Combining Enumeration and Deductive Techniques in order to Increase the Class of Constructible Infinite Models

RICARDO CAFERRA† AND NICOLAS PELTIER‡

46, avenue Félix Viallet, 38031 Grenoble Cedex, France

A new method for building infinite models for first-order formulae is presented. The method combines enumeration techniques with existing deductive (in a broad sense) ones. Its soundness and completeness w.r.t. the class of models that can be represented by equational constraints are proven. This shows that the use of enumeration techniques strictly increases the power of existing methods for building Herbrand models that are not complete in this sense. Some strategies are proposed to reduce the search space. We give examples and show how to use this approach for building interactively a model of a formula introduced by Goldfarb in his proof of the undecidability of the Gödel class with identity. This formula is satisfiable but has no finite model.

© 2000 Academic Press

1. Introduction

The possibility of systematic model building in first-order logic has existed at least since the introduction of the tableaux method (see, for example, Smullyan, 1968; Fitting, 1990). Some striking results in interactive model building have been obtained less than 20 years ago, using general-purpose theorem provers (Winker, 1982). However, it is only less than 10 years ago that results on automated model building have started to be regularly published. Various kinds of methods have been proposed (see, for example, Caferra and Zabel, 1992; Fermüller et al., 1993; Slaney, 1993; Bourely et al., 1994; Zhang and Zhang, 1995; Fermüller and Leitsch, 1996) for representing (possibly infinite) interpretations and for building automatically models of first-order formulae. Of course, none of them can be complete w.r.t. full first-order logic. The existing model building methods proposed so far can be divided into three distinct categories.

— Enumeration methods (Slaney, 1993; McCune, 1994; Slaney, 1994; Zhang and Zhang, 1995; Peltier, 1997b; Zhang, 1997; Peltier, 1998b). The principle of these methods is a very simple one: enumerate a class of interpretations (usually interpretations defined on a finite domain) and check whether these interpretations validate the considered formula. These kinds of approaches are very costly and powerful heuristics have to prune the search space, for making them practical. The main interest of these methods is that they are, by definition, complete for a given class of interpretations, which is usually not the case for other approaches.

†E-mail: Ricardo.Caferra@imag.fr
‡E-mail: Nicolas.Peltier@imag.fr

0747–7171/00/020177 + 35 $35.00/0 © 2000 Academic Press
Saturation methods (Tammet, 1991; Fermüller et al., 1993; Fermüller and Leitsch, 1996). These methods use standard proof procedures (tableaux, resolution...) or a refinement of them. In these methods, model building occurs as a post-processing step after the proof procedure terminates and concludes that the proposed formula is satisfiable, because no new consequences can be drawn. Due to this basic restriction, saturation methods are intrinsically applicable only to some specific decidable classes of formulae. For tableaux-based methods, or refinement of them, such as the positive hyper-resolution tableaux, the SATCHMO procedure (Hasegawa et al., 1992; Baumgartner et al., 1996; Bry and Yahya, 1996; Manthey and Bry, 1988; Paramasivam and Plaisted, 1998) etc., the model is obtained as a side effect of the proof procedure (when it terminates). This is easy to see, because each branch of a tableau is (potentially) a partial model of the formula at hand. For resolution-based methods, some special techniques have to be developed in order to extract a model from the saturated set of clauses obtained by resolution. The method by Fermüller and Leitsch (1996, 1998) based on hyper-resolution and that by Tammet (1991) based on the use of ordering strategies can be classified in this category, as the hyper-linking procedure proposed by Plaisted (see, for example, Chu and Plaisted, 1994). One of the main advantages of these techniques is that they can be very efficient, since they take advantage of the particular properties of some classes of formulae. Moreover, they can be implemented as a post-processing in existing theorem-provers.

Simultaneous search for refutations and models (Cafera and Zabel, 1992, 1993; Bourely et al., 1994; Peltier, 1997c). These methods are extensions of proof procedures. Their main originality is that the model is not obtained as a by-product of the refutation process, but is built during the search for a refutation in a systematic (dual) way. The approach captures the standard attitude of a human being faced with a conjecture: trying simultaneously to prove it or to disprove it (by giving a counter-example). The original method is called RAMC (Refutation And Model Construction). The principle is to add to proof procedures some new rules called model construction rules or disinference rules, allowing us to build a model of the initial formula. These rules are used to implement what we call a non-consequence relation, i.e. a relation between some formulae and the formulae that are not consequences of them. This approach has been developed in the context of resolution (Cafera and Zabel, 1992) and in that of tableaux (Cafera and Zabel, 1993; Peltier, 1997b,c).

Each approach has its advantages and drawbacks, though some of them are more general than others. It is therefore natural to try to combine them to obtain more advantages and less drawbacks. In particular, enumeration methods have an important advantage: they are complete w.r.t. the class of interpretations considered, i.e. if the formula at hand has a model in this class, then the method will eventually find it. But such methods are, in general, very costly. On the other hand, deductive method are often more efficient, since they focus on the considered formula (they are in some sense “goal-oriented”). However, they are not complete, i.e. they miss some interpretations. Combining enumeration and deduction can be very useful for improving the capabilities of model building procedures.

So far, enumeration methods have been defined only for finite model building. Though this restriction seems natural, we show in this paper that these kinds of approaches can also be useful to build infinite models. More precisely, we propose an enumeration-
based method allowing us to build automatically (representations of) infinite Herbrand models of first-order formulae. The proposed method can be seen as an extension of enumeration-based finite model builders to infinite models. As in our previous works, infinite interpretations are represented using constraints (equational formulae interpreted on the Herbrand universe). These interpretations are called eq-interpretations (eq as equational). Eq-interpretations were introduced in Caferra and Zabel (1991). Roughly speaking, a partial interpretation is said to be an eq-interpretation if it can be represented by an equational formula, interpreted on the Herbrand universe.

Then, we propose different methods for pruning the search space. In particular, we show how to use deductive procedures, such as RAMC, in order to guide the search for a model. We also give some strategies and we prove their completeness. Our approach can be seen as a contribution to the combination of model building procedures: our method is based on combination of RAMC’s inference and disinference rules with enumeration of the set of eq-interpretations.

The method has been implemented (Peltier, 1997b) and allowed us to build semi-automatically models for some formulae for which, as far as we know, no other method works.

**WHY LOOK FOR NEW TECHNIQUES FOR INFINITE MODEL BUILDING?**

In this section, we motivate the interest of using enumeration-based methods for building eq-models by showing the limits of existing approaches.

From the theoretical point of view, the methods that have been introduced for infinite model building are basically limited for two reasons.

---

The first one is the limit of the expressive power of the representation of interpretations (see, for example, Peltier, 1997a,d). We have proposed elsewhere to extend it by using tree automata and term schematization techniques (see Peltier, 1997a). Other model representation techniques, based on the use of term grammars, have also been proposed in Matzinger (1997).

---

The second limitation is that, even if the formula admits a model representable by equational formulae, the methods may not be able to generate this interpretation. This is well illustrated by the following very simple example.

**Example 1.1.** We call $S$ the following set of 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 equality on the Herbrand universe. $S$ has only one Herbrand model $M$: $P(x, y)$ is true iff $x = y$. $M$ is an eq-interpretation, however no finite representation of $M$ can be generated by using inference rules. Indeed, it is possible to generate the clauses

\[
P(0, 0), P(succ(0), succ(0)), P(succ(succ(0)), succ(succ(0))), P(succ^n(0), succ^n(0)).
\]
but it is impossible to derive the clause $\forall x. P(x, x)$ which is necessary for giving a finite description of the model but is not a logical consequence of the initial set of formulae (it is only an inductive consequence of it). The disinference rules proposed in Caferra and Zabel (1992) will fail as well.

**Remark 1.1.** We have chosen this very simple example for the sake of clarity. Of course, in this case, it is easy to see that ordering strategies (with an obvious ordering) can be used to show that the set of clauses is satisfiable. However, these strategies will not build explicitly a model of the set of clauses: in order to use ordering strategies for model building we would have to give a method for extracting representations of a model from a set of clauses obtained by saturation (as in Fermüller et al., 1993).

Eq-interpretations are “natural” ones in the framework of the ideas underlying our approach. It would therefore be interesting to capture all of them. We will prove that by combining RAMC with enumeration techniques (based on splitting and simplification of clauses) any eq-interpretation can eventually be reached.

The rest of the paper is organized as follows.

1. First, we recall some necessary well-known notions of Logic and Automated Deduction. In particular, we briefly recall some necessary definitions and theorems from Comon and Lescaune (1989).
2. In Section 3, we recall some definitions and results from Caferra and Zabel (1992); Caferra and Peltier (1997) and Peltier (1997b) concerning the use of equational constraints to represent models of first-order formulae. These results are necessary for a full understanding of the present work. We include them in the present paper in order to make it self-contained.
3. In Section 4, we introduce the new enumeration-based method and we prove its main properties (soundness and completeness w.r.t. the class of representable interpretations). The method terminates and returns a model for each satisfiable formula admitting an interpretation representable by equational constraints. Many interesting classes of satisfiable formulae fall into this category: for example, the Bernays–Schönfinkel class (Dreben and Goldfarb, 1979), OCC1N, PVD (Fermüller et al., 1993) etc.
4. In Section 5, we show how to combine this enumeration method with the RAMC method for simultaneous search for refutations and models expounded in Caferra and Zabel (1992). A brief description of the RAMC procedure is given in order to make this paper self-contained. We show how to use RAMC to prune the search space. It allows to detect counter-models and to guide the enumeration of the set of interpretations.
5. In Section 6 we give very simple examples illustrating how this combination works.
6. In Section 7 we propose some strategies for guiding the choice of the partition of the Herbrand base. This choice is crucial in practice for efficiency reasons. The idea behind these strategies is to use the information deduced from the failure of the model building process in order to guide the choice of a new partition. The completeness of this strategy w.r.t. the set of eq-models is proven.
(7) In Section 8, we show how to extend our method to first-order formulae others than conjunction of clauses. This extension is motivated by the fact that the skolemization rule, which is essential to transform first-order formulae into clausal form does not preserve models. Hence it is possible that a formula \( F \) has an eq-model but that the corresponding skolemized formula has no eq-model. In this case, it is necessary to apply our method on the initial formula and not on its clausal form.

(8) In Section 9 we prove that the problem of checking whether a formula has an eq-model is an undecidable one. Therefore there cannot be any syntactic characterization of the class of formulae tractable by our method.

(9) In Section 10, we give an example showing the practical interest of the presented method: we show how to use this approach to build interactively a model of a formula introduced by Goldfarb (1984) for proving the undecidability of the Gödel class with identity.

2. Preliminaries

2.1. Basic definitions and notations

We assume the reader is familiar with the usual terminology of automated deduction and first-order logic. We briefly review some of the basic notions used throughout this work.

Let \( \Sigma \) be a set of functional symbols, \( \Omega \) be a set of predicate symbols and \( V_N \) be an (countable) infinite set of variables. Let “arity” be a function mapping each symbol in \( \Sigma \cup \Omega \) to a natural number (the arity of the symbol). Function symbols of arity 0 are called constants. \( x, y, z, \ldots \) will denote tuples of variables. The set of terms \( T(\Sigma, V) \) is defined as usual over the alphabet \( \Sigma, V \) (where \( V \subseteq V_N \) is a set of variables). If \( V \) is empty, \( T(\Sigma, V) \) is denoted by \( T(\Sigma) \).

An atom is of the form \( P(t_1, \ldots, t_n) \), where \( P \in \Omega \), \( \text{arity}(P) = n \) and \( \forall i \in [1..n], t_i \in T(\Sigma, V) \). A literal is either an atom or the negation of an atom. If \( p \) is a literal, \( \neg p \) denotes the literal with the same predicate symbol and the same arguments as \( p \) but with different sign. A clause is a finite set (or a disjunction) of literals. First-order formulae are built as usual over atoms by using the logical symbols \( \lor, \land, \neg, \exists, \forall \ldots \). By \( \text{Var}(E) \) we denote the set of (free) variables occurring in the expression (term, clause, atom, literal, \ldots) \( E \). If \( \text{Var}(E) = \emptyset \) (i.e. \( E \) does not contain free variables), then \( E \) is called ground. Substitutions will be represented by sets: \( \{x_1 \rightarrow t_1, \ldots, x_n \rightarrow t_n\} \).

The notion of substitution is defined as usual. The result of applying a substitution \( \sigma \) to an expression (term, atom, clause etc.) \( E \) is denoted by \( E\sigma \). The domain of a substitution \( \sigma \) is the set of variables \( x \) such that \( x\sigma \neq x \) (denoted \( \text{Dom}(\sigma) \)). If for all variables \( x \in \text{Dom}(\sigma) \), \( x\sigma \) is ground then \( \sigma \) is called ground.

