PARALLEL PROCESSING OF RESOLUTION

Takahira YAMAGUCH1*, Yoshikazu TEZUKA** and Osamu KAKUSHO*

* The ItistiLute of Scientific and Industrial Research, Osaka University, 8-1 Mihoqaoka, Ibaragi, Osaka 567, Japan
** Faculty of Engineering, Osaka University, ?-'\ Yamadaoka, Suita, Osaka 565, Japan

ABSTRACT

In this paper, PArallel Resolution Algorithm (PARA) is described to improve the execution efficiency in resolution process. PARA consists of two parts: parallel unification and generation of a resolvent. The first part is characteristic of PARA, which partitions whole set of expressions W into independent clusters as pre-processing and unifies each cluster in parallel. The efficient implementation for the processing peculiar to PARA is presented and checked by means of the experiment in comparison of execution efficiency of resolution. Experimental results show PARA is very effective in occurrence of many clusters.

1.INTRODUCTION

Unification in first-order logic is to match corresponding arguments in two predicates. This processing is important in resolution [1] and the efficient implementation for it has been the subject, of much investigation [2]-[5I. But this research in unification was out of parallel-process ing.

With regard to unification (resolution) parallelism, two representative ways, the post-processing way and the pre-processing way, would be considered. In case of the former, after unification, the consistency of substitution must be checked and each substitution component must be composed. This processing would be computed with unification and so the post-processing way is considered to cost much. On the contrary, in case of the latter, the pre-processing way to partition a set of expressions firstly could cost little, because this processing could be computed without unification.

from this consideration, in this paper, PARA is presented as one of the pre-processing ways. The characteristic of PARA lies in partitioning whole set of expressions W (the pairs of corresponding arguments in resolved literals) into clusters (sets of pairs of arguments such that each set has no variables in common) as pre-processing and unifying each cluster independently in parallel.

2.PARALLEL RESOLUTION ALGORITHM

Resolution consists of unification and generation of a resolvent. It costs more time to execute unification and so unification must be executed efficiently. The unification problem can be expressed as simultaneous equations [5] and the solution of them can be considered as most general unifier (mgu). The ordinary serial unification is process which solves simultaneous equations sequentially. They could, however, be divided Into subsets of equations which are parallel-processed independen tly. Parallel resolution consists of two parts: parallel-processing of unification and generation of a resolvent. Flg.1 shows PARA. Firstly PARA partitions whole sets of expressions W into clusters of WlAWm which include no variables in common as pre-processing. Secondly, after this pre-processing, PARA tries to unify each cluster independently in parallel. If all clusters are unifiable, PARA obtains the mgu of W by uniting the mgus of all clusters. Otherwise PARA concludes that W is not unifiable. Finally, after having obtained the mgu of W, PARA generates the resolvent and Clustering Information (which will be described later).

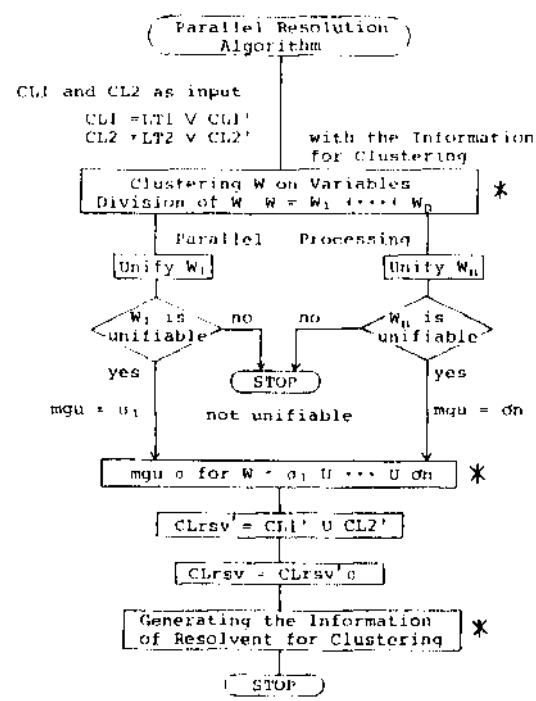

Fig.1 PArallel Resolution Algorithm (PARA)

3.EFFICIENT PROCESSING FOR PECULIAR PHASE IN PARA

The phases with * sign in Fig.1 are peculiar to PARA. Among these phases, clustering of W and generation of Clustering Information must be processed efficiently.

