THE KNOWLEDGE REPRESENTATION FOR A STORY UNDERSTANDING AND SIMULATION SYSTEM

HITOSHI OGAWA*, JUNICHIRO NISHI** AND KOKICHI TANAKA*

* FACULTY OF ENGINEERING SCIENCE, OSAKA UNIVERSITY
TOYONAKA, OSAKA 560, JAPAN
** MATSUSHITA ELECTRIC INDUSTRIAL CO.,LTD.
KADOMA, OSAKA 571, JAPAN

Abstruct There exist many difficult problems to understand a situation and an event described in sentences or a story. One of them is to treat with more than one subject and their relations. Another is the comprehension of movement of the subjects and their effects to the others. In this pater, micro-actor is used as the knowledge representation in which such the problems mentioned above are solved. The micro-actor is an artificial intelligence module for knowledge representation, which is realized Hewitt's actor concept.

A large problem is often solved by a group of specialists. Each specialist has his own knowledge and technique. A specialist can accomplish independently a small work communicating with the others. The specialist is implemented in the form of micro-actor on a computer. The micro-actor is independent of the others, and communicating with the others using one kind of action: sending message to another micro-actor.

We discuss the following four problems to understand stories and the approaches to them: (1) depth of understanding sentences to comprehend a story, (2) a method to deal with an event which happens on the specific condition, (3) synchronization of the events which occur at same time and (4) treatment of the event which involves more than one object.

1. Introduction

The common issues of the various themes in artificial intelligence are the knowledge representation, and the mechanism of understanding and inference. The reason is that the problems in each theme is solved well using the knowledge known by human in a computer. We will discuss a story understanding to investigate how the knowledge is affected by the change of a world state with the passing of time.

For right understanding of a story, it is necessary to check the world knowledge obtained from input sentences using common sense. This paper describes the system which simulates the actions of objects in a story for understanding an event and forecasting the results of the story. For the simulation, the system must deal with the action of individual object and the relation amang them. A micro-actor is adoped as the knowledge representation in which the problems mentioned above are solved. It is indepenent of the others, and communicating with the others using one kind of action: sending message to another one.

Chapter 2 describes about the micro-actor and its features. In chapter 3, four of the problems in story understanding are discussed. Chapter 4 shows the constructon of the simulation system and its example.

2. Micro-actor

A micro-actor was proposed as the implementation of some of abilities of Actor (Hewitt, C. 1973) in 1977 [7]. The micro-actor described in this paper is more powerful as it is supplemented some strong faculties. This chapter discuss the concept of the micro-actor, then explains the structure and the feature of it.

2.1 Concept of micro-actor

In a large system, it is difficult to add a new faculty to it, delete one from it, and modify it. A solution of such a crucial issue is that a large system is divided into many specialists. Each specialist has his own knowledge and technique to accomplish a small work independently. Only message passing is the interaction method between the specialists. A large problem is solved as follows: It is divided into the subproblems. If there does not exist a specialist which can solve a subproblem, the subproblem is further divided into the subproblems. When a specialist solves a problem, it communicates with the others to get his necessary information, to send the result of inference, and to check the consistency of solution. Finally, the large problem can be solved.

A specialist is implemented in the form of micro-actor on computers. A proper definition of each micro-actor leads to a good execution of a system composed of the micro-actor. The concept of actor can be applied in various regions by means of assigning a suitable role to the micro-actor. For example, if a computer is regarded as a micro-actor, a powerful method will be supplied for the implementation of the parallel process or the distributed process using more than one computer. This paper describes that the micro-actors are implemented on a computer for problem solving and knowledge representation.

2.2 Construction of Micro-Actor

A micro-actor is composed of a "script" and an "acquaintance" (Fig. 1). The script describes the behavior which the micro-actor should take when it receives a message. That is, it shows the procedural knowledge to use the data in the acquaintance. The acquaintance stores the data that the micro-actor knows; for example, its native values, its attributes, the name of the other micro-actors it directly knows about, and the relation with them. That is, it shows the declarative knowledge.

the name of a micro-actor
script the procedural knowledge
acquaintance the declarative knowledge

Fig. 1 Construction of micro-actor.

