知識処理のための2階層モデリング手法と複合多機能型知識処理言語 S-LONLI の開発†

芳賀博英†† 中所武司†† 大藤淑子††

知識処理システムの構築においては，基本技術として(1)知識処理システムの構築のためのモデリング手法と(2)多種多様な知識を自然に表現できる知識表現言語の開発が重要である．そのためにルール型や論理型，フレーム型などの各種の知識表現言語が提案されているが，個々の言語単独では多種多様な知識を自然に表現することは難しい．また複数の知識表現を利用することのできる言語もいくつか提案されているが，言語仕様が複雑になり十分使いこなすことが難しい．本論文ではこれらの問題を解決することを目的として，(a)対象となるシステムをマクロなレベルとミクロなレベルに分けてモデリングする新しい2階層モデリング手法を提案し，(b)2階層モデリング手法に基づき，マクロなレベルの記述をオブジェクト指向型表現で，ミクロなレベルの記述を論理型表現で行う複合型知識表現言語 S-LONLI とその処理系を開発，評価した．本言語はオブジェクト指向型表現の階層構造とメッセージパッシングにより，システムの大域的構造を自然に表現でき，論理型表現を用いて個々の事実と規則を厳密に表現できる．またオブジェクト指向型表現と論理型表現の境界を明確にしたことにより，言語仕様が単純になり，記述が容易になる．本言語を用いて記述実験を行い，2階層モデリング手法の有効性を検証し，知識処理システムの構築が容易になることを確認するとともに，事象駆動型プログラミングへの拡張方式を提案した．

1. はじめに

第5世代計算機プロジェクトの発足以来，知識情報処理技術の研究，開発が盛んになり，すでにいくつかの知識処理システムが実現されている．このような知識処理システムの構築においては，(1)知識処理システムを構築する際に，対象をどのようにモデリングすればよいかというモデリング手法と，(2)対象の持つ知識をできるだけ自然に表現できる知識表現言語の開発が重要である．

知識処理の研究が進むに従っていろいろな知識表現技法およびプログラミングの方法論が提案されてきた．たとえばルール型知識表現[1]は，人間の「認知-行動サイクル」を基礎とした知識表現方式である．この方式では，人間の知識を「条件とそれが満たされたときの行動」という形式でモデル化している．したがって人間の持つノウハウ的な知識の表現には適した方式である．しかし制御構造が明確でないので，決まりきった手順などの知識を表現するのにはあまり適さない．論理型知識表現[2]は人間の思考を形式化する手段としての論理学を利用し，三段論法を推論規則の基本とする知識表現方式である．そのために各知識が明確な意味を持ち，事実と規則の厳密な表現が可能であるが，すべての知識を論理の枠組みの中で整理しなければならず，例外的な知識やノウハウ的な知識の記述が難しい．フレーム型知識表現[3]は，ある種の構造を持った静的な知識の典型的な枠組みを表現するための方式である．人間の持つ多くの典型的な状況に対応して，フレームと呼ばれるデータ構造を用意し，必要に応じて値を埋め込むことのできる機構や手続き付加機構を用意している．したがって構造が明確で静的な知識を表現するのには良いが，動的に変更される知識や構造がよくわからない知識を表現するのは難しい．

一方，実用的な知識処理システムが開発されていく過程で，これらの単独の知識表現方式だけでは実際の知識処理システムの知識をすべて有効に表現できないことが明らかになってきた．たとえばプラントの運転を行うようなルールベースシステムでも，異常診断やオペレータのノウハウ的な運転規則などはルール形式で記述すると良いが，定型的な運転操作などは手順的な記述で行いたい場合が多い．しかしルール形式で手順的な知識を記述すると不自然な記述となる．

従来の知識表現方式は，いずれも人間の持つ知識の一つの側面にのみ注目した表現方式である．したがってこれらの知識表現方式においては，ある特定の知識を表現するのには適しているが，異なった種類の知識を一つの知識表現方式ですべて表現するのには無理がある．そこで複数の知識表現方式が利用できる複合型知識表現言語がいくつか提案されてきた．複数の知識

† The Development of 2-Level Modeling Method for Knowledge Processing and the Hybrid Knowledge Programming Language S-LONLI by HIROHIDE HAGA, TAKESHI CHUSHO and YOSHIKO OTO (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

表現を利用するための第1の方式は，多くの知識表現方式を包含する大規模な知識表現言語を設計するという方式である．つまり多くの知識表現の方法を一つに融合し，一つの言語の中で多くの知識表現可能な言語を設計することである．この方式の言語の例としては，LOOPS[6]やTAO[7]などのLispを基礎とした言語や，ESP[8]，MANDALA[9]などの論理型言語を基礎とした知識表現言語がある．しかしながら，このような融合方式には，次のような実用上の問題がある．

（1） 各知識表現はそれぞれ特有の設計哲学，つまり「どのような観点から知識をモデリングするか」という方法論を持っている．しかし一つの言語の中で各種の知識表現を利用できると，どのような観点から対象をモデリングすればよいのかがわかりにくくなる．

