A Model Generation Theorem Prover Handling Finite Interval Constraints

Hähnle, Reiner
Department of Computing Science, Chalmers Technical University

Shirai, Yasuyuki
Department of Intelligent Systems, Graduate School of Information Science and Electrical Engineering, Kyushu University : Graduate Student | Information Technologies Development Department, Mitsubishi Research Institute, Inc.

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

https://doi.org/10.15017/1515680

出版情報：九州大学大学院システム情報科学紀要. 5 (2), pp.167-172, 2000-09-26. 九州大学大学院システム情報科学研究院
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 5 巻　第 2 号　平成 12 年 9 月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.5, No.2, September 2000

A Model Generation Theorem Prover Handling Finite Interval Constraints

Reiner HÄHNLE* , Yasuyuki SHIRAI** and Ryuzo HASEGAWA***

(Received June 16, 2000)

Abstract: Model generation theorem proving (MGTP) is a class of deduction procedures for first-order logic that were successfully used to solve hard combinatorial problems. For some applications, the representation of models in MGTP and its extension CMGTP is too redundant. In this paper, we suggest to extend members of model candidates in such a way that a predicate p can have not only terms as arguments, but at certain places also subsets of totally ordered finite domains. The ensuing language and deduction system relies on constraints based on finite intervals in totally ordered sets and is called IV-MGTP. It is related to constraint programming and many-valued logic, but differs significantly from either. We show soundness and completeness of IV-MGTP. First results with our implementation show considerable potential of the method.

Keywords: Theorem proving, Model generation, Constraint propagation, Interval/Extraval representation, Extended interpretation

1. Introduction

Model generation theorem proving (MGTP) is a class of deduction procedures for first-order logic in conjunctive normal form (CNF) that have been successfully used to solve hard combinatorial problems [6]. The procedural semantics of first-order CNF formulas as defined by MGTP is based on bottom-up evaluation. MGTP operates on range-restricted non-Horn rules over positive literals, and it is proof confluent. Together, this ensures completeness even without backtracking. MGTP is closely related [4] to hypertableaux [1]. CMGTP [9] is an extension of MGTP that allows negated atoms in rules and approximates their declarative semantics with an additional inference rule. Very efficient implementations of (C)MGTP were realized [5].

In MGTP (CMGTP), interpretations (called *model candidates*) are represented as finite sets of ground atoms (literals). In many situations, this turns out being too redundant: take, for example, variables I, J ranging over the domain dom $= \{1, \ldots, 4\}$, and interpret $\leq$, $+$ naturally. A rule like "$p(I)$, $I + J \leq 4 \rightarrow q(J)$" splits into three model extensions: $q(1)$, $q(2)$, $q(3)$, if $p(1)$ is present in the current model candidate. Now assume we have the rule "$q(I)$, $q(J)$, $I \neq J \rightarrow \bot$" saying that q is functional in its argument and, say, $q(4)$ is derived from another rule. Then all three branches must be refuted separately.

Totally ordered, finite domains occur naturally in many problems. In such problems, situations such as the one just sketched are common. In the present paper, we set out to enhance MGTP with mechanisms to deal with them efficiently. Informally, we suggest two extensions of MGTP: first, the arguments of predicates can contain certain finite domain constructors based on intervals. The rule above could be rephrased as "$p([I, I]) \rightarrow q([1, 4 - I])$", where $[i, j]$ denotes the set $\{k \in \text{dom} \mid i \leq k \leq j\}$. Second, to make full use of this extended language, elements of model candidates M are generalized as well. In the example, if $p(1) \in M$, then the rule is triggered with $\{I \leftarrow 1\}$ and results in the single extension of M with $q(\{1, 2, 3\})$, rather than in three extensions as before. One could even write a more general rule like "$p([1, I]) \rightarrow q([1, 4 - I])$", which has two advantages: it is still useful, if p is underdetermined in the current model candidate, and (a suitably extended version of) pattern matching can be used on the term $[1, I]$. Because intervals play a central role, we gave the name IV-MGTP to the present extension of MGTP.