(i) Script

The script is specified with a set of pairs of message patterns and rules as follows:

```
((message patter,n1 rule,11 ... rule,1m)
 (message patter,n2 rule,21 ... rule,2n)
 (message patter,np rule,p1 ... rule,pq)).
```

(ii) Message Pattern

There are two types for pattern:
(1) (MES: ?%N content TO: ?CONT).
(2) (RE: ?%N content).
?%N is only matched with the massage number %n, where n is a positive integer. A message number is assigned to an input sentence or a message for distinguishing it from the others as a tag. The message number is also used to distinguish a datum from the others stored in the same micro-actor. The atoms prefixed by "?" (e.g. ?CONT) are the variables for pattern matching The prefix "!" is used instead of the prefix "?" to refer to the pattern matching variables for their values; for example, !CONT. "MES" means a message and "RE" means a reply. "TO:" means that the continuation is sent a reply. Assign "NO-ONE" to !CONT in the case that the answer is not required. Assign "ME" to !CONT when the reply is demanded. This is equal to the subroutine control.

(iii) Rule

A rule is a set of programs, which carries out a job. The rule consists of three parts as follows:

```
((P-C: program,p1 ... program,pq)
 (N-C: program,n1 ... program,nr)
 (C-E: program,c1 ... program,cs)).
```

"P-C" is an abbreviation for pre-condition. "N-C:" means next-condition which is an option. "C-E:" is an abbreviation for caused-events in which the messages are sent to the other micro-actors when both the pre-condition and the next-condition are satisfied. The form of the rule is given by Dr. Yonezawa[12], and it is suggested that the form of the rule is useful to deal carried out in order of the appearance of them in the pre-condition and the

(Frame_Name
$(S_1\ (F_{11}\ (D_{111}\ C_{111})\ \cdots\ (D_{11h}\ C_{11h})))$
$(S_2\ (F_{21}\ (D_{211}\ C_{211})\ \cdots\ (D_{21i}\ C_{21i})))$
$(S_3\ (F_{31}\ (D_{311}\ C_{311})\ \cdots\ (D_{31j}\ C_{31j}))$
$\quad (F_{32}\ (D_{321}\ C_{321})\ \cdots\ (D_{32k}\ C_{32k})))$
.....
.....
$(S_s\ (F_{s1}\ (D_{s11}\ C_{s11})\ \cdots\ (D_{s1m}\ C_{s1m}))))$

Fig. 2 Frame representation for the acquaintance.

next-condition. On the other hand, the programs in the caused-events can be executed concurrently.

(iv) Program

A program is written in the restricted LISP notation. The user can freely use the LISP functions which do not have influence upon the pointers to lists and atoms; e.g., MEMBERS, ASSOC and so on. The variables are restricted to the form !m or !!n (Both m and n are positive integers) for the LISP functions which have influences upon the pointers; e.g., SETQ and so on. This restriction prevents the behavior of a micro-actor from giving unitentional effect to other micro-actors. The variable !m is used as well as PROG variable. The user can operate the basic function (FGET, FPUT and FREMOVE) to deal with the acquitance. The function FGET is used to obtain the data from the acquaintance. The functions FPUT and FREMOVE may put the data to it and remove the data from it, restrictively. The user uses the basic function (FGET, FPUT and FREMOVE) to deal with the acquaintance. These functions differ from those of FRL[9,10] in point of the necessity of the frame name. The functions in this paper deals with only its acquaintance. Therefore, the execution of the basic function does not affect the other micro-actors. A specification for message passing is as follows: (=> the name of a micro-actor a message).

(v) Acquaintance

The frame is employed to represent the acquaintance as shown in Fig. 2. The respective substructures of a frame are named; Slot, Facet, Datum and Comment. A slot shows which property the data in it are connected with. A facet shows kinds of the data; e.g., values, constraints, procedures and so on. A datum is elementary information associated with the facet, and a comment is additional information for the datum. More than one facet can be included in a slot.

2.3 Behavior of Micro-Actor

The micro-actor becomes active and tries to play its role when it receives a message M1. If the message pattern P1 is found, a rule R1 is searched for, which is one of rules making a pair with the pattern P1, and whose pre-condition should be satisfied. When there is the next-condition in R1, all programs in the next-condition should be evaluated. If the next-condition is satisfied, the casued-events are executed. If there is no rule whose pre-condition is satisfied concerning the pattern P1, the micro-actor tries to find another message pattern which is matched with M1.

