論　文

μ-actor の実現と知識表現の構造*

小　川　　均**　木　島　裕　二**　田　中　幸　吉**

Abstract

The knowledge representation in a natural language question answering system may be better to have the following facilities: The creation, modification and deletion of the knowledge are easy. The knowledge is also easily used. And it could be specified at various levels of detail.

The idea of an actor module (C. Hewitt 1973) seems suitable for realizing the knowledge representation. We introduce micro-actor, a subset of actor, to construct the date structure in a Natural Language System. The knowledge will be procedually embedded in the micro -actor. In a Q. A. System based on the micro-actor, we can freely use not only predefined words, but also new words that have been specified in term of predefined words. If there are words enough to represent a world, we can use the restrictive Japanese as a formal language.

1. まえがき

T. Winograd が言語理解システム SHRDLU を作成して以来[6]，数多くの言語理解システムおよび質問応答システムが作成されてきたが，システム中の知識の表現法，および，知識をいかに計算機中に記憶するかが問題となりつつある[7]．本論では，計算機中での知識表現の構造について考察する．

問題解決，自然語処理等を行う人工知能システムにおいては，対象物や，それらの関係，それらの処理に関する知識をどのように表現し，蓄積し，利用するかということが重要な問題となる．知識の計算機における表現方法には2種類あると考えられる．1つは，「地球は丸い」「4は偶数である」というような，事実や真であると考えられる事柄の断言である．他は，「Xが2で割り切れるなら，Xは偶数である」「スイッチを押せば電灯がつく」のような事柄の関係や動作の定義である．知識はこの2種類のうち1つまたは両方の方法を用いて表現される．

質問応答システムでは，会話中に対象となる世界の新しい知識を追加し，既存の知識を変更し，削除する必要がある．さらに，その知識は簡単に利用されなくてはならない．質問応答に自然語を用いた場合，入力される知識は自然語文（単語の組合わせによる文）によって表現される．このようなシステムでは，あらかじめ使用する単語をすべて用意しておくことはできない．新しい単語は会話中に定義され，その後自由に使用される．したがって，会話中に新しい単語を簡単に定義でき，定義された単語があらかじめ与えられている単語と同様に扱われる機能が必要となる．また，処理を行った結果を後に利用する場合もあり，処理の結果も簡単に利用できなければならない．例えば，「Xを3で割り，その余りが1ならば…」のような文では，「Xを3で割った」時，答としての余りの値を簡単に利用できる状態で記憶しておかなければならない．

以上のことから，自然語処理における知識構造は次のような機能を要求される．

1．事実の断言と，事柄の関係や動作の定義が両方表現できる．
2．追加，変更，削除が簡単にできる．
3．あらゆる実行レベルから効率よく利用できる．

以上の機能を満足する知識構造を構成するため，人

---

* Implementation of μ-actor for Knowledge Representation Structure by Hitoshi OGAWA, Yuji KIJIMA, and Kokichi TANAKA (Department of Information & Computer Sciences, Faculty of Engineering Science, Osaka Univercity).

** 大阪大学基礎工学部

工知能用モジュール actor[1)-3)] の概念を導入し，自然語処理を対象とした知識表現について考察する．actor は従来の人工知能言語（例えば，CONNIVER, QA-4, PLANNER 等）の特徴の大部分を含む人工知能用モジュールである．actor で構成されたシステムでは，そのシステムの動きが明確に定義され，しかも，その構成が簡単に行うことができる．actor については文献 1)～3) および 4) を参照されたい．

本論では，actor の機能のうち主要な部分を実現する micro-actor (μ-actor) を定義し，各 μ-actor を各単語に対応した自然語処理システムについて述べる．システム中では，μ-actor は手続的に埋め込まれた知識表現として用いられている．