（2） 一つの言語の中で多くの知識表現を可能にするために，言語仕様が複雑になる．そのため修得が難しく，機能を十分使いこなすことができなくなる．

もう一つの方式としては個々の知識表現言語を分離し，相互にデータ交換を許す方式である．この方式の言語としては，フレームを基礎としたKRYPTON[5]がある．KRYPTONはフレームの構造を規定するT-BOXとT-BOXのフレーム型データを用いて，種々の関係を一階述語論理に似た構成で記述するA-BOXの記述に分離している．これは，データとプロシジャの記述を分離したことに相当する．このように記述を分離したことによって，宣言的な情報と構造的な情報を明確に区別することができるが，その結果ある知識に対する記述が分散化し，保守性などの点で問題が残る．

これらの問題を解決するために，本論文では第2章で，複数の知識表現方式の間の境界を明確にし，かつ個々の表現方式のモデリング手法の有効性を損なわずに複数の表現方式を利用できるようにするために，知識処理システムをマクロなレベルとミクロなレベルの2階層でモデリングし，マクロなレベルのモデリングでは，ある程度まとまった働きをする知識の集合を表す知識モジュールの間の関係を記述し，ミクロなレベルのモデリングは個々の知識モジュールの機能に適した知識表現で行う，という2階層モデリング手法を提案している．第3章でこの2階層モデリング手法に基づき，マクロなレベルのモデリングをオブジェクト指向表現で，ミクロなレベルのモデリングを論理型表現で行う知識処理言語S-LONLI (Super-LONLI) を提案している．第4章でS-LONLIの処理系を開発し，記述実験による評価を行っている．

2. 2階層モデリングに基づく複合型知識表現言語の設計思想[16]

第1章で述べたような問題点を解決するために，われわれは複数の知識表現言語を一つの言語に統合するのではなく，むしろ個々の言語の特徴を生かしつつ，相互の境界を明確にすることにより，複数の知識表現言語を利用できるようにするアプローチを採用した．

このアプローチを実現するために，われわれは2階層モデリングという新しいモデリング手法を提案する．2階層モデリング手法では，まずある程度の機能を果たす知識の集合である知識モジュールを想定し，その知識モジュール相互の関係を記述することにより，対象のマクロなレベルでのモデリングを行う．この段階では，個々の知識モジュールの内部の詳細には立ち入らず，その知識モジュールが行うことのできる外部仕様のみに着目する．次に個々の知識モジュールの内部のモデリングを行う．これは，外部仕様の実現法を考えることに相当する．このようにすることにより，知識ベースをトップダウン的に作成していくことができる．

2階層モデリングの手法を用いるためには，知識モジュールの間の関係を記述する言語と知識モジュールの内部を記述する言語が必要になる．われわれは知識モジュール間の関係を記述する言語として，オブジェクト指向型言語[4]を用いることとした．これは，オブジェクト指向型言語の次のような性質が，知識モジュール間の関係の記述に適していると考えたからである．

（1） 知識モジュールをオブジェクトにより自然に表現できる．オブジェクト指向型言語は現実の世界を自然にモデリングできるので，複雑な知識処理システムのモデリングには適している．

（2） オブジェクトにはスロットとメソッドが記述できるので，個々の知識モジュール固有の推論を記述することが可能である．

（3） オブジェクト指向型言語におけるオブジェクトの階層構造と性質の継承機能により，知識ベースの静的な構造が記述できる．

（4） オブジェクト間のメッセージ送信機能により，知識ベース内の知識モジュールの動的な関係

が記述できる.

(5) 上の(1)から(4)の性質は，オブジェクトの外部仕様的な特徴であり，オブジェクトの内部表現に依存しない.

(5)よりオブジェクト内の記述には，その知識モジュールの性質や機能に従って，各種の知識表現方式が利用できる．つまり個々の知識表現の境界として，この知識モジュール（＝オブジェクト）をとり，オブジェクトの内部は適切な知識表現言語で記述することによって，ひとつの言語の中で複数の知識表現方式を利用できる複合型知識表現言語を得られる.

この2階層モデリング手法と，それに基づく知識表現言語は次のような特徴を持っている.

(a) 個々の知識表現方式の持つプログラミング方法論や設計哲学などが混乱しない．ある知識モジュールを記述する時には，ある特定の知識表現方式を用いるので，そのモジュールの記述に際しては，そこで用いられている表現方式のプログラミング方法論なり設計哲学だけを考慮すれば良いことになる．これは知識モジュールの単位として，独立性の高いオブジェクトを用いたことによる.

(b) 知識表現言語の言語仕様が複雑にならない．ある知識モジュールの記述には，そのモジュールで利用される知識表現言語の言語仕様に，オブジェクト指向処理用の手続きを加えた言語を知っているだけで良い.

(c) 新しい知識表現方式を追加する場合，それを知識オブジェクトにカプセル化し，他のオブジェクトは自分の持っている他オブジェクトへのアクセス手続きを通してのみアクセスするので，オブジェクト内の実現を意識する必要がない．したがって，機能の拡張による言語の変更が最小限で済み，拡張性が高い

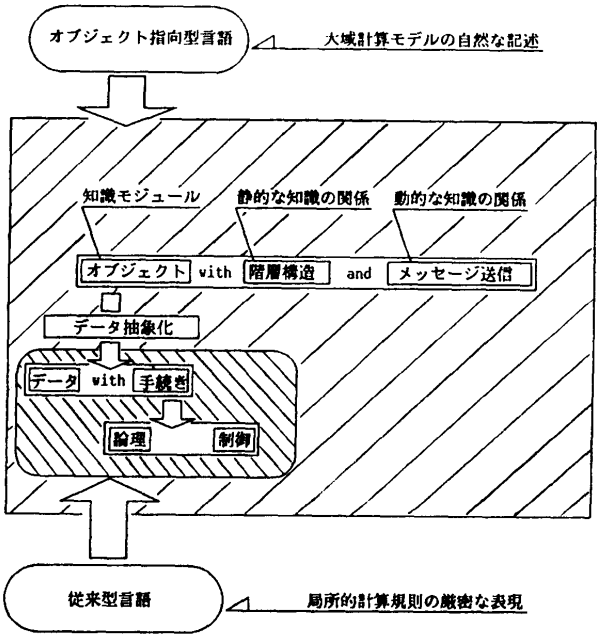


図 1　2階層モデルの概要
Fig 1　An overview of the 2-level modeling method.