Since the system holds the COMMON-SCRIPT which is necessary for all micro-actor, the users may specify the micro-actor by the peculiar script. The COMMON-SCRIPT has the message pattern (MES: ?%N ?X TO: ?CONT) and acts as follows:
(a) If the variable !X (corresponding to ?X) indicates one of the basic functions (FGET, FPUT and FREMOVE), then it is put into practice.
(b) If !X shows one of the extension functions (i.e., there exists a micro-actor corresponding to it), the rules are obtained from the micro-actor corresponding to the extension function, and they are performed. The variables !!n (n = 1, ..., 5) are corresponed to the arguments; for example, !!1 means the first argument.
(c) If !X indicates a set of rules, then !X is accomplished.

2.4 Demon for Micro-actor

A demon function is newly added to the micro-actor to build a story understanding system. Micro-actors correspond with men, animals and objects which appear in a story. Each action is described in the script. The acquaintances show peculiar values and the relation to the others. It is, however, difficult to get a result of the simulation of the world situation obtained from a story. For example, assume that there is a micro-actor Taro (names of micro-actors are underlined) which corresponds to a man named Taro. Consider the method that Taro sends a message to a certain micro-actor when he arrives at place X. In this case, Taro must see if his place is place X whenever Taro changes his place. To do this well, the demon function is necessary, which always checks if there exists the specific situation or not, and behaves an appropriate action: In this example, once the demon function comes to know that Taro is at place X, it sends a message to a certain micro-actor.

We can use demon functions using is the frame representaion like FRL, and two kind methods of the demon functions mentioned below. Two of new demon functions are made in consideration of the activation method of micro-actors: The micro-actors are activated when then receive messages. Only a post-action method is adopted in the present system.

Pre-action A pre-action demon is described in the form of rules in a pre-action slot in the acquaintance. When a micro-actor receives a message, rules in the pre-action slot are evaluated before the micro-actor begins to execute its role according to the script.

Post-action A post-action demon is described in the form of rules in a post-action slot in the acquaintance. After the micro-actor finished to execute its role according to the script, the rules in the post-action slot are evaluated.

2.5 Features of Micro-actor

The knowledge representation using the micro-actors can establishes both its modularity and easibility of interaction. This section states the features of a micro-actor using examples.

The declarative knowledge and the procedural one The knowledge is represented in either the declarative form or the procedural one. The following example shows how to represent the knowledge in the micro-actor. Assume micro-actor Human which shows the knowledge about a human being. Though we have a great deal of knowledge about a human being, two simple examples will be shown. One is represented as the declarative knowledge: "A human is a kind of animal." The representation of this knowledge is to put value ANIMAL to Ako slot in the acquaintance of micro-actor Human. The other is represented as the procedural knowledge: "When there exists an obstacle (e.g., tree, wall, etc.) in his way, he jumps over it if it is low, otherwise, he go a long way around." This knowledge is shown in the form of rule, and the rules are put to either the script, or Pre-condition slot or Post-action slot in the acquaintance of micro-actor Human.

Modularity It is essential to be able to represent a great deal of knowledge by a group of small modules. As mentioned in 2.1, the micro-actor can independently accomplish a small work, and only message passing is the interaction method between one and another. The messages received by a micro-actor describes what the micro-actor must do. The messages sent from a micro-actor shows what the micro-actor needs. We can understand the action of a micro-actor from only the messages sent from or received by the micro-actor. Therefore, we may only check the messages sent or received to change the micro-actor. Furthermore, we can easily understand the micro-actor system inspecting the messages.

The moduality brings the construction in which a classification and a hierarchical structure of knowledge is easily represented. The following shows the example about the representation of knowledge in a hierarchical structure.

The obstacles to the human (e.g., John and Tom) are generally trees, fire, wall, and so on. Dogs are, however, obstacles to Tom if he dislikes them. In the micro-actor system, these obstacles are specified in a hierarchical structure as shown in Fig. 3. In Fig. 3, the names and the acquaintances are shown in boxes.