A position is a finite sequence of integers. We denote by \( \Lambda \) the empty sequence and by “.” the concatenation operator between sequences. The set of positions in a term (or in an atomic formula) \( t \) is denoted by \( \text{Pos}(t) \) and inductively defined as follows: \( \text{Pos}(t) = \{\Lambda\} \), iff \( t \) is a variable; \( \text{Pos}(f(t_1, \ldots, t_n)) = \{\Lambda\} \cup \{i.p/p \in \text{Pos}(t_i)\} \).

If \( p \in \text{Pos}(t) \), we denote by \( t_p \) the term at position \( p \) in \( t \). This is formally defined as follows. \( t_\Lambda = t \) and \( f(t_1, \ldots, t_n)_{i.p} = t_{i.p} \).

A strategy for the application of a non-deterministic procedure is a rule for making choices. It is fair if no choice is indefinitely delayed.
2.2. EQUATIONAL FORMULAE

We briefly recall some necessary notions from Comon and Lescanne (1989).
An equational formula is simply a first-order formula containing as a predicate symbol only “=” (the equality predicate).

**Definition 2.1.** Let $F$ be an equational formula. A ground substitution $\sigma$ of variables $\text{Var}(F)$ is said to be a solution of $F$ iff $F\sigma$ is valid in the empty theory. The set of solutions of $F$ is denoted $\text{Sol}(F)$. Two equational formulae $F_1$ and $F_2$ are said to be equivalent iff $\text{Sol}(F_1) = \text{Sol}(F_2)$.

A basic property of equational formulae is that

**Theorem 2.1.** (Comon and Lescanne, 1989) There exists an algorithm checking whether a given first-order\footnote{With negation, quantifier and connectives.} equational formula has solutions.

An improved version of the constraint-solving algorithm has been implemented. The system, called ECS\textsubscript{ATINF}, is described in Peltier (1998c) and can be downloaded from the WEB (http://www-leibniz.imag.fr/ATINF/).

2.3. INTERPRETATIONS

In what follows “interpretations” means Herbrand interpretations.

A partial interpretation $I$ is a set of ground literals such that, for all atoms $P$, $P \in I \Rightarrow \neg P \notin I$. If, moreover, $P \notin I \Rightarrow \neg P \in I$, then $I$ is said to be total (a total interpretation is a Herbrand interpretation in the usual sense). An interpretation $J$ is said to be an extension of $I$ iff $I \subseteq J$.

A partial interpretation does not necessarily assign a truth value to each ground atom. The notion of validity in a partial interpretation is then naturally defined as follows: a partial interpretation validates a first-order formula $F$ iff all its total extensions validate $F$.

For each partial interpretation $I$, and for each predicate symbol $P$ of arity $n$, we denote by $I(P)^+$ (resp. $I(P)^-$) the set of $n$-uples of ground terms $(t_1, \ldots, t_n)$ such that $P(t_1, \ldots, t_n)$ is true (resp. false) in $I$. The truth values true and false will be, respectively, denoted by $\top$ and $\bot$.

3. Representing Herbrand Interpretations

The choice of the model representations is capital. This problem is very clearly explained in Fermüller and Leitsch (1996): (“little”) finite models can be represented by standard multiplication tables, but when dealing with infinite Herbrand models, formalisms allowing us to represent arbitrary models on the Herbrand universe cannot exist (the class of sets of ground terms is not countable). It is therefore necessary to choose a suitable representation of infinite (and “big” finite) interpretations. It has to be expressive enough to represent a class of models as large as possible, but it should also
allow us to perform effectively some basic operations on the models: for example, compute the truth value of an atom in the model, check whether the model validates a given formula. . . .

We have chosen in our approach to use equational formulae to represent sets of (ground) terms or atoms and interpretations. The reasons of this choice are mainly the following (see also Caferra and Zabel, 1991, 1992; Caferra and Peltier, 1997).

— The class of sets of literals that can be represented by such formulae is closed under the usual Boolean operations: union, intersection, complement, projection. . . .
— Most of the problems about the represented sets (for example the evaluation problem or the emptiness problem† . . . ) can be solved using existing constraints-solving algorithms.
— The class of representable interpretations can be very easily extended, simply by extending the class of equational formulae allowed (see Peltier, 1997a, for an extension to terms with integer exponents and Peltier, 1997d, for an extension to membership constraints).

The notion of eq-interpretation is precisely defined as follows.

**Definition 3.1.** A subset \( E \) of \( T(\Sigma)^n \) is called an eq-set iff there exists an equational formula \( \mathfrak{F}(E) \) such that

\[
(t_1, \ldots, t_n) \in E \iff \{ x_i \rightarrow t_i / 1 \leq i \leq n \} \in \text{Sol}(\mathfrak{F}(E)).
\]

An interpretation (resp. model) \( I \) is said to be an eq-interpretation (resp. eq-model) iff for each predicate symbol \( P \), \( I(P)^+ \) and \( I(P)^- \) are eq-sets. The set of first-order formulae having an eq-model is denoted by \( \mathcal{C}_{\text{eq-model}} \).

**Remark 3.1.** Many “interesting” classes of formulae belong to the \( \mathcal{C}_{\text{eq-model}} \) class. In particular, any satisfiable formulae for the Bernays–Schönfinkel class (class of prenex formula of the form \( \exists x_1, \ldots, x_n. \forall y_1, \ldots, y_m. M \)), the OCC1N and PVD classes (Fermüller et al., 1993), etc.

Moreover, \( \mathcal{C}_{\text{eq-model}} \) also contains some classes of formulae for which no ordering restriction of resolution can be a decision procedure (Klingenbeck, 1996).

Any “good” model representation formalism must allow us to perform effectively some basic operations on the interpretations.

We propose below an algorithm allowing us to calculate some of the solutions of a formula \( F \) in an eq-interpretation \( I \). This algorithm is incomplete, i.e. it does not compute all the possible solutions. However, it is complete for total interpretations. The underlying idea is to translate the formula \( F \) into a purely equational formula, having the same set of solutions as the original one.

More precisely, Definition 3.2 below introduces two formulae \( \phi^+_M(F) \) and \( \phi^-_M(F) \) expressing conditions that are sufficient (but in general not necessary) to evaluate \( F \) to true (resp. false) in a partial eq-interpretation \( M \).

†The emptiness problem is the problem of deciding whether a given set of literals is empty.
Remark 3.2. An example is included in the following definition in order to help the reader to grasp the meaning of the equational formulae defined in it.

Definition 3.2. Let $\mathcal{M}$ be a partial eq-interpretation and $\mathcal{F}$ a formula. $\phi^+_{\mathcal{M}}(\mathcal{F})$ and $\phi^-_{\mathcal{M}}(\mathcal{F})$ are equational formulae defined as follows.

- If $\mathcal{F}$ is of the form $P(\bar{t})$, then
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \exists \bar{x}. \bar{t} = \bar{t} \land \overline{\mathcal{F}}(\mathcal{M}(P)^+),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \exists \bar{x}. \bar{t} = \bar{t} \land \overline{\mathcal{F}}(\mathcal{M}(P)^-),$$
  where $\overline{\mathcal{F}}$ are the variables corresponding to $\overline{\mathcal{F}}(\mathcal{M}(P)^+)$ and $\overline{\mathcal{F}}(\mathcal{M}(P)^-)$ (resp. $\overline{\mathcal{F}}(\mathcal{M}(P)^+)$ and $\overline{\mathcal{F}}(\mathcal{M}(P)^-)$) the equational formulae corresponding to the eq-set $\mathcal{M}(P)^+$ (resp. $\mathcal{M}(P)^-$). $\phi^+_{\mathcal{M}}(\mathcal{F}(\bar{t}))$ clearly formalizes the idea that $\mathcal{F}(\bar{t})$ is evaluated to true (resp. false) on ground terms $\bar{t}$ such that $\bar{t} \in \mathcal{M}(P)^+$ (resp. $\bar{t} \in \mathcal{M}(P)^-$).

Example 3.1. Let $\mathcal{F} \equiv P(x, x)$. Let $\mathcal{M}$ be a partial eq-interpretation defined by: $P(x_1, x_2)$ is true if $x_1 = a$ and $\exists u. x_2 = g(u)$. Then $\overline{\mathcal{F}}(\mathcal{M}(P)^+)$ $\equiv x_1 = a \land \exists u. x_2 = g(u)$ hence $\phi^+_{\mathcal{M}}(\mathcal{F}) \equiv \exists x_1, x_2. x_1 = x_1 \land x_2 = x_2 \land x_1 = a \land \exists u. x_2 = g(u)$, i.e. $\phi^+_{\mathcal{M}}(\mathcal{F}) \equiv \exists x_1, x_2. x_1 = a \land x_2 = a \land \exists u. a = g(u)$, i.e. $\phi^+_{\mathcal{M}}(\mathcal{F}) \equiv \bot$.

- If $\mathcal{F}$ is of the form $\mathcal{F}_1 \lor \mathcal{F}_2$:
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \phi^+_{\mathcal{M}}(\mathcal{F}_1) \lor \phi^+_{\mathcal{M}}(\mathcal{F}_2),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \phi^-_{\mathcal{M}}(\mathcal{F}_1) \land \phi^-_{\mathcal{M}}(\mathcal{F}_2).$$

- If $\mathcal{F}$ is of the form $\mathcal{F}_1 \land \mathcal{F}_2$:
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \phi^+_{\mathcal{M}}(\mathcal{F}_1) \land \phi^+_{\mathcal{M}}(\mathcal{F}_2),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \phi^-_{\mathcal{M}}(\mathcal{F}_1) \lor \phi^-_{\mathcal{M}}(\mathcal{F}_2).$$

- If $\mathcal{F}$ is of the form $\exists x. \mathcal{F}_1$:
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \exists x. \phi^+_{\mathcal{M}}(\mathcal{F}_1),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \forall x. \phi^-_{\mathcal{M}}(\mathcal{F}_1).$$

- If $\mathcal{F}$ is of the form $\forall x. \mathcal{F}_1$:
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \forall x. \phi^+_{\mathcal{M}}(\mathcal{F}_1),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \exists x. \phi^-_{\mathcal{M}}(\mathcal{F}_1).$$

- If $\mathcal{F}$ is of the form $\neg \mathcal{F}_1$:
  $$\phi^+_{\mathcal{M}}(\mathcal{F}) := \phi^-_{\mathcal{M}}(\mathcal{F}_1),$$
  $$\phi^-_{\mathcal{M}}(\mathcal{F}) := \phi^+_{\mathcal{M}}(\mathcal{F}_1).$$

Theorem 3.1. Let $\mathcal{M}$ be a partial eq-interpretation, and $\mathcal{F}$ be a first-order formula. Let $\sigma$ be a substitution. If $\sigma \in \text{Sol}(\phi^+_{\mathcal{M}}(\mathcal{F}))$, then $\mathcal{M} \models \mathcal{F} \sigma$. If $\sigma \in \text{Sol}(\phi^-_{\mathcal{M}}(\mathcal{F}))$, then $\mathcal{M} \models \neg \mathcal{F} \sigma$. 


Example 3.2. Let \( \mathcal{F} \equiv P(x) \lor \neg Q(x,y) \). Let \( \mathcal{I} \) be the partial Herbrand eq-interpretation defined by: \( P(x) \) is true in \( \mathcal{I} \) if \( x = a \), and \( Q(x,y) \) is false in \( \mathcal{I} \) if \( x = y \). Then \( \phi^+_I(\mathcal{F}) \equiv (x = a \lor x = y) \). For any solution \( \sigma \) of \( \phi^+_I(\mathcal{F}) \), we have \( \mathcal{I} \models \mathcal{F}\sigma \).

If \( \mathcal{I} \) is total, the converse of Theorem 3.1 is also true:

Theorem 3.2. If \( \mathcal{I} \) is total then \( \phi^+_I(\mathcal{F}) \lor \phi^-_I(\mathcal{F}) \equiv \top \).

Proof. By structural induction on \( \mathcal{F} \).

— Atomic formula. By definition, \( \sigma \in \text{Sol}(\phi^+_I(\mathcal{F})) \) iff \( \sigma \in \mathcal{I}^+(P) \). Since \( \mathcal{I} \) is total, this is equivalent to \( \forall \sigma \in \mathcal{I}^+(P) \), hence to \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F})) \).

— If \( \mathcal{F} = \mathcal{F}_1 \lor \mathcal{F}_2 \), and if \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F})) \), then \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F}_1)) \) and \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F}_2)) \). By the induction hypothesis, it implies that \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F}_1)) \) and \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F}_2)) \), i.e. \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F})) \).

— The proof is similar if \( \mathcal{F} = \mathcal{F}_1 \land \mathcal{F}_2 \).

— If \( \mathcal{F} = \neg \mathcal{F}' \), then \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F})) \) implies \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F}')) \) hence (by the induction hypothesis) \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F}')) \) and \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F})) \).