2. Standard Definitions

Terms and atoms are defined as usual over a **signature** Σ consisting of **constant symbols** C_Σ, **function symbols** F_Σ, **predicate symbols** P_Σ. Variables from a set $TVar$ occurring in terms are called **term variables** to distinguish them from variables occurring in constraints. The set of variable free or **ground terms** is denoted $Term^0_\Sigma$, the

* Department of Computing Science, Chalmers Technical University, 41296 Gothenburg, Sweden

** Department of Intelligent Systems, Graduate Student & Information Technologies Development Department, Mitsubishi Research Institute, Inc., Tokyo 100-8141

*** Department of Intelligent Systems

set of variable free or **ground atoms** with $Atom^0_\Sigma$. A **literal** is either an atom or an expression $\neg L$, where L is an atom.

Clauses are written in **rule notation** and are expressions of the form $C \rightarrow D$, where C and D are finite sequences of atoms or literals. C is called the **antecedent** of $C \rightarrow D$ and D its **succedent**. Using explicitly named atoms such a rule is written $L_1, \ldots, L_k \rightarrow M_1; \ldots; M_l$. Write $\top$ for the empty antecedent and $\bot$ for the empty succedent.

As usual, **ground literals, rules, clauses** are variable free expressions. A rule is **range-restricted**, if all variables occurring in its succedent occur already in its antecedent. Therefore, a range-restricted rule with empty antecedent must be ground. The first occurrence (reading from left to right) of a variable in a rule is called **free**, all other occurrences are **bound**. In a range-restricted rule, only bound variables occur in the succedent.

An **MGTP program** P is a finite set of range-restricted rules. If literals are allowed to occur in rules, one obtains the language of **CMGTP**.

A **domain** is a finite, totally ordered set N, which is represented by a set of natural numbers of the form $\{1, \ldots, n\}$. This implies that domains are *homogeneous*, that is, initial segments with identical length from different domains are indistinguishable. **Constraint variables** $I, J, \ldots$ from a set $CVar$ hold elements from a domain N, while **domain variables** $U, V, \ldots$ from a set $DVar$ hold subsets of a domain N. The notions of range-restrictedness and free/bound variable are defined exactly as for term variables.

3. Syntax and Semantics of IV-MGTP

3.1 The language of IV-MGTP

In order to enhance MGTP with finite domain constraints, we adopt the notation: $p(t_1, \ldots, t_r, S_1, \ldots, S_m)$ for what we call a **constrained atom**. This notation is motivated from the point view of signed formula logic (SFL) [7] and constraint logic programming (CLP) over finite domains [8].

Constrained atoms explicitly stipulate subsets of domains and thus are in **solved form**. The language of IV-MGTP needs to admit other forms of atoms, in order to be practically useful.

For a start, an IV-MGTP atom $p(\ldots)$ may contain domain variables from $DVar$ at the constraint places of p. Our domains are totally ordered; we take advantage from this by admitting **interval** and **extraval** notation to specify subsets of domains. In addition, the boundary values of intervals and extravals can be constraint variables from $CVar$. To summarize, an **IV-MGTP atom** is an expression $p(t_1, \ldots, t_r, \kappa_1, \ldots, \kappa_m)$, where the κ_i have one of the following forms:

1. $\{i_1, \ldots, i_r\}$, where $i_j \in N$ for $1 \leq j \leq r$ (κ_i is in **solved form**);
2. $]\iota_1, \iota_2[$, where $\iota_j (j = 1, 2) \in N \cup CVar$; the intended meaning is $]\iota_1, \iota_2[= \{i \in N \mid i < \iota_1 \text{ or } i > \iota_2\}$;
3. $[\iota_1, \iota_2]$, where $\iota_j (j = 1, 2) \in N \cup CVar$; the intended meaning is $[\iota_1, \iota_2] = \{i \in N \mid \iota_1 \leq i \leq \iota_2\}$;
4. $U \in DVar$.

