

THE THEOREM PROVER USING A PARALLEL PROCESSING SYSTEM

Hitoshi Ogawa*, Tadahiro Kitahashi** and Kokichi Tanaka*

* Faculty of Engineering Science
Osaka University
Toyonaka, Osaka 560
JAPAN

** Toyohashi University of Technology
Toyohashi, Aichi 440
JAPAN

1. INTRODUCTION

An AND/OR graph is often used to solve problems. It abstractly expressed the concept of dividing one problem into several sub-problems which are independently solved. This makes it possible that the sub-problems are solved in a parallel process. In this paper, D-graph deduction is proposed, which is based on a reductio ad absurdum using D-graph (a kind of directed AND/OR graph). The D-graphs are corresponded to the predicate logic formulas which represent the axioms and the negation of the theorem proved.

If several sub-problems share the same variables in problem solving, then the way in which one problem is solved may affect the solution of the others. This prevents sub-problems from being solved independently. There is the same issue in the D-graph deduction. The use of micro-actors settles such an issue. The micro-actors are the modules which compose basic structure for the knowledge representation. They can be applied to search a D-graph, because the D-graph basically consists of OUT-AND arcs. It will be shown that the sub-problems are solved independently in a micro-actor system in which the micro-actors correspond to the nodes in the D-graph and the common variables.

2. D-GRAPH DEDUCTION

2.1 First Order Predicate Logic and D-graphs

The authors will propose a formal deduction method using a directed AND/OR graph, called D-graph deduction, which is based on a reductio ad absurdum argument for the first order logic using D-graphs. Nodes in the D-graph are corresponded to the predicate symbols, their negations, TN (which means true), and FN (which means false). A set of arcs indicates a clause (a predicate logical formula).

[Definition 1] A label (α, β) is attached to arc (A, B) corresponding to the formula $A(\alpha) \rightarrow B(\beta)$, as shown in Fig. 1. α and β are the arguments of the predicates A and B, respectively. Thus, a label to an arc is an ordered pair of variable vectors, whose components are the variables and Skolem functions of predicates corresponding to the start and the goal nodes.

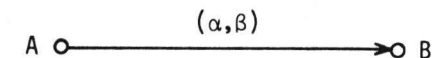


Fig. 1 D-graph representing $A(\alpha) \rightarrow B(\beta)$.

Let Greek lower-case letters express the tuple of constants, variables and Skolem functions.

[Definition 2] (Rules for translating clauses to a D-graph)

(1) Select a clause from a set of support, where the set of support is a non-empty subset K of the initial set S of clauses such that S-K is satisfiable. Let the clause be represented $L_1(\lambda_1) \vee \dots \vee L_h(\lambda_h)$.

The clause may be modified as follows:

$$\begin{aligned} & \text{True} \rightarrow L_1(\lambda_1) \vee \dots \vee L_h(\lambda_h) \\ &= (\text{True} \rightarrow L_1(\lambda_1)) \vee \dots \vee (\text{True} \rightarrow L_h(\lambda_h)). \end{aligned}$$

Nodes $L_1, \dots, L_h$ in the D-graph are regarded as OUT-AND nodes concerning TN, and arcs $(TN, L_1), \dots, (TN, L_h)$ are grouped by a linking symbol. Attached to the arcs are the labels $(, \lambda_1), \dots, (, \lambda_h)$ (See Fig. 2 (a)).

(2) Let $L_1(\lambda_1) \vee \dots \vee L_m(\lambda_m)$ be a clause which has more than one literal, excluding the clause chosen at (1). This clause can be translated as follows:

$$\begin{aligned} & \neg L_i(\lambda_i) \rightarrow L_1(\lambda_1) \vee \dots \vee L_{i-1}(\lambda_{i-1}) \\ & \quad \vee L_{i+1}(\lambda_{i+1}) \vee \dots \vee L_m(\lambda_m) \end{aligned}$$

$$= (\neg L_i(\lambda_i) \rightarrow L_1(\lambda_1)) \vee \dots \\ \vee (\neg L_i(\lambda_i) \rightarrow L_{i-1}(\lambda_{i-1})) \vee (\neg L_i(\lambda_i) \rightarrow L_{i+1}(\lambda_{i+1})) \\ \vee \dots \vee (\neg L_i(\lambda_i) \rightarrow L_m(\lambda_m)).$$

The D-graph representing this formula consists of OUT-AND arcs and nodes concerning $\neg L$ as shown in Fig. 2 (b). The above operation must be performed for all i ($i = 1, \dots, m$).

