Two Approaches for Finite-Domain Constraint Satisfaction Problems

Yasuyuki Shirai
Ryuzo Hasegawa
Institute For New Generation Computer Technology
{shirai,hasegawa}@icot.or.jp

Abstract

We have developed two types of systems; CP and CMGTP, for finite-domain constraint satisfaction problems. CP is based on the constraint logic programming scheme, and is written in SICStus Prolog. CP has achieved high performance on quasigroup (QG) existence problems in terms of the number of branches and execution time. CP succeeded in solving a new open quasigroup problem. On the other hand, CMGTP is a slightly modified version of our theorem prover MGTP (Model Generation Theorem Prover [1]), enabling negative constraint propagation and is written in SICStus Prolog and KL1 (parallel logic programming language). CMGTP has exhibited the same pruning ability as CP for QG problems. CMGTP can be used as a general constraint solver for finite-domains on which we can write down constraint propagation rules with CMGTP input clauses directly.

1 Introduction

We present an CLP based system CP and an extension of the MGTP system called CMGTP (Constraint MGTP), both of which can solve finite-domain constraint satisfaction problems efficiently such as quasigroup (QG) existence problems [2] in finite algebra.

In 1992, M. Fujita and J. Slaney[5] first succeeded in solving several open QG problems by using FINDER and MGTP. Later, it was shown that other systems such as DDPP or CHIP could solve QG problems more efficiently. Such research has revealed that the original MGTP lacks negative constraint propagation ability. This motivated us to develop an experimental system CP based on the CLP (constraint logic programming) scheme.

We found that the constraint propagation mechanism used in CP can be realized by the slightly modified MGTP system, called CMGTP.

2 Quasigroup Problems

A Quasigroup is a pair $\langle Q, \circ \rangle$ where Q is a finite set, $\circ$ a binary operation on Q and for any $a, b, c \in Q$,

$$a \circ b = a \circ c \Rightarrow b = c$$
$$a \circ c = b \circ c \Rightarrow a = b.$$

The multiplication table of this binary operation $\circ$ forms a latin square (shown in Fig.1).

In a quasigroup, we can define the following inverse operations $\circ_{ijk}$ called *(ijk)-conjugate*:

$$x \circ_{123} y = z \iff x \circ y = z$$
$$x \circ_{231} y = z \iff y \circ z = x$$
$$x \circ_{312} y = z \iff z \circ x = y$$

$\circ$	1	2	3	4	5
1	1	3	2	5	4
2	5	2	4	3	1
3	4	5	3	1	2
4	2	1	5	4	3
5	3	4	1	2	5

Figure 1: Latin square (order 5)

S–square : (123)-conjugate

$\circ$	1	2	3
1	V_{11} $(A_1\ A_2\ A_3)$	V_{12} $(B_1\ B_2\ B_3)$	V_{13} $(C_1\ C_2\ C_3)$
2	V_{21} $(D_1\ D_2\ D_3)$	V_{22} $(E_1\ E_2\ E_3)$	V_{23} $(F_1\ F_2\ F_3)$
3	V_{31} $(G_1\ G_2\ G_3)$	V_{32} $(H_1\ H_2\ H_3)$	V_{33} $(I_1\ I_2\ I_3)$

Figure 2: The variables in a third-order latin squares

Multiplication tables of the inverse operations defined above also form latin squares.

We have been trying to solve 7 categories of QG problems (called QG1, QG2,..., QG7), each of which is defined by adding some constraints to original quasigroup constraints. For example, QG5 constraint is defined as $\forall ab \in Q.\ ((ba)b)b = a$.

3 CP

While in CLP languages, domain variables are used to represent constraints, in CP, domain element variables are introduced as well as domain variables. Fig.2 shows the variables in a third-order latin square used in CP where domain variables V_{ij} range over $\{1,2,3\}$ $(1 \leq i,j \leq 3)$ and domain element variables $A_k, B_k, \ldots, I_k$ range over $\{yes, no\}$ $(1 \leq k \leq 3)$.

Let V be a domain variable whose domain is $\{1,2,\ldots,n\}$, and $(A_1, A_2, \ldots, A_n)$ be a vector of domain element variables w.r.t. V. The value of A_i determines whether $V = i(A_i = yes)$ or $V \neq i(A_i = no)$.

For QG problems, we maintain three squares according to (1,2,3)-, (2,3,1)- and (3,1,2)-conjugates. Domain element variables in these 3 squares can be linked (unified) with each other. (shown in Fig.2 and Fig.3).