2. Micro-Actor

Hewitt の actor は universal なものを目標としているため，相当複雑な構造になっている．これは，あらゆるプログラム言語に対して適応するだけでなく，ハードウェア化またはファームウェア化を目標としているためである．しかし，本研究では，リスト処理言語 LISP を用いてプログラミングするため，actor の正確な実現は困難である．たとえば，コルーチン，並列処理の擬似的な動作は可能であるが，そのために効率的な実行ができなくなる．したがって，actor の概念を実現した μ-actor を考え，自然語処理用データベースとして使用する．μ-actor は，actor の機能を縮少したものであるが，その主な特徴を実現している．また，μ-actor を遂行する LISP システムまたは計算機が並列処理可能であれば，μ-actor を用いたシステムは並列処理可能である．本章では，μ-actor について，その構造，動作について述べる．

2.1 μ-actor の構造

μ-actor は **Fig. 1** に示すように，ボックスで表現し，最上部には μ-actor の名前を書く．μ-actor の内部は次の 3 つの部分に分かれる．

1．制御部：送られたメッセージと行動情報をパタン照合することにより μ-actor の行動を決定する．そして，その行動の結果が意図されたことを満足するかどうかチェックする．もし，満足すれば返答し，そうでなければ別の行動を決定しようとする．この部分は actor の intention と monitor の両方の機能を持つ．LISP でプログラムされ，各 μ-actor に共通したものである．

2．行動情報部：行動に関する情報を記述してある

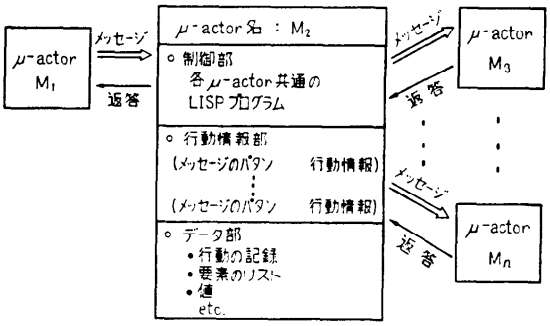

Fig. 1 A micro-actor model corresponding to a ward.

部分である．μ-actor は受け入れることのできるメッセージのパタンと，これに対応する行動を組にして行動情報部に記憶している．これは事柄や動作を定義する知識に対応している．行動情報部の内容は，μ-actor が構成される時に記述されるが，その後でも，情報を追加することができる．μ-actor は LISP を用いて記述するが，新しく定義される単語の場合，および，会話中に情報が追加される場合には，日本語表現も許される．ただし，本研究で用いる日本語表現は，後述するように非常に制限された形である．

3．データ部：データの記憶領域であり，事実を断言する知識に対応する．データ部はその μ-actor の固有の値，属性，および，行動中に得た情報で必要なものを記憶しておく部分である．例えば，動詞を表わす μ-actor の場合には，行動の記録が残される．名詞に対応する μ-actor の場合には，単語の属性に関する包含関係や，値が蓄えられる．

以上が，μ-actor の構造である．これからわかるように，μ-actor は actor の機能のうち，monitor と intention を備えている．その他の機能はすべて LISP インタプリタに依存している．

μ-actor はメッセージが送られると直ちに実行を始めるので，scheduler は非常に簡単になる．そして，μ-actor 中に現われる名前が ALIST 中に存在するか，または，アトムとして APVAL の特性を持っておれば，LISP インタプリタによりその値が与えられる．これは binder の役割を果たす．μ-actor が動作する際には，そのプロセージャがインタプリタにより計算機中のコア内に割り当てられる．これは banker の機能である．

以上のような構造の μ-actor モデルを使用したシステムには，次の利点がある．

1．知識を手続的に埋め込むことができるため，次

のことが可能である.

(a) 知識は必要な時のみ使用される.

(b) 任意の詳細化レベルで μ-actor を明細化できる. すなわち, 希望通り詳しくも簡単にでも μ-actor を明細化でき, 知識を必要なだけ μ-actor に詰めることができる.