— If \( \mathcal{F} = \exists x. \mathcal{F}' \), and if \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F})) \), then by definition for all terms in \( T_s(\Sigma) \), we have \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F}'(x \mapsto t))) \) (where \( x \in \mathcal{V}_s \)). Hence by the induction hypothesis \( \forall t \in T_s(\Sigma), \sigma \in \text{Sol}(\phi^-_I(\mathcal{F}(x \mapsto t))) \), i.e. \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F})) \).

— The proof is similar for \( \mathcal{F} = \forall x. \mathcal{F}' \). □

Example 3.3. Let \( \mathcal{F} \equiv P(x) \lor \neg Q(x,y) \). Let \( \mathcal{I} \) be the total Herbrand eq-interpretation defined by: \( P(x) \) is true in \( \mathcal{I} \) if \( x = a \) (and false otherwise), and \( Q(x,y) \) is false in \( \mathcal{I} \) if \( x = y \) (and true otherwise). Then \( \phi^+_I(\mathcal{F}) \equiv (x = a \lor x = y) \) and \( \phi^-_I(\mathcal{F}) \equiv (x \neq a \land x \neq y) \). Therefore \( \phi^+_I(\mathcal{F}) \lor \phi^-_I(\mathcal{F}) \equiv \top \).

Corollary 3.1. Let \( \mathcal{I} \) be a total eq-interpretation and \( \mathcal{F} \) a formula. Let \( \sigma \) be a substitution of domain \( \text{Var}(\mathcal{F}) \).

\[
\mathcal{I} \models \mathcal{F}\sigma \Leftrightarrow \sigma \in \text{Sol}(\phi^+_I(\mathcal{F}))
\]

\[
\mathcal{I} \nmodels \mathcal{F}\sigma \Leftrightarrow \sigma \in \text{Sol}(\phi^-_I(\mathcal{F})).
\]

Proof. It suffices to show that \( \mathcal{I} \models \mathcal{F}\sigma \) implies that \( \sigma \in \text{Sol}(\phi^+_I(\mathcal{F})) \) (the converse is a consequence of Theorem 3.1). Assume that \( \mathcal{I} \models \mathcal{F}\sigma \) and that \( \sigma \notin \text{Sol}(\phi^+_I(\mathcal{F})) \). Then, by Theorem 3.2, we have \( \sigma \in \text{Sol}(\phi^-_I(\mathcal{F})) \). Therefore \( \mathcal{I} \nmodels \neg \mathcal{F}\sigma \) by Theorem 3.1. Hence \( \mathcal{I} \) validates simultaneously \( \mathcal{F}\sigma \) and \( \neg \mathcal{F}\sigma \), which is impossible. □

Decidability result. Finding the truth value of a first-order formula in a total eq-interpretation is therefore reduced to solving a first-order equational formula in the Herbrand universe (which is a decidable problem by Theorem 2.1).
Remark 3.3. The corresponding model checking algorithm has been implemented. The system ECS\_ATIN\_F allows the user to define very easily eq-interpretations, and to evaluate first-order formulae in these interpretations (see Peltier, 1998c, and http://www-leibniz.imag.fr/ATINF for details).

We recall the notion of constrained clauses (or \emph{c-clauses}).

Definition 3.3. A \emph{constrained clause} (or a \emph{c-clause} for short) is a couple \([C : X]\) where:

- \(C\) is a clause (in the standard sense);
- \(X\) is an equational formula.

If \(C\) is unit, then \([C : X]\) is called a \emph{c-literal}.

Roughly speaking, a constrained clause denotes the set of its ground instances. A total interpretation \(\mathcal{I}\) \emph{validates} a \emph{c-clause} \([C : X]\) iff for all substitution \(\sigma \in \text{Sol}(X)\), \(\mathcal{I} \models C_\sigma\).

If \(X \equiv \top\), \([C : X]\) is equivalent to the clause \(C\). Hence standard clauses are particular cases of c-clauses.

A satisfiable set of unit c-clauses \(S\) can be seen as the representation of the partial Herbrand interpretation \(\mathcal{I}\), defined as follows. For all ground literal \(L\), \(L \in \mathcal{I}\) iff there exists a c-clause \([L' : X]\) in \(S\) and a solution \(\sigma \in \text{Sol}(X)\), such that \(L'\sigma = L\).

Conversely, each partial Herbrand interpretation \(\mathcal{I}\) can be represented by the set of unit c-clauses \(S = \{[P(x_1,\ldots,x_n) : \phi^+_{\mathcal{I}}(P(x_1,\ldots,x_n))] \cup \{\neg P(x_1,\ldots,x_n) : \phi^-_{\mathcal{I}}(P(x_1,\ldots,x_n))\}\}.

Hence, in the following, we will identify satisfiable sets of unit c-clauses and partial Herbrand eq-interpretations.

Notation 3.1. We introduce a few notations.

Let \(L_1 = [P(\bar{t}) : X]\) and \(L_2 = [P'(\bar{s}) : Y]\) be two unit c-clauses. We denote by \(L_1 \setminus L_2\) the unit c-clause defined as follows:

\[
L_1 \setminus L_2 = \begin{cases} [P(\bar{t}) : X \land \forall \bar{x} (\neg Y \lor \bar{x} \neq \bar{t})] & \text{if } P = P' \\ L_1 & \text{if } P \neq P' \end{cases}
\]

where \(\bar{y}\) are the variables in \(L_2\). The use of set notation ("\(\setminus\)") is motivated by the fact that the set of ground literals denoted by \(L_1 \setminus L_2\) is the same as the set difference of those denoted by \(L_1\) and \(L_2\).

Let \(L_1\) and \(L_2\) be two \emph{c-literals}. We say that \(L_1 \subseteq L_2\) iff \(L_1 \setminus L_2\) is equivalent to \(\top\) (i.e. the set of ground literals denoted by \(L_1\) is included in the set of ground literals denoted by \(L_2\)).

Let \(L_1 = [P(\bar{t}) : X]\) and \(L_2 = [P'(\bar{s}) : Y]\) be two unit c-clauses. We denote by \(L_1 \cap L_2\) the unit c-clause defined as follows:

\[
L_1 \cap L_2 = \begin{cases} \top & \text{if } P \neq P' \\ [P(\bar{t}) : X \land \exists \bar{s} (Y \lor \bar{s} = \bar{t})] & \text{if } P = P' \end{cases}
\]

(i.e. if \(P\) and \(P'\) are the same predicate symbols).
where \( \overline{y} \) are the variables in \( L_2 \). Obviously the set of ground literals denoted by \( L_1 \cap L_2 \) is the intersection of those denoted by \( L_1 \) and \( L_2 \).

The following definition will be used in the next sections.

**Definition 3.4.** A \( c \)-clause \( [C : \mathcal{P}] \) is in *normal form* iff \( \mathcal{P} \) is a conjunction of disequations of the form \( \bigwedge_{i=1}^{n} x_i \neq t_i \) where \( x_i \) are distinct variables and \( \forall i \leq n. x_i \neq t_i \). The *depth* of a \( c \)-clause \( [C : \mathcal{P}] \) in normal form is the maximal depth of the terms in \( C \) plus the depths of the terms in \( \mathcal{P} \).

**Theorem 3.3.** Any set of clauses can be automatically transformed into a set of clauses in normal form.

**Proof.** It suffices to use the constraints-solving rules in Comon and Lescanne (1989) then the structural simplification rules in Caferra and Zabel (1992) with an obvious strategy. □

4. Enumerating Eq-interpretations

It is clear that the set of eq-interpretations is recursively enumerable. Moreover, it has been proven (see Theorem 3.1 and Caferra and Peltier, 1997; Peltier, 1997b) that the problem of finding the truth value of a first-order formula in a total eq-interpretation is decidable. Hence, an obvious (and very naive) algorithm for building an eq-model of a given formula consists of enumerating all eq-interpretations on the considered signature and testing if they validate the formula at hand. The main advantage of this approach is that it can deal with all formulae having an eq-model. Hence, in contrast to other Herbrand model building methods, the capabilities of ours are *independent* from the syntactical properties of the formula, and only depends on its *semantics*.

Obviously, such a brute-force method is useless in practice. Hence, we must reduce the search space, by reducing the number of interpretations to be considered. To achieve this goal, we *combine* deductive and enumeration methods in a new method called EQMC (EQ-Model Construction).

In this section, we give a formal definition of the enumeration process. For this purpose, we assume given a procedure called *Deduce*, that will be defined later (in Section 5). Informally, this procedure corresponds to the deductive part of the method. Its aim is twofold. First, it is used to detect that a given interpretation is a model or a counter-model of the initial formula. Second, it is used to reduce the search space by deriving new consequences and non-consequences of the initial formula.

**Remark 4.1.** We would like to emphasize the *modularity* of our approach. The enumeration process is clearly independent from the procedure *Deduce* (used to reduce the search space). Hence several different procedures can be combined with it: resolution procedures, our method RAMC for simultaneous search for refutation and models (Caferra and Zabel, 1992; Bourely et al., 1994), etc.

4.1. Representation sets

We first introduce a key concept in our method: the notion of a *representation set*. 
Definition 4.1. A set of positive unit $c$-clauses (i.e., positive $c$-literals) $D$ is called a representation set iff the following conditions are satisfied.

- For all pair $(L_1, L_2) \in D^2$, $L_1 \cap L_2 = \emptyset$. This condition ensures that the set of ground atoms denoted by two distinct $c$-literals are pairwise disjoint.
- For all ground literal $L_g$ there exists a $c$-literal $L \in D$ such that $L_g$ is an instance of $L$. This condition means that all ground literals can be captured by $D$.

These two requirements guarantee that a representation set $D$ defines a partition of the Herbrand base on the signature $\Sigma$.

Then it is possible to represent some eq-interpretations by giving a truth value to each element of a given equivalence class in the partition. This is possible only if the considered interpretation is compatible with $D$, that is to say if two atoms belonging to the same class of the partition have the same truth value in the interpretation.

Definition 4.2. A unit $c$-clause $C$ is said to be compatible with a given set of $c$-literals $D$ (or $D$-compatible) iff for each $c$-literal $L$ such that $L \in D$ or $\neg L \in D$ we have: if $L \cap C \neq \emptyset$, then $L \subseteq C$. A set of unit $c$-clauses $S$ is said to be $D$-compatible iff each $c$-clause in $S$ is $D$-compatible. An eq-interpretation $M$ is said to be $D$-compatible iff there exists a representation of $M$ by a set of $D$-compatible $c$-clauses.

Example 4.1.

$$D = \{ P(x, y), Q(a), Q(b), Q(f(x, x)), [Q(f(x, y)) : x \neq y] \}$$

is a representation set of the signature $\Sigma = \{ a, b, f \}$ and $\Omega = \{ P, Q \}$. The interpretation

$$I = \{ P(x, y), Q(b), [\neg Q(x) : x \neq b] \}$$

is $D$-compatible. The set of $c$-clauses

$$J = \{ P(a, y), [\neg P(x, y) : x \neq a], Q(b), [\neg Q(x) : x \neq b] \}$$

is not $D$-compatible: $P(x, y) \cap P(a, y) \neq \emptyset$ and $P(x, y) \subseteq P(a, y)$. Moreover, it is clear that the interpretation denoted by $J$ is not $D$-compatible.

Definition 4.3. A set of $c$-literals $D$ is said to be compatible with a formula $F$ iff there exists a model of $F$ that contains $D$ (i.e. if $D$ is a partial eq-interpretation and $D \neq \neg F$).

4.2. Generating representation sets

Definition 4.1 states properties of representation sets but does not give hints as how to generate them. The first problem we have to solve is how to generate suitable partitions of the Herbrand base, i.e., how to generate representation sets. We will use the following rules, operating on representation sets.

$$\text{GR}_\Sigma: D \cup \{ [L : X] \} \rightarrow D \cup \{ [L : X \land x = f(\bar{z})] / f \in \Sigma \}$$

If there exist at least two symbols $f$ such that the formula $X \land x = f(\bar{z})$ is satisfiable, and if $x \in \text{Var}([L : X])$, and $\bar{z}$ are new variables.

$$\text{GR}_=: D \cup \{ [L : X] \} \rightarrow D \cup \{ [L : X \land t = s], [L : X \land t \neq s] \}$$

if $X \land t \neq s \neq \bot$ and $X \land t = s \neq \bot$, if $t, s$ are two terms occurring in $[L : X]$. 
The conditions restricting the application of the rules insures that this application really modify the sets of ground literals, denoted by the representation set. Without these restrictions, applying the GR_Σ and GR_= rules could leave the representation unchanged.

We denote by GR the system \{GR_Σ, GR_=\}. If \( D \) is a representation set, then the set \( D' \) obtained by applying the GR rules on \( D \) is a representation set.

**Example 4.2. (The system GR)** Let \( \Omega = \{P\}, \Sigma = \{a, f\} \), and \( D_0 = \{P(x)\} \). We apply the GR_Σ rule on the variable \( x \) in \( D_0 \). We obtain (after simplification) the set \( D_1 = \{P(a), P(f(u, v))\} \). Then we apply the rules GR_= on the terms \( u \) and \( v \). We obtain: \( D_2 = \{P(a), P(f(u, u)), [P(f(u, v)) : u \neq v]\} \).