Further forms of constraints might be useful, but are not considered for now. One has to find a trade-off between implementability and usability.

IV-MGTP rules are defined like MGTP rules, but are based on IV-MGTP atoms. By way of extension of the corresponding mechanism in MGTP [6], we allow an arbitrary ground decidable or *condition* `cond` over term and *constraint* variables to occur in the antecedent. The term and constraint variables occurring in condition must be bound.

A finite set of IV-MGTP rules satisfying the conditions above is called an **IV-MGTP program**.

For each $p \in P_\Sigma$ with constrained arguments, an IV-MGTP program contains a **declaration** line of the form "declare $p(t, \ldots, t, j_1, \ldots, j_m)$". If the i-th place of p is t, then the i-th argument of p is a standard term; if the i-th place of p is a positive integer j, then the i-th argument of p is a constraint over the domain $\{1, \ldots, j\}$.

Each IV-MGTP atom $p(t_1, \ldots, t_r, \kappa_1, \ldots, \kappa_m)$ consists of two parts, the standard **term part** $p(t_1, \ldots, t_r)$ and the **constraint part** $\langle \kappa_1, \ldots, \kappa_m \rangle$. Each of r and m can be 0. The latter, $m = 0$, is in particular the case for a predicate that has no declaration. By this convention, every MGTP program is also an IV-MGTP program.

3.2 Formal Semantics of IV-MGTP

Substitutions. A **ground substitution** σ in IV-MGTP maps term variables to ground terms, constraint variables over a domain N to N, and domain variables over a domain N to $2^N - \{\emptyset\}$.

Interpretations. Let $\mathbb{N}^+$ denote the positive integers. Then any domain is contained in $\mathbb{N}^+$. For simplicity, we assume in the following $\mathbb{N}^+ \subseteq C_\Sigma$. With this in mind, assume "declare $p(t, \ldots, t, j_1, \ldots, j_m)$", then any subset of $(Term^0_\Sigma)^r \times \{1, \ldots, j_1\} \times \cdots \times \{1, \ldots, j_m\}$ is an **IV-MGTP pre-interpretation** of p. In other words,

any standard Herbrand interpretation (restricted to suitable domains for the constraint part) is also an IV-MGTP pre-interpretation. In an **IV-MGTP interpretation I**, only the constraint part is handled slightly non-standard: the constrained arguments of p are **functional**, so the same must be true for $\mathbf{I}(p)$. If p is declared as above, then for all $\langle t_1,\ldots,t_r\rangle \in (Term^0_\Sigma)^r$ there is at most one $\langle i_1,\ldots,i_m\rangle \in \{1,\ldots,j_1\}\times\cdots\times\{1,\ldots,j_m\}$ such that $\langle t_1,\ldots,t_r,i_1,\ldots,i_m\rangle \in \mathbf{I}(p)$.

Note that the constraint part of an IV-MGTP ground atom can be assumed in solved form, because variable free intervals and extravals can be trivially converted to sets of domain values.

Satisfaction. An IV-MGTP ground atom $L = p(t_1,\ldots,t_r,S_1,\ldots,S_m)$ is **satisfied** by an IV-MGTP interpretation $\mathbf{I}$ ($\mathbf{I}\models L$) iff there are $i_1 \in S_1,\ldots,i_m \in S_m$ such that $\langle t_1,\ldots,t_r,i_1,\ldots,i_m\rangle \in \mathbf{I}(p)$. A set of IV-MGTP ground atoms M is satisfied iff $\mathbf{I}\models L$ for each $L\in M$. Note that an atom of the form $p(\ldots,\{\},\ldots)$ is unsatisfiable.