図1に2階層モデリングの概念図を示す．このモデリングでは，対象のシステムは，構成要素間の関係を示すレベル（図1の■部分）と各要素の機能を示すレベル（図1の■部分）の二つの階層でモデリングされる．われわれはこのような特徴を持つ2階層モデリングを基礎として，知識モジュール内の記述言語として論理型言語を用いた複合多機能型知識表現言語 Super-LONLI（以下 S-LONLI と略す）を開発した.

3. 知識処理言語 S-LONLI の言語仕様と処理方式

3.1 構文仕様[11]

S-LONLI では，知識処理プログラムは個々の知識モジュールに対応するオブジェクト定義の集合として記述される．オブジェクトには，複数のオブジェクトの共通の性質を記述したクラスオブジェクトと，個々の実体に相当するインスタンスオブジェクトがある．クラスオブジェクトにはスロットの宣言とメソッドの定義がなされている．インスタンスオブジェクトには，スロットの値が宣言されている．図2に S-LONLI のプログラムの構成を示す．表1はプログラムを構成するキーワードの一覧表である.

スロットとは，個々のオブジェクトの状態を示す変数である．スロットにはクラススロットとインスタンススロットがあり，それぞれ別々に宣言する．スロットには表 2 に示すような属性を付けられる.

メソッドもスロットと同様に，クラスメソッドとインスタンスメソッドの2種類がある．メソッドの記述

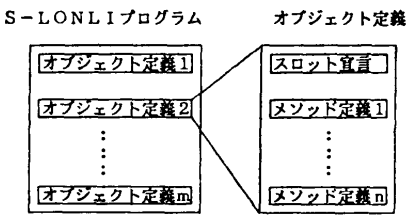


図 2　S-LONLI のプログラムの基本構造
Fig. 2　Basic structure of S-LONLI program.

表 1 S-LONLI のプログラムのキーワード
Table 1 Key words of S-LONLI program.

キーワード	機能
attribute	オブジェクトの属性（クラスオブジェクトかインスタンスオブジェクトか）を指定する
supers	定義しているクラスオブジェクトの親オブジェクトを指定する
meta	定義しているインスタンスオブジェクトのメタクラスを指定する
c_vars	定義しているクラスオブジェクトのクラス変数を宣言する
i_vars	定義しているクラスオブジェクトから動的に生成されるインスタンスオブジェクトのインスタンス変数を宣言する
c_method	定義しているクラスオブジェクトのクラスメソッドを定義する
i_method	定義しているクラスオブジェクトから動的に生成されるインスタンスオブジェクトのインスタンスメソッドを定義する

表 2 変数の属性
Table 2 Attributes of the variables.

属性名称	機能
if_accessed	この属性の付いた変数にアクセス（読出し）に行ったときに，読出し操作の前に，所定のメソッドを実行
if_deleted	この属性の付いた変数の値を削除しようとした時に，削除操作の前に，所定のメソッドを実行
if_replaced	この属性の付いた変数に値を書き込もうとした時に，書込み操作の前に所定のメソッドを実行
initial	この属性の付いた変数の初期値は所定の値であることの宣言
instance_of	この属性の付いた変数は，別のクラスオブジェクトのインスタンスであることの宣言

には，オブジェクト指向プログラミング用の組込み述語を追加した論理型言語 HI-UX PROLOG（以下単に PROLOG と言う）[14]により行う．

S-LONLI のメソッドは一般に次の形式をしている．

```
method m ;                          ①
  m :-…, n, …… ;                    ②
    ⋮
  n :-…… ;                          ③
end ;                               ④
```

①はこれから定義するメソッドの名称を宣言している．ここで宣言された名称（この場合ならmという名称）のみが，このオブジェクトの外から見える．②はこのメソッドの定義の本体である．メソッド定義の本体は，メソッド名称を示す頭部と本体部から構成されている．③はこのメソッドの定義だけに用いる補助メソッドの定義である．このように定義された補助メソッドは，このオブジェクトの外部からは見えない．④はこの m というメソッドの定義の終了を示している．

表 3 オブジェクト指向プログラミング用組込み述語
Table 3 Built-in predicates of S-LONLI for the object-oriented programming.

