Model Generation with Boolean Constraints

Koshimura, Miyuki
Department of Intelligent Systems, Faculty of Information Science and Electrical Engineering, Kyushu University

Fujita, Hiroshi
Department of Intelligent Systems, Faculty of Information Science and Electrical Engineering, Kyushu University

Hasegawa, Ryuzo
Department of Intelligent Systems, Faculty of Information Science and Electrical Engineering, Kyushu University

https://doi.org/10.15017/1515730

出版情報：九州大学大学院システム情報科学紀要. 6 (2), pp.157-160, 2001-09-26. Faculty of Information Science and Electrical Engineering, Kyushu University
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第6巻 第2号 平成13年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.2, September 2001

Model Generation with Boolean Constraints

Miyuki KOSHIMURA*, Hiroshi FUJITA* and Ryuzo HASEGAWA*

(Received June 15, 2001)

Abstract: We present a simple method for eliminating redundant searches in model generation. The method employs *Boolean Constraints* which are conjunctions of ground instances of clauses having participated in proofs. Boolean Constraints work as sets of lemmas with which duplicate subproofs and irrelevant model extensions can be eliminated. The method has been tentatively implemented on a constraint logic programming system. We evaluated effects of the method by proving some typical problems taken from the TPTP problem library.

Keywords: Theorem proving, Constraint solver, Folding-up, Proof condensation

1. Introduction

The model generation procedure tries to construct Herbrand models for a given clause set and determines its satisfiability. It maintains a set M of ground atoms called a model candidate, finds violated clauses that are not satisfied under M, then extends M to satisfy them, and repeats this process until a model is found or all model candidates are rejected.

There are two types of redundancy in model generation: One is that the same subproof tree may be generated at several descendant nodes after a case-splitting occurs. Another is caused by unnecessary model candidate extensions with irrelevant clauses. We embedded both folding-up [3] and proof condensation [5] into model generation for eliminating these redundancies by analyzing dependency in a proof [2]. The embedded function examines the structure of proof in order to append a solved subproof-tree to an open branch.

This paper presents yet another method to eliminate the redundancies on the basis of semantical information. If the current model candidate conflicts with a set of instances of clauses that have participated in model generation so far, we can reject the model candidate without further exploration. We call the set a *Boolean Constraint*. It is worth noting that the Boolean Constraint consists of only ground instances of clauses and all atoms in model candidates are ground. Therefore, a conflict test is essentially propositional theorem proving.

In this work, we utilize a constraint solver [7] on Boolean expressions for the test, though we could utilize model generation itself or other proving methods in principle. The main reason for utilizing the constraint solver is that it can compute a simple (canonical) form of the Boolean Constraint which is incrementally updated as the proof progresses. Since the constraint solver reduces the Boolean Constraint as simple as possible, it can detect the conflict efficiently.

* Department of Intelligent Systems

2. Model Generation

Throughout this paper, a *clause* $\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$ is represented in implicational form: $A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m$, where A_i $(1 \leq i \leq n)$ and B_j $(1 \leq j \leq m)$ are atoms; the left hand side of "$\rightarrow$" is said to be the *antecedent*; and the right hand side of "$\rightarrow$" the *consequent*.

A clause is said to be *positive* if its antecedent is $\top$ $(n = 0)$, and *negative* if its consequent is $\bot$ $(m = 0)$; otherwise it is *mixed* $(n \neq 0, m \neq 0)$. A clause is said to be *violated* under a set M of ground atoms if the following condition holds with some ground substitution σ: $\forall i(1 \leq i \leq n) A_i\sigma \in M \wedge \forall j(1 \leq j \leq m) B_j\sigma \notin M$.

A model generation proof procedure is sketched in **Fig. 1**. Given a set S of clauses, MG tries to construct a model by extending the current model candidate M so as to satisfy violated clauses under M (model extension). When a negative clause is violated under M, MG rejects M because there is no way of extending M (model rejection). If no clause is violated under M, we conclude M is a model of S, that is, S is satisfiable (model finding).