3.1 CLUSTERING INFORMATION

Since W is partitioned based on variables, Clustering Information will have to include the following information in order to execute clustering of W efficiently .

(1) Variable Information (of argument)
(2) Cluster Information (of literal)

Variable Information shows what variables each argument of literal has. Variable Information is stored in a form easy to treat. Cluster Information shows how one literal can be divided. Clustering Information consists of the above two pieces of information and the number of them. This is stored corresponding to each literal, as shown in Fig.2.

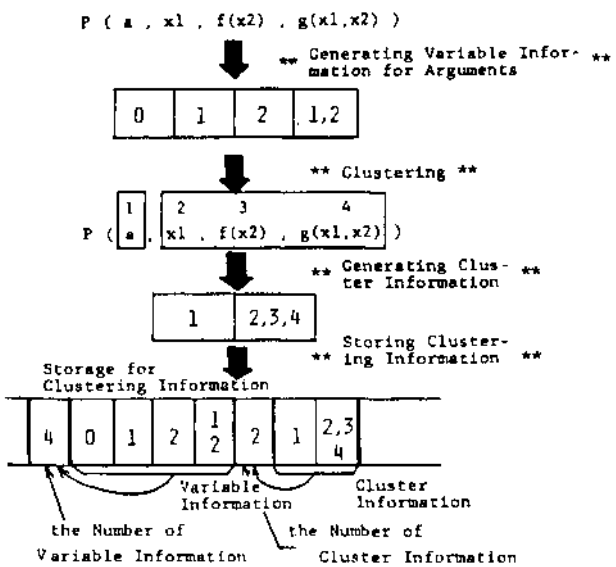

Fig.2 Data Structure for Clustering Information

3.2 CLUSTER OPERATION ALGORITHM

Clustering of W is executed by means of applying cluster operation to Cluster Information of resolved literals. Cluster operation is the process which makes clusters for parallel unification. It matches n-th component of one Cluster Information with all components of the other and combines matched components. P(a,g(x1),x2,f(g(x2))) and P(x4,x3,f(x4),f(x3)) turn out not to be divided divided by means of applying cluster operation to Cluster Information of them.

3.3 CLUSTERING INFORMATION GENERATION ALGORITHM

Generating Clustering Information is executed by means of making use of Variable Information of parent clauses.

*** Clustering Information Generation Algorthim ***

Step 1 : Delete Variable Information of resolved literals of parent clauses. The rest is Variable Information of CLrsv' which is an incomplete resolvent which mgu is not applied.

Step 2 : Generate the pairs of the variable number (natural number corresponding to each variable) included in "term" of mgu and the variable number corresponding to the substituted variable of mgu.

Step 3 : Apply Variable Information of mgu to Variable Information of CLrsv' and generate Variable Information of CLrsv which is a complete resolvent.

Step 4 : Generate Cluster Information of first literal by means of clustering Variable Information.

Step 5 : Connect Variable Information of and Cluster Information of and let them Clustering Information of CLrsv.

4.COMPARISON OF PARA AND SRA

To compare PARA with SRA (Serial Resolution Algorithm), 4 logic programs [6] have been run on two algorithms using theorem proving system "SENRI" [6]. The run time for each process has been measured with executing SNL resolution and the total time for resolution process has been finally compared. Table 1 shows the experimental results.

PARA has about 2.1-fold improvement in unification process and about 1.8-fold improvement in resolution process over SRA on the average.

Comparing the run time for generating resolvents in two algorithms, the run time of PARA is about twice one of SRA. This implies that the run time for generating Clustering Information is as small as one of generating resolvents and so does not affect the total time so badly.

Since the redundancy for generating Clustering Information adds to resolution process, the improvement rate of resolution process becomes worse. The improvement rate of resolution process is, however, around 0.7-times of the number of clusters. Moreover the improvement rate of both unification process and resolution process rises in proportion to the number of clusters.

5. COMMENTS

In this experiment, some pairs of resolved literals, in which PARA becomes effective or ineffective, emerged definitely. Table 2 reports these literal pairs in detail.

(1) *Literal Pair 1* (in which resolution succeeds and PARA becomes effective)
-> *MIC0M(FM8,FUJITSU,6809,218000)*
MIC0M(XI, X2 , X3 ,218000)

Literal Pair 1 is divided into 4 clusters at the pre-processing. Since the mgu of each cluster has only substitution component, the run time for substitution is short. So parallel resolution becomes very effective.