述語名	機能
gen	クラスオブジェクトからインスタンスオブジェクトを生成する述語
send	メッセージをオブジェクトに送る述語
change	スロットの値を変更する述語
do	クラスを指定してメソッドを実行する述語
kill	オブジェクトを削除する述語

メソッド定義において，②あるいは③は通常の PROLOG のプログラムと同じ形式をしている．メッセージ送信やその他のオブジェクト指向用の組込み述語も，通常の PROLOG の述語と全く同じ形式をしているので，②と③は通常の PROLOG のプログラミングと全く同じ形式で記述できる．**表 3** はオブジェクト指向用プログラミングのために提供されている組込み述語の一覧表である．**図 3** に S-LONLI で記述したプログラムの例を示す．これは，内部にカウントを示すスロットを持ち，up, down, show などのメッセージに対応するメソッドを持つ“カウンタ”というオブジェクトの定義である．

3.2 意味仕様[12]

3.2.1 オブジェクトの解釈

S-LONLI のオブジェクトを論理型言語の側面から見ると，次のようなことが言える．

オブジェクトは一つの世界（world）を表現していると考えることができる．世界とはいくつかの論理式の集合である．通常の PROLOG では，内部データベースがこの世界に相当する．そしてその後ろに，論理式を解釈／実行するインタプリタが付随している．オブジェクト指向型言語では通常複数のオブジェクトが定義されるが，その一つ一つが独立した世界を構成している．したがって論理型言語から見ると，オブジェクト指向型言語は多重世界を実現していることになる．各世界においてはその世界固有の処理が定義されており，基本的には他の世界（オブジェクト）とは無

```
define_frame counter ;                                  /*オブジェクト定義の開始*/
    attribute class ;                                   /*クラスオブジェクト定義の
                                                          宣言*/
    supers root ;                                       /*親クラスはルートである
                                                          ことの宣言*/
    i_vars count(initial(0)) ;                          /*インスタンス変数の宣言*/
    i_method up;                                        /*インスタンスメソッドの定義
                                                          開始*/
        up :- NewCount is count + 1, change(count, NewCount), show ;
        show :- write('Value ==> '), ,write(count) ;
    end ;                                               /*インスタンスメソッドの定義
                                                          終了*/
    i_method set ;                                      /*次のインスタンスメソッドの
                                                          定義開始*/
        set(V) :- change(count,V) ;
    end ;                                               /*インスタンスメソッドの定義
                                                          終了*/
end.                                                    /*オブジェクト定義の終了*/
```

図 3 S-LONLI のプログラム例
Fig. 3 Sample program of S-LONLI.

関係である.

オブジェクトにおけるスロットとは，そのオブジェクトの状態を表現するものであるが，それを論理的に解釈すると次のようになる．ある時点において，あるオブジェクトのスロット s の値が s_1 であるということは，そのオブジェクトが定義する世界において，「『スロット s の値が s_1 である』ということが真である」という論理式が成立すると解釈することができる．したがってスロット s の値が s_1 から s_2 に変化するということは，それまで成立していた論理式「スロット s の値が s_1 である」ということが偽となり，新たに「スロット s の値は s_2 である」が真になったということに相当する．あるいは，「スロット s の値が s_1 である」世界が消え，「スロット s の値は s_2 である」という新しい世界が生成されたという風に解釈することもできる．

オブジェクトを世界であると考えると，メッセージ送信は，ある世界への証明の依頼と考えることができ，論理的意味は demo 述語[17]の実行であると考えることができる．このようにオブジェクト指向の概念は「オブジェクトが世界である」という解釈を行うことによって，論理型言語の枠組みで自然に解釈することができる．

3.2.2 階層と継承の解釈[16]

階層構造と継承規則の解釈については，閉世界仮説 (closed world assumption) の観点から次のように解釈することができる．一般に継承の解釈としては，次の二つの解釈が可能である．

(1) コピールール：この解釈では，あるオブジェクトの上の階層にあるオブジェクトに定義されたスロットとメソッドは，すべて下のオブジェクトにコピーされると解釈する．図 4 のようなオブジェクトの階層構造においては，階層的に上にある c_1, c_2 にあるスロット s_{11}, s_{12}, s_{21} はすべてオブジェクト c_3 にコピーされたものと解釈する．メソッドの解釈も同様に行う．

(2) パスルール：この解釈では，上の階層にあるオブジェクトに定義されたスロットとメソッドは直接は継承されず，オブジェクト間の階層関係だけが存在する．そして，あるオブジェクトが自分で処理ができないメッセージを送られた時には，階層関係に従って自分の上位のオブジェクトにそのメッセージを再送信することによって処理を行う．

オブジェクト指向型言語という側面から見た場合，この二つの解釈は同じ意味を持っている．しかし論理型言語という側面から見ると，意味が異なっている．コピールールは，一つのオブジェクトの中に，実行の対象と成るメソッドがすべて含まれているので，一つ

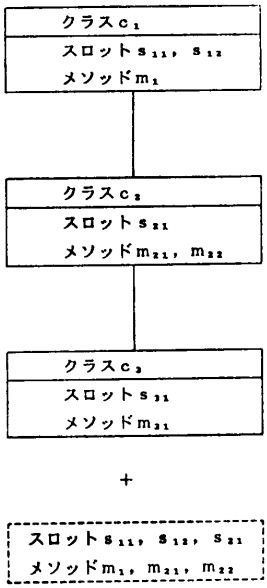


図 4 オブジェクトの階層とスロット，メソッドの継承の様子
Fig. 4 Object hierarchy and the inheritance of the slots and methods.

