Parallel Resolution Algorithm for Logic Machine

Takahira Yamaguchi　Seiichiro Dan
Seiichi Uchinami　Yoshikazu Tezuka
(Faculty of Engineering Osaka University)

A B S T R A C T

In this paper, we present Parallel Resolution Algorithm (PARA) for the improvement of execution efficiency in resolution process. PARA clusters W (the pairs of corresponding arguments in resolved literals) by means of variables as preprocessing and tries to unify each cluster independently in parallel. The experiment in comparison of execution efficiency shows PARA has about 1.77-fold improvement over SRA (Serial Resolution Algorithm). Finally we show PARA is very effective in occurence of plual sets of clusters.

1. Introduction

Some parallelisms exist in Programming Language Based on Predicate Logic. These ones are (1) And parallelism, (2) Or parallelism, (3) Stream parallelism, (4) Resolution parallelism. However the detailed study of Resolution parallelism hasn't been found so far. So in this paper, we present Parallel Resolution Algorithm and show the efficiency of this algorithm in experimental comparison with ordinary serial resolution algorithm.

2. Parallel Oriented Unification Algorithm [2][3]

The unification probelm can be written as simultaneous equations and the solution of them can be considered as mgu. The ordinary serial unification is a process which solves simultaneous equations sequencially. However we can divide them into subsets of equations (clusters) which are parallel-processed independently. Parallel Oriented Unification Algorithm (POUA) is based on above notion and Fig.2 shows how POUA is executed.

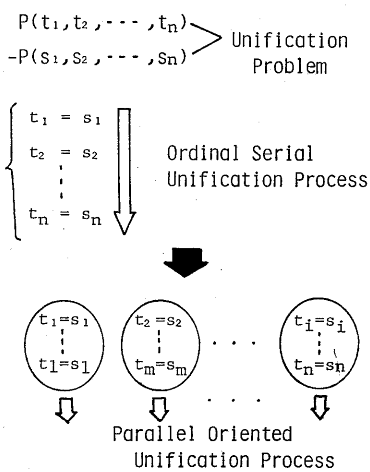

Fig.1 Unification Process

3. Parallel Resolution Algorithm

Parallel resolution process adds resolvent generating one to parallel unification one. Fig.3 shows Parallel Resolution Algorithm (PARA). Firstly PARA clusters W (the pairs of corresponding arguments in resolved literals) by means of variables and generates clusters of W_1-W_n which are parallel-processed independently, 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 Informations of it. Clustering Informations include Variable Informations and Cluster Informations. Variable Informations show what variables each argument has. Cluster Informations show how one literal can be divided.

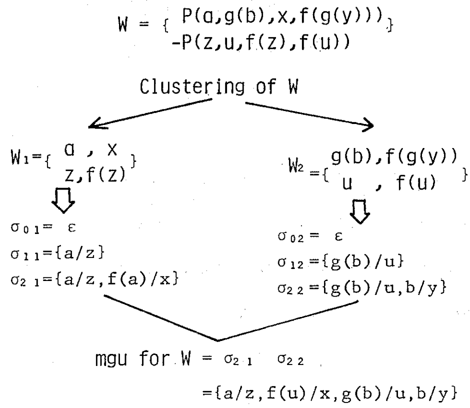

Fig.2 An Example of Parallel Oriented Unification Process

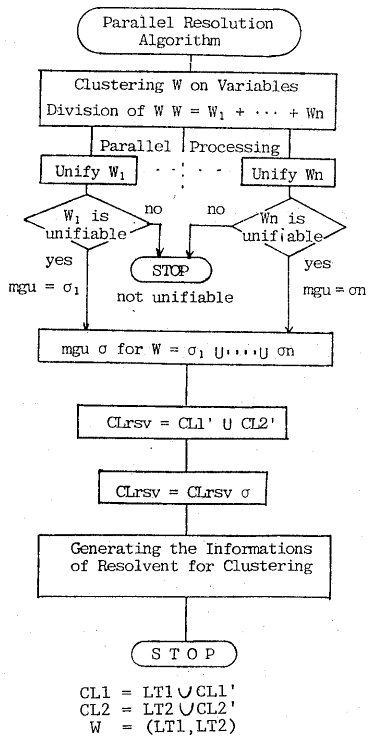

Fig.3 Parallel Resolution Algorithm

4. Efficient Algorithm for Peculiar Processes

Since clustering of W and generation of clustering informations are peculiar processes which don't exist in the ordinary resolvent algorithm, they must be processed efficiently. So we present efficient algorithms for two processes.

