Extending Resolution for Model Construction*

Ricardo Caferra     Nicolas Zabel
Laboratoire d'Informatique Fondamentale et d'Intelligence Artificielle
IMAG-CNRS
46, Av. F. Viallet 38031 Grenoble — France
e-mail: caferra@lifia.imag.fr (uucp)

Keywords: Theorem Proving, Model Construction, Equational Problems, Unification, Disunification, Decision Procedures.

Abstract

A method is proposed to systematize the simultaneous search for a refutation and Herbrand models of a given conjecture. It is based on an extension of resolution using equational problems and the inference system included in the method is proved to be sound and refutational complete. For some classes of formulae the method is indeed a decision procedure. Some examples of model construction — including one for which other resolution based decision procedures fail to detect satisfiability — are developed in detail.

Models are built by constructing relations on Herbrand universe. The relationship between these models and finite ones is established. The class of these constructible relations is precisely characterized. Some of the rules introduced, in order to extend resolution, are essentially new. Their necessity in constructing models is proved. A brief comparison with existing methods which bear similarity with ours, either in the use of constraints (a particular case of equational problems) or in the search of a model, shows the originality of our proposal.

1 Introduction

Non-theoremhood detection, and more precisely counter-example construction, have deserved little attention in the field of Automated Deduction (in fact, insignificant with respect to efforts devoted to refutation or proof methods).

In the literature, there exist few works in which strategies have been incorporated to resolution and paramodulation in order to use it as a decision procedure [12, 16], but these strategies detect satisfiability by the impossibility of inferring new clauses and do

*This work has been supported by the MEDLAR BRA Esprit project and the PRC - IA - CRNS — MRT.
not build effectively models for satisfiable sentences. On the contrary, our method is based on setting up conditions avoiding new clauses to be generated.

At the same time, model construction with the help of automated theorem provers is considered as one of the most outstanding success in the field [2] and some remarkable work has been done in this direction (see e.g. [20, 18]). This research seems to be particularly important when using theorem provers as assistants in mathematical research [19].

Instead of a new strategy, we propose to intervene in the core of the inference rule. Binary resolution tries to identify conditions (substitutions) making identical (disregarding negation symbol) two literals. Our approach is to keep this research (for refutation), but also to consider conditions preventing two literals to be identical (disregarding negation symbol). The key for doing that is considering equational problems. Solutions of these equational problems, non necessarily explicitly computed, are coded apart and define a kind of “dynamic sorts” for n-tuples of variables. These “sorts” are successively refined in order to produce pure literals which are then used to define Herbrand models of satisfiable set of formulae. The new extended approach we adopt has mainly two interesting and deep consequences:

- Simultaneous search for refutation and models is introduced in a very natural way (see sections 3 and 4).
- We do not need to define a new notion of mgu adapted to our calculus (this feature has also structure-sharing implementational consequences).

## 2 Definitions and Fundamental Properties

In the following, FOL is an abbreviation for First Order Logic; \( \Sigma \) denotes the ranked alphabet of all functional symbols (we assume \( \Sigma \) contains at least a 0-ary function symbol), \( \Omega \) the ranked alphabet of all predicates symbols, \( V \) an infinite set of variables; \( \Sigma, V, \Omega \) share no element; a rank function named \textit{arity} is assumed.

\( \tau(V, \Sigma) \) denotes the set of terms whose variables are in \( V \subseteq V \), and their functional symbols in \( \Sigma \); if \( V \) is empty, we write \( \tau(\Sigma) \).

An equation in \( \tau(V, \Sigma) \) is the formula \( t = s \); a disequation in \( \tau(V, \Sigma) \) is the formula \( t \neq s \) where \( t, s \in \tau(V, \Sigma) \). \( T, \perp \) denote respectively the true, and the false formula. \( \overline{x} \) denotes a tuple of terms, its projections are noted \( x_i \). If \( \overline{x}, \overline{y} \) denote n-tuples, then \( \overline{x} = \overline{y} \) (\( \overline{x} \neq \overline{y} \)) is an abbreviation for \( \wedge_{i=1}^n x_i = y_i \) (resp. \( \vee_{i=1}^n x_i \neq y_i \)) We assume known the ordinary definitions used in the field (see e.g. [10, 15]).

In order to make this paper self contained, we need to recall some fundamental definitions in [7] (in a slight different formulation).

**Definition 2.1 (Equational Problems)**

- A \textit{system} is a pure equational, quantifier free formula in n.n.f.
- An \textit{equational problem} \( P \) is a rectified formula of the form \( \exists \overline{w}. \forall \overline{y}. M(\overline{w}, \overline{x}, \overline{y}) \) where \( M(\overline{w}, \overline{x}, \overline{y}) \) is a system, and the variables in \( \overline{x} \) are, obviously, the free variables in the formula.