Pseudo parallel processing can be easily implemented. This secton will show the example of that John and Tom play at a tug of war (Two person draw a rope in the opposite direction.). John and Tom are assigned to micro-actor John and micro-actor Tom, respectively. To simulate the tug of war, it is necessary that John and Tom are activated simultaneously. This is, however, impossible of implementation using our facilities. Therefore, the example is simulated with pseudo parallel processing.

We also use micro-actor SIMTW which has the knowledge about a tug of war. Fig. 4 shows the relation of the micro-actors.

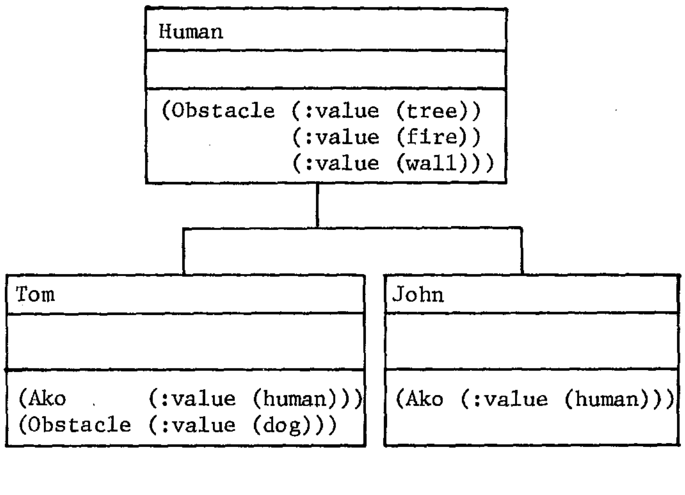

Fig. 3 An example of a hierarchical structure.

SIMTW cannot be active until it receive the messages from both Tom and John. If it receives two messages, then it check the situation of the tug of war: e.g., if Tom and John draw a rope in opposite directions with the same might, then the tug is continued. If one of them is more powerful than the other, then he becomes a winner. When one of them falls down, there is no power to draw a rope. Then he is loser. ...

Parallel processing is possible if Tom and John are independently implemented using two micro-computers. In our system, we use one computer to implement the micro-actor system. Therefore, necessary is the micro-actor for synchronization of the action of Tom and of John. The detail will be described in chapter 4.

3. Problems in Story Understanding

There exist many difficult problem to understand a story. In this chapter, we discuss four main problem in them, and approaches to them.

(1) How deeply should a sentence be understand to comprehend a story?

A consistent meaning should be obtained from a sentence neither in too detail nor in too rough in order to comprehend a story. The level of detail at which a sentence is understood depends upon the context and one's interest. The example will be given to us using the following sentence: "John was walking." One may think that the sentence means that John advances. This interpretation leads the reason enough to understand "John arrived at a town." However, it is not the reason enough to understand "John was tired." One must infer as follows: Action "walking" consists of the relaxation of muscles and the tension of them. One of reasons for his fatigue may be the repetition of the tension and the relaxation of muscles. We, therefore, infer

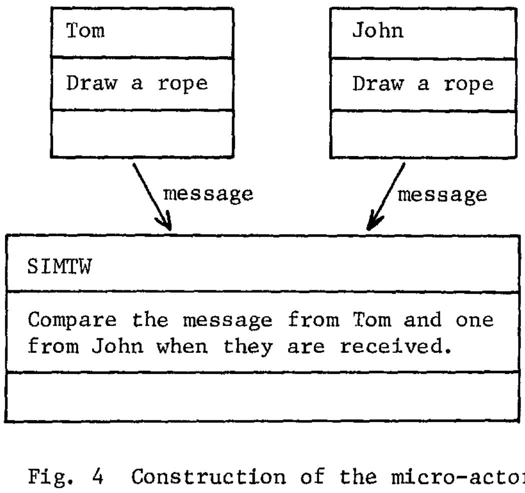
Tom
Draw a rope
John
Draw a rope
message
message
SIMTW
Compare the message from Tom and one from John when they are received.

Fig. 4 Construction of the micro-actors for a tug of war.

that his fatigue is caused by the long distance walking until we find another reason for it.

To implement te above inference, we adopted a frame representation to be easy to represent knowledge in a hierarchical structure, and developed the convenient method to deal with such knowledge.

(2) How to deal with an event which happens on the specific condition.