A ground IV-MGTP rule is satisfied by $\mathbf{I}$ iff at least one of the atoms in its antecedent is not satisfied by $\mathbf{I}$ or at least one of the atoms in its succedent is satisfied by $\mathbf{I}$. Obviously, $C\rightarrow D$ is satisfied iff $\overline{C}$ is satisfied or D is satisfied. An IV-MGTP rule r is satisfied by $\mathbf{I}$ iff $\mathbf{I}\models r\sigma$ for every ground substitution σ. Finally, an IV-MGTP program P is satisfied by $\mathbf{I}$ iff $\mathbf{I}\models r$ for all $r\in P$.

We stress that in IV-MGTP we did not essentially change the notion of an Herbrand interpretation What we did change is the notion of an atom. While in classical logic a maximal, consistent set of ground literals specifies exactly one interpretation, this particular relationship becomes one-many in the case of IV-MGTP.

4. The IV-MGTP Deduction Process

4.1 Model candidates

In the MGTP deduction process, a list of current model candidates is kept that represent Herbrand interpretations. In MGTP model candidates can be simply identified with sets of ground atoms. The same holds in IV-MGTP, only that some places of a predicate contain a ground constraint in solved form (that is: a subset of a domain) instead of a ground term. Note that, while in MGTP one model candidate containing ground atoms $\{L_1,\ldots,L_r\}$ trivially represents exactly one possible interpretation of the set of atoms $\{L_1,\ldots,L_r\}$, in IV-MGTP one model candidate represents many IV-MGTP interpretations which differ in the constraint parts.

Thus, in the following high-level description of the IV-MGTP deduction process, model candidates can be conceived as sets of constrained atoms of the form $p(t_1,\ldots,t_r,S_1,\ldots,S_m)$, where the S_i are subsets of the appropriate domain. If M is a model candidate, $p(t_1,\ldots,t_r)$ the ground term part, and $\langle S_1,\ldots,S_m\rangle$ the constraint part in M, then define $M(p(t_1,\ldots,t_r)) = \langle S_1,\ldots,S_m\rangle$.

Formally, a **model candidate** M is a *partial* function that maps ground instances of the term part of constrained atoms which is declared as "$p(t,\ldots,t,j_1,\ldots,j_m)$" into $(2^{\{1,\ldots,j_1\}}-\{\emptyset\})\times\cdots\times(2^{\{1,\ldots,j_m\}}-\{\emptyset\})$. Note that $M(p(t_1,\ldots,t_r))$ can be undefined.

4.2 Conjunctive matching

Let $r = C\rightarrow D\in P$ be an IV-MGTP rule, M a model candidate. We say that **conjunctive matching (CJM) can be applied to (the antecedent) of** r **and** M if there is a ground substitution σ such that for each atom of the form $p(t_1,\ldots,t_r,\kappa_1,\ldots,\kappa_m)\in C$: (i) $M(p(t_1,\ldots,t_r)\sigma) = \langle S_1,\ldots,S_m\rangle$; (ii) For all $1\le i\le m$: $S_i\subseteq\kappa_i\sigma$; (iii) `cond` part in the antecedent of $r\sigma$ is satisfied.

4.3 Inconsistency

A model candidate M **is inconsistent with** a constrained ground atom of the form $p(t_1,\ldots,t_r,S_1,\ldots,S_m)$ iff $M(p(t_1,\ldots,t_r)) = \langle S'_1,\ldots,S'_m\rangle$ and $S_i\cap S'_i = \emptyset$ for some $i\in\{1,\ldots,m\}$.

4.4 Subsumption

The task of the subsumption check is to avoid extension steps that would not further constrain the current model candidate. Formally, we say that a constrained ground atom $p(t_1,\ldots,t_r,S'_1,\ldots,S'_m)$ **is implied** by model candidate M iff $M(p(t_1,\ldots,t_r)) = \langle S_1,\ldots,S_m\rangle$ and $S_i\subseteq S'_i$ for $1\le i\le m$.

Let D be the succedent of an IV-MGTP rule to which conjunctive matching can be applied in M with substitution σ, then $D\sigma$ is said to be **subsumed** by M iff there is a constrained atom L in $D\sigma$ such that L is implied by M.