Consider the following set of clauses $S1$:

$C1 : \top \rightarrow p(a) \vee p(c) \qquad C2 : p(a) \rightarrow q(b)$

$C3 : p(X) \wedge q(Y) \rightarrow r(X,Y) \vee r(X,X) \vee r(Y,X)$

$C4 : p(X) \wedge q(Y) \rightarrow r(s(X),Y) \vee r(X,X) \vee r(Y,X)$

$C5 : p(X) \wedge r(s(X),Y) \rightarrow r(Y,s(X))$

procedure $MGTP(S) : Res$; /* Input(S):Clause set, Output(Res):satisfiability of S */
return$(MG(\emptyset))$;
procedure $MG(M) : Res$; /* Input(M): Model candidate*/
1. (Model rejection) If a negative clause $A_1 \wedge \ldots \wedge A_n \rightarrow \bot \in S$ is violated under M with a ground substitution σ, **return** *unsatisfiable*;
2. (Model extension) If a positive or mixed clause $A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m \in S$ is violated under M with a ground substitution σ,
   for $(i = 1; i \leq m; i++)$ {
      if $(MG(M \cup \{B_i\sigma\}) = satisfiable)$ **return** *satisfiable*;
   }
   return *unsatisfiable*;
3. (Model finding) If neither 1 nor 2 is applicable, **return** *satisfiable*;

Fig.1 Model generation procedure.

$C6: \; r(s(X), Y) \wedge r(Y, s(X)) \rightarrow r(X, X)$
$C7: \; r(X, X) \rightarrow r(s(X), X) \vee r(X, s(X))$
$C8: \; p(X) \wedge q(Y) \wedge r(Y, X) \rightarrow r(X, X)$
$C9: \; r(s(X), X) \rightarrow \bot$
$C10: r(X, s(X)) \rightarrow \bot \qquad C11: p(c) \rightarrow \bot$

Figure 2 shows a proof-tree for $S1$. The inner nodes of a proof-tree except the root node are labeled with atoms used for model extension. A branch or a path from the root to a node corresponds to a model candidate. A leaf labeled with $\bot$ indicates that the corresponding model candidate has been rejected. $S1$ is unsatisfiable because all leaves of its proof-tree are labeled with $\bot$.

The procedure MG in **Fig.** 1 can be proved sound and complete in the sense that MG examines only models containing the model candidate M[4].

Theorem 1 *Let S be a set of clauses and M be a set of ground atoms. Then $MG(M)$ return unsatisfiable if and only if there is no model containing M.*

Let BC be a set of ground instances of violated clauses in S that have been used for model rejection and extension. If $BC \cup M$ is unsatisfiable, $S \cup M$ is unsatisfiable. In this case, according to Theorem 1, we can reject M without further proving. This rejection mechanism can reduce search spaces by orders of magnitude. **Figure** 3 shows a model generation procedure in which the rejection mechanism is embedded. The framed parts are embedded ones. We call the procedure *model generation with Boolean Constraints* because BC is essentially the conjunction of propositional clauses and can be treated as a Boolean expression.

Initially, the set BC is set to the empty set ((1)). BC is updated whenever ground instances of clauses are used for model extension or model rejection ((3),(4)). BC is used for model rejection prior to performing normal model rejection and extension ((2)). This rejection works as folding-up to eliminate duplicates subproofs. BC is also used for model rejection testing whenever each extension $MG(M \cup \{B_i\sigma\})$ is finished ((5)). This rejection test works as proof condensation to avoid unnecessary model extensions.

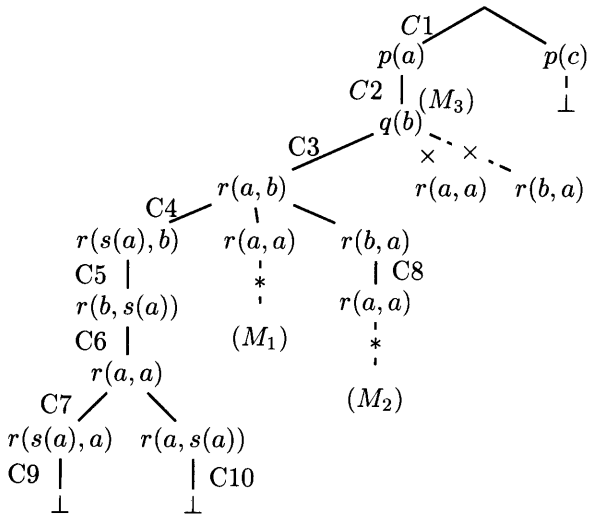

Fig.4 Eliminating redundant branches.

Figure 4 shows a proof tree for $S1$ obtained by model generation with Boolean Constraints. The mark $*$ indicates a branch pruned by operation (2), while the mark $\times$ indicates that by operation (5).