2. 新しく知識を追加する場合, その知識を扱うための特殊な機能が必要であっても, システム全体を変更することなく, その知識に相当する μ-actor のみに, その機能を組み込んだり, さらに, いくつかの μ actorを付け加えることにより実現することができる.

3. システムに変更を加える時には, 書き換えられた μ-actor の受け渡しするメッセージが今までのものと矛盾しない. または, 今までのものを包含することだけに注意すれば自由に変更することができる.

4. 個々の μ-actor の動作は, メッセージ転送以外は他の μ-actor と独立しているためにディバッグが容易である. すなわち, 送られるメッセージに対応して正しい動作をするようにプログラムすればよい. また, 各々の μ-actor の動作が正しいことを確認すれば, システム全体の動作も保証されると考えられる.

2.2 μ-actor の動作

μ-actor においては, プロセージャとデータの構造には区別がなく, それぞれ半ば独立している物体 (これを μ-actor と呼ぶ) として存在する. システム全体はこの μ-actor の集合である. 各 μ-actor 間では, メッセージを転送するという一種類の行動のみが許され, 他の交渉法 (他の μ-actor の実行により起こる副作用等) は一切ない. μ-actor はメッセージが送られることにより動作を起こし, 行動情報を参照しながらそれ自身の役割を果たそうとし, 必要に応じてメッセージを他の μ-actor へ送る. そして, 返答を待ち, 返答が得られれば再び行動を続ける. 行動が終了するとメッセージを送ってきた μ-actor へ返答を返す. Fig. 1 でメッセージの転送は「⇒」, 返答の転送は「→」で示す. 広義の意味では返答もメッセージである.

データの動きをする μ-actor は, 送られた質問に対して適切な答を返すという動作をする. また, プロセージャに対応する μ-actor は行動が依頼された時に, その依頼に応じた行動を行う. 必要な場合には, 他の μ-actor へメッセージを送り, 必要なデータを尋ねたり, 行動の一部を依頼する. そして, 各 μ-actor からの返答をすべて受け取った後, その行動によって得られた結果, または, 行動の実行に失敗したという返答を出す. このようにして, システム全体の動作は, μ-actor の連鎖的な行動によって遂行される.

各 μ-actor 間の交渉にメッセージ転送のみが許されているが, このために次の特徴がある. 3は, 本研究では実現しなかったが, 並列処理可能な計算機または LISP システムを用いて μ-actor を遂行した場合には実現可能である.

1. μ-actor 間のメッセージ転送が一般的コントロールの基本である. 関数呼出し, 変数からのデータ参照等の操作はすべてメッセージ転送という一種類の行動により実行される.

2. μ-actor は他の μ-actor と互いに直接会話が可能であり, そのための他の μ-actor によるパイプを必要としない.

3. μ-actor へメッセージを送ることは全く副作用がない. このため, 最大限の並列処理が可能になり, μ-actor が混乱せずに同時に数個の会話をすることができる.

3. 知識ベース

本システムの知識ベースは, μ-actor を単語を対応させることにより構成される. 本章では, システムへの入力文と μ-actor 間に転送されるメッセージ, 単語 μ-actor の構成方法, 定義方法について述べる.

3.1 入力文とメッセージ

本研究の目的は知識構造としての μ-actor の動作を考察することなので, 他の部分 (自然語解析, 生成部分) を簡単にする. したがって, 入力文は以下のような制限された日本文を用いる. これらの日本文は, 普通の日本文を各単語に分ち書きし, 動詞の格により語順を統一した形をしている. メッセージには入力文と同じ形式が用いられるが, Schank の CD グラフ[5]のような内部表現でもよい. ここで擬似日本文を用いたのは使用者にとって分かりやすいからである.

入力文は BNF 表現を用いると次のようになる.

