of c-clauses in which each member contains each equivalence class denoted by a c-literal L, or the one denoted by $\neg L$). The simplification rules are used to detect counter-models and to prune the search space. If no model is found, then another partition is tried, finer than the initial one.

The following two rules are used by the strategy.

$\{ \llbracket P \ : \mathcal{X} \rrbracket \} \quad \rightarrow \quad \{\llbracket P \ : \mathcal{X} \wedge s = t \rrbracket, \llbracket P \ : \mathcal{X} \wedge s \neq t \rrbracket\}$

where s and t are two subterms in $\llbracket P \ : \mathcal{X} \rrbracket$

$\{ \llbracket P \ : \mathcal{X} \rrbracket \} \quad \rightarrow \quad \{\llbracket P \ : \mathcal{X} \wedge x = f(\overline{y}) \rrbracket / f \in \Sigma\}$

Where Σ is the signatures for function symbols and $x, y \in \mathcal{V}ar(\llbracket P \ : \mathcal{X} \rrbracket)$.

The following theorem states the main technical result of this paper. It proves that our method can build models for any formula having a model representable by equational formulae ($\mathbf{BuildMod}_1(S)$ is the name of the procedure implementing our method).

Theorem 1 *If* $\mathbf{BuildMod}_1(S)$ *returns* S' *then* S' *is (the representation of) a model of* S*. If* S *has a eq-model then any fair application of* $\mathbf{BuildMod}_1$ *returns an eq-model of* S*.*

Remark 2 *There exist several interesting classes of formulae with the property that any satisfiable set of c-clauses in these classes have an eq-model. This is the case for example for the classes* PVD*,* OCC1N*, the Bernay-Schönfinkel class and more generally any class decidable by hyperresolution (see [3]). Our method can build models for any satisfiable formula in these classes.*

3 Conclusion and future work

The use of mb-splitting appears to be critical in building models for difficult formulae, as for example a satisfiable formula used by Goldfarb in proving the undecidability of Gödel class with identity [4].

Two main technical problems are open by this work:

- Find criteria for "good" partitioning of the Herbrand universe in order to capture larger classes of models or to improve efficiency.
- Find criteria allowing to decide, faced to a conjecture, to switch RAMC from simultaneous search for refutations and models as originally to exclusive model building search as in the present work.

References

[1] R. Caferra and N. Peltier. A new technique for verifying and correcting logic programs. To appear in the Journal of Automated Reasoning, 1997.

[2] R. Caferra and N. Zabel. A method for simultaneous search for refutations and models by equational constraint solving. *Journal of Symbolic Computation*, 13:613–641, 1992.

[3] C. Fermüller and A. Leitsch. Hyperresolution and automated model building. *Journal of Logic and Computation*, 6(2):173–203, 1996.

[4] W. D. Goldfarb. The unsolvability of the Gödel class with identity. *Journal of Symbolic Logic*, 49(4):1237–1252, 1984.

[5] N. Peltier. Increasing Model Building Capabilities by Constraint Solving on Terms with Integer Exponents. *Journal of Symbolic Computation*, 24:59–101, 1997.