From: AAAI-99 Proceedings. Copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Sacre : a Constraint Satisfaction Problem Based Theorem Prover

Jean-Michel Richer Jean-Jacques Chabrier
LIRSIA, Université de Bourgogne
9 Avenue Alain Savary, BP 47870, 21078 Dijon cedex, France
{richer,chabrier}@crid.u-bourgogne.fr

Abstract

The purpose of this paper is to present a new approach for solving first-order predicate logic problems stated in conjunctive normal form. We propose to combine resolution with the Constraint Satisfaction Problem (CSP) paradigm to prove the inconsistency and find a model of a problem. The resulting method benefits from resolution and constraint satisfaction techniques and seems very efficient when confronted to some problems of the CADE-13 competition.

Introduction

From a general point of view we can classify methods for solving first-order predicate calculus problems stated in conjunctive normal form, into two categories. The first one is *consistency searching* or *proof searching* and is syntax-oriented. It consists of raising a contradiction from a set of clauses by applying inference rules. For example, the theorem prover Otter (McCune 1994) uses *resolution, unit-resolution, hyperresolution*, while Setheo (Bayerl & Letz 1987) is based on *model elimination* (Loveland 1978).

The second one, *satisfiability checking*, also called *model finding*, is related to semantics and tries to find a model or a counterexample of a problem. In this last category we can draw a distinction between *saturation* and *extension* approaches. In the former case, we iteratively generate ground instantiations of the problem and test ground clause sets for unsatisfiability with a propositional calculus prover (Chu & Plaisted 1997). In the latter case, we try to build a model of the problem by assuming new ground facts. One of the first satisfiability approaches was the saturation approach of Gilmore (Gilmore 1960) which proved to be very inefficient. We believe it is because this kind of approach has to tackle with the whole Herbrand base while only a part of it is necessary.

Another kind of approach is the combined approach implemented in the theorem prover Satchmo (Manthey & Bry 1988) which uses syntactic and semantic features to solve problems. It can be qualified as an extension approach. However Satchmo is based on the *model generation* reasoning paradigm (Bry & Yahya 1996). Satchmo suffers from certain drawbacks. The first one is *range restriction* requiring that each head variable must occur in the body of a clause. The second drawback is the fact that Satchmo can sometimes choose a clause irrelevant to the current goal to be solved and thus causes unnecessary model candidate extensions. This may result in a potential explosion of the search space. Nevertheless, some improvements can be made, such as *relevancy testing* (Loveland, Reed, & Wilson 1995) to avoid unnecessary case splittings.

Apart from those semantic approaches, Finder (Slaney 1995) searches for finite models of first order theories presented as sets of clauses. Falcon (Zhang 1996), for which model generation is viewed as constraint satisfaction, constructs finite algebras from given equational axioms. Finite models are able to provide some kind of semantic guidance that helps refutation-based theorem provers find proofs more quickly (Slaney, Lusk, & McCune 1994).

It is also possible to combine *resolution* with rewrite techniques so as to guide the search and design more efficient inference rules, such as the *problem reduction format* (Loveland 1978) or the *simplified problem reduction format* (Plaisted 1982), that permits the deletion of unachievable subgoals, or its extension the *modified problem reduction format* (Plaisted 1988).

The key novelty introduced in this paper is the combination of *resolution* with the *Constraint Satisfaction Problem* (CSP) paradigm, so as to solve first-order predicate calculus problems, stated in conjunctive normal form. This combination is not fortuitous. First, consistency searching and model finding are both common problems related to logic and CSPs. Second, the CSP techniques have proved to be very powerful to solve large combinatorial problems by applying strategies and heuristics that help guide the search and improve the resolution process by efficiently pruning the search space. The resulting method, called *Sacre*[1] is based

Copyright ©1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

[1]for SAtisfaction de Contraintes et REsolution - Constraint satisfaction and resolution

on a unique forward chaining rule and combines constraint satisfaction heuristics and techniques together with theorem-proving techniques and is able to prove the inconsistency or find a model of a problem. It is, to our knowledge, the first attempt of this kind ever tried in this direction.