4-1 Cluster Operation Algorithm

Clustering of W is executed by means of doing cluster operation to cluster informations of resolved literals. Cluster operation is the process which forms clusters for parallel unification. It matches n-th component of one cluster information with all components and combines matched components (Fig.4). Fig.5 shows W isn't divided by means of doing cluster operation to cluster informations of P(a,g(x1), x2,f(g(x2))) and -P(x4,x3,f(x4),f(x3)).

4-2 Clustering Informations Generation Algorithm

Generating clustering informations is executed by means of making use of variable informations of parent clauses. Fig.6 shows the general-flow of this algorithm and Fig.8-1~8-3 show how each phase of this algorithm is executed in the example of Fig.7.

*** Clustering Informations Generation Algorithm ***

Phase (1) : Extract LT12 and LT22 from variable informations for parent clauses and connect them (Fig.4-1).

Phase (2) : Generate the pairs of variable number and variable number with

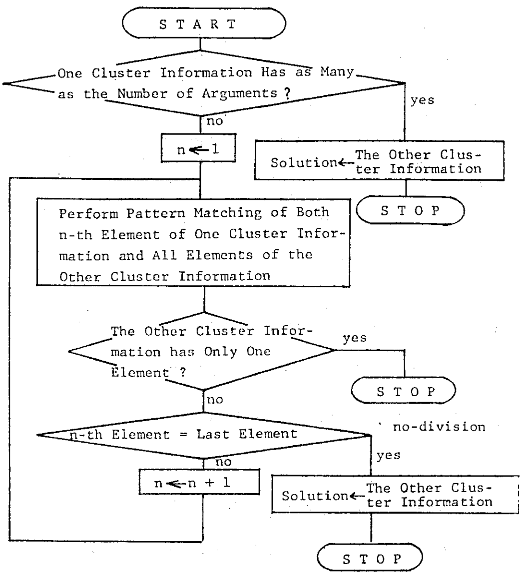

Fig.4 Cluster Operation Algorithm

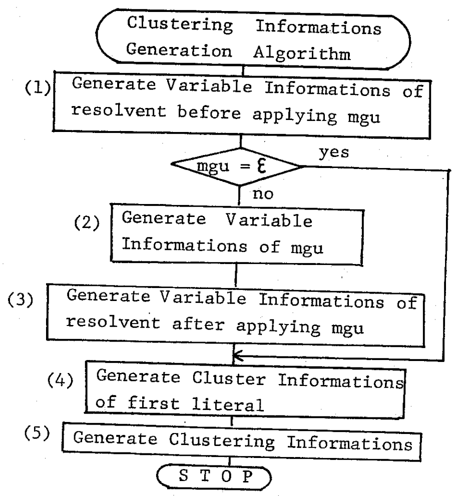

Fig.6 Clustering Informations Generation Algorithm

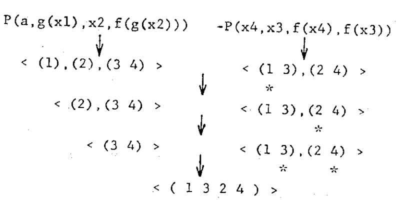

Fig.5 Example for Cluster Operation

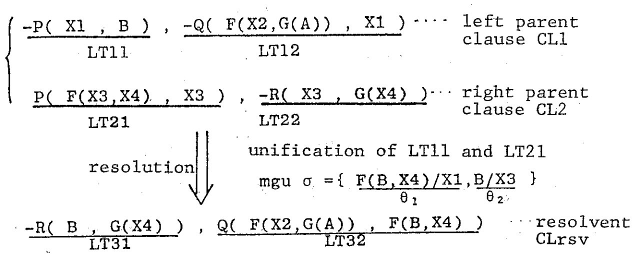

Fig.7 An example of resolution process

term to substitution components of mgu (Fig.8-2).

Phase (3) : Apply variable informations of mgu to variable informations of CLrsv' and generate variable informations of CLrsv (Fig.8-3)

Phase (4) : Generate clustering informations of first literal by means of clustering variable informations of it.

Phase (5) : Connect variable informations of (3) and cluster informations of (4) and let them clustering informations of the resolvent CLrsv.