To answer the questions about a story, it is necessary to refer to the knowledge base and to forecast the event which will happen in the future. One of methods for a forecast is the simulation of the story. In the simulation, there exests an event which happens only on the special condition. The simulation is performed to investigate at where Tom will arrive using the information obtained from the story, when the system receives the question "Can Tom arrive at a town?" If there is obstacles on his way, we cannot predict Tom's behavior. Therefore, whenever he changes his position, the system should see if he is in the town. The demon function is useful to check this rather than programming his behavior in the script. In this paper, the rules are in the post-action slot, which send a messge to micro-actor Answer if Tom is in the town. These rules are activated after being applied some rules in the script of Tom.

(3) How to describe the knowledge changed frequently.

When there exist some subjects which move at the same time, the simulation system must change simultaneously their position. However, it is impossible to implement the parallel processing by using a CPU machine. For example, assume the scene that Tom and a dog are walking oppositely. The simulation system changes their positions in order: first Tom moves then the dog moves, or in the opposite order. The different results are brought in the different ordering when there is the restriction that they must not come across. To settle the above issue, we use the time-flag in order to represent the situations of them. The method is a little unconvenient for describing the situation per time. It, however, needs neither the specific faculty to remember its careers, nor the memory space to hold new situations for the micro-actors because they are changed at the same time.

Fig. 5 shows the list of the acquaintance of micro-actor Tom. Sn is the abbreviation of Stepn(Suffix n is a positive integer.). S1 indicates the time when the system is the initial state. Suffix n is increased one by one as a certain time passes. Place slot indicates the present place of Tom, and Fig. 5 says that he was in P16 (Place 16) at S1 and in P17 at S2. In ACT slot, the action of Tom is described. According to Fig. 5, Tom is going to P20.

```
(AKO    (:VALUE (HUMAN)))
(MIGHTY (:VALUE (5)))
(LIVING (:VALUE (ANIMAL)))
(HEIGHT (:VALUE (LITTLE)))
(PLACE  (S1 (:VALUE (P16)))
        (S2 (:VALUE (P17))))
(ACT    (S1 (:V (MOVE)) (:G (P20)))
        (S2 (:V (MOVE)) (:G (P20))))
```

Fig. 5 The acquaintace of micro-actor Tom.

(4) How to deal with the event which involves more than one subject.

There may be more than one subject in an event stated in a story. When they are related with one another, we must skillfully deal with the relations of them. There are two kinds of relations between the subjects in a story. One is cooperation (e.g., Tom and John push a box together), and the other is opposition (e.g., Tom and John play at a tug of war).

There are many ambiguous sentences dealing with cooperation. In the case of sentence "Tom and John made a shed for dog in cooperation," there are some interpretation as follows:
(1) They divided the labor: e.g., One of them bought materials and the other constructed the shed.
(2) They worked together for all jobs.
(3) One of them worked according to the other's direction.
The above example says that we can consider many complicated relation for cooperation in a simple sentence. In this paper, we use the second interpretation for simplicity.

Both cooperation and opposition is dealt in the similar way in this simulation system. Examples will be given using the following sentences:
(1) Tom and John play at a tug of war.
(2) Tom and John push a box together.

In order to deal with sentence (1), we make micro-actor SIMTW which stores the knowledge about a tug of war: e.g. "A winner is a person who is more powerful than the other." "A winner's opponent is a loser." and so on. When SIMTW receives both a message from Tom and one from John, it compares the power of Tom with one of John, and judges a winner. A Coop (which means cooperation) slots are added to the acquaintances of Tom and John to store the name of his opponent.

The treatment of sentence (2) is similar to one of sentence (1). We make micro-actor SIMPUSH which compares the sum of power of Tom and John with the weight of the box, when it receives both a message from Tom and one from John. SIMPUSH sends micro-actor BOX a message to get its weight. If the sum of power of two person is larger than the weight of the box, SIMPUSH sends messages to Tom, John and Box to change their position.

4. Simulation System

In the last chapter, we discussed the main problems in story understanding and the methods using micro-actors to settle them. In order to confirm the validity of these methods, we simulate how the world described in a story will change with the passing of time. The simulation is shown as an animation. In this chapter, we state the simulation system consisting of micro-actors and its executive examples.