(2) *Literal Pair 2* (in which resolution fails and PARA becomes effective)

-> *MICOM(MZ80B,SHARP,Z80A,278000)*
MICOM(X1 , X2 , X3 ,218000)

Literal Pair 2 is divided into 4 clusters at the pre-processing and the computation terminates at the time when 4-th cluster proves not to be unifiable. So parallel resolution becomes very effective.

(3) *Literal Pair 3* (in which resolution fails and PARA becomes ineffective)

-> *MAKER(OKI ,TOKYO, E)*
MAKER(SHARP, X1 ,X2)

Literal Pair 3 proves not to be unifiable at the first argument in SRA. So PARA becomes ineffective due to the extra time for clustering. Besides Literal Pair 3, literal pairs which are not divided at the pre-processing are similar.

(4) *Literal Pair*(in which resolution succeeds and PARA becomes ineffective)

This literal pair is one which is not divided at the pre-processing(not shown in Table 2).

Table 1 Comparison of SRA and PARA

Comparison items \ Problems		MEMBER	APPEND	LOOP	DB
The number of resolution trial		6	5	3	57
The number of resolution success		4	3	8	4
The number of clusters for pairs of resolved literals		1-2	2	2-3	3-4
S R A	The total time for unification process	6000	10245	29605	11820
	The total time for generating resolvents	490	485	1980	1335
	The total time for resolution process	6490	10730	31585	13155
P A R A	The total time for unification process	4220	5775	12550	4560
	The total time for generating resolvents	1025	1290	3830	1785
	The total time for resolution process	5245	7065	16380	6345
I R	Improvement rate in unification process	1.42	1.77	2.36	2.59
	Improvement rate in resolution process	1.24	1.52	1.93	2.07

Table 2 Comparison of Some Pairs of Resolved Literals

Pair of resolved literals \ Comparison items	Literal Pair 1	Literal Pair 2	Literal Pair 3
T_{SRA}	1075	1035	60
T_{PARA}	535	195	150

unit time : usec
machine : ACOS system 1000 (NEC)
S R A : Serial Resolution Algorithm
P A R A : PArallel Resolution Algorithm
I R : Improvement Rate

4 logic programs have literal pairs inconvenient for PARA as well as convenient for PARA and so the experiment is fair. Since the decrease in cost caused by convenient literal pairs is over the increase in cost caused by Inconvenient literal pairs, PARA is effective to ordinary problems which include inconvenient literal pairs as well as convenient literal pairs.

6.CONCLUSION

PARA has been introduced for the improvement of execution efficiency for resolution process. It is characteristic of PARA to partition whole set of expressions W into independent clusters as pre-processing and unify each cluster in parallel.

The experimental results in comparison of execution efficiency for resolution show the following.

(1) The processing peculiar to PARA has been implemented efficiently.
(2) The improvement rates of both unification process and resolution process by parallel-processing rise in proportion to the number of clusters.
(3) Since the decrease in cost caused by convenient literal pairs is over the increase in cost caused by inconvenient literal pairs, PARA is effective to ordinary problems.

Finally, unification in this paper is limited to be basic unification in [2]. Parallel-processing of refined unification in [4] or [5] would, however, be possible and further improvement could be expected.

REFERENCES

(1) Kowalski, R.A. :" Logic for Problem Solving ",North Holland (1979)
(2) Robinson, J.A. :" A Machine Oriented Logic Based on Resolution Principle ", J.ACM,12, pp.23-41 (1965)
(3) Robinson, J.A. :" Computational Logic : The Unification Computation ", Machine Intelligence 6, pp.63-72,Edinburg Univ.Press (1971)
(4) Paterson, M.S. and Wegman,M.N. :" Linear Unification ", Proc.8th Annual ACM Symp. on Theory of Computing, pp.181-186 (1976)
(5) Martelli, A. and Montanari, U. :" An Efficient Unification Algorithm ", ACM Trans, on Programming Lang, and Syst.,4,2 pp.258-282 (1982)
(6) Yamaguchi, T. :" The Efficient Implementation for Predicate Logic Based System " Univ. of Osaka, Dr. dissertation (dec. 1983) (in Japanese)