入力文::=(＊ **文**)

文::=**単文**／**条件文**／**定義文1**／**定義文2**

条件文::=(**単文** ナラバ **重文**)

定義文1::=(**単文** ヲ **重文** ト テイギスル)

定義文2::=(**名詞** ヲ **名詞句** ト テイギスル)

重文::=(**文**)／(**文**)＋**重文**

ここで, **単文**は, 文中の動詞が唯一であり, その動詞の格のわく組み (Case Frame) の要素がすべて満さ

```
変数Xは4である：(＊ 'X' ハ 4 デアル)          (文1)
「SはTで割り切れるか？」を「SをTで割る．余りは0であるか？」と定義する：
(＊ (?S ハ ?T デ ワリキレル カ?) ヲ
    (S ヲ T デ ワル) (アマリ ハ 0 デアル カ?)
    ト テイギスル)                              (文2)
Xは2で割り切れるか？：
(＊ X ハ 2 デ ワリキレル カ?)                   (文3)
```

Fig. 2 Japanese sentences and input sentences.

れ，語順が統制されている文である．**Fig. 2** に示すように，本論で扱う単文は主格がない．それは行動体が計算機であると仮定したためであり，計算機以外のものが行動体となる文章を扱う場合には主格を加える．この場合も，主格がない場合と基本的には同様の行動を行う．

"＊" は入力 μ-actor を表わす．入力 μ-actor は入力文をどの μ-actor へ送るのか決定する．文中の単語に対しては（定数，変数名，助詞および定義される単語を除いて）すべて対応した μ-actor が存在しなければならない．定数，変数に対応した μ-actor はあらかじめ用意しておくことができない．したがって，普通の変数名は最初に現われる時に "'" を用いて表わし，パタン照合用変数名はその前に "?" を用いる．たとえば，変数Xは 'X'，パタン照合用変数Yは?Y と表わす．対応した μ-actor がない単語中，変数，助詞以外で，数字で表わされるものは定数，その他は定義される単語と解釈する．

メッセージの形は（N L）である．Lは**文**である．Nはメッセージの系統を示すもので，処理結果を記憶する際に他のデータと区別するための目印となる．本システムでは入力 μ-actor により文番号として与えられる．例えば，後述するように "μ-actor 商" は割算の結果として得られる商を次々と蓄えるが，その際に文番号と結果を組にしておく．そうすることにより，多くの演算の答を整理して記憶することができ，これまで実行した割算すべてについて商に関する質問に返答することができる．特別に指定しない場合は文番号として %0 を用いる．

3.2 名詞 μ-actor の構成

名詞の定義には，(a)その固有値の定義，(b)その単語の特性の定義の2種類がある．

(a) 固有値の定義

ある単語の示す物の固有値（例えば，変数の値）の定義は文1 (Fig. 2) のように「…である」と断言することにより行われる．これは，事実の断言である．文1の処理について述べる．最初に文1は入力 μ-actor へ転送される．入力 μ-actor は，対応する μ-actor がない変数名Xと定数4を MAKEACTOR に送りそれぞれ μ-actor **X** と **4** を作成する．このとき，**4** のデータ部に値4を入れる．次に，μ-actor **デアル** に文1を送る．**デアル**はパタン照合により何を行うのか判断し，行動を起こす．ここでは，**X** に値として **4** を持つようにメッセージを送る．**X** では **4** に値を聞き，その値4をデータ部に記憶し，**デアル**に操作が完了したことを返答する．**デアル**も同様に，入力 μ-actor へ返答し，動作が完了したことを伝える．**Fig. 3** にメッセージの伝達を示す．