The paper is organized as follows : in section 2, we will set forth some basic definitions of constraint satisfaction problems. The next section is devoted to the use of CSPs in propositional calculus. Section 4 presents the *Sacre* approach. The last section exhibits some results for some of the problems of the CADE-13 competition.

Constraint satisfaction problems

The past 10 years have witnessed the development of efficient algorithms for solving Constraint Satisfaction Problems (CSPs). Examples of CSPs include propositional theorem proving, map coloring, planning and scheduling problems.

Definition 1 - CSP - *: A CSP (Montanari 1974) is traditionally defined by a set of variables $X = \{x_1, \ldots, x_n\}$ ranging over a finite set of domains $D = \{d_1, \ldots, d_n\}$ that need to satisfy a set of constraints $C = \{c_1, \ldots, c_m\}$ between variables.*

A constraint is *satisfied* if there exists an assignment of its variables such that the relationship between the variables holds. A CSP is *consistent* if there exists an assignment of $\mathcal{X}$ such that all the constraints of $\mathcal{C}$ are satisfied. Given a CSP we can check if there exists a solution (consistency checking), find a solution or all the solutions (model finding) or find an optimal solution for a given criterion (optimization problem).

Example 1- Consider the following CSP :

$$(CSP_1)\left\{\begin{array}{ll} \mathcal{X} = & \{x_1, x_2, x_3\}, \\ \mathcal{D} = & \{d_1, d_2, d_3\}, \text{ with} \\ & \quad d_1 = d_2 = d_3 = \{0, 1, 2\} \\ \mathcal{C} = & \{x_1 + x_2 = x_3\}, \end{array}\right.$$

This CSP is consistent and its solutions are :

$$\begin{array}{ll} S_1 = & \{(0,0,0),(0,1,1),(1,0,1), \\ & (1,1,2),(0,2,2),(2,0,2)\} \end{array}$$

Propositional calculus and CSPs

The satisfiability problem (SAT) in propositional calculus consists in determining whether there exists an assignment of the propositional variables of a set of clauses that renders the set satisfiable. The translation of a set of propositional clauses into a CSP is quite obvious. Propositional variables become the variables of the CSP and range over the boolean domain. Clauses of the form $p_1 \vee \ldots \vee p_k$ (where the p_is are literals) are translated into cardinality constraints (Van-Hentenryck & Deville 1991) $\#\langle\alpha, \beta, p_1, \ldots, p_k\rangle$ which state that at least α and at most β propositional variables must be instantiated to true. Not only does the cardinality constraint model more concisely problems but it also improve the efficiency of the resolution process (Chabrier, Juliard, & Chabrier 1995). For example, the following set of clauses :
$\{p \vee q \vee r, \neg p \vee \neg q, \neg p \vee \neg r, \neg q \vee \neg r\}$ can be represented by $\#\langle 1, 1, p, q, r\rangle$. Among the algorithms developed to solve the satisfiability problem, the methods relying on CSP techniques have proved to be far more efficient than the basic Davis and Putnam procedure (Davis & Putnam 1960). We can draw a distinction (Chabrier, Juliard, & Chabrier 1995) between systematic approaches like C-SAT (Dubois *et al.* 1993), non-systematic approaches such as GSAT (Selman, Levesque, & Mitchell 1991) and hybrid approaches like *Score* (Chabrier, Juliard, & Chabrier 1995).

Systematic approaches are complete. They rely on a backtrack algorithm that tries to find a solution by successively instantiating variables while constraints are satisfied. Non-systematic approaches start from an initial instantiation (also called a configuration) of the variables and try to make local changes to the configuration until a solution is found following heuristic criteria. This kind of approach is incomplete, for heuristics may cause the algorithm to be stuck in a local optimum, but tends to be more efficient than systematic approaches because they correspond to a more focused search. Finally, hybrid approaches start from an initial configuration of the variables generated for example by a Min-Conflict algorithm (Minton *et al.* 1992) and try to repair it using a backtrack algorithm to avoid testing the same configuration twice.