4.1 Planning

It is necessary for simulating to specific attribute, location, action etc. of the objects in a story. In this system, micro-actors are made corresponding to objects in a story, and the necessary is stored in them.

There are the micro-actors which have general knowledge about the objects (e.g. person, dog, tree and so on) in this system. These micro-actors give the micro-actors corresponding to the objects the default values of general action and property. The simulation is based on the information given to the micro-actors corresponding to the objects. For example, micro-actor John is made according to sentence "John is a human". John is obtain the properties of a human: e.g. he is little size for a kind of mammal, he can move freely, he plays games, and so on. Of cource, they are modified easily when new information is given.

If a sentence describes what is concerning with the micro-actor made already, a information in the sentence is given to the micro-actor. The information is represent either in the script or in the acquaintance.

Information about action is described in the script. For example, micro-actor Walk sends micro-actor John the rules for walk, according to "John walks to P20". (P20 indicate a place whose discriminative number is 20.) The rules represent the necessary actions to move toward the goal and, if necessary, call the other rules: e.g. to jump an obstacle, to pass aside it, and so on.

Information indirectly relating to action is represented in the form of demon. In the case that "John dislikes a dog", the post-action slot stores the rules that John goes away from a dog when the dog comes near him. When John is walking, his action does not change until he meets with a dog. He, however, avoids it and walks to his goal if he comes across a dog.

There is the issue whether John comes across the dog or not. This system divides a world into 15(=5x3) places. This reason is just the convenience for the indication of animation. We use these place to check if two objects meet together or not. Two objects "meet together" if they are in the same place. They "come near" if their places are neighboring. Otherwise, they "are kept apart".

4.2 Simulation

The last section describes a method of planning the action of each micro-actor. The activation of micro-actors must be planed well, because micro-actors should act in a pseudo parallel for the simulation. This system adopts a micro-actor Sinchronizer for a pseudo parallel processing of micro-actors. Sinchronizer synchronously sends a message to a micro-actor corresponding to an animate object. Namely, it sends each of the micro-actors only one message to activate it for every frame of the animation.

The actions of the planned micro-actors simulate the change of the world given from a story. A mean processing part is necessary for us to know the results of the simulation and the mean indicated by world state obtained in the simulation. For example, in the case that two person play at a tug of war, the change of the world is indicated as the content that strong person A can draw near a rope (with weak one B). Such world state, however, means that A has won and B has lost the game. The mean processing part is also implemented using micro-actors. The interaction form the simulation part to the mean processing part is a message passing from a micro-actor in the simulation part to a micro-actor in the mean processing one. In the example of a tug of war, a demon is adopted to send a message when one player can draw a rope (with the other player), and a micro-actor in the mean processing part is prepared to received the message and to judge the winner of the game. This organization, furthemore, makes it possible to explain some actions (For example, that John dislikes a dog is the reason why he released the rope and moved to another place.). Fig. 6 shows the construction of the two parts.

The two parts are implemented on PANAFACOM U-1500. We constructed the animation indication system to monitor the simulation. This sysem consists of two parts: Picture Display System (PDS) and Picture Instruction Generator (PIG). The PIG generates the instructions to display the objects in the simulation in the animation style. As the PIG receives messages including only essential information (name and action: e.g. ((A John) (V walk)), the PIG acquirs the detail information (e.g. type, size, etc.) to display the animation from knowledge base. Then the PIG makes the instructions and sends them to the PDS. The PDS displays the animation according to the instructions. The PIG and the PDS are implemented on PANAFACOM U-1500 and NEC

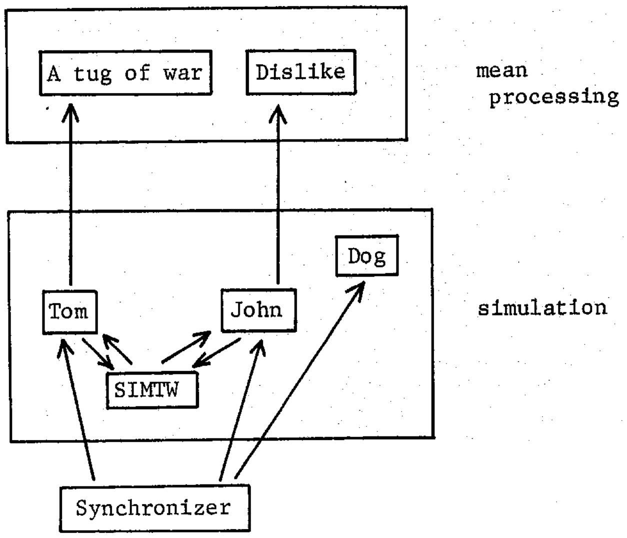

Fig. 6 The relation of mean processing part and simulation part.