\( D_0, D_1, D_2 \) are representation sets.

Let \( D_0 \) be a representation set. We show below that for all eq-interpretations \( I \), any application of the GR rules leads to a representation set \( D \) such that \( I \) is \( D \)-representable.

This property is not true in general and we have to impose further restrictions on the application of the rules in order to insure that they are applied in a fair way.

**Definition 4.4.** Let \( s \) be a strategy for the system GR. \( s \) is said to be fair iff for all infinite sequence, \( D_1 \rightarrow_s D_2 \rightarrow_s \cdots \rightarrow_s D_n \rightarrow_s \cdots \) and for all \( L_1 \in D_1 \), we have:

- if the rule GR_= is applicable on \( L_1 \) with two terms \( t \) and \( s \), then there exists \( i \) such that for all \( L_i \in D_i \) the rule GR_= is not applicable on \( L_i \) on the literals \( t \) and \( s \);
- if the rule GR_Σ is applicable on \( L_1 \) with \( x \), there exists \( i \) such that for all \( L_i \in D_i \), GR_Σ is not applicable on \( L_i \) with \( x \).

**Example 4.3.** The strategy consisting in applying the rules on terms occurring at positions of minimal length is fair.

The following theorem gives simple conditions ensuring that an eq-interpretation will be representable by a partition of the Herbrand universe.

**Theorem 4.1.** Let \( (D_i)_{i \in \mathbb{N}} \) be a fair derivation using the GR rules. Let \( I \) be an eq-interpretation. There exists \( k \in \mathbb{N} \) such that \( I \) is \( D_k \)-representable.

The proof of Theorem 4.1 needs a few definitions and lemmata.

**Definition 4.5.** Let \( n \in \mathbb{N} \). A c-literal \([P : X]\) is said to be \( n \)-maximal iff the following conditions are satisfied.

- For all position \( p \) in \( P \) such that \(|p| \leq n, P|_p\) exists and is not a variable.
- For all pairs of positions \( p_1, p_2 \) such that \(|p_1| \leq n \) and \(|p_2| \leq n \), either \( P|_{p_1} = P|_{p_2} \land X \equiv \top \) or \( P|_{p_1} = P|_{p_2} \land X \equiv \bot \).

We denote by \( D_n \) the set of all distinct \( n \)-maximal literals.

The set of \( n \)-maximal literals has an interesting property: any eq-interpretation is \( D_n \)-compatible for some \( n \). The proof of this assertion needs the following lemma.
LEMMA 4.1. Let \([L(\overline{t}) : X]\) be a c-clause in normal form of depth lower than \(n\). Let \([L(\overline{s}) : Y]\) be an \(n\)-maximal literal. If there exists a solution \(\sigma\) of \(Y \land \overline{t} = \overline{s} \land X\), then for all solutions \(\theta_1\) of \(Y\), there exists a solution \(\theta_2\) of \(X\) such that \(\overline{t} \theta_2 = \overline{s} \theta_1\).

PROOF. Without loss of generality we assume that \([L(\overline{s}) : Y]\) is in normal form. Let \(\overline{\sigma} = \text{Var}(\{L(\overline{t}) : X]\)) and \(\overline{\sigma} = \text{Var}(\{L(\overline{s}) : Y]\)).\) Assume that \(\overline{t} = \overline{s} \land X \land Y\) is not equivalent to \(\bot\). Let \(\theta_1\) be a solution of \(Y\). Assume that there exists a solution \(\sigma\) of \(X \land Y\) such that \(\overline{t} \sigma = \overline{s} \sigma\). Let \(x \in \overline{\sigma}\). Let \(q_1, q_2\) two positions in \(L(\overline{t})\) such that \(L(\overline{t})|_{q_1} = L(\overline{t})|_{q_2} = x\). \(\sigma \in \text{Sol}(Y \land L(\overline{s})|_{q_1} = L(\overline{s})|_{q_2})\). Hence \(Y \land L(\overline{s})|_{q_1} = L(\overline{s})|_{q_2} \neq \bot\). Since \(|q_1| \leq n\) and \(|q_2| \leq n\), and since \([L(\overline{s}) : Y]\) is \(n\)-maximal, \(Y \land L(\overline{s})|_{q_1} = L(\overline{s})|_{q_2}\) is either \(\top\) or \(\bot\), hence equal to \(\top\). We denote by \(t_x\) a term at a position \(q\) (arbitrarily chosen) in \([L(\overline{s}) : Y]\) such that \(L(\overline{t})|_q = x\).

Let \(\theta_2\) a substitution of \(\overline{\sigma}\) defined by \(x \theta_2 = t_x \theta_1\). Assume that \(\theta_2 \not\in \text{Sol}(X \land Y \land \overline{t} = \overline{s})\). Let \(q\) be a position of \(L(\overline{t})\). \([L(\overline{s}) : Y]\) is \(n\)-maximal hence the term at position \(q\) in \([L(\overline{s}) : Y]\) is not a variable (since \(|q| < n\)). Therefore \(\overline{s} = \overline{t}\) is reduced by decomposition either to \(\bot\) (which is impossible since \(\overline{t} = \overline{s}\) has at least one solution) or to a conjunction of equations \(x = t\) where \(x \in \overline{\sigma}\). Hence by definition of \(\theta_2\) and \(t_x\), we have \(\theta_2 \in \text{Sol}(\overline{t} = \overline{s} \theta_1 \land Y \theta_1\)).

Let \(x\) be a variable in \(\overline{\sigma}\) such that \(X\) contains a dissequation of the form \(x \neq s\). \(x\) occurs at a position \(q\) in \(L(\overline{t})\). Since \([L(\overline{s}) : Y]\) is \(n\)-maximal, \(Y \land t_x \theta_1 \neq s \theta_2\) is either \(\bot\) (impossible since \(\sigma\) is a solution) or is equivalent to a disjunction of dissequations of the form \(y \theta_2 \neq t'\) where \(y\) is a variable in \(s\) and \(t'\) a term in \(t_x \theta_1\), i.e. of the form \(t_y \theta_1 \neq t'\). Since \([L(\overline{s}) : Y]\) is \(n\)-maximal, \(t_y \theta_1 \neq t'\) is either \(\bot\), or \(\top\), hence is \(\top\). Hence \(\theta_2 \in \text{Sol}(X)\).

Therefore \(\theta_2 \in \text{Sol}(X \land \overline{s} = \overline{t})\). □

LEMMA 4.2. Let \(I\) an eq-interpretation. There exists \(n \in \mathbb{N}\) such that, for all \(n\)-maximal sets \(D\), \(I\) is \(D\)-representable.

PROOF. Without loss of generality, we assume that \(I\) is in normal form. Let \(n\) the maximal depth of the terms in \(I\). We show that \(I\) is \(D_n\)-compatible. Let \([L(\overline{s}) : Y]\) be a \(n\)-maximal literal. Assume that there exists a substitution \(\sigma\) of \(Y\) such that \(I \models L(\overline{s})\sigma\). By definition, there exists a c-clause \([L(\overline{t}) : X] \in I\) such that \(\sigma\) is a solution of \(\exists \overline{x}. X \land Y \land \overline{s} = \overline{t}\). By Lemma 4.1, we conclude that for all solutions \(\theta_1\) of \(Y\), there exists a set solution \(\theta_2\) of \(X\) such that \(\overline{t} \theta_2 = \overline{s} \theta_1\). By definition, \(I \models L(\overline{t}) \theta_2\), hence \(I \models L(\overline{s}) \theta_1\). □

PROOF. (OF THEOREM 4.1) By Lemma 4.2, there exists \(n\) such that \(I\) is \(D\)-representable, for all \(n\)-maximal sets \(D\). Since the number of terms at a position lower than \(n\) is finite, and since the application of the rules is fair, there exists \(k \in \mathbb{N}\) such that the normal form of \(D_k\) is \(n\)-maximal. Hence \(I\) is \(D_k\)-representable. □

Theorem 4.1 gives a way of enumerating representation sets, hence of enumerating the partitions of the Herbrand base: it suffices to choose an arbitrary representation \(D\), then to apply the GR rules on \(D\). Theorem 4.1 insures that we will eventually obtain a partition suitable for the formula for which we want to build a model, in the sense that if the formula admits an eq-model, then it admits an eq-model compatible with this partition of the Herbrand base.

1For example, one can choose the set \(\{P(x_1, \ldots, x_n) / P \in \Omega, \text{arity}(P) = n\}\), where \(x_1, x_2, \ldots\) are new variables, which is obviously a representation set.
4.3. Specifying the Method

Informal Presentation

The method can be roughly specified as follows.

1. Choose a finite partition of the Herbrand base (defined by the choice of a representation set).
2. Enumerate the set of interpretations that are compatible with this partition, by choosing a truth value for each element of the partition.
3. If no eq-interpretation can be obtained, the process is iterated with another partition of the Herbrand base. This new partition is obtained by applying the rules GR_α and GR_Σ (using a fair strategy). As we will see in Section 7 the application of these rules can be guided by the informations deduced from the preceding step.

The choice of the truth value of each element of the partition is made by a method similar to the well-known Davis and Putnam’s procedure (1960): it consists of decomposing the problem $F$ into two sub-problems obtained by adding to $F$ the literals $P$ and $\neg P$, where $P$ belongs to the chosen representation set. This process is iterated on each literal in the representation set, in order to enumerate the set of $D$-compatible interpretations. In order to reduce the search space, it is necessary to detect, as soon as possible, that a partial interpretation is a counter-example (counter-model) of the formula. To do that, we use the procedure Deduce, operating on first-order formulae (or sets of c-clauses). Its aim is to:

— check the compatibility of the partial interpretation generated so far with the considered formula; and
— guide the application of the decomposition rule, by generating consequences and non-consequences of the formula.

The specification of such procedure is given in Figure 1.

During the enumeration of eq-interpretations, we have to take into account the fact that the procedure Deduce does not necessarily preserve equivalence. Hence, we might have to modify, during the search for a model, the considered representation set in order to take into account the new facts generated by the deduction procedure. This problem is well illustrated by the following example (in the framework of the RAMC method).

Example 4.4. Let $S$ be the following set of c-clauses.

$$\{ [P(x) \lor Q(x) : x \neq b], \]

$$

$P(a),

$\neg P(b) \lor Q(x)$.

Let $D = \{ P(x), Q(x) \}$. $S$ has a $D$-compatible eq-model $I = \{ P(x), Q(x) \}$. However, if we apply the Gpl rule (one of the RAMC’s rules, see Section 5) in order to simplify the set of c-clauses $S$ we obtain the c-clause $\neg P(b)$. $S \cup \{ \neg P(b) \}$ does not have any $D$-compatible eq-model. However, $S \cup \{ \neg P(b) \}$ admits the following eq-model $\{ \neg P(b), [P(x) : x \neq b], Q(x) \}$ which is compatible with the set $\{ [P(x) : x \neq b], Q(x), \neg P(b) \}$. 
Procedure Deduce

INPUT:
- a formula $F$
- a partial eq-interpretation $I$

OUTPUT:
- a formula $F'$
- a partial eq-interpretation $I'$

such that:
- if $I$ is total then either $\text{Deduce}(F, I) = (\bot, I)$ or $\text{Deduce}(F, I) = (\top, I)$ (C1)
- for all $D$, if there exists a $\mathcal{J}_D$-compatible model of $F$
  then there exists a $\mathcal{J}_D$-compatible interpretation $J \supseteq I'$ with $J \models F'$ (C2)
- $I \subseteq I'$ (C3)
- $F' \wedge I' \Rightarrow F \wedge I$ (C4)
- If $F \wedge I$ is satisfiable then $F' \wedge I'$ is satisfiable. (C5)

Figure 1. The procedure Deduce.

the EQMC (EQ-model Construction) procedure

We introduce the following notation.

Notation 4.1. Let $S = \{L_1, \ldots, L_n\}$ a set of c-literals, and $L$ a c-literal. We denote by $\mathcal{J}_S(L)$ the literal

$$L \land \neg L_1 \land \ldots \land \neg L_n.$$ 

Similarly, for all set of c-literal $S'$, we denote by $\mathcal{J}_S(S')$ the set $\{\mathcal{J}_S(L)/L \in S'\}$.

Intuitively, $\mathcal{J}_S(L)$ is the literal obtained by deleting from the set of ground literals denoted by $L$ each ground literal whose complement is in $S$.

Definition 4.6. A procedure is said to be admissible iff it terminates and it satisfies the specification depicted on Figure 1.

Informally, these conditions ensure the following properties.