(b) 単語の特性の定義

物の特性および事柄の成立条件の定義は，主に**定義文2**によってなされる．例えば，"四角形の上に三角形があるものを家と定義する" "2で割り切れるものを偶数と定義する" 等である．処理の際に**定義文2**中の**名詞**に対応する μ-actor が存在しなければ，新しい μ-actor を作る．そして**名詞**に対応する μ-actor の行動情報部には，肯定文のパタンと疑問文のパタンを作り，それぞれに対応する行動を対にして入れる．例えば，肯定文パタンとして (?A ハ グウスウ デアル) と疑問文パタンとして(?A ハ グウスウ デアル カ?) を作る．行動は両方ともAが偶数であるかどうか確める動作である．この方法は，**定義文1**の処理（用語の定義）と同じであるので，次節で詳細に述べる．

3.3 用言の定義

用言の定義は**定義文1**を用いて記述される．定義される**単語**の**重文**中での使用を許すが，これにより再帰的定義が可能となる．定義文は μ-actor **テイギスル**に送られる．**テイギスル**では，**単文**の中心語の μ-actor が存在しなければそれに対応する μ-actor を作る．そして，その行動情報部に，**単文**をパタンとし，**重文**と組にして加える．文2 (Fig. 2) について述べよう．文2を受けた入力 μ-actor は**テイギスル**に入力文を送る．テイギスルは，「割り切れる」が未定義であるこ

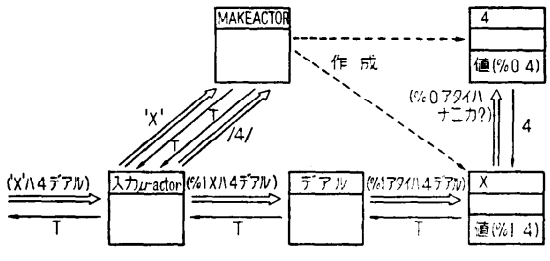

Fig. 3 A diagram for the computation of sentence **1.**

ワリキレル
◦ 制 御 部
◦ 行動情報部 ((?S ハ ?T デ ワリキレル カ?) ((S ヲ T デ ワル) (アマリ ハ 0 デアル カ?)))
◦ データ部

Fig. 4 Micro-actor "Divisible".

とから，以下の方法で μ-actor **ワリキレル**を作る．**ワリキレル**を **Fig. 4** に示す．

μ-actor は Fig. 1 に示すように形式化されているので以下のようにすれば簡単に μ-actor の作成ができる．制御部は各単語 μ-actor に共通した部分であるから，制御部のみ構成され，その他の部分を空白にしたものを用意する．μ-actor を作成する時には μ-actor 名と行動情報部，データ部に適当なものを代入するだけでよい．

4. 自然言語処理システム

前章では μ-actor を用いた知識ベースの構成について述べたが，本章では，これらの知識ベースの処理動作，および，多義語の処理，反意語間の関係について述べる．

自然言語処理システムでは，各単語に対応した単語 μ-actor の他に，入力 μ-actor，MAKEACTOR，MATCH，INF 等システムを維持するための μ-actor がある[4]．これらの説明は紙面の都合上省略する．また，以下の例では，「である」「割る」「引く」「商」「余り」に対応する μ-actor が用意されているものとする．μ-actor はその名前で示し，図中の μ-actor を表わすボックスの内容は説明に必要なもののみを示す．