一つのオブジェクトに対して閉世界仮説が成立する．パスルールでは，あるオブジェクトの中でメソッドの実行に失敗したら，そのオブジェクトの上位オブジェクトにメッセージを再送するので，一つのオブジェクトだけでは閉世界仮説が成立しない．閉世界仮説はPROLOG のセマンティクスと密接に関連しており，S-LONLI のメソッド記述と PROLOG のセマンティックギャップをできるだけ避けるために，われわれは閉世界仮説との親和性から，コピールールを採用した．

継承規則のコピールール解釈により，メソッド記述の意味について，次の規則が成立する．

（a） メソッドの起動において，メソッドの名称のみが記述されている場合には，そのメソッドは自分または自分よりも階層的に上のオブジェクトからコピーされたメソッドを意味する．

（b） メソッドの起動において，メッセージ送信の形式で記述されている場合には，そのメソッドはメッセージ送信先に定義されたメソッドを意味する．

3.3 処理方式[13]

図5に S-LONLI の処理方式の概要を示す．この図に示すように，S-LONLI の処理系は，

（1） S-LONLI のソースプログラムを実行形式中間語に変換するフレームコンパイラ

（2） 実行形式中間語を解釈実行する S-LONLI インタプリタ

の二つから構成されている．また実行形式中間語は，PROLOG の節で表現されている．これらのことにより，移植性の向上，開発期間の短縮，PROLOG コンパイラの利用による処理速度の向上などがはかれる．

実行形式中間語には次の3種類の節がある．

（a） クラス表現節：クラスオブジェクトの情報のうちメソッド本体の情報以外の情報を表現する節

（b） インスタンス表現節：インスタンスオブジェクトの情報を表現する節

（c） メソッド表現節：メソッド本体を表現する節

（a）と（b）は本体部を持たない PROLOG の事実節，（c）は頭部と本体部を持つ PROLOG の規則節によって表現されている．**図6**は図3で示したカウンタのプログラムを実行形式中間語で表現したものである．このうち，“class” という述語名を持つ節はクラス表現節，“method” という述語名を持つ節はメソッド表現節，“instance” という述語名を持つ節はこのクラスから生成した “c” というインスタンスを表現したインスタンス表現節である．

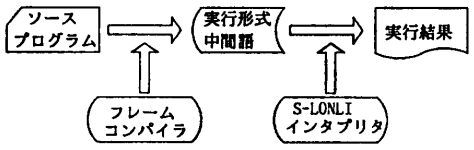


図 5 S-LONLI 処理系の基本構造
Fig. 5 Structure of S-LONLI processor.

```
class(counter,instance,[]).
class(counter,i_methods,[up,set,down,reset]).
class(counter,c_methods,[]).
class(counter,supers,[root]).
class(counter,i_vars,initial,0).
method(counter,up,global,instance,Self):-
    getValue(Self,count,_count),
    send(self,is(NewCount,+(_count,1)),self),
    send(self,change(count,NewCount),self),
    am(counter,up,show).
method(counter,show,up,instance,Self):-
    send(self,write('Value ==> '),self),
    getValue(Self,count,_count,_count),
    send(self,write(_count),self).
method(counter,set(V),global,instance,Self):-
    send(self,change(count,V),self).
```

図 6 中間語の一例
Fig. 6 Sample of the intermediate code.

S-LONLI の処理系はすべて，PROLOG で記述されている．その規模はフレームコンパイラ，S-LONLI インタプリタともに約1キロラインである．

4. 記述実験と評価

4.1 記述実験

S-LONLI を用いて，いくつかのシステムを記述し機能の評価を行った．記述したシステムとしては，簡単なマルチウインドウシステム，フェリーの運航のシミュレーション，ハミング問題，在庫管理システム，事象駆動型システムなどがある．本論文ではシステムのモデリングのしやすさの評価という観点から在庫管理システム，知識処理の応用の評価という観点から，複数のオブジェクトが協調して一つの問題解決に当たる事象駆動型システムの記述例を述べる．

（1） 在庫管理システム[10]の記述

この問題は次のような問題である．

複数の種類の品物が混載されたコンテナが倉庫に運び込まれる．一方倉庫には複数の種類の品物の出庫依頼が来る．出庫依頼に対して，倉庫に品物がある場合には出庫するが，ない場合には在庫不足を記録しておいて，必要な品物が入庫した時に出庫す

る．このような処理を行う倉庫の受付係のプログラムを作成せよ．

ここではまず次の五つのオブジェクトを作成した．これらはいずれも現実の世界に現れる「モノ」をオブジェクトに対応させたものである．したがって機能も現実の世界の「モノ」が持つ機能をそのまま自然に実現すれば良い．

受付係：工場や営業所などからの入庫／出庫依頼を受け，倉庫係に指示を行うオブジェクトである．

倉庫係：受付係からの指示に従って，品物を入庫／出庫したり，在庫票，在庫不足票などを管理し，更新するオブジェクトである．

積荷票：入庫の際に品物に付属しているオブジェクトである．これは倉庫係に渡されて，倉庫係はこのデータを基にして自分の管理する各種の帳票を更新する．

在庫リスト：受付係の持つ在庫リストを表現するオブジェクトである．品物の入／出庫の際に倉庫係によって更新される．

在庫不足リスト：受付係の持つ在庫不足リストを表現するオブジェクトである．出庫の際に在庫不足があれば，受付係がこのリストに記述する．また品物の入庫があれば更新される．