- The procedure is “correct” in the sense that it transforms a couple $(F, I)$ into a couple $(F', I')$ such that:
  - the satisfiability of the formula is preserved.
  - $F \wedge I$ is a logical consequence of $F' \wedge I'$ (C4).
  - if $F$ admits a $\mathcal{J}_D$-compatible model, then $F'$ admits a $\mathcal{J}_D$-compatible model (C2). This property guarantees that each eq-model can be obtained, which insures the completeness of the procedure EQMC (see Theorem 4.4).
  - Each total model or counter-model is detected by the method (C1). The search for a model is therefore stopped.
  - Monotonicity, $I \subseteq I'$. This property insures the termination of the procedure EQMC (see Lemma 4.3).
Procedure BuildMod

**INPUT:**
- a formula \( F \).
- a representation set \( D \).

**OUTPUT:**
- an eq-model \( M \) of \( F \),
- or a message **no model found**.

\[
\begin{align*}
S & := \{(F, \emptyset)\} \\
\text{while} & \ S \neq \emptyset \land \forall (F', I) \in S, F' \neq \top \\
\text{begin} & \ 
\text{choose a rule } \rho \text{ in } R_D \\
& \text{Apply } \rho \text{ on } S \\
\text{end} \\
\text{if} & \ S = \emptyset \\
\text{then return (no model found)} \\
\text{else } & \{\exists (\top, I) \in S\} \\
& \text{return } (I) \\
\end{align*}
\]

**Figure 2.** The **BuildMod** procedure.

Let \( D \) be a representation set and \( F \) a formula, the algorithm **EQMC** searches for a \( D \)-**compatible** model of \( F \). It is specified by a set of rewriting rules operating on sets of couples \( (F, I) \) where \( F \) is a formula and \( I \) a partial interpretation. \( F \) is the considered formula and \( I \) represents the interpretation built so far by the method. Initially the set of couples contains only the couple \( (F, \emptyset) \), where \( F \) is the initial formula.

(1) **Deduction rule.** It consists of applying the **Deduce** procedure on a couple \( (F, I) \).
(2) **Decomposition rule.** The problem \( (F, I) \) is divided into two sub-problems obtained adding, respectively, \( P \) and \( \neg P \), to \( I \), where \( P \) is a \( c \)-literal of \( D \). It should be noted that as already mentioned the procedure **Deduce** can modify the interpretation \( I \) and, in particular, can generate \( c \)-literals that are not \( D \)-**compatible**. Hence it is not sufficient to apply the decomposition rule on each \( c \)-literal in \( D \): we have to remove \( c \)-literals whose complement occurs in \( I \) (this is the goal of the interpretation of \( J_S(L) \)).
(3) **Cleaning rule.** It simply suppresses couples of the form \((\bot, I)\).

\( R_D \) denotes the system composed by the rules \{**Deduction**, **Cleaning**, **Decomposition**\} below.

**Deduction.** 
\[
S \cup \{(F, I)\} \rightarrow S \cup \{(F', I')\}
\]
if **Deduce**\((F, I) = (F', I')\)

**Cleaning.** 
\[
\{((\bot, I)) \cup S \rightarrow S
\]

**Decomposition.** 
\[
S \cup \{(F, I)\} \rightarrow S \cup \{(F, I \cup \{J_T(P)\}), (F, I \cup \{J_T(\neg P)\})\}
\]
if \( P \in D, J_T(P) \neq \emptyset, J_T(\neg P) \neq \emptyset \)

The procedure **EQMC** is formally specified in Figure 3.
Procedure EQMC \{ EQ-Model Construction \}
\begin{algorithmic}
\Input a formula $\mathcal{F}$.
\Output (if any):
\State an eq-model $\mathcal{M}$ of $\mathcal{F}$,
\State or a message no model found.
\Begin
\State $\mathcal{D} := \{P(\pi) / P \in \Omega\}$
\Repeat
\State or $\mathcal{D} := \text{GR}(\mathcal{D})$
\State or $\mathcal{M} := \text{EQMC}(\mathcal{F}, \mathcal{D})$
\If{$\mathcal{M} \neq$ no model found}
\State return($\mathcal{M}$)
\EndIf
\If{$\text{GR}$ is not applicable and $\text{EQMC}(\mathcal{F}, \mathcal{D}) =$ no model found}
\State return(no model found)
\EndIf
\EndRepeat
\End
\end{algorithmic}

Figure 3. The EQMC procedure.

4.4. the properties of the EQMC procedure

In this section, we list some basic properties of the EQMC procedure.

Theorem 4.2. If $\text{EQMC}(\mathcal{F})$ returns $\mathcal{I}$, then $\mathcal{I}$ is an eq-model of $\mathcal{F}$.

Proof. By definition of Deduce $(C_4)$, and by induction of the number of applications of the rules, we have for each iteration and for all $(\mathcal{F}', \mathcal{I}) \in \mathcal{S}$: $\mathcal{I} \wedge \mathcal{F}' \Rightarrow \mathcal{F}$. If $\text{EQMC}(\mathcal{F})$ returns $\mathcal{I}$, we have $\mathcal{I} \wedge \top \Rightarrow \mathcal{F}$, i.e. $\mathcal{I} \models \mathcal{F}$. $\square$

The following theorem proves that our method builds models for each formula of the $\mathcal{C}_\text{eq-model}$ class. The proof of this result needs the following lemmata.

Lemma 4.3. Any fair application of $\text{BuildMod}_1$ terminates.

Proof. It is immediate since the number of distinct $c$-literals in $\mathcal{D}$ is finite. Indeed, assume that there exists an infinite fair derivation. Since the application of the rules is fair and since $\mathcal{D}$ is finite, the decomposition rule will be applied on each literal in the representation set. Then, after a finite number of applications of the rules, all the couples in $\mathcal{S}$ will be of the form $(\mathcal{F}', \mathcal{I})$, where $\mathcal{I}$ is total. By applying the Deduce rule, we obtain only couples either of the form $(\bot, \mathcal{I})$, or of the form $(\top, \mathcal{I})$ (by $(C_1)$). $\square$

Lemma 4.4. If $\mathcal{F}$ admits a $\mathcal{D}$-compatible eq-model, then $\text{BuildMod}_1(\mathcal{F}, \mathcal{D})$ returns a model of $\mathcal{F}$.
Proof. Assume that $F$ admits a $D$-compatible eq-model. Then we show, by induction on the number of rules applications in the BuildMod$_1$ procedure, and by the definition of Deduce, that there exist $(F', I) \in S$, and an $J^*_F(D)$-compatible interpretation validating $F'$.

Indeed:

— The property is true for the initial set of couples, since $F$ admits a $D$-compatible model and $(F, \emptyset) \in S$.
— Assume that the property is true for $n$ iterations. Then there exists $(F', I) \in S$, and an interpretation $J$ $J^*_F(D)$-compatible validating $F'$. If the deduction rule is applied, the property is preserved, by (C2). If the cleaning rules is applied, then the property is trivially preserved. If the decomposition rule is applied, we have $P \in D$ hence $J^*_F(P) \in J^*_F(D)$. Hence either $J^*_F(P) \in J$ or $\neg J^*_F(P) \in J$. Assume for example that $P' = J^*_F(P) \in J$. Then $(F \wedge I \cup \{P'\})$ has a $J^*_{F \cup \{P'\}}(D)$-compatible eq-model. Hence the property is satisfied for $n+1$ applications of the rules.

Therefore, if BuildMod$_1$ terminates, $S$ must contain a couple of the form $(\top, I)$. □

Theorem 4.3. If $F$ has an eq-model, then any fair application of BuildMod$_1(F)$ returns an eq-model of $F$.

Proof. Let $F$ a formula of $C_{eq}$-model. The application of the rules in GR is fair, hence we obtain (after a finite number of applications) a set $D$ such that $F$ has a $D$-compatible model. By Lemma 4.4, we deduce that the procedure BuildMod$_1$ returns a model of $F$. □

5. The Choice of the Procedure Deduce

It remains to specify the choice of the procedure Deduce. Obviously, any procedure satisfying the specifications can be used.

In this section, we assume that the formula $F$ considered is a set of $c$-clauses. We briefly recall our calculus RAMC (Caferra and Zabel, 1992; Peltier, 1997b), for simultaneous search for refutations and models, and we show this procedure is admissible, i.e. that it satisfies the above conditions. This allows us to combine in a very natural way this method with the enumeration techniques presented in Section 4.3.

5.1. the RAMC procedure

The RAMC procedure (Caferra and Zabel, 1992; Bourely et al., 1994; Peltier, 1997b) is an extension of the resolution method. Instead of clauses, we consider constrained clauses. The standard inference rules (resolution, factorization), and contraction rules (subsumption, elimination of tautologies) can be adapted in a very natural way to constrained clauses. Then we define new rules called disinference rules, that are not inference rules in the usual sense.

We present some representative rules of this method. They belong to one or the other of two categories: the refutation rules (or rc-rules) and the model building rules (or mc-rules or dis-inference rules).
Refutation rules.

The binary c-resolution
Let \( \neg P(t) \lor c'_1 : X \) and \( P(\overline{t}) \lor c'_2 : Y \) be two c-clauses \( c_1 \) and \( c_2 \). The rule of binary c-resolution (abbreviated bc-resolution) on \( c_1 \) and \( c_2 \) upon \( \neg P(t) \) and \( P(\overline{t}) \) is defined as follows:

\[
\begin{array}{c}
\neg P(t) \lor c'_1 : X \\
\vdash c'_1 \lor c'_2 : X \land Y \land \overline{t} = \overline{t} \\
\end{array}
\]

The binary c-factorization
The binary c-factorization (abbreviated bc-factorization) of the c-clause \( c = [P(t) \lor c' : X] \) upon \( P(t) \) and \( P(\overline{t}) \) is defined as follows:

\[
\begin{array}{c}
P(t) \lor P(\overline{t}) \lor c' : X \\
\vdash P(\overline{t}) \lor c' : X \land \overline{t} = \overline{t} \\
\end{array}
\]

In Caferra and Zabel (1992) it is proved that the bc-resolution and the bc-factorization are sound and refutationally complete.

Remark 5.1. Rules similar to c-resolution and c-factorization appeared also in Kirchner et al. (1990).

Model building (or dis-inference) rules.

The model building rules aim at building a model of the initial set of c-clauses. We present some of them here.

The unit bc-disresolution
The rule of unit bc-disresolution computes constraints in order to prevent application of bc-resolution between a c-clause \( c_2 \) and a unit c-clause \( c_1 \).

Let \( c_1 : [\neg P(t) : X] \) be a unit c-clause and \( c_2 : [P(\overline{t}) \lor c'_2 : Y] \) be a c-clause: the rule of unit bc-disresolution (or bc-disresolution) on \( c_2 \) with \( c_1 \) upon \( P(t) \) and \( P(\overline{t}) \) is defined as follows (where \( x = \text{Var}(X) \cup \text{Var}(\neg P(t)) \)):

\[
\begin{array}{c}
\neg P(t) : X \\
\vdash P(\overline{t}) \lor c'_2 : Y \land \forall x. [\neg X \lor \overline{t} = \overline{t}] \\
\end{array}
\]

This rule illustrates clearly the usefulness of constraints: it is very easy to state, as a constraint, the conditions allowing to discard particular ground instances as explained above.

The unit bc-dissubsumption
The unit bc-dissubsumption rule computes constraints preventing a c-clause \( c_1 \) from being subsumed by a unit c-clause \( c_2 \). It is defined formally as follows (where \( x = \text{Var}(X) \cup \text{Var}(P(t)) \)):

\[
\begin{array}{c}
c_1 : [P(t) \lor c' : Y] \\
c_2 : [P(\overline{t}) : X] \\
c_3 : [P(\overline{t}) \lor c' : Y \land \forall x. [\neg X \lor \overline{t} = \overline{t}] \\
\end{array}
\]

Generating Pure Literals: the GPL rule
A literal \( L \) is said to be pure in a set of c-clauses \( S \) if its complementary (i.e. \( \neg L \)) does not appear in clauses of \( S \). Obviously, if \( L \) is pure in \( S \) and \( S \) is satisfiable, then \( S \cup \{ L \} \)
is satisfiable. The aim of the GPL rule is to generate such pure literals into the set of c-clauses. For doing that, it computes constraints preventing the application of binary resolution between \( L \) and any complementary literal in \( S \).

The GPL rule is defined as follows:

\[
\frac{\{P(t) \lor c' : X\} \quad S}{\{P(t) : X_{\text{pure}}\}}
\]

where \( X_{\text{pure}} = \bigwedge_{\forall y. \neg P(s) \lor r : Y \in S} \neg P(y) \lor r : Y \). where \( y \) are all the variables in \( \text{Var}(\{P(s) \lor r : Y \}) \).

We denote by \( DSub(S_1, S_2) \) the set obtained by applying the bc-dissubsumption rule to \( S_2 \) using c-clauses in \( S_1 \). We denote by \( N_{\text{RAMC}}(S) \) the set of c-clauses obtained by applying (as long as possible) the factorization, constraint simplification and bc-dissubsumption rules on \( S \).