4.5 Model candidate update

Besides rejection, subsumption, and extension of a model candidate, in IV-MGTP there is a fourth possibility not present in MGTP.

Example 1 *Let* $D = p(\{1,2\})$ *and assume* $M(p) =$

$\langle\{2,3\}\rangle$. *Neither is the single atom in D inconsistent with M nor is it subsumed by M. Yet the information contained in D is not identical to the that in M and it can be used to refine M to* $M(p) = \langle\{2\}\rangle$.

In general we define model candidate update or the addition of information to model candidates as follows: let $L = p(t_1, \ldots, t_r, S_1, \ldots, S_m)$ be a constrained ground atom consistent with model candidate M; $p' = p(t_1, \ldots, t_r)$ is the ground term part of L. Then the **update** $M + L$ of M with L is:

$$(M + L)(q) = \begin{cases} \langle S_1, \ldots, S_m \rangle \\ \quad q = p',\ M(p') \text{ undefined} \\ \langle S_1 \cap S_1', \ldots, S_m \cap S_m' \rangle \\ \quad q = p',\ M(p') = \langle S_1', \ldots, S_m' \rangle \\ M(q) \text{ otherwise} \end{cases}$$

Example 2*Let us redo Example 1 in plain MGTP. The constrained atom* $D = p(\{1,2\})$ *corresponds to the disjunction* $D' = p(1); p(2)$ *of classical atoms, while* $M = \{p(\{2,3\})\}$ *corresponds to* two *MGTP model candidates:* $M' = \{p(2)\}$, $M'' = \{p(3)\}$. *Moreover, the functionality axiom "*$p(I), p(J), I \neq J \rightarrow \bot$*" is added. D' is subsumed by M', so M' is unchanged. On the other hand, M'' is rejected (in the MGTP sense) by D' and functionality, so we are left with M' only. But M' contains exactly the information of* $M + p(\{1,2\}) = p(\{2\})$.

We see that model candidate update is really a combination of subsumption and rejection.

4.6 IV-MGTP procedure

The following definition gives the procedural semantics of IV-MGTP programs. It is modeled after the high-level description of the standard MGTP procedure [2]. Let $M_\emptyset$ be the **undefined model candidate**, where $M_\emptyset(p(t_1, \ldots, t_r))$ is undefined for all $t_1, \ldots, t_r \in Term_\Sigma^0$ and "declare $p(t, \ldots, t, j_1, \ldots, j_m)$". For a given IV-MGTP program P, let $\mathcal{M}$ be a set of IV-MGTP model candidates, inductively defined as follows:

Initialization $M_\emptyset \in \mathcal{M}$.

Update $M \in \mathcal{M}$ is not rejected, CJM can be applied to the antecedent of $C \rightarrow D \in P$ and M with substitution σ, $D\sigma$ is not subsumed by M, and not all constrained atoms in $D\sigma$ are inconsistent with M. Let $D'\sigma \neq \bot$ consist of the atoms in $D\sigma$ consistent with M.

Then M is **extendible (with** $D'\sigma$**)**, the elements of $\mathcal{M}' = \bigcup_{L \in D'\sigma}(M + L)$ are called **immediate successor of** M, and $\mathcal{M}$ becomes $(\mathcal{M} - \{M\}) \cup \mathcal{M}'$.

Rejection CJM can be applied to the antecedent of $r = C \rightarrow D \in P$ and M with substitution σ and all constrained atoms in $D\sigma$ are inconsistent with M(This is in particular the case when $D = \bot$). Then M is **rejected.**

Those elements of $\mathcal{M}$ that are neither rejected nor extendible are the **IV-MGTP models** of P.

5. Soundness and Completeness

We have proved the following soundness property of the IV-MGTP procedure.