上のオブジェクトのうち，在庫リストと在庫不足リストは，記述される内容と使い方が違うだけで，オブジェクトとしては同じ構造のものである．そこで，オブジェクト指向型言語の継承機能を用いて，これらの二つのオブジェクトの上位オブジェクトとして，一般的なリストを表現する次のオブジェクトを定義した．

リスト：リストという汎用的なデータ構造を表現するオブジェクト．システムに現れるリストはすべてこのオブジェクトのサブオブジェクトとする．

これらのオブジェクトは図7に示すような階層構造を持っている．図8は倉庫係のオブジェクトのソースコードの一例である．個々のメソッドは論理型表現で記述されている．

（2） 事象駆動型プログラミングの記述

次に事象駆動型システムの記述について述べる．オブジェクト指向型言語を用いて事象駆動型システムを記述する方法としては，次の二つが考えられる．

（a） 全体の実行を制御するオブジェクトを作成し

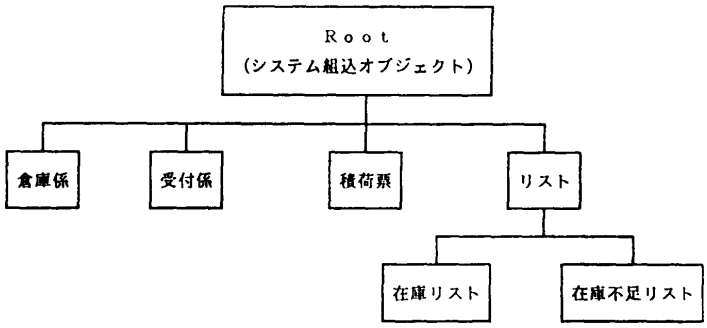


図 7 “在庫管理問題”におけるオブジェクト構造
Fig. 7 Object hierarchy of the “store control problem.”

```
define_frame 受付係
    i_vars 受付係名称,
           在庫リスト(instance_of(リスト)),
           在庫不足リスト(instance_of(順序リスト));
    i_method 出庫依頼;
        出庫依頼(_名称,_数量,_出庫先):-
              send(在庫リスト,在庫検査(_名称,_答)),
              answer_user(_出庫先,_名称,_数量,_答);
        answer_user(_出庫先,_数量,[]):-
              send(在庫不足リスト,不足登録(_出庫先,_名称,_数量));
        answer_user(_出庫先,_数量,在庫(_名称,_在庫数,_リスト)):-
              _数量>_在庫数,!,_不足数 is _数量-_在庫数,
              send(在庫不足リスト,不足登録(_出庫先,_名称,_不足数));
        answer_user(_出庫先,_数量,在庫(_名称,_在庫数,_リスト)):-
              出庫処理(_数量,_リスト_更新リスト),answer_1(_在庫数,_数量,_名称);
                 :
    end;
    i_method 入庫;
        入庫(_名称,積荷リスト):-
              在庫不足リスト更新(積荷リスト,更新後積荷リスト),
              send(在庫リスト,更新(_名称,_更新後積荷リスト));
        在庫不足リスト更新([],[]);
        在庫不足リスト更新([積荷(_名称,_数量)|_残りの積荷],
                           [積荷(_名称,_残り数量)|_新しい残りの積荷]):-
              send(在庫不足リスト,不足票ピックアップ(_名称,_不足リスト),
              リスト更新(_数量,_不足リスト,_更新不足リスト,_不足数),
              send(在庫不足リスト,追加(_更新不足リスト));
              在庫不足リスト更新(_残りの積荷,_新しい残りの積荷);
                                    :
    end;
end.
```

図 8 “受付係”のプログラムの一部
Fig. 8 Some part of the program of “uketsuke-gakari.”

て，ある事象が起こったときに，各オブジェクトが自分がその事象に対して動くべきかどうかを実行制御オブジェクトに問い合わせる方法．

(b) 実行制御オブジェクトがシステムの状態をすべて把握して，事象が起こるたびに動くべきオブジェクトにメッセージを送信する方法．

今回の記述実験では，次のような問題に対して，(a)，(b)両方法を用いた記述実験を行った．

ある大きさの部屋に一匹のネコと数匹のネズミが閉じ込められている．この状態でネコはネズミを追いかける．ただしネコは近眼で自分を中心としたある領域しか見えない．この領域にネズミがいればそれを追いかける．いなければランダムに部屋を動く．ネズミはランダムに動く．この状況をシミュレートするプログラムを作成せよ．

今回の記述実験では，実行制御オブジェクトとネコとネズミを表すオブジェクトを作成した．これらのオブジェクトはすべてルートオブジェクトのすぐ下に配置される．ネズミは自分の位置を変えるたびに実行制御オブジェクトに位置を知らせる．またネコは移動するたびに自分の見える範囲にネズミがいるかどうかを実行制御オブジェクトに問い合わせ，いる場合にはそれを追いかけ，いない場合にはランダムに部屋を移動する．このようなプログラミングをすることにより，事象駆動型システムの記述を行うことができた．