Finally, Unit\((S)\) is the set of unit c-clauses in \( S \).

**Notation 5.1.** For any set of c-clauses \( S \), \( \text{RAMC}(S) \) denotes a set of c-clauses obtained by applying (as long as possible) the factorization, constraint simplification and bc-dissubsumption rules on \( S \).

**5.2. admissibility of the RAMC procedure**

**Notation 5.2.** For all sets of c-clauses \( S \), we denote by \( \text{decompose}(S) \) the following couple of sets of c-clauses.

\[
\text{decompose}(S) = (S \setminus \text{Unit}(S), \text{Unit}(S))
\]

if \( \text{Unit}(S) \) is satisfiable, \((\bot, \emptyset)\) otherwise.

**Theorem 5.1.** The procedure \( \text{Deduce}_{\text{c-clausal}}(S, \mathcal{I}) \) defined by:

\[
\text{Deduce}_{\text{c-clausal}}(S, \mathcal{I}) = \text{decompose}(\text{RAMC}(N_{\text{RAMC}}(S \cup \mathcal{I})))
\]

is admissible.

**Proof.** Let \( \mathcal{I} \) be a total eq-interpretation and \( S' = \text{RAMC}(S \cup \mathcal{I}) \).

If \( \mathcal{I} \models S \), we have \( DSub(\mathcal{I}, S) = \emptyset \). Otherwise \( \mathcal{I} \not\models S \), hence \( \square \) is deduced from \( S \) by c-resolution.

— Assume that there exists a \( \mathcal{J}_{\mathcal{I}}(\mathcal{D})\)-compatible eq-model \( \mathcal{J} \) of \( S \). We prove that there exists a \( \mathcal{J}_{\mathcal{I}}(\mathcal{D})\)-compatible eq-model of \( S' \). Two cases must be distinguished.

If the considered rule preserves the equivalence, the proof is immediate. Therefore, it suffices to consider the case of the GPL rule. Let \( E \) be a set deduced from \( S \) by the GPL rule. Let \( J' = J_E(J) \cup E \). Let \( C \) be a ground instance of a clause in \( S \). \( J \models C \) hence there exists \( L \in J \) such that \( L \in C \). If \( \neg L \in E \), then we have \( L \in J' \), hence \( J' \models C \). Otherwise, by definition of Gpl, \( E \models C \) hence \( J' \models C \). It remains to prove that \( J' \) is \( \mathcal{J}_{\mathcal{I}}(\mathcal{D})\)-compatible. Let \( L_1, L_2 \) be two ground literals belonging to the same literal of \( \mathcal{J}_{\mathcal{I}}(\mathcal{D}) \). Then \( L_1, L_2 \) does not occur in \( \neg E \) (since \( \mathcal{I} \)).

\(^1\)It is easy to see that the non-deterministic application of these rules terminates (Peltier, 1997b).
contains $E$, by definition). Hence $J(L_i) = J'(L_i) \ (i = 1, 2)$. $J$ is $\mathcal{J}_T(D)$ compatible. Moreover, $L_1$ and $L_2$ does not occur in $\neg I$ (since $I \subseteq I'$). Hence $J(L_1) = J(L_2)$ and $J'(L_1) = J'(L_2)$. Therefore, $J'$ is $\mathcal{J}_T(J)$-compatible.}

6. Example

The following example, deliberately simple, combines the use of the EQMC procedure with the RAMC method.

Example 6.1. Let $S$ be the following set of $c$-clauses, adapted from Bourely et al. (1994) (problem SYN303-1 of the Tptp library, see Suttner and Sutcliffe (1996)).

\[
\begin{align*}
[P(x,y) \lor \neg P(f(x),f(y)) : \top] \\
[\neg P(x,y) \lor P(f(x),f(y)) : \top] \\
[\neg P(a,f(x)) : \top] \\
[\neg P(f(x),a) : \top] \\
[P(a,a) : \top].
\end{align*}
\]

Let $D_0 = \{P(x,y)\}$. The application of the decomposition rule on $D_0$ leads to the two sets of $c$-clauses: $S \cup \{P(x,y)\}$ and $S \cup \{\neg P(x,y)\}$. They are obviously unsatisfiable (it suffices to apply the resolution rule in order to obtain a contradiction). Hence the application of the procedure BuildMod returns no model found. Hence, we have to apply the system GR in order to obtain a representation set allowing us to represent a larger class of eq-interpretations. We apply the GR rules on the terms $x,y$ in $P(x,y)$. We obtain

\[
\{P(x,x), [P(x,y) : x \neq y]\}.
\]

Now we apply the decomposition rule on $\{P(x,y)\}$, which produces the sets $S \cup \{P(x,y)\}$ and $S \cup \{\neg P(x,y)\}$. $S \cup \{\neg P(x,y)\}$ contains two contradictory $c$-literals $P(a,a)$ and $\neg P(x,y)$, hence the procedure Deduce($S \cup \{\neg P(x,y)\}$) returns \bot. Applying the Deduce rule on $S \cup \{P(x,y)\}$ gives (after simplifying by bc-dissubsumption and constraint resolution).

\[
\begin{align*}
[P(x,y) \lor \neg P(f(x),f(y)) : x \neq y] \\
[\neg P(x,y) \lor P(f(x),f(y)) : x \neq y] \\
[\neg P(a,f(x)) : \top] \\
[\neg P(f(x),a) : \top] \\
[P(x,x) : \top].
\end{align*}
\]

Now we can apply the GMPL rules, giving the $c$-literal

\[
\{[\neg P(x,y) : x \neq y]\}.
\]

This literal together with the literal $P(x,x)$ gives a model of $S$:

\[
\{P(x,x), [\neg P(x,y) : x \neq y]\}.
\]

Remark 6.1. Though very simple, Example 6.1 is interesting because $S$ does not have
any finite model. Moreover, to the best of our knowledge, no other published model building procedure can build a model for this problem (hyper-resolution will not terminate, and ordering strategies will only detect that the problem is satisfiable and will not build explicitly a model of the set of clauses).

7. Some Strategies for the EQMC Method

Non-deterministic application of the proposed rules can be very costly. Hence, we have to propose some special techniques to reduce the search space. In particular, it is clear that the use of structure-sharing techniques for representing terms and clauses is essential. It allows us to avoid representing twice the same clause, if it occurs in two distinct branches. Moreover, it also allows to apply rules simultaneously in each branch containing the parent c-clauses. This is very important in order to reduce redundancy.

Strategies are necessary in order to guide the application of the rules, and to improve the efficiency of the EQMC method. We also need to propose criteria for guiding (1) the choice of the rule to be applied (deduction or decomposition), (2) the choice of the literal on which to apply the deduction rules and (3) the choice of the representation set (i.e. for guiding application of the GR rules).

7.1. The Order of Application of the Rules

First, it is necessary to give some criteria in order to guide the choice of the rules decomposition, cleaning, and deduction. For practical reasons, it is preferable to apply first the cleaning and deduction rules, because they do not increase the number of pairs to be considered, unlike the decomposition rule. Obviously this strategy is not complete, since we can indefinitely apply the Deduce rules. Therefore we have to fix some limits on the application of this procedure, for example, fixing bounds on the computing time or on the number of applications of the rule in order to preserve the completeness of the method.

7.2. Choice of the c-literal for the Decomposition Rule

It is clear that any literal \( P \in D \) such that \( \text{Deduce}(\mathcal{F}, I \cup I_F(P)) \) is of the form \((\bot, I')\) should be chosen as soon as possible for applying the decomposition rule. Indeed, one of the two pairs generated by the decomposition rule will be immediately eliminated by using the cleaning rule (hence the number of branches will not increase). Therefore, it is interesting to identify, before applying the decomposition rule, which are the c-literals that are incompatible with a formulae \( \mathcal{F} \) in \( S \), and apply the decomposition rule on these literals (which can reduce significantly the search space). More generally, it is also possible to compute the sets of unit c-clauses that are incompatible with the set of formulae and to choose the c-literal occurring in sets of minimal size.

7.3. A Restriction Strategy for the GR Rules

The choice of the application of the rules \( \text{GR}_\infty \) and \( \text{GR}_\Sigma \) is crucial, because the non-deterministic application of these rules increase very quickly the size of the representation set. It is necessary to define some criteria in order to guide the choice of the literal and the choice of the position in which the rules will be applied. A possible strategy is briefly
described below. It is based on the use of the information deduced from the procedure **Deduce** and on the analysis of the **refutation** of the formula. For the sake of clarity, we assume that the RAMC procedure is used as a deductive procedure (though the proposed method can be easily extend to other procedures).

**Notation 7.1.** Let \( D \) be a representation set. Let \( F \) be a formula such that **BuildMod**\(_1\) \((F, D) = \) **no model found**. Let \( \{[[\Box : X_i]]/1 \leq i \leq n\} \) be the set of empty c-clauses deduced during the application of the **BuildMod**\(_1\) procedure. We denote by \( Y_{F, D} \) the following formula.

\[
Y_{F, D} = \bigwedge_{i=1}^{k} \exists \pi. X_i \sigma_i
\]

where the \( \sigma_i \) are renaming of variables from \( D \) by new variables, \( \pi \) are the variables of \( F \).

**Remark 7.1.** The formula \( Y_{F, D} \) can be seen as an explanation of the fact that \( D \) is incompatible with \( F \).

The **GR**\(_\Sigma\) rule.

**Notation 7.2.** Let \( x \) be a variable in \( D \). Let \( X^x \) be a formula obtained by replacing each occurrence of \( x \) in \( X \) (and each occurrence of the variables obtained from \( x \) by renaming of the c-clauses in \( D \)) by a (unique) variable \( y \).

If there exists a term \( t \) such that \( Y_{Y, F, D} \{y \rightarrow t\} \) is satisfiable, then it is useless to apply the rule **GR**\(_\Sigma\) on \( y \). Indeed, it is clear that each set of literals obtained by applying the **GR**\(_\Sigma\) rule on \( y \) contains at least one literal incompatible with the initial formula.

**Example 7.1.** Let \( F \) be a set of c-clauses containing the following set of c-clauses

\[
\{P(u, u), \neg P(u, f(u)), Q(b, f(v)), \neg Q(a, f(a))\}
\]

Let \( D = \{P(x, x), [P(x, y) : x \neq y], Q(x, x), [Q(x, y) : x \neq y]\} \). It is clear that \( F \) does not have any \( D \)-compatible interpretation. Indeed, the Decomposition rule adds either the c-clause \([Q(x, y) : x \neq y]\) (which contradicts the c-literal \( \neg Q(a, f(a)) \)) or \([-Q(x, y) : x \neq y]\) which contradicts \( Q(b, f(v)) \).

The c-literals \([Q(x, y) : x \neq y]\) and \([-Q(x, y) : x \neq y]\) are incompatible with \( F \). We obtain by c-resolution the following set of c-clauses.

\[
[\Box : x = b \land y = f(v) \land x \neq y]
\]

and (after renaming \( \{x \rightarrow x', y \rightarrow y'\}\)):

\[
[\Box : x' = a \land y' = f(a) \land x' \neq y']
\]

We have \( Y_{Y, F, D} = (x = b \land y = f(v) \land x \neq y) \land (x' = a \land y' = f(a) \land x' \neq y') \). We replace \( y \) and \( y' \) by a new variable \( y'' \). We obtain: \( Y_{Y, F, D} = \{(x = b \land y'' = f(v) \land x \neq y'') \land (x' = a \land y'' = f(a) \land x' \neq y'')\} \). Here \( Y_{Y, F, D} \) is satisfiable, hence the **GR**\(_\Sigma\) rule is not applicable on \( y \). However, if we replace \( x \) and \( x' \) by a new formula \( x'' \) in the formula \( Y_{F, D} \), the obtained formula \( Y_{Y, F, D} = (x'' = b \land y = f(v) \land x'' \neq y) \land (x'' = a \land y' = f(a) \land x'' \neq y'') \) is unsatisfiable.
Hence GR$_\Sigma$ can only be applied on the literal $[Q(x, y) : x \neq y]$ and on the variable $x$, which leads to the set:

$$D' = \{P(x, x), [P(x, y) : x \neq y], Q(x, x), [Q(a, y) : a \neq y], [Q(b, y) : b \neq y], [Q(f(u), y) : y \neq f(u)]\}.$$ 

By applying the Decomposition rule, we obtain the following model:

$$\{P(x, x), [\neg P(x, y) : x \neq y], [\neg Q(a, y) : a \neq y], [Q(b, y) : b \neq y]\}.$$ 

This strategy gives a criterion for guiding the choice of the variable on which the GR$_\Sigma$ rule can be applied.

The rule GR$_\Sigma$.