The *Sacre* approach

Score, that has proved to be fairly efficient for random SAT problems (3-SAT) and structured problems (Ramsey, Pigeon-hole) (Chabrier 1997), led us naturally therefore to venture out beyond the limits of propositional calculus. To some extent, *Sacre* can be considered as an attempt at extending *Score* to first-order logic.

A new original approach to theorem proving for first-order logic based on a constraint representation of predicate calculus problems was defined in (Richer & Chabrier 1997).

This approach originates from the observation that a logic problem in conjunctive normal form is able to be expressed as a special case of a CSP, that we call CSP^T,

Definition 2 - CSP^T - *A CSP^T is a kind of CSP defined to represent the set of terms T of a problem in logic expressed in conjunctive normal form.*

The main idea of our work was to transform a predicate calculus problem into a CSP^T and solve it using heuristic techniques related to the resolution of Constraint Satisfaction Problems and thus take advantage of the efficiency of CSP techniques.

The major stumbling block in trying to translate a set of clauses of the predicate calculus into a CSP concerns the representation of literals as variables ranging over sets of terms. To take into account the characteristic of CSP^T, we introduce the notations of Ψ-domain and Ψ-variable to help represent respectively the domains and variables of CSP^T. These notations rely on a membership interpretation of literals. For example, when we write $man(socrate)$ we mean that $socrate$ is a member of the set of men. In the same way, $\neg man(tweety)$ means that $tweety$ does not belong to the set of men but that the concept of man applies to $tweety$. We can then think of the concept of man in terms of two separate and complementary subsets (or Ψ-subdomains) that capture the notion of a boolean interpretation. A Ψ-domain is then the set composed of these two subsets (see fig. 1).

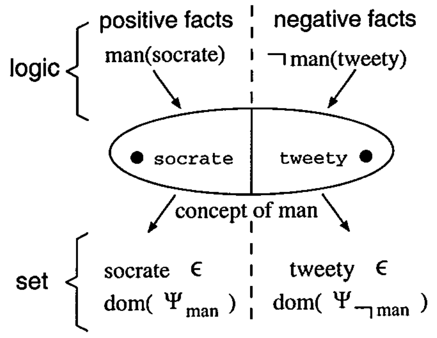

Figure 1: Notion of Ψ-domain

Furthermore, any element of a Ψ-domain can not belong to both subsets without raising an inconsistency. More precisely, if p is a predicate symbol we will note $dom(\Psi_p)$ and $dom(\Psi_{\neg p})$ the subsets (or Ψ-subdomains) respectively related to literals p and $\neg p$. The Ψ-domain of p is noted $dom(\Psi_{p,\neg p})$ and is such that

$$\begin{array}{rll} dom(\Psi_{p,\neg p}) = & < dom(\Psi_p), dom(\Psi_{\neg p}) > & \\ with & dom(\Psi_p) \cap dom(\Psi_{\neg p}) & = \emptyset \end{array}$$

where the notation $E =< A, B >$ expresses the fact that the set E is composed of two disjoint subsets A and B.

The notion and notation of Ψ-variable is then straightforward. A Ψ-variable represents the translation of a literal into a variable. Each literal p (resp. $\neg p$) is associated to a Ψ-variable, noted Ψ_p (resp. $\Psi_{\neg p}$) ranging over $dom(\Psi_{p,\neg p})$. Any literal $p(t)$ where t is a list of terms represents an occurrence of the variable Ψ_p for which the values that Ψ_p can be assigned to are restricted to values of $dom(\Psi_{p,\neg p})$ that unify with t and is noted $\Psi_p^{[t]}$. Provided with this formalism we can describe a linear time transformation of a predicate calculus problem stated in conjunctive normal form into a CSP^T for which unit clauses will form the Ψ-domains of the CSP and non-unit clauses will be transformed into cardinality constraints so as to turn clauses into constraints.