4.2 評価と考察

(1)の記述例では，現実の世界とプログラムの構造が比較的よく一致しており，オブジェクト指向型プログラミングを採用したことにより，プログラム作成が容易になっている．在庫管理システムの記述においては，2階層モデルとオブジェクト指向型プログラミングを採用したことにより，全体のモデリングが容易になっている．図7に示したオブジェクトはすべて現実の世界に現れている「モノ」であるが，このようなオブジェクトの構成を考えることが，プログラムのモジュール分割を行うことに相当している．したがってマクロなレベルでのモデリングを行うときには，現実の世界に対応させて行えば良いことになり，プログラムの構造の設計が容易になっている．次に各オブジェクトの機能の実現に関しては，オブジェクト内の記述言語が単一の言語であるので，LOOPS や TAO, ESP などとは異なり，一つのオブジェクトを記述している限りは単一の言語によるプログラミングとほとんど同じようにプログラミングができる．S-LONLI の場合はオブジェクト内の記述言語が論理型言語であるので，論理型言語のことだけを知っていればプログラミングが可能である．たとえば図8の“出庫依頼”というメソッドの本体部は“send”を組込述語であると考えれば，普通の PROLOG のプログラムと同じ形式，意味を持っている．このようにマクロなレベルとミクロなレベルでのモデリングを分離したことにより，システム全体の構成が容易になり，個々のオブジェクトの記述も容易になる．またオブジェクト指向プログラミングを採用したことにより，静的なフレームを基礎とした言語，たとえば KRYPTON などに比べ，記述したシステムの柔軟性が向上している．この例の場合で言うならば，積荷はインスタンスオブジェクトを生成するだけでよく，静的なデータ構造（たとえば KRYPTON の T-BOX）を変更する必要はない．また，積荷の種類が増加した場合にも，システム全体に影響を及ぼすデータ構造を変更する必要はなく，種類の増加に関与するオブジェクトの，メソッドの変更だけで処理でき，保守性が向上している．

(2)の記述例では，実行制御オブジェクトという特殊なオブジェクトを導入することによって記述を行った．しかし，このような実行制御オブジェクトを設けることには，次のような問題がある．

(a) 実世界には存在しない余分なオブジェクトが現れる．

(b) 実行制御オブジェクトはシステムのすべての状態を知っており，機能が不自然である．

そこで，実際の世界に現れないオブジェクトを記述する必要がないという点において，事象駆動型システムの記述をより自然に行うために，次のような機能拡張を行った[15]．

システムの組込みオブジェクトとして，上で述べたような実行制御を行うオブジェクトを提供する．このオブジェクトは，別のオブジェクトを含むことができる一種の空間であり，その中には事象が漂っている．また各オブジェクトは漂っている事象を捉えるためのアンテナと，漂っている事象を弁別するためのチューナを持っている．オブジェクトについているアンテナは，常に空間に漂っている事象を監視しており，自分に同調する事象があればそれを捕まえてオブジェクト内に取り込む．事象駆動的なプログラミングを行いたい場合には，チューナの同調条件とその時の処理をあらかじめ記述しておき，実行時に事象駆動プログラミングに関連するオブジェ

クトをこの実行制御を行うオブジェクトの中に入れる.

このような実行制御を司るオブジェクトを導入することによって,利用者が不自然なオブジェクトを導入することなく,事象駆動的なプログラムを行うことができる.先のネコとネズミの問題においては,ネズミが移動するたびに,自分の位置を事象として空間に放出する.ネコの持つアンテナの同調条件としては,ネズミから放出された事象の中の位置情報と自分の位置から,その事象を放出したネズミが捕獲可能かどうかを判断するようなものとし,捕獲可能であればそのネズミを追いかけるようにすればよい.このようにすれば事象駆動型プログラミングをより自然に実現することができる.

S-LONLI で採用したオブジェクト指向型プログラミングは,オブジェクトがメッセージを受け取らない限りなにも実行しない,という意味で,受動的なプログラミングであると言える.これは,C, Fortran, PROLOG などの言語にも言えることであり,プロシジャコールによって起動されない限り,プロシジャは何もしない.一方事象駆動型プログラミングは,プロシジャ実行の主体がみずから自分の実行の可否を判定し行動を起こす,という意味で,先の受動的なプログラミングとは対照的な能動的なプログラミングである.S-LONLI において,このような能動的プログラミング機能を持つ組込オブジェクトを提供したことにより,現在の知識処理システムにおいて主流となっているルールベースシステムなども自然に記述できる.これによって広範な知識処理システムの記述が可能になる.

5. まとめ

知識処理システムの構築のためのモデリング手法と,複数の知識表現方式の使用が可能な知識表現言語の開発を目的として,マクロなモデルとミクロなモデルで対象をモデリングする新しい2階層モデリング手法を提案し,それを基にした複合多機能型知識処理言語 S-LONLI を開発し,記述実験による評価を行った.

2階層モデリング手法により,あるまとまった機能を果たす知識の集合である知識モジュールごとに,その機能に最適な知識表現方式が選択でき,一つのシステムの中で複数の知識表現方式を使用することが可能になった.本手法は従来の複合多機能型言語のように,複数の知識表現方式を使える一つの大きな言語を設計するのではなく,個々の知識表現の記述体系とそれらの知識表現言語で記述された知識の間の関係を記述する記述方式を提供することによって,個々の知識表現言語の間の境界を明確にして複数の知識表現方式の使用を可能にするものである.

S-LONLI は階層構造と継承機能およびメッセージ送信機能により対象が自然にモデリングできるオブジェクト指向型言語でマクロなモデルを記述し,個々の事実と規則を厳密な形式で記述できる論理型言語でミクロなモデルを記述する.このような機能を持つ S-LONLI による記述実験により,次のことを確認した.