BC becomes $BC_1 = \{C1, C2, C3\sigma_1, C4\sigma_1, C5\sigma_1, C6\sigma_1, C7\sigma_2, C9\sigma_2, C10\sigma_2\}$ after the second branch from the left has been rejected where $\sigma_1 = \{X \leftarrow a, Y \leftarrow b\}$ and $\sigma_2 = \{X \leftarrow a\}$. Then, the next model candidate M_1 to be solved is $\{p(a), q(b), r(a, b), r(a, a)\}$. However, since $BC_1 \cup M_1$ is unsatisfiable, M_1 is rejected. After the model extension under $r(b, a)$ with clause $C8\sigma_2$

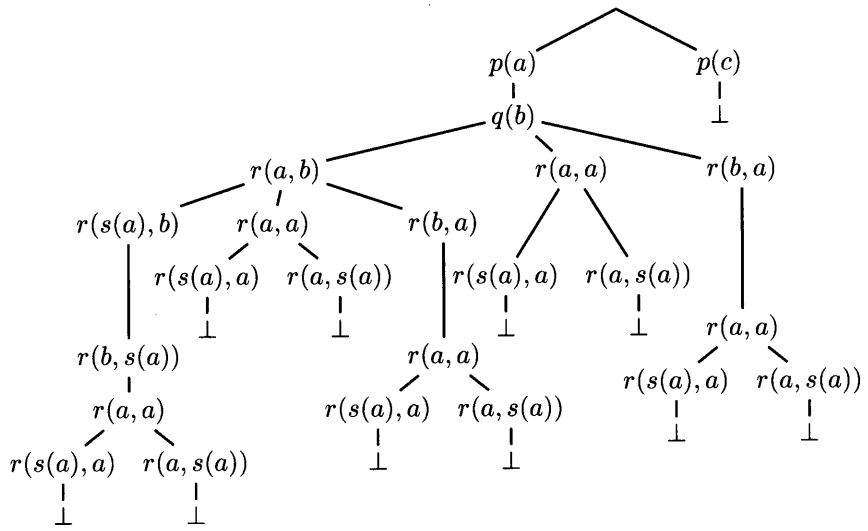

Fig.2 A normal proof-tree of $S1$.

procedure $MGTP(S) : Res$;

$\boxed{BC := \emptyset;\ (1)}$

return$(MG(\emptyset))$;

procedure $MG(M) : Res$;

1. (Model rejection by BC)
   $\boxed{\text{If } (BC \cup M \text{ is unsatisfiable}) \textbf{ return } \textit{unsatisfiable};\ (2)}$
2. (Model rejection by negative clauses) If a negative clause $(A_1 \wedge \ldots \wedge A_n \rightarrow \bot) \in S$ is violated under M with a ground substitution σ,
   $\boxed{BC := BC \cup \{A_1\sigma \wedge \ldots \wedge A_n\sigma \rightarrow \bot\};\ (3)}$
   return *unsatisfiable.*
3. (Model extension) If a positive or mixed clause $(A_1 \wedge \ldots \wedge A_n \rightarrow B_1 \vee \ldots \vee B_m) \in S$ is violated under M with a ground substitution σ,
   $\boxed{BC := BC \cup \{A_1\sigma \wedge \ldots \wedge A_n\sigma \rightarrow B_1\sigma \vee \ldots \vee B_m\sigma\};\ (4)}$
   for $(i = 1; i \leq m; i++)$ {
   if $(MG(M \cup \{B_i\sigma\}) = satisfiable)$ **return** *satisfiable*
   $\boxed{\textbf{elseif } (BC \cup M \text{ is unsatisfiable}) \textbf{ return } \textit{unsatisfiable}\ (5)}$
   }
   return *unsatisfiable*;
4. (Model finding) If neither rule is applicable, **return** *satisfiable*;

Fig.3 Model generation with Boolean Constraint.

has been performed, BC becomes $BC_2 = BC_1 \cup \{C8\sigma_2\}$. The corresponding model candidate M_2 is $\{p(a), q(b), r(a,b), r(b,a), r(a,a)\}$. In this case, $BC_2 \cup M_2$ is unsatisfiable as well, so that M_2 is rejected.

On the other hand, $BC_2 \cup \{p(a), q(b)\}$, that is, $BC_2 \cup M_3$ is unsatisfiable. Therefore, the exploration of $r(a,a)$ and $r(b,a)$ below $q(b)$ can be eliminated. Thus, we obtain a proof-tree which has 12 inner nodes while the normal proof-tree shown in **Fig.** 2 has 23 inner nodes.

