A STRUCTURE FOR THE REPRESENTATION OF KNOWLEDGE
A PROPOSAL FOR A MICRO-ACTOR

Hitoshi OGAWA, Kokichi TANAKA

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

I. Introduction

A question-answering system is a composite of subsystems of language analysis, inference and language generation. In the inference system, a useful and flexible structure is needed for the representation of knowledge. There are two ways of representing knowledge in computers: the assertion of facts and theorems. A micro-actor, which possesses some of the attributes of actors (C. Hewitt 1973), is introduced to construct a knowledge representation structure. Knowledge is procedurally embedded in micro-actors so it is easy to represent theorems and, since each micro-actor has a stack for data, it is also possible for various kinds of facts to be stored.

Actor is indeed universal and, although complicated, Is compatible with all programming languages and hardware. However, it is not so easy to realize precisely such a complicated system. For example, a pseudo-implementation of co-routine processing or parallel processing will reduce the efficiency of the system. We, therefore, introduce a micro-actor based on the concept of actor, and use it as the knowledge base. Micro-actor retains the main characteristics of actor. Using a LISP system on a parallel processor, a micro-actor system will be easily implemented.

II. Action of Micro-Actors

In a micro-actor system, there is no distinction between procedures and data. The system is a collection of independent objects (micro-actors) which act as either procedures or data. The action of a micro-actor is defined in terms of only one kind of action: sending messages to the other micro-actors.

When a micro-actor receives a message, it becomes active, and tries to play its role. It sends messages to other micro-actors if necessary, and waits for their replies. When its action is finished, it responds to the requester (the micro-actor sending the message). In Fig. 1, sending a message is illustrated with "$\Longrightarrow$", and receiving an answer with "$\longleftarrow$". An answer is also a message in a wider sense. For example, a micro-actor which acts as datum returns the answer to the question. A micro-actor which acts as a procedure performs its own function for the requested message and sends messages to other micro-actors requesting data or asking them to perform a part of the function. In this way, the job is carried out through the sequence of message sending.

Adoption of only one kind of communication (message sending) between micro-actors has the following advantages:

(1) Only one kind of behavior (message sending) is the basic unit of control in the system. Invoking a function, referring to the value of a variable, and other behavior are all considered to be special cases of this behavior.
(2) Micro-actors can directly communicate with other micro-actors——there is no need to have any special channel for communication.
(3) Sending messages has no side-effects. Because a micro-actor can converse with the several micro-actors simultaneously, a high degree of parallel processing is possible.

III. Structure of Micro-Actor

As shown in Fig. 1, a micro-actor is represented as a box whose top is filled with its name. A micro-actor is composed of three parts:

(1) Control part: This determines the action of the micro-actor by matching the sent message with the patterns of script (see below). The control part also checks whether the result of the action satisfies the request, and if so, replies to the requester. Otherwise, it will decide to take another action. The control part is considered to be the union of the 'intentions' and 'monitors' of Hewitt's actors. It is programmed in LISP, and is common to all micro-actors.

(2) Script: This expresses theorems. A micro-actor stores several pairs composed of the pattern of the receivable message and the corresponding action. Usually, the content of the script is stored when the micro-actor is constructed, and more information can be added at any

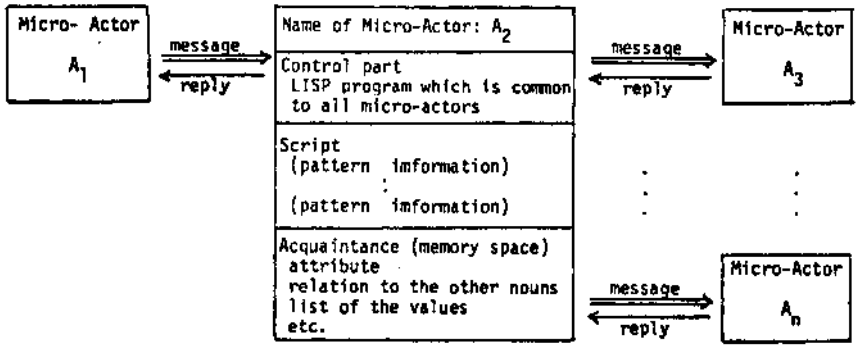

Fig.1
A Word Micro-Actor Model

time. It is written in LISP and the restricted Japanese.

(3) Acquaintance: This part is a memory space. A micro-actor stores facts or data, for example, the particular values, attributes or other information which should be remembered, and which express the assertion. For example, a verb micro-actor stores the record of its actions and some noun micro-actor holds its particular value.

