ICOT Technical Memorandum: TM-1303

TM-1303

Constraint Propagation of CP and CMGTP: Experiments on Quasigroup Problems

by
R. Hasegawa & Y. Shirai

June, 1994

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5

Institute for New Generation Computer Technology

Constraint Propagation of CP and CMGTP: Experiments on Quasigroup Problems (Extended Abstract)

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

1 Introduction

Quasigroup (QG) existence problems[B89] in finite algebra are typical finite-domain constraint satisfaction problems, which have a reputation as being combinatorially intensive.

Several attempts have been made to solve open quasigroup problems [SFS93][FSB93]. M. Fujita and J. Slaney[FSB93] first succeeded in solving some open QG problems by using MGTP[FH91] and FINDER. It was found later that these problems can be solved more efficiently with the Davis & Putnam theorem prover DDPP developed by M. Stickel and the constraint logic programming (CLP) system, CHIP[CS89], developed at ECRC.

Such research has shown that MGTP lacks negative-constraint propagation ability. This motivated us to develop two types of systems: CP (Constraint Propagation) and CMGTP (Constraint MGTP). In this paper, we introduce the systems and show their effectiveness in solving QG problems.

2 CP

2.1 Key Features of CP

CP, which is based on the CLP scheme, is a very compact program for solving finite domain problems written in SICStus Prolog on Sparc workstations. CP has the following key features.

- Domain and domain element variables. For QC problems, we use three squares according to (1,2,3)-, (2,3,1)- and (3,1,2)-conjugates.
- A constraint propagation mechanism which uses the freeze facility of SICStus Prolog.

2.2 Variable Maintenance in CP

Figure 1 shows the variables in a third-order latin square used in CP for solving quasigroup problems, where domain variable V_{ij} ranges over $\{1,2,3\}(1 \geq i,j \geq 3)$ and domain element variable X_{ij}^{k} ranges over $\{yes, no\}(1 \geq k \geq 3)$.

Domain variables have the same meaning as in ordinary CLP. However, CP also introduces domain element variables for quick constraint propagation. If a domain element variable X_{ij}^{k} is bound to *yes*, V_{ij}'s value is fixed to k; if bound to $= no$, V_{ij} should not take k; and if it remains unbound, V_{ij} may take k.

In general, a variable V has the domain $\{1,2,\ldots,n\}$ with the corresponding domain element variables $\{X_1, X_2, \ldots, X_n\}$. From the finite domain property, if $n-1$ variables of $\{X_1, X_2, \ldots, X_n\}$

$\circ$	1	2	3
1	V_{11} $(X_{11}^1 X_{11}^2 X_{11}^3)$	V_{12} $(X_{12}^1 X_{12}^2 X_{12}^3)$	V_{13} $(X_{13}^1 X_{13}^2 X_{13}^3)$
2	V_{21} $(X_{21}^1 X_{21}^2 X_{21}^3)$	V_{22} $(X_{22}^1 X_{22}^2 X_{22}^3)$	V_{23} $(X_{23}^1 X_{23}^2 X_{23}^3)$
3	V_{31} $(X_{31}^1 X_{31}^2 X_{31}^3)$	V_{32} $(X_{32}^1 X_{32}^2 X_{32}^3)$	V_{33} $(X_{33}^1 X_{33}^2 X_{33}^3)$

Figure 1: The Variables in a third-order latin square

except for X_k are bound to *no*, then X_k is bound to *yes* and variable V is bound to k. If V is bound to k, then X_k is bound to *yes* and $X_j (j \neq k)$ is bound to *no*.

For QG problems, inverse functions play a significant role in constraint propagation. Each function defines a different latin square, and domain element variables can be shared by these squares. Using shared variables facilitates constraint propagation like :

$$a \circ_{123} b = c \rightarrow b \circ_{231} c = a, c \circ_{312} a = b \quad (1)$$
$$a \circ_{123} b \neq c \rightarrow b \circ_{231} c \neq a, c \circ_{312} a \neq b \quad (2)$$

where $\circ_{231}$ and $\circ_{312}$ are inverse operations of $\circ_{123}$. Ordinary CLP does allow constraint propagation like (1), and (2) is not possible, in general, because domain elements cannot be handled directly.