(3) A unit clause $L(\lambda)$, which is not selected at (1), may be modified to $\neg L(\lambda) \rightarrow \text{False}$. $L(\lambda)$ is therefore represented as the D-graph illustrated in Fig. 2 (c).

(4) $L(\lambda)$ and $\neg L(\lambda)$ are always inconsistent, so the clause set containing them is unsatisfiable. They are indicated by IN-AND arcs and nodes concerning FN, as shown in Fig. 2 (d).

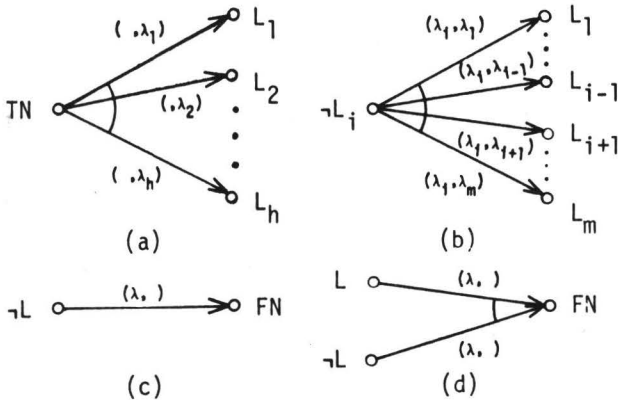


Fig. 2 D-graph

- (a) $\text{True} \rightarrow L_1(\lambda_1) \vee \dots \vee L_h(\lambda_h)$
 (b) $\neg L_i(\lambda_i) \rightarrow L_1(\lambda_1) \vee \dots \vee L_{i-1}(\lambda_{i-1}) \vee L_{i+1}(\lambda_{i+1}) \vee \dots \vee L_m(\lambda_m)$
 (c) $L(\lambda)$
 (d) $(L(\lambda), \neg L(\lambda))$.

2.2 Completeness of D-graph Deduction

D-graph deduction is carried out by communicating the state of a node to its successor and by referring to the labels. The following define the state of a node and its communication rule, and the relation between a node and FN in the D-graph.

[Definition 3] Suppose e , with label (α, β) , denotes the arc from node A to node B, and that γ represents the state of A. If there exists the most general unifier (m.g.u.) π of α and γ , the state γ of A is said to satisfy the label condition and e is said to be available. Then the state of B is $\beta\pi$. When the state γ of A does not satisfy the label condition of e ,

nothing is conveyed to B, i.e., this is equivalent to that saying arc e does not exist.

[Definition 4] (Achievability for a D-graph) Achievability is defined between any node and FN as follows:

(1) If a node A is connected to FN by an arc (A, FN) which is not an IN-AND arc and is available, then $A \rightarrow \text{FN}$.

(2) Suppose that the set of arcs $(A, N_1), \dots, (A, N_m)$ forms an OUT-AND arcs concerning A. If each of the arcs is available and $N_j \rightarrow \text{FN}$ ($j = 1, \dots, m$), then $A \rightarrow \text{FN}$.

(3) If there is a sequence of arcs $(\neg A, N_1), (N_1, N_2), \dots, (N_n, A)$ (or $(A, N_1), (N_1, N_2), \dots, (N_n, \neg A)$) and all arcs are available together with the most general unifier π for the states of A and $\neg A$, then $A \rightarrow \text{FN}$ ($\neg A \rightarrow \text{FN}$). If there are nodes $M_1, \dots, M_m$ which have an OUT-AND relation with $\neg A$ (A), FN must be achievable from them and the state of M_i ($i = 1, \dots, m$) applied must satisfy the label condition.

(4) $A \rightarrow \text{FN}$ is only generated by applying the above rules.

The D-graph deduction is carried out to determine whether FN is achievable from TN in D-graph which represents the axioms and the negation of the theorem. The states of nodes and the labels of arcs are employed to deal with the arguments of the predicates. The completeness of the D-graph deduction is guaranteed by Theorem 1.

[Theorem 1] Let G be the D-graph representing a set of S of clauses. S is unsatisfiable if and only if $\text{TN} \rightarrow \text{FN}$ in G.