A similar strategy can be defined for guiding the application of the rule GR$_\Sigma$. Let $t$ and $s$ be two terms. We denote by $\mathcal{X}^+(t, s)$ the constraint: $\bigwedge_{i=1}^k t\sigma_i = s\sigma_i$ and by $\mathcal{X}^-(t, s)$ the constraint $\bigwedge_{i=1}^k t\sigma_i \neq s\sigma_i$. We apply the GR$_\Sigma$ rule iff $\mathcal{Y}_F, D \land \mathcal{X}^+(t, s)$ and $\mathcal{Y}_F, D \land \mathcal{X}^-(t, s)$ are simultaneously unsatisfiable.

We provide below a formal definition of the algorithm generating new representation sets using the above strategy.

**Procedure** GR$_{\text{restricted}}$

**INPUT**
- a set of c-clauses $F$
- a formula
- a representation set $D$

**OUTPUT**
- a new representation set $D'$
- or a message *no rule can be applied*

**begin**

$$\mathcal{Y} := \mathcal{Y}_F, D$$

$$\mathcal{Y}_F, D = V_{\text{ar}}(F)$$

**while** $\mathcal{Y} \not\equiv \top$ and there exists an applicable rule $\rho$ in GR

**choose** an applicable rule $\rho \in \text{GR}$

if $\rho = \text{GR}_\Sigma$

then

**choose** a variable $y$ in $D$

if $\mathcal{Y}_F, D \equiv \perp$

then

**Apply** the GR$_\Sigma$ rule on the variable $y$ in $D$

**return** the obtained representation set

else

$\mathcal{Y} = \mathcal{Y}_F, D$

else

% $\rho = \text{GR}_\Sigma$

**choose** two terms $s, t$ in a literal in $D$

if $(\mathcal{Y}_F, D \land \mathcal{X}^+(s, t) \equiv \perp) \land (\mathcal{Y}_F, D \land \mathcal{X}^-(s, t) \equiv \perp)$

then

**Apply** the GR$_\Sigma$ rule on the terms $s, t$ in $D$

**return** the obtained representation set

else

$\mathcal{Y} = (\mathcal{Y}_F, D \land \mathcal{X}^+(s, t)) \lor (\mathcal{Y}_F, D \land \mathcal{X}^-(s, t))$
It remains to prove that the method is still complete when this strategy is used to prune the search space. The proof is based on the following lemma.

**Lemma 7.1.** Let \( \mathcal{F} \) be a set of clauses and let \( \mathcal{D} \) be a representation set. Assume that \( \mathcal{D} \) is irreducible w.r.t. the GR restricted algorithm applied on terms at position lower than \( n \). Then for any \( n \)-maximal representation set \( \mathcal{D}' \) and for any \( \mathcal{D}' \)-compatible eq-model \( \mathcal{M} \) of \( \mathcal{F} \), \( \mathcal{M} \) is \( \mathcal{D} \)-compatible.

**Proof.** Assume that there exists an \( n \)-maximal representation set \( \mathcal{D}' \) and a \( \mathcal{D}' \)-compatible model \( \mathcal{M} \) of \( \mathcal{F} \) such that \( \mathcal{M} \) is not \( \mathcal{D} \)-compatible.

If \( \text{GR}_{\mathcal{D}} \) returns **no rule found**, then there exists a set of variables \( x_1, \ldots, x_m \) occurring in \( \mathcal{D} \) and a set of pairs of terms \( \{(s_i, t_i) | i \leq m'\} \) in \( \mathcal{D} \) such that the formula:
\[
\mathcal{Y}_{\mathcal{F}, \mathcal{D}} (x_1, \ldots, x_m) \land \mathcal{X}^+(s_1, t_1) \land \cdots \land \mathcal{X}^+(s_{m'}, t_{m'})
\]

is satisfiable.

By Theorem 4.1, there exist a sequence of application \( \mathcal{D} \rightarrow_{\rho_1} \mathcal{D}_1 \rightarrow \cdots \rightarrow_{\rho_k} \mathcal{D}_k \) of the rules in \( \mathcal{G} \) such that \( \mathcal{D}_k \) is \( n \)-maximal. Moreover, these rules are applied at position of depth lower than \( n \).

We prove, by induction on \( k \), that this is impossible.

If \( k = 0 \), the proof is immediate.

If \( k > 0 \), we distinguish two cases.

1. \( \rho_1 = \text{GR}_{\mathcal{D}_1} \). \( \rho_1 \) is applied on a variable \( y \) occurring in a literal \( [L : \mathcal{X}] \) in \( \mathcal{D} \).

Let \( \delta \) be a function mapping each \( c \)-literal \( P \) in \( \mathcal{D} \) to \( \{P, \neg P\} \). By definition of \( \mathcal{Y}_{\mathcal{F}, \mathcal{D}} \) there exists a conjunct \( \mathcal{Z} \) occurring in the conjunction \( \mathcal{Y}_{\mathcal{F}, \mathcal{D}} \) and \( k \) renaming \( \sigma_1, \ldots, \sigma_k \) of the variables in \( \mathcal{D} \) such that \( [\varnothing : \mathcal{Z}] \) can be obtained from \( S = \delta(\mathcal{D}) \sigma_1 \land \delta(\mathcal{D}) \sigma_2 \land \mathcal{F} \) by applying \( \text{RAMC} \)'s rules. Let \( S' = \{[C : \mathcal{X} \land \mathcal{Z}] | [C : \mathcal{X}] \in S\} \). We know that the \( c \)-clause \( [\varnothing : \mathcal{Z}] \) can be derived from \( S \), therefore \( [\varnothing : \mathcal{Z} \land \mathcal{Z}'] \) can be derived from \( S' \).

Since \( \rho_1 \) is applicable on \( \mathcal{D} \) on a variable occurring at a position of depth lower than \( n \), \( y \) must occur in \( \{x_1, \ldots, x_m\} \). Hence \( \mathcal{Z}' \) is satisfiable. Therefore \( S' \) is unsatisfiable. Let \( L_1, \ldots, L_n \) be the literals obtained from \( [L : \mathcal{X}] \) by applying the \( GR_{\mathcal{D}_2} \) rule on the variable \( y \). Since \( L_1 \land \cdots \land L_n \equiv L \), there exists \( i \) such that
\[
[L : \mathcal{Z}'] \subseteq L_i.
\]

Therefore, the set of \( c \)-clauses \( S'' = \{L_i\} \cup \{S' \setminus \{L : \mathcal{X} \land \mathcal{Z}'\}\} \) is unsatisfiable. Let \( \delta' \) be any mapping of \( \mathcal{D} \cup \{L_1, \ldots, L_n\} \setminus \{[L : \mathcal{X}]\} \). We have \( S'' \subseteq \delta' \mathcal{D}' \), hence \( \delta' \mathcal{D}' \) is unsatisfiable. Moreover, there exists a derivation using \( \text{RAMC} \)'s rules from \( S'' \cup \delta' \mathcal{D} \) to \( [\varnothing : \mathcal{Z}'] \). \( \mathcal{D}' \) can be transformed into a \( n \)-maximal representation set by a derivation of length \( k - 1 \). By the induction hypothesis, this is impossible.

2. \( \rho_1 = \text{GR}_{\mathcal{M}} \). The proof is similar. □

**Theorem 7.1.** The procedure EQMC restricted by the above strategy is complete w.r.t. the class of eq-interpretations (i.e. if \( \mathcal{F} \) has an eq-model, then EQMC(\( \mathcal{F} \)) returns a model of \( \mathcal{F} \)).
Proof. The proof follows directly from Lemma 7.1 and Theorem 4.3. □

8. Extension to First-order (Non-clausal) Formulae

In the previous sections, only formulae in clausal normal form were considered. As usual, this was implicitly justified by the fact that any formula \( F \) can be transformed into a set of \( c \)-clauses \( S = \text{clause}(F) \) such that:

(1) \( S \) satisfiable \( \iff \) \( F \) satisfiable;
(2) if \( M \) is a model of \( S \), then \( M \models F \).

From a purely deductive point of view we can always consider sets of \( c \)-clauses instead of first-order formulae, because this transformation preserves the satisfiability of the formula. Moreover, it is possible to use renaming (Plaisted and Greenbaum, 1986; Boy de la Tour, 1992; Egly and Rath, 1996) in order to avoid increasing the size of the formula. From a model building point of view, clausal form transformation is not innocuous, because it does not preserve the models of the formula. Due to the skolemization procedure, there can exist interpretations that are models of the initial formula, and not of the corresponding clausal form. Worse, it is possible that a formula \( F \), having an eq-model \( M \), is transformed into a set of \( c \)-clauses \( S \) with no eq-model. Indeed, skolemization reduces the number of possible models.

Example 8.1. Consider, for example, the following formula.

\[
P(a, a) \land \neg P(b, a) \land \forall x, x'. \exists y. P(x', x) \iff \neg P(x', y)
\]

\( F \) admits the following eq-model (on the signature \( \Sigma = \{a, b\} \)):

\[
\{P(a, a), \neg P(a, b), \neg P(b, a), P(b, b)\}
\]

The corresponding clausal normal form of \( F \) is the following.

\[
\begin{align*}
\{ & [P(a, a) : \top] \\
& [\neg P(a, b) : \top] \\
& [\neg P(x', x) \lor \neg P(x', f(x', x)) : \top] \\
& [P(x', x) \lor P(x', f(x', x)) : \top].
\end{align*}
\]

This set of \( c \)-clauses does not have any eq-model (see, for example, Klingenbeck, 1996).

Hence, transforming the problem into clausal normal form can in some cases prevent the construction of the model. In order to solve this problem, we propose to generalize the method EQMC to non-clausal first-order formulae. The only thing we have to do is to provide a new definition of the procedure \textbf{Deduce}. Indeed, the RAMC method cannot be applied since it only deals with formulae in clausal normal form. The idea is the following: instead of using the method RAMC, we define a new operator denoted by \text{Normalize}\(_y(x)\), allowing us to simplify a first-order formula in a partial interpretation. This operator will be defined by using the formulae \( \phi^+ \) and \( \phi^- \) defined in Section 3.

More precisely:
Definition 8.1. Let $F$ be a formula, $I$ a partial eq-interpretation. We denote by $\text{Normalize}_I(F)$ the formula obtained by replacing each literal $A$ in $F$ by $\neg \phi - I(A) \land (\phi + I(A) \lor A)$.

Theorem 8.1. Let $F$ be a formula, $I$ a partial eq-interpretation. We have

$$I \models F \iff I \models \text{Normalize}_I(F).$$

Moreover, if $I$ is total, $\text{Normalize}_{FI}$ is purely equational.

Now, we can give the definition of the procedure Deduce.

Theorem 8.2. The procedure $\text{Deduce}_{1\text{st order}}$ defined by:

$$\text{Deduce}_{1\text{st order}}(F, I) = (\text{Normalize}_I(F), I)$$

is admissible.

Proof. The proof is immediate.$\square$

Since the procedure is admissible, all the properties of the method, as stated in Section 4.3, (and in particular the completeness w.r.t. the $\mathcal{C}_{\text{eq-model}}$ class) are preserved.

9. Undecidability of the Problem “$F \in \mathcal{C}_{\text{eq-model}}$”

As EQMC is complete w.r.t. eq-models, in order to show that it is a complete for satisfiability detection for a given class of formulae, it suffices to show that all satisfiable formulae in this class have an eq-model.\footnote{Obviously since first-order logic is undecidable, no algorithm can be complete for satisfiability detection for all formulae.} Hence a question naturally arises: does there exist an algorithm deciding whether a set of $c$-clauses has an eq-model? Such an algorithm would be of great practical interest since it would give a syntactical characterization of the class of formulae tractable by our approach. Unfortunately, this problem is undecidable as evidenced by the following theorem. On the other hand, this limitation provides evidence, in some sense, that the class of eq-models is a “reasonable size” one.

Theorem 9.1. The problem “$F \in \mathcal{C}_{\text{eq-model}}$?” is undecidable.

Proof. We reduce the problem “$F \in \mathcal{C}_{\text{eq-model}}$?” to the well known (undecidable) Post’s correspondence problem.

Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ be two sequences of strings on a given set of symbols $V$. Let $\Sigma = \{0, \text{succ}\} \cup V$.

For all strings $s$ we denote by $l(s)$ the length of $l$ and if $i \leq l(s)$, we denote by $s(i)$ the $i$th symbol in $s$.

We consider the following set of formulae.

(1) $P_1(0, 0)$.
(2) $P_2(0, 0)$. 
Proof. Immediate from Corollary 9.1. It is not possible to give a syntactic characterization of the \( \mathcal{C}_{eq-model} \) class.

**Proof.** Immediate from Theorem 9.1. \( \square \)
10. Interactive Model Building of a “Big” Formula

We show in this Section (see also Peltier, 1997a,b) how to use the EQMC method to build a model for a satisfiable formula introduced by Goldfarb in his proof of the unsolvability of the Gödel class with identity.