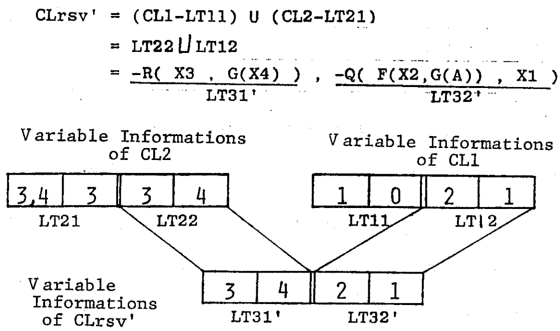

Fig.8-1 The processing of phase (1)

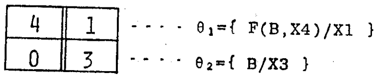

Fig.8-2 The processing of phase (2)

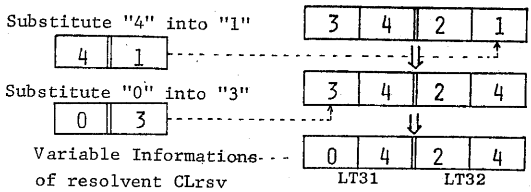

Fig.8-3 The processing of phase (3)

5. Comparison of SRA and PARA and its Remarks

To compare PARA with SRA(Serial Resolution Algorithm), we ran 4 logic programs [3] on two algorithm by modifying Theorem proving System "SENRI" [1]. We measured run time of each process with executing SNL resolution and compared the total time of resolution process finally. Table 1 shows the experimental result.

Comparing the total times of generating resolvents of two algorithm, we see that one of PARA is about twice one of SRA. This suggests that run time of generating clustering informations is as small as one of generating resolvents and doesn't affect the total run time so badly.

Comparing improvement rates of unification process and resolution process, since the redundant process to generate clustering informations participates in resolution process, one of resolution process becomes worse. However increase time by generating clustering informations is muvh smaller than decrease time by parallel processing an unification. So improvement rate of resolution process is about 0.8 times as much as one of unification process and 0.6-0.7 times as many as the number of clusters.

6. Comments

6-1 The literal pairs in which PARA becomes effective or ineffective

In above experiment, some pairs of resolved literal in which PA A becomes effective or ineffective emerged definitely.

(1) Literal Pair 1 in which resolution succeeds and PARA becomes effective

MICOM(FM8, FUJITSU,6809,218000)
MICOM(X1, X2 , X3 ,218000)

Literal Pair 1 is divided into 4 clusters at the pre-processing. Since mgu of each cluster has only substitution component, altering structure in each cluster is unnecessary. So parallel resolution became very effective.

(2) 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 S R A and PAR A

Comparison items \ Problems		MEMBER	APPEND	LOOP	D B
The number of times of resolution trial		6	5	34	57
The number of times of resolution success		4	3	8	4
The number of clusters for pairs of resolved literals		1～2	2	2～3	3～4
SRA	The total time of unification process	6,000	10,245	29,605	11,820
SRA	The total time of generating resolvents	490	485	1,980	1,335
SRA	The total time of resolution process	6,490	10,730	31,585	13,155
PARA	The total time of unification process	4,220	5,775	12,550	4,560
PARA	The total time of generating resolvents	1,025	1,290	3,830	1,785
PARA	The total time of resolution process	5,245	7,065	16,380	6,345
IR	Improvement rate in unification process	1.42	1.77	2.36	2.59
IR	Improvement rate in resolution process	1.24	1.52	1.93	2.07

unit time : μsec
machine : ACOS system 1000 model 40

S R A : Serial Resolution Algorithm
PAR A : Parallel Resolution Algorithm
I R : Improvement Rate

(3) 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 computation terminates at the time when 4-th cluster proves not to be unifiable. So parallel resolution became very effective.

(4) 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 takes an extra time of clustering. Besides Literal Pair 3, literal pair which is not divided at the pre-processing is contained naturally.

Table 2 Comparison of some pairs of resolved literal

pairs of resolved literal / Comparison items	Literal Pair 1	Literal Pair 2	Litearl Pair 3
T_{SRA}	1 0 7 5	1 0 3 5	6 0
T_{PARA}	5 3 5	1 9 5	1 5 0

unit time : u sec
machine : ACOS system 1000 model 40

6-2 Efficiency improvement by making use of variable informations

We can make use of variable informations with arguments and mgu for efficiency improvement of resolution process.

(1) Efficiency improvement for occur check

We can execute occur check efficiently by logical product of variable informations with corresponding arguments (Fig.9-1)

(2) Efficiency improvement for composition of substitution

We can execute composition of substitution by logical product of variable informations with substitution components (Fig.9-2).