4.1 処 理 動 作

四則計算やその他の演算を実行する場合には，その結果として他の μ-actor に影響を及ぼす．本システムでは，この影響は他の μ-actor へのメッセージ転送という形で明確に表わされる．また，会話中に定義された単語も，あらかじめ与えられている単語と同様に使用することができる．ワリキレルが使用された場合について，文3 (Fig. 2) を例にとって述べる．先に文1，文2が処理されたとする．この時の文番号を %3 とする．入力 μ-actor から μ-actor **ワリキレル**へ文3が送られると**ワリキレル**はパタン照合により，文2で定義された動作を行うことを決定する．最初に，「**X**を2で割る」を実行しようとし，**ワル**にこの文を送る．**ワル**もパタン照合により動作を決定し，最初に **2** と **X** にその値を尋ねる．**2** は常に値2を返答するが，**X** はメッセージと同じ文番号を持つデータがあればその値を，なければ最も新しく入った値を返答する．この場合は値4を返答する．次に**ワル**は 4÷2 を計算し，商と余りを得る．そして，**ショウ**と**アマリ**にそれぞれ値 (%3 2) と (%3 0) を送り，記憶させる．次に**ワリキレル**は「余りは0であるか？」を実行しようとし，この文を**デアル**に転送する．**デアル**は**アマリ**に値が0かどうか尋ねる．**アマリ**のデータ部には，割算の結果がすべて記憶されているのが文番号により，必要な結果を正確に見つけることができる．ここでは文番号が %3 なので，値として0が抽出され，返答として T (Yes) が**デアル**に返される．この返答と**デアル**は**ワリキレル**に返し，「**X**は2で割り切れる」ことがわかる．以下の動作の概略を **Fig. 5** に示す．

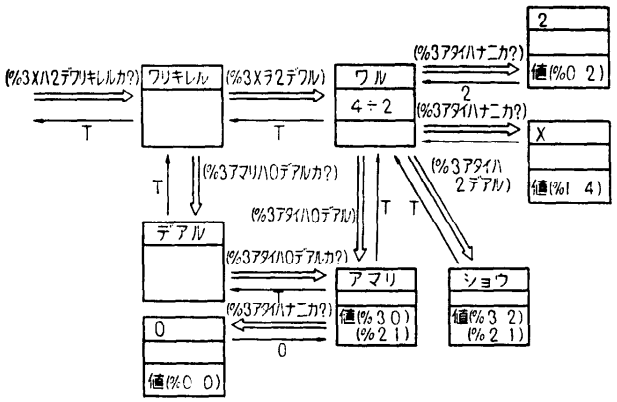

Fig. 5 A diagram for the computation of sentence 3.

4.2 多義語に対する処理

1つの単語に対応して μ-actor が1つ存在する．したがって，多義語に対応する μ-actor は，その単語がどの意味で使用されているのか判断しなければならない．μ-actor は2つの方法により意味を判別する．1つは，その品詞および動詞によって決定される格構造によるもので，メッセージを行動情報部のパタンと照合することにより判別する．他は，上の方法で判別できない時にセマンティックにより区別するもので，照合が成功したすべてのパタンについて，対応する行動を行う．文章が一意に決まるならば，唯一の行動が成功する．例えば，動詞「引く」は次の3種類の意味を持つ；(1)数字の引算，(2)線を描くこと，(3)物体の移動，これら3種類の「引く」に対して唯一の μ

-actor ヒクしか存在しない．この μ-actor は(1)に対して，パタン (?X カラ ?Y オ ヒク) を持つ．(2)に対しては (?X ヲ ヒク) または (?X カラ ?Y ニ ?Z ヲ ヒク) であり，(3)に対しては(?X ヲ ヒク) または，(?X ニ ?Y ヲ ヒク) である．パタン (?X ヲ ヒク) 以外のパタンに照合する入力文は唯一に意味が決定され，実行される．実行の際，メッセージの内容を意味的に処理を行い，条件が合えば実行し，そうでなければ失敗する．例えば，「山から3を引く」という文は(1)のパタンに照合するが，?X, ?Y は数字または変数名でなければならない．ところが，山は数字でも変数名でもないため，実行は失敗する．

パタン (?X ヲ ヒク) に入力文が照合した場合は(2)と(3)の2通りの意味が考えられる．両方の場合について上と同様にメッセージの内容を意味的に処理する．ここでは ?X の内容が存在していなければ(2)，実在の物体を表わしているならば(3)と解釈される．