Definition 3 - Transformation into a CSP^T - *A first-order predicate calculus problem P stated in conjunctive normal form can be transformed into a CSP over the set of terms T of P (noted CSP^T), by applying the following rules :*

- *for each predicate p of P :*
  - *$\mathcal{X}$ is made up of the variables Ψ_p and $\Psi_{\neg p}$ that take their values in $dom(\Psi_{p,\neg p})$ and related to literals of the form $p(t)$ and $\neg p(t)$.*
  - *$\mathcal{D}$ is made up of the domains $dom(\Psi_{p,\neg p})$ defined over T; each $dom(\Psi_{p,\neg p})$ is separated into two subdomains $dom(\Psi_{p,\neg p}) = dom(\Psi_p) \cup dom(\Psi_{\neg p})$ such that $dom(\Psi_p) \cap dom(\Psi_{\neg p}) = \emptyset$,*
    - *$dom(\Psi_p) = \{t, such\ thatp(t)\}$, represents the set of terms that are true under a partial interpretation of P,*
    - *$dom(\Psi_{\neg p}) = \{t, such\ that \neg p(t)\}$, represents the set of terms that are false under a partial interpretation of P,*
  - *$\mathcal{C}$ initially contains the consistency constraints of the form $\#\langle 1, 1, \Psi_p, \Psi_{\neg p}\rangle$, that maintain consistency at the logical level;*
- *unit clauses of P initially define the domains :*
  - *$dom(\Psi_p) = \{t_0\}$, for each unit clause $p(t_0)$ of P,*
  - *$dom(\Psi_{\neg p}) = \{t_0\}$, for each unit clause $\neg p(t_0)$ of P,*
- *non-unit clauses are transformed into cardinality constraints. A clause $p_1(t_1) \vee \cdots \vee p_k(t_k)$ is transformed into $\#\langle 1, n, \Psi_{p_1}^{[t_1]}, \ldots, \Psi_{p_k}^{[t_k]}\rangle$ thus following the Ψ-variable notation.*

We give here a simple example of the representation of a logic problem P as a CSP^T :

(P)

$$\left\{ \begin{array}{l} p(a) \\ \neg p(X) \vee q(f(X)) \\ \neg q(f(f(X))) \end{array} \right.$$

(CSP^T)

$$\left\{ \begin{array}{l} \mathcal{X} = \{\ \Psi_p, \Psi_{\neg p}, \Psi_q, \Psi_{\neg q}\ \}; \\ \mathcal{D} = \left\{ \begin{array}{ll} dom(\Psi_{p,\neg p}) = & \{a\} \cup \{\} \\ dom(\Psi_{q,\neg q}) = & \{\} \cup \{f(f(X))\} \end{array} \right. \\ \mathcal{C} = \{\ \#\langle 1, 2, \Psi_{\neg p}^{[X]}, \Psi_q^{[f(X)]}\rangle\ \} \end{array} \right.$$

Resolution

In (Richer 1999) it was clearly underlined that the resolution of CSP^T was not compatible with the standard resolution approach of CSPs which considers that domains are finite and not extendable. The resolution of CSP^T relies on a domain extension that compels the introduction of the notion of an extended CSP.

Definition 4 - Extended CSP - *An extended CSP, noted CSP$_{ext}$, has the ability to extend its domains through constraint satisfaction.*

For example, the assignment $(x_1 = 1, x_2 = 2, x_3 = 3)$ for (CSP_1) is not valid for a standard approach because

3 does not belong to d_3 but is acceptable for an *extended* approach and the value 3 is added to domain d_3. In the remainder of this paper we will only consider CSP^T_{ext}.