Theorem 1 (Soundness)*If P is a satisfiable IV-MGTP program, then P has an IV-MGTP model.*

As for the completeness, IV-MGTP cannot be complete with respect to the semantics that has been used up to now. The reason is essentially the same as for incompleteness of resolution and hypertableaux with unrestricted selection function [4]. It can be demonstrated with the simple example $P = \{\top \rightarrow p, \neg q \rightarrow \neg p, q \rightarrow \bot\}$. The program P is unsatisfiable, yet deduction procedures based on selection of only antecedent (or only succedent) literals cannot detect this. Likewise, the incomplete treatment of negation in CMGTP comes up with the incorrect model $\{p\}$ for P. The example can be transferred to IV-MGTP. Assume p and q are defined "declare $p(2)$" and "declare $q(2)$". The idea is to represent a positive literal p with $p(2)$ and a negative literal $\neg p$ with $p(1)$. Consider

$$P' = \{\top \rightarrow p(\{2\}), q(\{1\}) \rightarrow p(\{1\}), q(\{2\}) \rightarrow \bot\} \qquad (1)$$

which is unsatisfiable (recall that p and q are functional), but has an IV-MGTP model, where $M(p) = \langle\{2\}\rangle$, and $M(q)$ is undefined.

The example shows that already the ground Horn case with one constrained argument causes problems. Exactly this case is handled in SFL, where Lu [7] suggested a non-standard semantics called *extended interpretations* for SFLP that characterizes a ceratin procedural semantics. This procedural semantics, defined for the one-argument Horn case, happens to be quite similar to that of IV-MGTP.

There are more IV-MGTP models than satisfying interpretations as is exemplified by (1). So our solution is to admit additional interpretations that account for the remaining IV-MGTP models.

The basic idea underlying extended interpretations (briefly: e-interpretations) is to introduce the disjunctive information inherent to con-

straints into the interpretations themselves. Recall that IV-MGTP interpretations essentially are normal Herbrand interpretations. In contrast to this, model candidates contain IV-MGTP ground atoms with constraints. In e-interpretations we allow the same, formally, if p is defined "declare $p(t,\ldots,t,j_1,\ldots,j_m)$", then an e-interpretation of p is a partial function $\mathbf{I}$ mapping ground instances of the term part of IV-MGTP atoms with predicate symbol p into $(2^{\{1,\ldots,j_1\}}-\{\emptyset\})\times\cdots\times(2^{\{1,\ldots,j_m\}}-\{\emptyset\})$.This is the same definition as used for model candidates. This means that we can use concepts introduced for model candidates for e-interpretations. In the following, we normally use M, M' for model candidates and $\mathbf{I}$, $\mathbf{I}'$ for e-interpretations, but it is sometimes convenient to use the same letter for both which we do without an explicit (but trivial) conversion function.

An extended interpretation $\mathbf{I}$ does **e-satisfy** an IV-MGTP ground atom $L{=}p(t_1,\ldots,t_r,S_1,\ldots,S_m)$ iff $\mathbf{I}(p(t_1,\ldots,t_r))$ is defined, has the value $\langle S'_1,\ldots,S'_m\rangle$, and $S'_i\subseteq S_i$ for all $1\leq i\leq m$. The concept of e-interpretation and e-satisfaction occurs already in [3], where it was used to simplify some definitions. Its relevancy for SFLP was first noticed in [7]. Equivalently, using the terminology of model candidates we could have expressed e-satisfaction as "CJM can be applied to L and $\mathbf{I}$". E-satisfaction of rules and programs is defined exactly as before, only relative to e-interpretations.

Our previous notation for updates (Section 4.5) can be generalized to cover extended interpretations: $\mathbf{I}+\mathbf{I}'=\mathbf{I}+\Sigma_{p\in\mathrm{Atom}^0_\Sigma}\mathbf{I}'(p)$.