(1) システム全体のモデリングにおいては,モデリング対象の世界にある「モノ」に対応したオブジェクトを定義することにより行えるので設計が容易になる.

(2) 個々のオブジェクトの記述においては,単一の言語によるプログラミングと同じ方法で行えるので,プログラミングが容易になる.

また知識処理システムのプログラミング技法として重要な事象駆動プログラミング機能をより自然に行うために,事象によってオブジェクトが起動することのできる特殊な空間を提供する方式を提案した.

謝辞 本研究の機会を与えていただいた(株)日立製作所システム開発研究所所長堂免信義氏,同研究所第5部部長石原孝一郎氏,S-LONLI の記述実験にご協力いただいた同研究所第5部吉浦裕氏,古賀明彦氏,本論文執筆時に貴重なご意見をいただいた同研究所第5部主任研究員絹川博之博士に深謝いたします.

参考文献

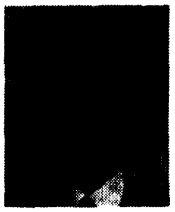
1) Barr, A. et al. 田中,淵訳:人工知能ハンドブック第1巻,共立出版,東京 (1983).
2) Kowalski, R.: *Logic for Problem Solving*, North-Holland , New York (1979).
3) Minsky, M.: A Framework for Representing Knowledge, in Winston, P. (ed.), *The Psychology of Computer Vision*, McGraw-Hill (1977).
4) 米澤:オブジェクト指向プログラミングについて,コンピュータソフトウエア, Vol. 1, No. 1, pp. 29-41 (1984).
5) Brachman, R. et al.: Krypton: A Functional Approach to Knowledge Representation, *IEEE Comput.*, Vol. 16, No. 10, pp. 67-73 (1983)
6) エキスパート・システム開発のツール—Xerox

PARC の LOOPS, 日経コンピュータ, 1983. 9. 19号, pp. 99-113 (1983).
7) Takeuchi, I. et al.: TAO—A Harmonic Mean of Lisp, Prolog and Smalltalk, *ACM SIGPLAN Notice*, Vol. 18, No. 7, pp. 65-74 (1983).
8) Chikayama, T.: Unique Features of ESP, *Proceedings of FGCS' 84*, pp. 292-298 (1984).
9) 古川ほか: MANDALA: Concurrent Prolog 上の知識表現システム, 情報処理学会知識工学と人工知能研究会資料, 32-1 (1983).
10) 山崎: 共通問題によるプログラム設計技法解説, 情報処理, Vol. 25, No. 9, p. 934 (1984).
11) 芳賀ほか: 知識情報処理システム記述言語 S-LONLI の提案, 日本ソフトウエア科学会第1回全国大会論文集, 2D-2 (1984).
12) 芳賀ほか: オブジェクト指向型言語と論理型言語の融合に関する一考察, 第34回情報処理学会全国大会論文集, 2J-1 (1987).
13) 芳賀ほか: On the Prototype Processor for Knowledge Information Processing System Description Language S-LONLI, 日本ソフトウエア科学会第2回全国大会論文集, 4B-5 (1985).
14) 広瀬ほか: 知識処理言語 "VOS 3/HI-UX PROLOG" の高速化方式とプログラミング環境, 日立評論, Vol. 69, No. 3, pp. 209-214 (1987).
15) 芳賀: オブジェクト指向型言語への事象駆動型プログラミング機能の導入の一方法について, 第33回情報処理学会全国大会論文集, 5D-7 (1986).
16) Chusho, T. et al.: A Multilingual Modular Programming System for Describing Knowledge Information Processing Systems, *Proceedings of IFIP '86*, pp. 903-908, North-Holland, Amsterdam (1986).
17) Bowen, K. A. et al.: Amalgamating Language and Meta-language in Logic Programming, in Clark, K. and Tarnlund, S. (eds.), *Logic Programming*, pp. 153-172, Academic Press, London (1982).

(昭和63年3月30日受付)
(昭和63年11月14日採録)

芳賀 博英 (正会員)

1954年生. 1978年同志社大学工学部電子工学科卒業. 1980年同大学院修士課程修了. 同年(株)日立製作所入社. 現在同社システム開発研究所関西分室において, 知的プログラム作成支援システム, プログラミング用エキスパートシステム, 知的 CAI 等のソフトウェア工学と知識処理技術の境界領域の研究開発に従事. 日本ソフトウェア科学会, 人工知能学会, The Association for Logic Programming 各会員.



中所 武司 (正会員)

昭和21年生. 昭和44年東京大学工学部電子工学科卒業. 昭和46年同大学院修士課程修了. 同年(株)日立製作所入社, 現在同社システム開発研究所主任研究員, 東京工業大学非常勤講師. 工学博士(東京大学). 入社以来, プログラミング方法論, 言語と処理系, テスト技法, 構造エディタ等のソフトウェア工学および知識処理言語, エキスパートシステム構築ツール等の知識工学の研究に従事. 昭和58年度情報処理学会論文賞, 昭和61年度大河内記念技術賞受賞. 著書「人工知能」,「プログラミングツール」(昭晃堂, 共著). 人工知能学会, 日本ソフトウェア科学会, IEEE Computer Society 各会員.

大藤 淑子

1961年生. 1984年津田塾大学学芸学部数学科卒業. 同年, (株)日立製作所入社. 同システム開発研究所所属. 知識ベースシステムの研究に従事. 日本ソフトウェア科学会会員.