Using shared (unified) variables facilitates constraint propagation like :

$$\forall abc.\ (a \circ_{123} b = c \Leftrightarrow b \circ_{231} c = a \Leftrightarrow c \circ_{312} a = b) \quad (1)$$

$$\forall abc.\ (a \circ_{123} b \neq c \Leftrightarrow b \circ_{231} c \neq a \Leftrightarrow c \circ_{312} a \neq b) \quad (2).$$

Ordinary CLP does allow constraint propagation like (1), but (2) is not possible in general because domain element variables cannot be handled directly. By this unification, CP can propagate negative information which can be overlooked in ordinary CLP systems.

I–square : (231)-conjugate

○	1	2	3
1	W_{11} $(A_1\ D_1\ G_1)$	W_{12} $(A_2\ D_2\ G_2)$	W_{13} $(A_3\ D_3\ G_3)$
2	W_{21} $(B_1\ E_1\ H_1)$	W_{22} $(B_2\ E_2\ H_2)$	W_{23} $(B_3\ E_3\ H_3)$
3	W_{31} $(C_1\ F_1\ I_1)$	W_{32} $(C_2\ F_2\ I_2)$	W_{33} $(C_3\ F_3\ I_3)$

R–square : (312)-conjugate

○	1	2	3
1	U_{11} $(A_1\ B_1\ C_1)$	U_{12} $(D_1\ E_1\ F_1)$	U_{13} $(G_1\ H_1\ I_1)$
2	U_{21} $(A_2\ B_2\ C_2)$	U_{22} $(D_2\ E_2\ F_2)$	U_{23} $(G_2\ H_2\ I_2)$
3	U_{31} $(A_3\ B_3\ C_3)$	U_{32} $(D_3\ E_3\ F_3)$	U_{33} $(G_3\ H_3\ I_3)$

Figure 3: The variables in (231)-, (312)-conjugate latin squares

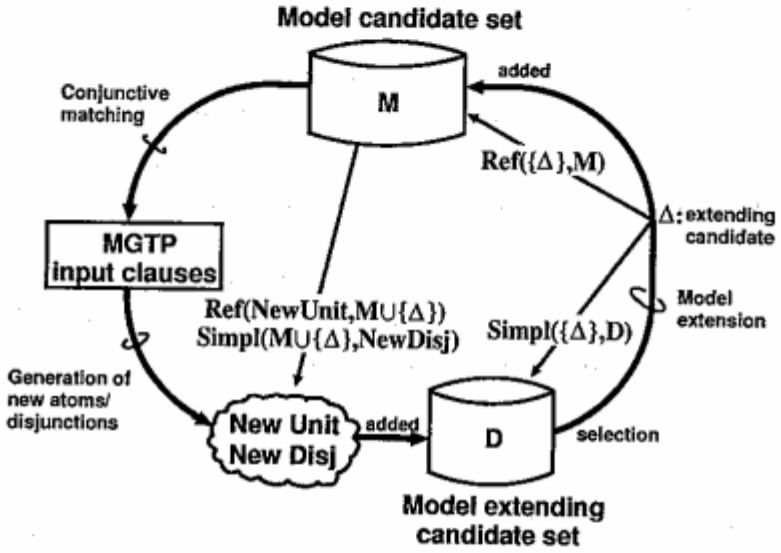

Figure 4: CMGTP model generation processes

4 CMGTP

Fig.4 shows the structure of CMGTP model generation processes which is basically the same as MGTP. The detail algorithm of MGTP is described in [1]. The differences between CMGTP and MGTP lie in the unit refutation processes and the unit simplification processes with negative atoms, introduced in CMGTP. We can use negative atoms explicitly to represent constraints in CMGTP. If there exist A and $\neg A$ in M then $false$ is derived by the unit refutation mechanism. If for a unit clause $\neg A \in M (A \in M)$, there exists a disjunction which includes $A(\neg A)$, then $A(\neg A)$ is removed from that disjunction by the unit simplification mechanism.

There are 2 refutation processes and 2 simplification processes added to the original MGTP:

- $Ref(\{\Delta\}, M)$
- $Ref(NewUnit, M \cup \{\Delta\})$
- $Simpl(\{\Delta\}, D)$

```
true → dom(1), dom(2), dom(3), dom(4), dom(5).
dom(M), dom(N) →
        p(M,N,1); p(M,N,2); p(M,N,3);
        p(M,N,4); p(M,N,5).
p(Y,X,V1), p(V1,Y,V2), p(V2,Y,V), {V\= X} → false.
p(X,X,V), {V\= X} → false.
p(X,Y1,V), p(X,Y2,V), {Y1\= Y2} → false.
p(X1,Y,V), p(X2,Y,V), {X1\= X2} → false.
p(X,5,Y), {X1 is X − 1, Y < X1} → false.
```

Figure 5: MGTP rules for QG5.5

- $Simpl(M \cup \{\Delta\}, NewDisj)$

where D is a *model extending candidate set*, M is a *model candidate set*, Δ is an atom that is picked up from D and put into M to perform conjunctive matching, and $NewUnit$ and $NewDisj$ are generated atoms and disjunctions by conjunctive matching.