Example 3 *Consider the single-rule programs P_1 containing $\top\rightarrow p(\{1\});p(\{2\})$ and P_2 containing $\top\rightarrow p([1,2])$. They cannot be distinguished with IV-MGTP interpretations: both are satisfied exactly by $\mathbf{I}_1=\{p(1)\}$ and $\mathbf{I}_2=\{p(2)\}$.The e-interpretations $\mathbf{I}'_1(p)=\langle\{1\}\rangle$ and $\mathbf{I}'_2(p)=\langle\{2\}\rangle$ corresponding to $\mathbf{I}_1$ and $\mathbf{I}_2$ satisfy P_1 and P_2 as well, but there is a further e-interpretation $\mathbf{I}'(p)=\langle\{1,2\}\rangle$ satisfying P_2, but not P_1.*

The relationship between interpretations and e-interpretations of IV-MGTP programs is stated in the next lemma. We say an e-interpretation (or a model candidate) $\mathbf{I}$ is **definite** iff $\mathbf{I}(p(t_1,\ldots,t_r))=\langle S_1,\ldots,S_m\rangle$ implies $|S_1|=\cdots=|S_m|=1$.

Lemma 1 *Let $\mathbf{I}$ be an IV-MGTP interpretation that satisfies the IV-MGTP program P. Then the e-interpretation $\mathbf{I}'$ defined as $\mathbf{I}'(p(t_1,\ldots,t_r))=\langle\{i_1\},\ldots,\{i_m\}\rangle$, if $\langle t_1,\ldots,t_r,i_1,\ldots,i_m\rangle\in\mathbf{I}(p)$, and that is otherwise undefined, e-satisfies P. If $\mathbf{I}'$ is a* definite *e-interpretation that e-satisfies P, then the interpretation $\mathbf{I}$ defined as follows satisfies P: $\langle t_1,\ldots,t_r,i_1,\ldots,i_m\rangle\in\mathbf{I}(p)$ iff $\mathbf{I}'(p(t_1,\ldots,t_r))=\langle\{i_1\},\ldots,\{i_m\}\rangle$*

Then we have proved the following completeness theorem.

Theorem 2 (Completeness) *An IV-MGTP program P having an IV-MGTP model M is e-satisfiable by M (viewed as an e-interpretation).*

6. Results

We developed an IV-MGTP prototype system in Java and made experiments on a Sun Ultra 5 under JDK 1.2. Here we consider channel routing problems in VLSI design. The results are compared with those on the same problems formulated and run with CMGTP [9] (also written in Java [5]).

Channel routing problems can be represented as constraint satisfaction problems, in which connection requirements (what we call *nets*) between terminals must be solved under the condition that each net has a disjoint path from all others. For these problems, many specialized solvers employing heuristics were developed. Our experiments are not primarily intended to compare IV-MGTP with such solvers, but to show the effectiveness of the interval/extraval representation and its domain calculation in the IV-MGTP procedure.

We consider a multilayer channel which consists of multiple layers, each of which has multiple tracks. We assume in addition, to simplify the problem, that each routing path contains no dog-legs and contains only one track. By this assumption, the problem can be formalized to determine the layer and the track numbers for each net with the help of constraints that express the two binary relations: *not equal* and *above*. *not equal*(N_1,N_2) means that the net N_1 and N_2 do not share the same track. *above*(N_1,N_2) means that if N_1 and N_2 share the same layer, the track number of N_1 must be larger than that of N_2 (trivially, the *not equal* relation includes the *above* relation).

For example, *not equal* constraints for nets N_1 and N_2 are represented in IV-MGTP as follows:

$p(N_1,[L,L],[T_1,T_1])$, $p(N_2,[L,L],[T_{21},T_{22}])$, $N_1\neq N_2\rightarrow p(N_2,[L,L],]T_1,T_1[)$

where the predicate p has two constraint domains: layer number L and track number T_i. We experimented with problems consisting of 6, 8, 10, and 12 net patterns on the 2 layers channel each of which has 3 tracks. The results are shown in **Table 1**.