10.1. the Gödel class with identity

The Gödel class is the class of first-order formulae of the form: \( \forall x, y. \exists z_1, \ldots, z_n. M \), where \( M \) is a quantifier-free formula without function symbols. The Gödel class without identity (i.e. where \( M \) does not contain the equality predicate) has been proven to be decidable (Gödel, 1932a) and finitely controllable.\(^1\) In Goldfarb (1984), Goldfarb has proven that the Gödel class with identity (i.e. containing \( \equiv \) in \( M \)) is undecidable, thus refuting a conjecture by Gödel (who claimed that his proof of the finite controllability of the Gödel class can be extended to the case with equality (Gödel, 1932b)). Goldfarb exhibited a first-order formula belonging to the Gödel class with identity, but having no finite models. This formula can be used to encode undecidable problems.

10.2. the original problem

Goldfarb’s formula is noted \( F \) in the following.

\[
F : \forall x. \forall y. \exists z_0. H
\]

where \( H \) is the conjunction of the following formulae.

1. \( Z(x) \land Z(y) \rightarrow x = y \)
2. \( Z(z_0) \land \neg S(z_0, x) \land \bigwedge_{d=1,2} P_3(x, z_0) \land P_3(x, y) \rightarrow y = z_0 \)
3. \( \exists z. S(z, x) \)
4. \( \neg Z(x) \land x \neq y \rightarrow \exists z. (S(x, z) \land \neg S(y, z)) \)
5. \( \exists z. [N(x, z) \land (Q(x, y) \rightarrow Q(z, y)) \land (R_1(x, y) \rightarrow R_1(z, y)) \land (R_2(x, y) \rightarrow R_2(z, y))] \)
6. \( N(x, y) \rightarrow \exists z. (P_2(x, z) \land P_2(y, z)) \)
7. \( N(x, y) \rightarrow \exists w, u (P_1(x, w) \land S(u, w) \land P_1(y, u)) \)
8. \( S(x, y) \rightarrow \exists z. (Q(z, x) \land P_2(z, y) \land P_1(z, z_0)) \)
9. \( Q(x, y) \rightarrow \exists z. (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \)
10. \( \bigwedge_{d=1,2} P_3(x, y) \land \neg Z(y) \rightarrow \exists z, w. (P_3(x, z) \land P_3(z, x) \land P_1(z, z_0) \land S(y, w)) \)
11. \( \bigwedge_{d=1,2} R_d(x, y) \rightarrow \exists z, w. (P_1(x, z) \land S(w, z) \land (P_3(y, w) \rightarrow P_2(x, z))) \)

As recalled above, this formula is satisfiable, but does not have any finite model. Hence any enumeration based finite model builder will fail on this problem. Herbrand model builders (such as the RAMC method (Cafera and Zabel, 1992; Bourley et al., 1994) or the method by Fermüller and Leitsch (1996)) fail as well, due to the presence of equality and to the transformation into clausal form (which does not preserve the models).

The model building process is divided into two steps.

1. A simplifying step. It reduces significantly the search space. It is interactive.
2. A model building step. It has been done using the EQMC method in a purely automatic way.

\(^1\) A class of first-order formula \( \mathcal{C} \) is said to be finitely controllable iff for all satisfiable formula \( F \in \mathcal{C} \), \( F \) has a finite model.
10.3. Simplifying the Problem

We first apply a simplifying step, which aims at reducing the search space, by reducing the scope of the quantifiers occurring in the formula. For doing that, it is necessary to eliminate the existential quantifier $\exists z_0$. This is done using skolemization. All occurrences of $z_0$ are replaced by the term $f(x, y)$ where $f$ is a new function symbol. We obtain the formulae $\forall x, y, H'$, where $H'$ is the conjunction of the following formulae.

\[
\begin{align*}
&\exists z, S(z, x) \\
&\exists z, (N(x, z) \land (Q(x, y) \rightarrow Q(z, y)) \land ((R_1(x, y) \rightarrow R_1(z, y)) \land (R_2(x, y) \rightarrow R_2(z, y))) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (S(w, z) \land (P_1(x, z) \land P_1(z, f(x, y))) \\
&\exists z, (P_2(x, z) \land P_2(x, z)) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (S(w, z) \land (P_1(x, z) \land P_1(z, 0))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)))
\end{align*}
\]

In particular, we have:

\[
\begin{align*}
&\forall x, y, H' \\
&\exists z, (S(w, z) \land (P_1(x, z) \land P_1(z, 0))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)))
\end{align*}
\]

We obtain (by resolution) the clause:

\[
\begin{align*}
&f(x, y) = f(x’, y’). \\
&\exists z, (S(w, z) \land (P_1(x, z) \land P_1(z, 0))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)))
\end{align*}
\]

\[f\] is therefore a constant function. Hence it is possible to introduce a new term of arity 0 (noted 0) and to replace all occurrences of $f(t, s)$ (for all terms $t, s$) by 0 (we use here the function introduction rule used for example by the theorem prover Otter (McCune, 1990)).

Remark 10.1. This step is not really necessary, but makes the obtained formula and the corresponding model much easier to understand.

Then we obtain the following set of formulae.

\[
\begin{align*}
&\exists z, S(z, x) \\
&\exists z, (N(x, z) \land (Q(x, y) \rightarrow Q(z, y)) \land ((R_1(x, y) \rightarrow R_1(z, y)) \land (R_2(x, y) \rightarrow R_2(z, y))) \\
&\exists z, (P_1(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z))) \\
&\exists z, (P_2(x, z) \land (S(y, z) \rightarrow P_2(x, z)))
\end{align*}
\]
We also use skolemization in order to eliminate the existential quantifier occurring in clause (3). A new functional symbol of arity 1 is introduced and we replace \( \exists z. S(z, x) \) by \( S(s(x), x) \).

Then the quantifier can be shifted into the formula.

We obtain:

\[
\begin{align*}
(1) & \quad \forall x, y. (Z(x) \land Z(y) \rightarrow x = y)) \\
(2) & \quad Z(0) \land \neg \forall x. S(0, x) \land \bigwedge_{\delta = 1, 2} \forall x, y. (P_3(x, 0) \land P_2(x, y) \rightarrow y = 0) \\
(3) & \quad \forall x, y. (\neg Z(x) \land x \neq y \rightarrow \exists z. (S(x, z) \land \neg S(y, z))) \\
(4) & \quad \forall x, y. (\exists z. [N(x, z) \land (Q(y, x) \rightarrow Q(z, y)) \land (R_1(x, y) \rightarrow R_1(z, y)) \land (R_2(x, y) \rightarrow R_2(z, y))]) \\
(5) & \quad \forall x, y. (\exists z. [N(x, z) \land (Q(y, x) \rightarrow Q(z, y)) \land (R_1(x, y) \rightarrow R_1(z, y)) \land (R_2(x, y) \rightarrow R_2(z, y))]) \\
(6) & \quad \forall x, y. (N(x, y) \rightarrow \exists z. (P_2(x, z) \land P_2(y, z))) \\
(7) & \quad \forall x, y. (N(x, y) \rightarrow \exists w, u. (P_1(x, w) \land S(u, w) \land P_1(y, u))) \\
(8) & \quad \forall x, y. (S(x, y) \rightarrow \exists z. (Q(z, x) \land P_2(z, y) \land P_1(z, 0))) \\
(9) & \quad \forall x, y. (Q(x, y) \rightarrow \exists z. (P_1(z, x) \land (S(y, z) \rightarrow P_2(x, z)))) \\
(10) & \quad \bigwedge_{\delta = 1, 2} \forall x, y. (P_3(x, y) \land \neg Z(y) \rightarrow \exists z, w. (R_3(z, x) \land P_2(z, x) \land P_1(z, 0) \land S(y, w))) \\
(11) & \quad \bigwedge_{\delta = 1, 2} \forall x, y. (R_3(x, y) \rightarrow \exists z, w. (P_1(z, x) \land S(w, z) \land (P_2(y, w) \rightarrow P_2(x, z))))
\end{align*}
\]

The conjunction of formulae (1)–(11) is denoted by \( F_{\text{simp}} \) in the following.

**SORTS INTRODUCTION**

Then we use a type inference algorithm in order to compute a set of sort symbols \( S \) and a function profile mapping each function and predicate symbols \( f \) to a profile of the form \( s_1, \ldots, s_n \rightarrow s \), such that \( F_{\text{simp}} \) is well typed. This operation preserves satisfiability and aims at reducing the search space.

We obtain the following profiles:

\[
S = \{S_1, S_2\}
\]

\[
s : \quad S_1 \rightarrow S_1
\]

\[
0 : \quad \rightarrow S_1
\]

\[
S : \quad S_1, S_1 \rightarrow \text{Boolean}
\]

\[
Z : \quad S_1 \rightarrow \text{Boolean}
\]

\[
R_1 : \quad S_2, S_2 \rightarrow \text{Boolean}
\]

\[
R_2 : \quad S_2, S_2 \rightarrow \text{Boolean}
\]

\[
Q : \quad S_2, S_1 \rightarrow \text{Boolean}
\]

\[
N : \quad S_2, S_2 \rightarrow \text{Boolean}
\]

\[
P_1 : \quad S_2, S_1 \rightarrow \text{Boolean}
\]

\[
P_2 : \quad S_2, S_1 \rightarrow \text{Boolean}
\]

**10.4. THE MODEL BUILDING PROCESS**

First of all we specify the domain of the interpretation, i.e., the set of functional symbols in the signature. This set obviously contains the function symbols occurring in \( F_{\text{simp}} \).

\[
\{0, s\}.
\]
Moreover, we also add a new function $g$ of profile

$$g : S_1, S_1 \rightarrow S_2.$$ 

$S_1$ is isomorphic to $\mathbb{N}$ and $S_2$ is isomorphic to $\mathbb{N}^2$.

**Remark 10.2.** The choice of this function $g$ is of course guided by an intuitive idea about the possible models of $F_{\text{simp}}$. This is the only part that really needs a human interaction (the other parts could be easily automated). The help of the user is essential here for specifying the domain of the interpretation, whereas the interpretation of the predicate symbols is obtained purely automatically.

Then we use EQMC in order to compute automatically the interpretation of predicate symbols of the signature. We obtain the following model.

$$Z(0) \text{ is true}$$
$$S(s(A), A) \text{ is true}$$
$$P_1(g(A, B), A) \text{ is true}$$
$$P_2(g(A, B), B) \text{ is true}$$
$$R_1(g(A, B), g(s(B), C)) \text{ is true}$$
$$R_2(g(A, B), g(C, s(B))) \text{ is true}$$
$$N(g(A, B), g(s(A), B)) \text{ is true}$$
$$Q(g(A, B), s(B)) \text{ is true}$$
$$Z(s(A)) \text{ is false}$$
$$S(A, B) \text{ is false if: } A \neq s(B)$$
$$P_1(g(A, B), C) \text{ is false if: } C \neq A$$
$$P_2(g(A, B), C) \text{ is false if: } C \neq B$$
$$R_1(g(A, B), g(C, D)) \text{ is false if: } C \neq s(B)$$
$$R_2(g(A, B), g(C, D)) \text{ is false if: } D \neq s(B)$$
$$N(g(A, B), g(C, D)) \text{ is false if: } C \neq s(A) \vee (B \neq D)$$
$$Q(g(A, B), C) \text{ is false if: } C \neq s(B)$$

Only slight human guidance is needed for this construction: the main part of the model building process (i.e. finding the interpretation of predicate symbols) is purely automatic. To the best of our knowledge, no other model builder is able to build models for this formula. The obtained model is isomorphic to the one given by Goldfarb.

### 11. Conclusion and Future Work

We have shown the flexibility of our deductive approach to model building by introducing a method for building models of first-order formulae. It combines deductive methods (using inference and disinference rules) with enumeration techniques, and is complete w.r.t. a particular class of models, called eq-models. There are many interesting classes of first-order formulae having an eq-model, those called Pvd, Occ1N, the Bernays–Schönfinkel class, the monadic class, etc. (see Peltier, 1997b, for details). We have shown that there cannot be any syntactic characterization of the class of formulae having an eq-model. The method has been implemented and allows us to build a model of a formula used by Goldfarb for proving the undecidability of the Gödel class with identity. This model has been built interactively, using the system EQMC ATINF. The model building
process requires some human guidance to be feasible: it is necessary to simplify the initial formula and to specify the domain of the model. This example is general enough to underline the main problems to be solved in the future in order to improve significantly the capabilities of the method proposed in this work. At present, we are mainly investigating two of them: finding criteria to choose the variables on which skolemization should be applied and finding strategies (or heuristics) to suggest the domain of the model. The main idea is that the system could use information deduced from a failure of the model enumeration process, in order to identify the cases in which new function symbols are needed. In case of failure, EQMCATINF would be able, instead of just changing the partition of the Herbrand base (as in Section 7), to modify the domain of the interpretations as well.

Acknowledgements

We thank the three referees for their detailed and useful suggestions and Ph. Jorrand for improving our English.

References

Combining enumeration and deductive techniques


Originally Received June 1998
Accepted June 1999