The principle of the general algorithm designed to solve the CSP^T_{ext} is to build a partial interpretation by iteratively satisfying constraints. The partial interpretation may be viewed as an attempt at constructing a counterexample for refuting the given hypothesis. Constraint satisfaction leads to the production of new values that do not appear in their related Ψ-domains. Following an extended approach, new values are added to their related domains to perform further deductions and increase the partial interpretation. The unsatisfiability of a CSP arises from the discovery of a value μ belonging to both Ψ-subdomains ($\mu \in dom(\Psi_p) \wedge \mu \in dom(\Psi_{\neg p})$) of a Ψ-domain $dom(\Psi_{p,\neg p})$. From a logic view point this is equivalent to generating two resolvents $p(t_1)$ and $\neg p(t_2)$ such that there exists a most general unifier σ of t_1 and t_2 ($\sigma(t_1) = \sigma(t_2)$).

```
function  $Solve(\mathcal{P} : CSP^T_{ext}) : boolean$
input  a $CSP^T_{ext} = \{\mathcal{X}, \mathcal{D}, \mathcal{C}\}$
output  $\mathcal{P}$ with extended domains
return  true if the problem $\mathcal{P}$ is consistent,
            false otherwise.
begin
    $V = \{\Psi_p \in \mathcal{V},\ dom(\Psi_p) \neq \emptyset\}$
    $consistency = true$
    while $V \neq \emptyset$ and $consistency$ do
        choose $\Psi_v \in V$
        $V = V / \{\Psi_v\}$
        $C(x_v) = \{\#(1, n, \Psi^{[t_1]}_{p_1}, \ldots, \Psi^{[t_{n-1}]}_{p_{n-1}}, \Psi^{[t_v]}_v\rangle\}$
        while $C(\Psi_v) \neq \emptyset$ and $consistency$ do
            choose $c \in C(\Psi_v)$
            $C(\Psi_v) = C(\Psi_v) / \{c\}$
            $consistency = propagate(V, c, \Psi_v)$
        end
    end
    return $consistency$
end
```

Figure 2: Resolution of a CSP^T_{ext}

The interesting point is that we do not confine ourselves to ground atoms. For example, we can add the term X to the Ψ-subdomain of a unary predicate p. This prevents us from enumerating the whole Herbrand universe by using subsumption.

The resolution procedure underlying the implementation of *Sacre*, based on constraint satisfaction, has been identified as a forward chaining rule applied to a set of contrapositives. To give the reader a yet clearer view of the resolution procedure, we shall refer to figures 2 and 3. The role of the *propagate* function is to determine the domains that are extended by the satisfaction of constraint c that contains the Ψ-variable Ψ_v.

```
function  $propagate(V, c, \Psi_v) : boolean$
input/output  $V$ a set of $\Psi$-variables
input  $c$ a constraint $\#(1, n, \Psi^{[t_1]}_{p_1}, \ldots, \Psi^{[t_{n-1}]}_{p_{n-1}}, \Psi^{[t_v]}_v\rangle$
input  $\Psi_v$ a $\Psi$-variable
return  true if the satisfaction of constraint $c$ did
            not lead to an inconsitency.
begin
    foreach $\mu \in dom(\Psi_{\neg v})$ do
        if the satisfaction of $c$ where $\Psi^{[t_v]}_v = \mu$
              leads to the extension of $dom(\Psi_q)$ then
            if $\mu$ already $\in dom(\Psi_{\neg q})$ then
                return false    // inconsistency
            else
                extend $dom(\Psi_q)$
                $V = V \cup \{\Psi_q\}$
            end
        end
    end
    return true
end
```

Figure 3: Satisfaction of a constraint with domain extension

Heuristic and strategy tuning

The major point in *Sacre* is that it is possible to combine several heuristics and restriction techniques issuing from theorem proving (such as weighting) or constraint satisfaction (like forward-checking). These heuristics help improve the efficiency of the resolution of the CSP^T_{ext}. For example, when choosing a variable it is possible to select a min-domain (choose the variable with the minimum number of values in its domain), max-domain, min-constraint or max-constraint heuristic. New values can be rejected if their weight exceeds an upper bound or, the satisfaction of a constraint can be ended if a maximum number of values has been reached. The major difficulty is to combine these heuristics together so as to sufficiently decrease the search space without restricting it to a space hiding the solution. It is also worth mentioning that it is possible to choose between a depth-first, breadth-first or depth-first iterative deepening search (Korf 1985).