Table 1 Experimental results for the channel routing problem

Number of Nets = 6

	IV-MGTP	CMGTP
models	250	840
branches	286	882
runtime(msec)	168	95

Number of Nets = 8

	IV-MGTP	CMGTP
models	1560	10296
branches	1808	10302
runtime(msec)	706	470

Number of Nets = 10

	IV-MGTP	CMGTP
models	4998	51922
branches	6238	52000
runtime(msec)	2311	3882

Number of Nets = 12

	IV-MGTP	CMGTP
models	13482	538056
branches	20092	539982
runtime(msec)	7498	31681

IV-MGTP reduces the number of models considerably. For example, we found the following model in a 6-net problem:

$$\{p(1,[1,1],[3,3]),\ p(2,[1,1],[1,1]),\ p(3,[1,1],[2,2]),\\ p(4,[2,2],[2,3]),\ p(5,[2,2],[1,2]),\ p(6,[1,1],[2,3])\}\ ,$$

which contains 8 ($= 1 \times 1 \times 1 \times 2 \times 2 \times 2$) CMGTP models. The advantage of using IV-MGTP is that the different feasible track numbers can be represented as interval constraints. In CMGTP, the above model is split into 8 different models. Obviously, as the number of nets increases, the reduction ratio of the number of models becomes larger. We conclude that IV-MGTP can effectively suppress unnecessary case splitting by using interval constraints, and hence, reduce the total size of proofs.

7. Summary

We have suggested the new approach to handle finite domain constraints in model generation manner, based on the interval/extraval representation. Our results on channel routing problems show considerable potential of the method.

Our framework is motivated by the SFLP. IV-MGTP atoms can be seen as a generalization of signed atoms in SFL to allow constraints in more than one place. As for the relation with CLP, bottum-up computation in IV-MGTP allows natural propagation of inconsistent constraints, while CLP needs other mechanisms to avoid redundant search such as dependency-directed backtracking. In addition, IV-MGTP gives the user explicit control over case-splitting in succedents of rules.

The elements of the constrained part in IV-MGTP literal are independent of each other, that is, a constrained part can only represent rectangles in domain spaces. We recognize the expansion to handle the dependency of domain spaces as a future work. Additionally, the language of IV-MGTP can be made richer. The more complicated constraints become, the more weight is shifted away from model generation towards constraint solving.

The performance of IV-MGTP should be improved. The strongly reduced number of generated models (see 6.) shows the IV-MGTP paradigm to be quite effective, but it needs to be more efficient to play out its full strength.

References

1) P. Baumgartner, U. Furbach, and I. Niemelä. Hyper tableaux. In *Proc. European Workshop: Logics in Artificial Intelligence, JELIA*, volume 1126 of *LNAI*, pages 1–17. Springer-Verlag, 1996.
2) H. Fujita and R. Hasegawa. A model generation theorem prover in KL1 using a ramified-stack algorithm. In *Proc. 8th Int. Conf. on Logic Programming, Paris/France*, pages 535–548. MIT Press, 1991.
3) R. Hähnle. *Automated Deduction in Multiple-Valued Logics*, volume 10 of *Int. Series of Monographs on Computer Science*. Oxford University Press, 1994.
4) R. Hähnle. Tableaux and related methods. In *Handbook of Automated Reasoning*. Elsevier Science Publishers, to appear, 2000.
5) R. Hasegawa and H. Fujita. A new implementation technique for model generation theorem provers to solve constraint satisfaction problems. Research Reports on Information Science and Electrical Engineering Vol. 4, No. 1, pp. 57–62, Kyushu University, 1999.
6) R. Hasegawa, H. Fujita, and M. Koshimura. MGTP: a model generation theorem prover—its advanced features and applications. In *Proc. Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1227 of *LNAI*, pages 1–15. Springer-Verlag, 1997.
7) J. J. Lu. Logic programming with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.
8) U. Montanari and F. Rossi. Finite Domain Constraint Solving and Constraint Logic Programming. In *Constraint Logic Programming: Selected Research*, pages 201–221. The MIT press, 1993.
9) Y. Shirai and R. Hasegawa. Two approaches for finite-domain constraint satisfaction problem: CP and MGTP. In *Proc. 12th Int. Conf. on Logic Programming*, pages 249–263. MIT Press, 1995.