2.3 Experimental Results on CP

Table 1 compares experimental results for QG problems on CP and other systems. The numbers of failed branches generated by CP are almost equal to DDPP and less than those from FINDER and MGTP. In fact, we confirmed that CP has the same pruning ability as DDPP by comparing the proof trees generated by CP and DDPP for QG5. 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.

3 CMGTP

3.1 Key Features of CMGTP

MGTP is a full-first order theorem prover based on the model generation method[MB88]. A merit of solving QG problems by MGTP is that they can be described in first-order form. This enables concise description. For example, in the case of problem QG5, MGTP only requires seven input clauses. However, MGTP also has the demerit that it cannot propagate negative constraints since it is based on forward reasoning and only uses positive atoms.

To overcome this inability, we developed CMGTP (Constraint MGTP) in SICStus Prolog, with a slight modification to original MGTP. CMGTP introduces the following key features:

- Negative literals and the integrity constraint, $P, \neg P \rightarrow false$.
- Extended MGTP rules, such as $p, \neg r \rightarrow \neg q$ and $\neg r, q \rightarrow \neg p$ as additions to the original rule $p, q \rightarrow r$.

Table 1: Comparison of experimental results using CP and other systems

Problem		Models	Failed Branches				Run Time(sec)			
			DDPP*	FINDER*	MGTP*	CP	DDPP*	FINDER*	MGTP*	CP
QG1	7	8	353	628		354	87	3		12.26
	8	16	97521	129258	180440	78079	10260	853	1894	3323
QG2	7	14	364	808	1128	642	80	4	48	23
	8	2	83087	119141		167397	8109	816		8177
QG3	8	18	1037	801	399	251	76	4	28	6
	9	0	46748	35473	312321	11030	5221	244	1022	283
QG4	8	0	970	989	3516	338	71	5	23	7
	9	178	58711	68550	315100	20295	6115	478	1127	456
QG5	9	0	15	40	239	16	25	2	12	1
	10	0	50	356	7026	46	66	8	66	2
	11	5	136	1845	51904	94	228	26	224	6
	12	0	443	13527	2749676	268	883	158	13715	14
	13	0				6466				467
	14	0				34835				2308
	15	0				130425				11218
	16	0				19382469				1831452
QG6	9	4	13	97	164	13	19	1	14	1
	10	0	65	640	2881	53	51	4	43	2
	11	0	451	4535	50888	474	299	25	248	13
	12	0	5938	73342	2429467	5573	5180	501	8300	134
QG7	9	4	9	62	37027	10	17	2	90	1
	10	0	40	289	1451992	37	47	4	2803	2
	11	0	321	1526		236	304	6		11
	12	0	2083	10862		1973	2291	146		113
	13	64	61612	141513		34206	99366	1984		2394

DDPP: Sparc2, FINDER: Sparc2, MGTP: PIM/m-256, CP: Sparc10 (*: [SFS93])

- Unit simplification is performed between unit literals in the model candidate set and disjunctive literals in the model-extending candidate set.

Figure 2 shows the CMGTP input clauses for QG5.5. As shown in this figure, constraint propagation rules can be written directly with MGTP input clauses.

3.2 Experimental Results on CMGTP

Table 2 compares experimental results for QG5 problems on CMGTP and CP with the same selection function. The numbers of failed branches are the same. We confirmed that CMGTP and CP generate identical proof trees when they use the same selection function. When it comes to CPU time, however, CMGTP is about 10 times slower than CP for problem order from 7 to 14.

Table 3 compares the execution time profiles of CP and CMGTP for QG5.10. For finite domain checking and candidate selection, both systems took almost the same time. The main speed difference occurs during term memory manipulation.