Other features

Sacre also applies to problems written in first-order logic with equality. Demodulation with Lex Recursive Path Ordering (Dershowitz 1987) has been implemented. Further versions will probably integrate paramodulation. One main feature of *Sacre* is the possibility to *direct* the search by orientating cardinality constraints. This ability, as in Prolog, is intended to compute a solution with very little searching and has the potential of being quite efficient compared to a non-directed approach. For example, problem NUM084.010 (evaluation of 10!) could not be solved in less than one second without taking this feature into account (see table 1).

Soundness and completeness

Unfortunately *Sacre* is incomplete, but a lack of completeness can generally lead to more efficiency. This seems to be the case considering the results obtained table 1.

Incompleteness is due to cardinality constraints that act as the *unit-resulting resolution* inference rule (or a forward chaining algorithm) which is sound but incomplete. Efficiency also stems from the lack of a *case splitting rule*. It is possible to design a case splitting rule that ensures completeness but causes a loss of efficiency (Richer 1999). This rule is not yet implemented in the current version of our solver.

Results

The *Sacre* method was implemented in C under Unix. In order to point out the real interest of our approach and the efficiency of our solver, we decided to tackle to some problems of the CADE-13 Automated Theorem Proving System competition (Suttner & Sutcliffe 1997).

Problem	Otter	Setheo	Sacre
BOO006-1	3	0	0
BOO012-1	8	0	0
BOO016-1	2	11	1
LCL196-1	7	6	23
LCL210-1	4	7	3
NUM003-1	0	0	0
NUM009-1	8	152	0
NUM284-1	0		0
PLA011-2		0	
PLA014-1		0	
RNG005-1	1	199	0
RNG038-2	0	94	8
RNG040-1	0	0	0
SET008-1	0	0	0
SET061-6	16		1
SET063-6	6		1
SET075-6	42		2
SET080-6	0	3	0
SET083-6	76		1
SET101-6	0	0	6
SET232-6		21	1
SYN200-1	0	0	0
SYN202-1	1	0	0
SYN271-1	0	0	1
Group	0	0	0
coloring	0	0	0

Table 1: Comparison between Otter, *Sacre* and Setheo.

We present in this section some comparatives results between *Sacre* and two other theorem provers Otter and Setheo. Otter (McCune 1994) is one of the most complete and efficient theorem prover using inference rules based on *resolution* (unit resolution, binary resolution, hyperresolution). Setheo (Bayerl & Letz 1987) uses *model elimination*. Otter and Setheo were chosen for comparison because they took part in the CADE-13 competition and obtained the best results.

Table 1 shows the results for some problems. Times are given in seconds. A resolution time of 0 second means that it took less than one second to prove inconsistency. Blanks mean that the problem could not be solved within 300 seconds. The tests were run on a Sun Sparc Ultra 1 workstation. The columns Otter and Setho respectively provide the results of Otter 3.0.4. in auto mode, and the best results of Setheo 3.3. with the "-dr" or "-wdr" option.

Sacre performs well on some kind of problems but there remains some problems out of *Sacre*'s scope. The problems PLA004-1 and PLA011-2 can not be solved by *Sacre* and Otter.

Conclusion and future work

In this paper we have shown that there exists a certain correspondence between first-order calculus problems in conjunctive normal form and constraint satisfaction problems. A new species of solvers can be built on this paradigm. Not only did we prove the validity of our approach but also its effectiveness with the *Sacre* prover. Improvements are still possible, and much work is under completion. We think it is possible to implement an oracle, like the autonomous mode of Otter(McCune 1994), able to determine the best strategies and heuristics for solving a given problem. The introduction of a *case splitting rule* that ensures completeness would probably be more efficient than the Satchmo case splitting rule because we are not forced to work with ground terms. In the case of predicate calculus problems with non-recursive clauses and non-extended domains (as it is the case for the Map Coloring problem), we are confronted to *standard* CSP. Moreover, the specification of the domains of predicates, instead of their computation, combined with a non-domain extension, can greatly reduce the search space. Further experimentation and improvements will determine how worthwhile are approcah is.

References

Bayerl, S., and Letz, R. 1987. Setheo : A sequential theorem prover for first-order logic. In *Esprit'87 - Achievements and Impacts, part 1*, 721–735. North-Holland.