4.3 反意語に対する処理

相互する意味を持つ2つの単語にそれぞれ対応する μ-actor 間の関係は，他の μ-actor と同様，単にメッセージを送り，仕事を依頼するだけである．「大きい」と「小さい」を例にとって述べよう．μ-actor **オオキイ**に (A ハ B ヨリ オオキイ カ?) が送られたとする．もし**オオキイ**がそれ自身でこの質問に答えられない場合に，**チイサイ**にこの仕事を依頼するように**オオキイ**の行動情報部に記述することができる．すなわち，パタン (?%N ?X ハ ?Y ヨリ オオキイ カ?) に対して (%N Y ハ X ヨリ チイサイ カ?) を実行するよう記述することができる．もし，**チイサイ**のデータ部にBがAより小さいという情報が入っておれば答として T (yes) が返され，AがBより大きいことがわかる．

5. むすび

我々の目標は，簡単に知識の定義ができ，かつその知識を利用し易いデータ構造を開発することにある．本システムでは，入力文は，統制されてはいるが日本語を用いているため人間にとって扱いやすい．また，すでに構成されている μ-actor はすべて利用可能であるため，定義された知識は常に使用可能である．したがって，計算機が理解，処理できる単語を基本単語とすれば，新しい知識を基本単語および定義された単語のみを使用して表現すれば，その知識は計算機に理解され，処理される．

μ-actor は Hewitt の actor の機能を縮小したものであるが，その主な特徴として **1.** で述べた3つの能力を有する．μ-PLANNER では宣言されたデータはすべて同じ記憶領域に記憶される．また，推論を行う際にどの手続的知識を使用するかは，主にパタン照合により決定される．直接，ある手続的知識を用いる場合には使用者によってその名前が指定されなければならない．一方，各 μ-actor はメッセージ転送により直接関連付けられており，手続的情報および宣言的情報はそれぞれ関係する μ-actor の行動情報部およびデータ部に蓄えられる．したがって，μ-actor は基本的には手続的知識を構成するが，セマンティックネットワークを表現することができ，またそれを取扱うこともできる．

今後の問題としては，3つ上げられる．最初は，入力文の自由度の拡張である．これには，日本文の解析能力が必要である．次は，並列処理への発展である．μ-actor は actor の scheduler の機能があれば並列処理可能な構造であるが，遂行する言語および計算機の制約から並列処理を実現するに至っていない．しかし，シュミレートにより μ-actor の動きを考察するのは可能である．最後は，文番号の処理である．本論では簡単に述べたが，並列処理および μ-actor の再帰呼出しを行う場合はもっと複雑な処理が必要であり，より深い考察が必要である．

終りに，本研究に関して熱心にご討論，ご助言していただいた大阪大学基礎工学部情報工学科田中研究室諸氏，特に北橋忠宏博士，ならびに同学科電子計算機室の工藤英夫氏，また，本論の知識構造を μ-actor と呼ぶことを快く承諾していただき，actor の文献を提供していただいた MIT の C. Hewitt 助教授に感謝します．

参考文献

1) C. Hewitt, P. Bishop, R. Steiger : A Universal Modular Actor Formalism for Artificial Intelligence, Proc. 3rd IJCAI, pp. 235～245 (1973).
2) C. Hewitt, B. Smith : Towards a Programming Apprentice, IEEE Trans. on SE. Vol. SE-1, No. 1, pp. 26～45 (1975).
3) C. Hewitt : Viewing Control Structures as Patterns of Passing Messages, A. I. Lab. MIT Working Paper 92 (1976).
4) 木島，小川，北橋，田中：自然言語処理における知識構造について，信学会技報，AL 76-44, pp. 51～58 (1976).

5) R.C. Schank : Conceptual Information Processing, p. 374, North-Holland Publishing Company (1975).
6) T. Winograd : Understanding Natural Language, p. 191, Academic Press, INC. (1972).
7) A Panel On Knowledge Representation, chaired by D. G. Bobrow, Proc. 5th IJCAI, pp. 983～992 (1977).

(昭和52年1月21日受付)
(昭和53年1月9日再受付)