P11	P12	P13	P14	P15
P16	P17	P18	P19	P20
P21	P22	P23	P24	P25

Fig. 8 Names of places in the display unit.

PC-8001, rspectively. The details of the animation display system will be presented at another chance.

4.3 Example

An input story is shown in Fig. 7. Places P16, P18 and P20 in Fig. 7 indicate the places an the display unit as shown in Fig. 8. After the sentences indicated in Fig. 7 are understood, the simulation begins when the following question is input:

"Can Tom become a winner of the game?" (9)

Since we do not have the analyzer of natural language (Japanese), sentences are given to the system in terms of Case grammer as shown in Fig. 9. The numbers in Fig. 9 correspond to the sentences in Fig. 7 and the question. Letter V, A, O, L, I and G indicate verb, subject, object, location, instrument and goal, respectively. List (M (? G)) means that the sentence is a question one.

John and Tom are human. (1)
Ropel is a rope. (2)
Spot is a dog. (3)
John, Tom and Ropel are at place P18. (4)
Spot is at place P20. (5)
John dislikes a dog. (6)
John and Tom play at a tug of war. (7)
Spot walks toward place P16. (8)

Fig. 7 Input sentences in a story.

```
1. ((V IS-A) (A (TOM JOHN)) (O HUMAN))
2. ((V IS-A) (A (ROPE1)) (O ROPE))
3. ((V IS-A) (A (SPOT)) (O DOG))
4. ((V IS-AT) (A (TOM JOHN ROPE1)) (L P18))
5. ((V IS-AT) (A (SPOT)) (L P20))
6. ((V DISLIKE) (A (JOHN)) (O DOG))
7. ((V PLAY-AT-A-TUG-OF-WAR)
    (A (TOM JOHN)) (I ROPE1))
8. ((V MALK) (A (SPOT)) (G P16))
9. ((M (? G)) (A (TOM))
    (V PLAY-AT-A-TUG-OF-WAR) (G WINNER))
```

Fig. 9 The input sentences and the question in the form of Case grammer.

Photo. 1 and Photo. 2 show two frames of the animation made from the sentences in Fig. 9. Photo. 1 indicates the state that John and Tom begin to play at a tug of war, and Spot is walking toward place P16. In Photo. 2, John ran away since Spot came near him. Therefore, Tom has won the game and is delighted.

5. Conclusion

There are two types for the representation of knowledge: the declarative one and the procedural one. The difference between them is which we attach importance to modularity or convenience of interaction. The micro-actor has been proposed as the method in which the modularity is established without sacrificing the possiblities for interaction.

This paper has treated of four problems to understand a story: (1) depth of understanding a sentence to comprehend a story, (2) a method to deal with an event which happens on the specific condition, (3) synchronization of the events which occur at the same time and (4) treatment of the event which involves more than one object. We have tried to solve these problems using micro-actors. The most difficult problem is the last one in the above. When the system deals with the last problem, it must have ability of the simulation in which the objects change their states in parallel. The convenient way for this is that a object affects only relevant objects. Since message passing is only the interaction method between micro-actors,

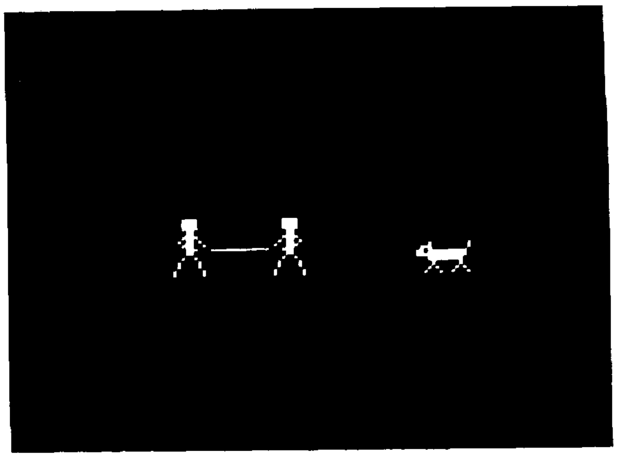

Photo. 1 The state that Spot comes near Tom and John who play at a tug of war.