Bry, F., and Yahya, A. 1996. Minimal model generation with positive unit hyper-resolution tableaux. In *Proceedings of the 5th Workshop on Theorem Proving with Tableaux and Related Methods*, Lectures Notes in Artificial Intelligence 1071, 143–159. New York: Springer-Verlag.

Chabrier, J.; Juliard, V.; and Chabrier, J.-J. 1995. SCORE(FD/B) : an efficient complete local-based search method for satisfiability problems. In *CP'95 Workshop - Studying and solving hard problems.*

Chabrier, J. 1997. *Programmation par contraintes : langages méthodes et applications sur les domaines booléens et entiers.* In *HDR.* Ph.D. Dissertation, LIRSIA, Université de Bourgogne.

Chu, H., and Plaisted, D. A. 1997. Clin-s. *Journal of Automated Reasoning* 18(2):183–188.

Davis, M., and Putnam, H. 1960. A computing procedure for quantification theory. *Journal of the ACM* 7:201–215.

Dershowitz, N. 1987. Termination of rewriting. *Journal of Symbolic Computation* 3:69–116.

Dubois, O.; André, P.; Boufkhad, Y.; and Carlier, J. 1993. Sat versus unsat. *In Second DIMACS Challenge.*

Gilmore, P. C. 1960. A proof method for quantification theory : its justification and realization. *IBM JRD* 28–35.

Korf, K. E. 1985. Depth-first iterative deepening : an optimal admissible tree search. *Artificial Intelligence* 27:97–109.

Loveland, D. W.; Reed, D. W.; and Wilson, D. S. 1995. Satchmore : Satchmo with relevancy. *Journal of Automated Reasoning* 14:325–351.

Loveland, D. W. 1978. *Automated Theorem Proving : A Logical Basis.* New York: North-Holland.

Manthey, R., and Bry, F. 1988. Satchmo : A theorem prover implemented in prolog. In *Proceedings of the 9th International Conference on Automated Deduction*, LNCS 310, 415–434. New York: Springer-Verlag.

McCune, W. W. 1994. *Otter 3.0 Reference Manual and Guide.*

Minton, S.; Johnston, M.; Philips, A.; and Laird, P. 1992. Minimizing conflicts : a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence* 58:161–205.

Montanari, U. 1974. Networks of constraints : Fundamental properties and applications to picture processing. *Information Science* 7:95–132.

Plaisted, D. A. 1982. A simplified problem reduction format. *Artificial Intelligence* 18:227–261.

Plaisted, D. A. 1988. Non-horn clause logic programming without contrapositives. *JAR* 4:287–325.

Richer, J.-M., and Chabrier, J.-J. 1997. Une approche de résolution de problèmes en logique basée sur des techniques de satisfaction de contraintes. In *JFPLC'97.*

Richer, J.-M. 1999. *Sacre : une approche de résolution en logique fondée sur des techniques de satisfaction de contraintes.* Ph.D. Dissertation, LIRSIA - Université de Bourgogne.

Selman, B.; Levesque, H.; and Mitchell, D. 1991. A new method for solving hard satisfiability problems. *In 10th NCAI* 440–446.

Slaney, J.; Lusk, E.; and McCune, W. 1994. Scott : Semantically constrained otter - system description. Technical Report TR-ARP-3-94, Centre for Information Science Research, Australian National University.

Slaney, J. 1995. *FINDER - Finite Domain Enumerator, Version 3.0 - Notes and Guide.*

Suttner, C., and Sutcliffe, G. 1997. The design of the CADE-13 atp system competition. *Journal of Automated Reasoning* 18(2):139–162.

Van-Hentenryck, P., and Deville, Y. 1991. The cardinality operator : A new logical connective for constraint logic programming. In Furukawa, K., ed., *Proceedings of the 8th ICLP, Paris, France 24-28 June 1991*, 745–759.

Zhang, J. 1996. Constructing finite algebras with falcon. *Journal of Automated Readoning* 17:1–22.