3. Implementation

The method is implemented on top of a constraint logic programming system B-Prolog[7] which supports constraint solvers over trees, Boolean, finite-domains and sets. We manipulate a set BC of ground instances of clauses through the constraint solver. Thus, BC is maintained within the constraint solver. When updating BC (**Fig.** 3(3) (4)), we tell $(\neg A_1\sigma \vee \ldots \vee \neg A_n\sigma) = TRUE$ or $(\neg A_1\sigma \vee \ldots \vee \neg A_n\sigma \vee B_1\sigma \vee \ldots \vee B_m\sigma) = TRUE$ to the

Table-1 Comparison of experimental results.
(Sun Ultra 60 450MHz)

Problem	B.C.	Time	Proof Tree	
		(seconds)	#B	#N
PUZ	no	T.O.	-	-
010-1	yes	261.55	2606	4467
PUZ	no	12.20	12546	17895
015-2.006	yes	0.50	113	205
PUZ	no	T.O.	-	-
018-1	yes	33.71	134	343
PUZ	no	T.O.	-	-
018-2	yes	111.38	272	618
PUZ	no	0.69	988	2509
028-5	yes	0.70	103	398
PUZ	no	T.O.	-	-
028-6	yes	4.05	365	1147
CIV	no	T.O.	-	-
008-1.002	yes	0.63	5	196
MSC	no	124.49	24583	171899
007-2.005	yes	55.63	112	2231
GEO	no	T.O.	-	-
013-3	yes	44.75	90	573
GEO	no	T.O.	-	-
033-3	yes	2.24	1	139
GEO	no	T.O.	-	-
051-3	yes	65.87	43	471
SYN	no	1.62	19683	29526
009-1	yes	0.01	3	14

PUZ018-2 is satisfiable.
Others are unsatisfiable.
T.O.: Time Out (10 minutes)
B.C.: Boolean Constraint
#B: No. of branches
#N: No. of nodes

constraint solver. On the other hand, when testing whether a conflict occurs (**Fig.** 3(2) (5)), we ask the constraint solver "Is $A = TRUE$ possible for all $A \in M$?" If they become all $TRUE$, $BC \cup M$ is satisfiable, otherwise, it is unsatisfiable.

Table 1 compares the proving performance on several typical problems taken from the TPTP library [6]. The problems were run on a SUN Ultra 60 (450MHz, 1GB, Solaris2.7) workstation with a time limit of 10 minutes and a space limit of 240MB. All problems exhibit the pruning effect of Boolean Constraint. Especially, all the 12 problems are solved with Boolean Constraints although 8 of 12 could not be solved without Boolean Constraints.

4. Future Work

In the current implementation, the most time consuming task is the conflict test (**Fig.** 3 (2) (5)). The cost of this task may be reduced by using Binary Decision Diagrams (BDDs). We are considering two approaches using BDD: One is to replace the constraint solver with BDD. Another is to use a BDD for representing a proof tree of model generation [1].

In the latter, all model candidates are simultaneously represented as the paths ending with a truth node in a BDD. With this representation, model candidates conflicting with the Boolean Constraint are automatically eliminated by standard BDD functions. Thus, the conflict test can be ignored. However, an implementation of the latter approach is more difficult than that of the former because BDD may create more model candidates than the model generation procedure, and it would be necessary to select a minimal one for efficiency. We are now developing a prototype for the former.

References

1) R. Hähnle. BDDs for Representation of Model Candidates in MGTP. a private talk at Kyushu University, September 1998.
2) M. Koshimura and R. Hasegawa. Proof Simplification for Model Generation and Its Applications. In M. Parigot and A. Voronkov, editors, *Proceedings of 7th International Conference, LPAR2000*, volume 1955 of *Lecture Notes in Artificial Intelligence*, pages 96–113. Springer, November 2000.
3) R. Letz, K. Mayr, and C. Goller. Controlled Integration of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, 13(3):297–337, December 1994.
4) I. Niemelä. A Tableau Calculus for Minimal Model Reasoning. In P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, editors, *5th International Workshop, TABLEAUX'96*, volume 1071 of *Lecture Notes in Artificial Intelligence*, pages 278–294. Spriger, May 1996.
5) F. Oppacher and E. Suen. HARP: A Tableau-Based Theorem Prover. *Journal of Automated Reasoning*, 4(1):69–100, March 1988.
6) G. Sutcliffe and C. Suttner. The TPTP Problem Library -CNF Release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, October 1998.
7) N.-F. Zhou. *B-Prolog User's Manual (Version 5.0)*, 2000. http://www.probp.com.