$Ref(U_1, U_2)$, where U_1, U_2 are set of atoms, returns $false$ if there exist $A \in U_1, B \in U_2$, s.t., A and B are complementary. In this case, the branch is terminated with *unsat*. $Simpl(U, D)$, where U is a set of atoms and D is a set of disjunctions, returns the simplified set of disjunctions by a set of atoms U. If $false$ is derived as a result of simplification, then $Simpl$ returns $false$, and the branch is terminated with *unsat*. The function Ref can be considered as a special case of *simpl* function where D is restricted to a set of atoms.

As a result, these functions guarantee that for any atom $A \in M$, A and $\neg A$ are not both in the current M, and disjunctions in the current D have already been simplified by all atoms in M.

Fig.5 shows the original MGTP rules for QG5.5. These rules can be rewritten into CMGTP rules in order to propagate negative information using negative atoms. For example, the original MGTP rule for QG5,

$$p(Y, X, A), p(A, Y, B), p(B, Y, C), X \neq C \rightarrow false$$

can be rewritten in CMGTP rules as follows:

$$p(Y, X, A), p(A, Y, B) \rightarrow p(B, Y, X).$$
$$p(Y, X, A), \neg p(B, Y, X) \rightarrow \neg p(A, Y, B).$$
$$\neg p(B, Y, X), p(A, Y, B) \rightarrow \neg p(Y, X, A).$$

In the above rules, negative information is propagated by using the last 2 rules.

In this sense, CMGTP can be considered as a meta language for representing constraint propagation.

5 Experimental Results

Table 1 compares the experimental results for QG problems on CP, CMGTP and other systems. The numbers of failed branches generated by CP and CMGTP are

Table 1: Comparison of experimental results (QG5)

	Failed Branches				
Order	DDPP*	FINDER*	MGTP*	CP	CMGTP
9	15	40	239	15	15
10	50	356	7026	38	38
11	136	1845	51904	117	117
12	443	13527	2749676	372	372
13				13927	13927
14				64541	64541
15				130425	130425
16				19382469	

DDPP: Sparc2, FINDER: Sparc2, MGTP: PIM/m-256, CP: Sparc10 (*: [5])

almost equal to DDPP and less than those from FINDER and MGTP. In fact, we confirmed that CP and CMGTP have the same pruning ability as DDPP by comparing the proof trees generated by these systems. The slight differences in the number of failed branches were caused by the different selection functions used.

For general performance, CP was superior to the other systems in almost every case. In particular we found that no model exists for QG5.16 by running CP on a Sparc-10 for 21 days in October 1993. It was the first new result we obtained. On the other hand, CMGTP is about 10 times slower than CP. The reason of this difference is caused mainly by the manipulation of term memory. We are now trying to make term memory efficient in CMGTP, and also to parallelize CMGTP processes on PIM/m and parallel UNIX machines.

References

[1] H. Fujita and R. Hasegawa. A Model-Generation Theorem Prover in KL1 Using Ramified Stack Algorithm. *Proc. of the Eighth International Conference on Logic Programming*, The MIT Press, 1991.

[2] F. Bennett. Quasigroup Identities and Mendelsohn Designs. *Canadian Journal of Mathematics 41*, 341–368, 1989.

[3] P. V. Hentenryck, Constraint Satisfaction in Logic Programming. The MIT Press,1989.

[4] R. Manthey and F. Bry, SATCHMO: a theorem prover implemented in Prolog. *Proc. of CADE 88*, Argonne, Illinois, 1988.

[5] J. Slaney, M. Fujita, and M. Stickel, Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems. To appear in Computers and Mathematics with Applications.

[6] M. Fujita, J. Slaney, and F. Bennett, Automatic Generation of Some Results in Finite Algebra. *Proc. of International Joint Conference on Artificial Intelligence,* 1993.

[7] R. Hasegawa, Y.Shirai, Constraint Propagation of CP and CMGTP: Experiments on Quasigroup Problems. *Workshop 1C (Automated Reasoning in Algebra), CADE-12*, 1994.