(3) Efficiency improvement for generating resolvent

We can access only necessary arguments for generating resolvent by logical product of variable informations with arguments and variable informations with mgu (Fig.9-3).

We need examine above efficiency improvements by experiment for further research.

7. Conclusion

Firstly PARA proves to be very effective in the case in which literal pair is divided into plural clusters and plural substitu-

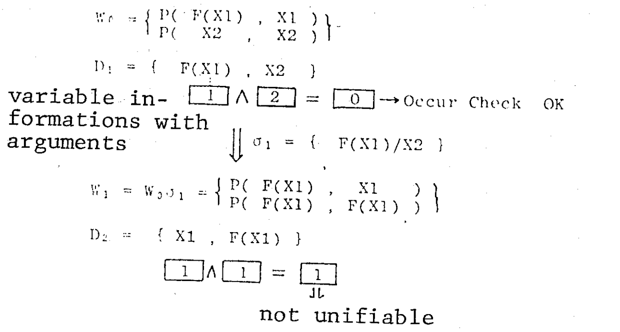

Fig.9-1 Efficiency improvement for occur check

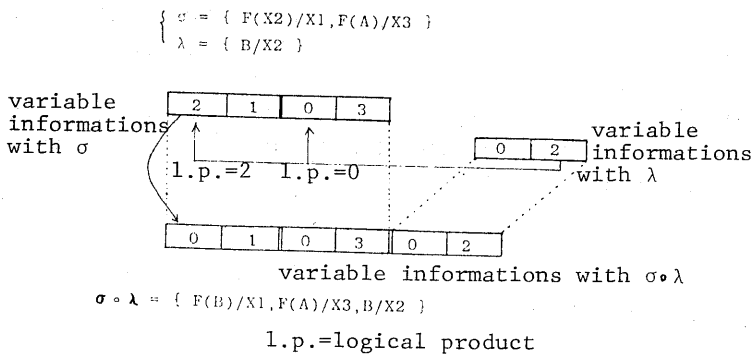

Fig.9-2 Efficiency improvement for composition of substitution

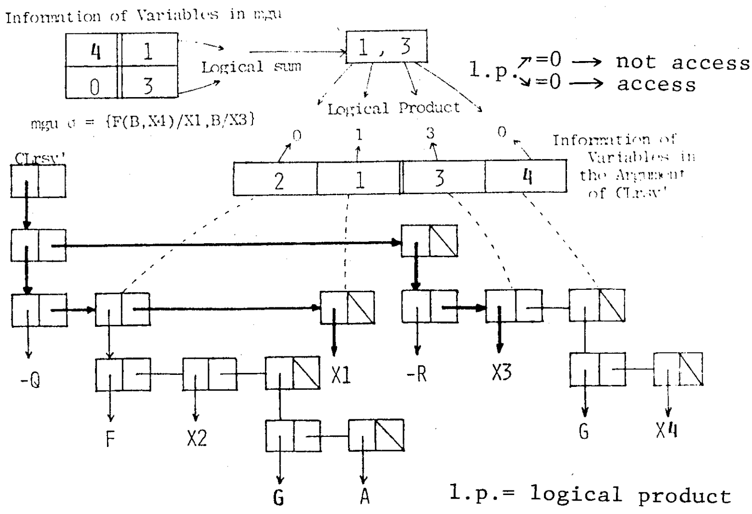

Fig.9-3 Efficiency improvement for generating resolvent

tions are done by deciding unifiability.

Secondly the experiment in comparison of execution efficiency shows PARA has about 1.77-fold improvement over SRA (the number of clusters is 1-4) and improvement rate becomes better in proportion to the number of clusters. So PARA considers to be effective for logic programs which have many arguments and in which each argument is independent mutually.

Finally since resolution process occupies much of the whole process in Programming Language Based on Predicate Logic, PARA contributes to the organization of Logic Machine.

References

(1) Yamaguchi, T., Nishioka, H., Uchinami, S., Tezuka, Y. :"The Organization of Theorem Proving System "SENRI"" IPSJ,WGAI, 22-1, Sep. 1981, (in Japanese)

(2) Yamaguchi, T., Uchinami, S., Tezuka, Y., :"Parallel Oriented Unification Algorithm and its Evaluation" IECE, Japan, AL82-23, pp.71-78, Jul. 1982, (in Japanese)

(3) Dan, S., Yamaguchi, T., Uchinami, S., Tezuka, Y., :"Experimental Evaluation of Parallel Oriented Unification Algorithm" IECE, Japan, AL83-17, pp.21-29 Jun. 1983, (in Japanese)