3. PARALLEL PROCESSING SYSTEM

In this section, a parallel processing system is described to solve the problem using micro-actors. The micro-actors are the modules for the knowledge base. The micro-actor is composed of a script and an acquaintance. The script consists of the pairs of the message pattern and the action corresponding to the pattern. The micro-actor has the acquaintance as a memory space to hold the data and the names of the other micro-actors it directly knows about. Since only message passing is the interactive method between the micro-actors, the action of a micro-actor never affects the other micro-actors. This enable us to construct a parallel processing system.

3.1 Node Micro-Actors and Variable Micro-Actors

We will define two kinds of micro-actors. One is a node micro-actor (NMA), which corresponds to a node in a D-graph. Scripts of NMAs are the same. That is, they have a program which sends messages to the other NMAs according to their acquaintances. An NMA stores the relation between the node indicated by the NMA and the other nodes, in ELIST in its acquaintance. An example of ELIST is as shown in Fig. 4 (a).

Another is a variable micro-actor (VMA), which corresponds to a common variable and is made under searching a graph. Since a micro-actor is independent of the other micro-actors except for message sending, a VMA is independent of the other VMAs and NMAs. However, several NMAs must be able to refer to one VMA. There are two kinds of actions of a VMA:

- (1) Add the sent data to the acquaintance.
 - (2) Reply the data which have flag sent.
- The basic data of the acquaintance is in form of (F V), where F is a set of flags which are similar to sentence number, and V is a value. (When the value is not given, V is NIL.)

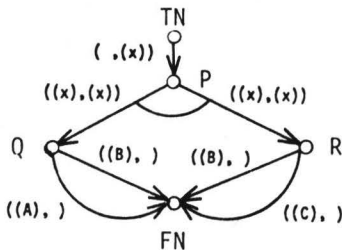
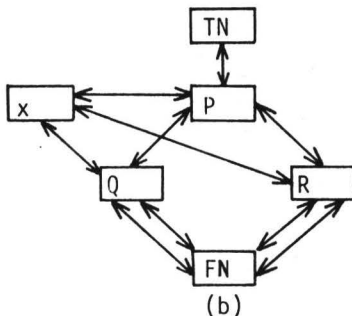


Fig. 3 A part of D-graph in Example 1.

TN-ELIST = ((NIL P (x)).
P-ELIST = (((?x) Q (x) R (x))).
Q-ELIST = (((A) FN NIL) ((B) FN NIL)).
R-ELIST = (((B) FN NIL) ((C) FN NIL)).

(a)



(b)

Fig. 4 ELISTs (a) and micro-actors (b) in Example 1.

[Example 1] The D-graph illustrated in Fig. 3 is a part of D-graph representing the following clauses:

axioms: $\neg P(x) \vee Q(x) \vee R(x)$, $\neg Q(A)$, $\neg Q(B)$,
 $\neg R(B)$, $\neg R(C)$.

negation of theorem: $P(x)$.

Where A, B and C are constants, and x is a variable. Fig. 4 (b) shows a micro-actor system corresponding to the D-graph in Fig. 3. Fig. 4 (a) shows ELISTs in the micro-actors. Graph searching begins when TN sends the messages and is carried out by sending message from NMAs to NMAs. NMAs perform one of three operations: expansion (sending the messages to successors), merging (merging several messages into one) and partial refutation (which is similar to ancestor resolution). FN replies to its ancestor. When a NMA receives a reply, the NMA checks the consistency of it. So the result of the system does not depend on the order of replies the messages which are sent concurrently. The system is stopped when a reply is sent to TN. Sending message from NMAs to NMAs makes the coroutine control, while sending message from NMAs to VMAs makes the subroutine control.

REFERENCES

- [1] Hewitt, C. & Smith, B. "Towards a Programming Apprentice." *IEEE Trans. on SE*. SE-1:1 (1975) 26-45.
- [2] Levi, G. & Sirovich, F. "Generalized AND/OR Graphs." *Artif. Intell.* 7:3 (1976) 243-259.
- [3] Ogawa, H. & Tanaka, K. "A Structure for Representation of knowledge -- A Proposal for A Micro-Actor --." In *Proc. IJCAI-77*. (1977) 248-249.
- [4] Ogawa, H., Nanba, H. and Tanaka, K. "An Active Frame for The Knowledge Representaion." In *Proc. IJCAI-79*. (1979).
- [5] Robinson, J. A. "A Machine-Oriented Logic Based on The Resolution Principle." *JACM* 12 (1965) 23-41.
- [6] Slagle, J. R. & Koniver, D. A. "Finding Resolution Proofs and Using Duplicate Goals in AND/OR Trees." *Information Science*, 3 (1971) 315-342.