The advantages of using micro-actors to represent knowledge are as follows:

(1) Knowledge is procedurally embedded in micro-actors, so that:
  (a) It can be used when and only when it is needed.
  (b) It can be specified at various levels of detail. That is, it can be specified in as much or as little detail as is desired.
(2) In the case where new knowledge is to be added to the system, even if some special mechanisms are needed for dealing with it, the whole system need not be changed. It is only necessary to embed the mechanisms in a new micro-actor, or to increase the other micro-actors which realize them.
(3) A micro-actor can be freely changed under the restriction that the messages sent or received by the new micro-actor should be consistent with those of the old one, or should subsume them.
(4) The action of each micro-actor is independent of the others except in the case of message sending, and there is no global state of all the micro-actors in the universe, so it is easy to debug the micro-actors. Furthermore, justification of each micro-actor may lead to justification of the whole system.

IV. Message and Sentence Number

A message is of the form of "N L", where L is a simple sentence and N is a sentence number. The sentence number indicates an order of the message, and can also be used as a marker to distinguish the result of a micro-actor's action from the others. For example, the micro-actor AMARI (remainder) stores the result of division to yield the pair consisting of the result and the sentence number. The micro-actor can retrieve all the remainders by means of the sentence number. So it can reply about the remainders of all divisions. Fig. 2 shows an example of a diagram of message flow. The following are initial conditions of the diagram:

(1) There existed five micro-actors; WARU (divide), DEARU (be), AMARI (remainder), SHOH (quotient), and TEIGISURU (define).
(2) When the system received "'X' is 4.", micro-actor *X* was made. Then *X* received the message "%1 Your value is 4." to have the value (%1 4).
(3) "Divide X by 3." was executed. WARU (divide) computed 4 + 3, then sent (%2 1) to both SHOH (quotient) and AMARI (remainder).
(4) WARIKIRERU (divisible) was made by TEIGISURU (define) which received "%2 'Is ?S diviseble by ?T.' is defined as 'Divide SbyT.' 'Is the remainder 0?'."

In Fig. 2, WARIKIRERU (divisible) sends the messages to both WARU (divide) and DEARU (be) referring to its script, when it receives "Is X divisible by 2?". WARU (divide) computes 4 * 2, and AMARI (remainder) and SHOH (quotient) store (%3 0) and (%3 2) respectively. AMARI (remainder) has two values but it returns the correct answer to DEARU (be) according to the sentence number.

ACKNOWLEDGEMENTS

The authors appreciate many helpful conversations with several graduate students and members of Prof. Tanaka's Laboratory of Osaka University, especially Dr. Kitahashi, Mr. Kijima, and Dr. Ross, In addition, we wish to thank Prof. Hewitt (MIT) for sending us his papers and allowing us to use the term "micro-actor".

REFERENCES

1) Hewitt, C, Bishop, P. and Steiger, T. : A Universal Modular Actor Formalism for Artificial Intelligence, Proc. 3rd IJCAI, pp. 235— 245 (1973)
2) Hewitt, C. : How To Use What You Know, Proc. 4th IJCAI, pp. 189 —190 (1975)

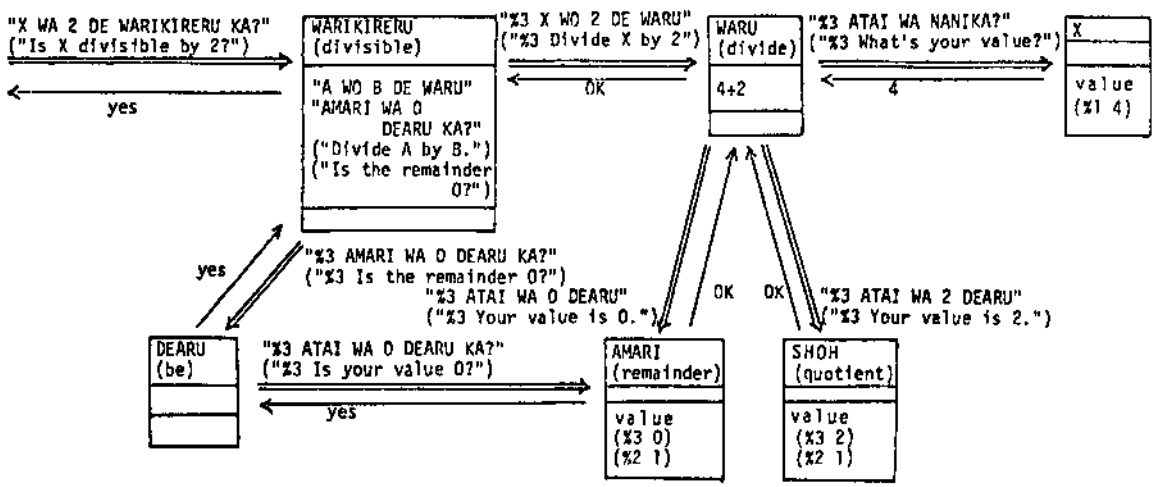

Fig. 2
An example of a Diagram of Message Flow