Another major speed difference occurs when updating the disjunction DB. CMGTP maintains disjunctive literals using a disjunction database. CP does not manipulate disjunctions explicitly.

In CP, constraint propagation is controlled by a freeze facility; in CMGTP, it is controlled by conjunctive matching. CMGTP handles unit conflict tests by conjunctive matching, while CP uses unification failure.

The current implementation of CMGTP is only a primary version. We expect that its performance can be further improved by a factor of 3 or 4.

$$true \rightarrow dom(1), dom(2), dom(3), dom(4), dom(5).$$
$$true \rightarrow p(1,1,1), p(2,2,2), p(3,3,3), p(4,4,4), p(5,5,5).$$
$$dom(M), dom(N), \{M \backslash = N\} \rightarrow p(M,N,1); p(M,N,2); p(M,N,3); p(M,N,4); p(M,N,5).$$
$$dom(M), dom(N), \{M \backslash = N\} \rightarrow p(M,1,N); p(M,2,N); p(M,3,N); p(M,4,N); p(M,5,N).$$
$$dom(M), dom(N), \{M \backslash = N\} \rightarrow p(1,M,N); p(2,M,N); p(3,M,N); p(4,M,N); p(5,M,N).$$
$$p(M,N,X), dom(M1), \{M1 \backslash = M\} \rightarrow \neg p(M1,N,X).$$
$$p(M,N,X), dom(N1), \{N1 \backslash = N\} \rightarrow \neg p(M,N1,X).$$
$$p(M,N,X), dom(X1), \{X1 \backslash = X\} \rightarrow \neg p(M,N,X1).$$
$$dom(X), dom(Y), \{X1 \text{ is } X-1, Y < X1\} \rightarrow \neg p(X,5,Y).$$

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

Figure 2: CMGTP rules for QG5.5

Table 2: Comparison of CMGTP with CP

QG5	M	CMGTP		CP	
		F.B.	Time	F.B.	Time
7	3	2	2.60	2	0.23
8	1	9	3.03	9	0.30
9	0	15	5.87	15	0.45
10	0	38	14.06	38	1.71
11	5	117	60.45	117	6.75
12	0	372	197.63	372	19.79
13	0	13914	9975.68	13914	1033.63
14	0	64541	42167.29	64541	5081.03

CMGTP, CP: Sparc 10

M: Number of models
F.B: Number of failed branches
Time: CPU time (second)

Table 3: Execution time profile

Modules	CMGTP	CP
Control	32061	1357
Term memory	354916	1967
Finite domain check	36050	32811
Update disjunction DB	86809	0
Constraint propagation	58899	5157
Unit conflict test	30542	0
Candidate selection	20883	23001
	620160	64293

4 Future Work

We have shown that both CP and CMGTP have the same constraint propagation ability as DDPP for pruning the search tree spaces in QG problems. We need to now improve CMGTP performance by refining implementation techniques. At present, CMGTP can handle forward checking in CLP. For further improvements, we are now investigating how to incorporate other facilities, such as the CLP lookahead mechanism, into CMGTP.

References

[FH91] H. Fujita and R. Hasegawa, A Model-Generation Theorem Prover in KL1 Using Ramified Stack Algorithm, In *Proc. of the Eighth International Conference on Logic Programming*, The MIT Press, 1991.

[B89] F. Bennett, Quasigroup Identities and Mendelsohn Designs, In *Canadian Journal of Mathematics* 41, pp.341-368, 1989.

[CS89] P. V. Hentenryck, *Constraint Satisfaction in Logic Programming*, The MIT Press,1989.

[MB88] R. Manthey and F. Bry, SATCHMO: a theorem prover implemented in Prolog, In *Proc. of CADE 88, Argonne, Illinois*, 1988.

[SFS93] J.Slaney, M.Fujita, and M.Stickel, Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems, 1993.

[FSB93] M. Fujita, J. Slaney, and F. Bennett, Automatic Generation of Some Results in Finite Algebra, In *Proc. of International Joint Conference on Artificial Intelligence*, 1993.