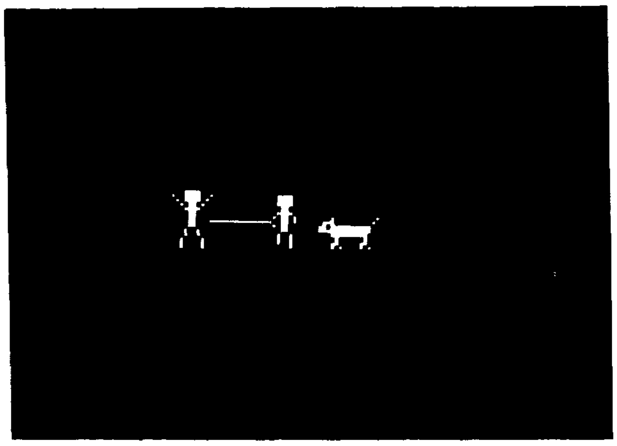

Photo. 2 The state that John ran away and Tom has won.

this method is very available for the parallel processing or the psudo parallel processing. Furthermore, not only the declarative knowledge but procedural one can be embedded in the micro-actors and the problems mentioned above have been solved with relative ease.

This paper has described the knowledge representation for a story understanding and a simulation system for it. In another paper, we have presented a tracing technique, using micro-actors, of blood vessels in stereorado-graphic images of a cerebrum-vascular system. Futhermore, some other systems have been constructed usig micro-actors with success [8].

The autors appreciate many helpful conversations with members of Prof. Tanak's Laboratory of Osaka University. Especially, they thank H. Karasawa and T. Takahashi. The micro-actor has been implemented in LISP 1.7 which was developed by S. Doshita et al. in Kyoto University.

References

[1] Hewitt, C., Bishop, P. and Steiger, T.: "A UNIVERSAL MOSULAR ACTOR FORMALISM FOR ARTIFICIAL INTELLIGENCE", Proc. of IJCAI-75, pp.235--245 (1973).
[2] Hewitt, C., and Baker, H.: "TOWARDS A PROGRAMMING APPRENTECE", IEEE Trans. on Software Engineering (1975).
[3] Kahn, K. M.: "AN ACTOR-BASED COMPUTER ANIMATION LANGUAGE", AI WORKING PAPER 120 (1976).
[4] Kahn, K. M.: "DIRECTOR GUIDE", AI MEMO 482 (1978).
[5] Lehnert, W.: "HUMAN AND COMPUTATIONAL QUESTION ANSWERING", Cognitive Science, Vol. 1, No. 1, pp.47--73 (1977).
[6] Minsky, M.: "FRAME SYSTEM"," in Winstom (ed.) Visual Information Processing (1975).
[7] Ogawa, H. and Tanaka, K.: "A STRUCTURE FOR THE REPRESENTATION OF KNOWLEDGE --A PROPOSAL FOR A MICRO-ACTOR--", Proc. of IJCAI-77, pp.248--249 (1977).
[8] Ogawa, H., Nanba, H. and Tanaka, K.: "ACTIVE FRAMES FOR THE KNOWLEDGE REPRESENTAION", Proc. of IJCAI-79, PP.668--675 (1979).
[9] Roberts, R. and Goldstein, I.: "THE FRL MANUAL", AI-MEMO 409, MIT (1977).
[10] Roberts, R. and Goldstein, I.: "THE FRL PRIMER", AI-MEMO 408, MIT (1977).
[11] Wilensky, R.: "WHY JOHN MARRIED MARY: UNDERSTANDING STORIES INVOLVING RECURRING GOALS", Cognitive Science, Vol. 2, pp.235--266 (1978).
[12] Yonezawa, A.: "A SPECIFICATION TECHNIQUE FOR ABSTRUCT DATA TYPES WITH PARALLELISM", Research Reports on Information Sciences in Tokyo Institute of Technology, No. C-17 (1978).