第 9 章

人工知能

9.1 知識ベースシステム

★

ルール形式の知識の集まりとして記述された
プログラムは，わかりやすい。

9.1.1 人工知能と知識工学

人工知能システムは，人間の頭脳の知的作用の一面をモデル化したものと考えられる。人工知能の研究は，後に Lisp を設計する J.McCarthy，プロダクションシステムを開発する A. Newell，フレーム理論を提唱する M. Minsky らが 1956 年にダートマス大学に集まり，コンピュータを用いた知能の実現に関する研究を人工知能と名付けたころから活発に行われてきた。しばらくは人工知能実現のための一般的原理の研究が中心となり，多くの基礎技術が開発されたが，実用化には至らなかった。

この反省から，人間の問題解決能力の基本となっている経験的知識に着目し，問題領域に固有の知識を利用する応用人工知能の研究が 1970 年前後から開始された。この分野の先駆者である E. Feigenbaum は，1977 年の第 5 回人工知能国際会議（The 5 th IJCAI：International Joint Conference on Artificial Intelligence）の招待講演でこの分野の研究を知識工学（Knowledge Engineering）と名付けた。

9.1.2 知識ベースシステム

知識工学の大きな特徴は，従来のコンピュータ応用技術では解決が難しい悪構造問題 (ill-structured problem) を扱えることである。すなわち，対象が数値データではなく記号データで表現され，解法のアルゴリズムが定式化できない問題をその分野の知識に基づいて解くことができる。このような知識工学を応用したシステムを一般に知識ベースシステム (knowledge-based system) と呼ぶ。特に特定分野の専門家の持つ専門知識を利用したシステムは，エキスパートシステム (expert system) と呼ばれる。

知識ベースシステムの概念図を 図 9.1 に示す。知識ベースには，

「花子は母親である」

という事実や，

「ある人が母親ならば，その人は女性である」

という規則などの知識が蓄えられている。推論エンジン（推論プログラム）は，この知識を利用して問題解決のための推論を行う。基本的な推論方式としては，前向き推論と後向き推論がある。前向き推論 (forward reasoning) は，「花子は母親である」という事実と「ある人が母親ならば，その人は女性である」という規則から，

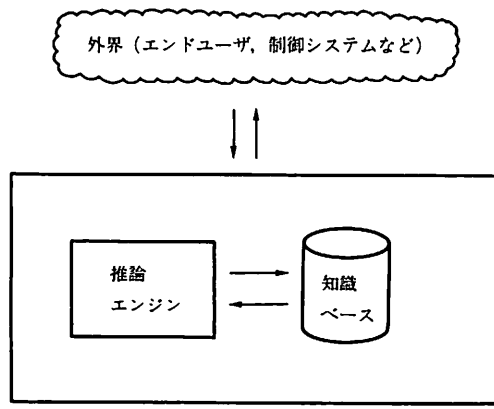


図 9.1 知識ベースシステムの概念モデル

「花子は女性である」

という結論を導く推論方式である。

一方，後向き推論（backward reasoning）は，

「花子は女性か？」

という問題を解くために，「ある人が母親ならば，その人は女性である」という規則を逆方向に利用する。すなわち，この規則を

「その人が女性であるためには，その人が母親であればよい」

と解釈することにより，「花子は女性か？」という与えられた問題を解く代わりに，

「花子は母親か?」

という問題を解くことを試み，

「花子は母親である」

という事実から，結局，与えられた問題の正しさを証明する推論方式である。

9.1.3 簡易プログラミング

このような知識ベースシステムやエキスパートシステムが種々の分野で実用になっているが，その理由は，"推論機能"というよりはむしろ"簡易プログラミング"としての魅力にある。職業的プログラマでない業務専門家が業務の言葉で表現できる利点が大きい。さらに，もっともよく用いられている知識表現であるプロダクションルールの場合は記述形式が「もし～ならば，～せよ」という1つのパターンに限定されており，かつ基本的には記述順序が自由であるので，業務知識の記述，追加，修正が容易である。

従来の手続き型プログラムが，N. Wirthの図式で，

プログラム＝アルゴリズム＋データ構造

と表現されたのに対応させると，知識ベースシステムでは，図9.1に示すよう

に,

知識ベースシステム＝推論エンジン＋知識ベース

となるが，アルゴリズムに対応する推論エンジンはすでに用意されている。したがって，ユーザは，データに対応する知識だけを記述すればよい。

本章では，実用的な観点から，広く利用されているプロダクションシステムおよびフレーム表現のプログラミングとしての簡易性について例題を中心に述べる。

9.2 プロダクションシステム

9.2.1 プロダクションシステムの特徴

プロダクションシステムは，1967年に米国カーネギーメロン大学のA. Newellらが，人間の認知モデルに基づく問題解決方法として考案したものである。実用になっているエキスパートシステムやその構築ツールの多くがこの機能を備えている。

プロダクションシステムの主な特徴は，以下のようなものである。

①知識は「if 条件 then 行動」というルール形式で簡潔に表現されるため，コンピュータに不慣れな人にとっても自然で理解しやすい。

②前向き推論機能により，初期条件が与えられると条件を満たすルールの実行を繰り返して結論を導くことができる。

③後向き推論機能により，目標が与えられると目標を結論とするルールの条件が満たされているか否かの判定を繰り返して目標の真偽を確認できる。

9.2.2 プロダクションルール

人間の世界には多種多様な知識が存在するが，「if 条件 then 行動」というルール形式で簡潔に表現されるノウハウ的な知識が重要な役割を持つ場合が多い。プロダクションシステムでは，このような知識をプロダクションルール (production rule)として記述する。これが手続き型プログラムの場合のプログラムに対応する。

本節では，以下のような例題を用いて説明する。

自動車自動運転エキスパートシステム（略称：自動運転 ES）の概要

①自動車は，**図 9.2** に示す地域内の指定されたルートを自動運転により移動する。

②知識は，**図 9.3** に示すような以下の 3 種類のプロダクションルールを用いる。

- ●交通規則に関するルール群
- ●運転操作に関するルール群
- ●ルート（経路）に関するルール群

③簡単のため，次のような前提を設ける。

- ・すべての交差点に信号がある。
- ・歩行者はいない。
- ・交差点に同時に到着する自動車は一方向からは 1 台以下とする。

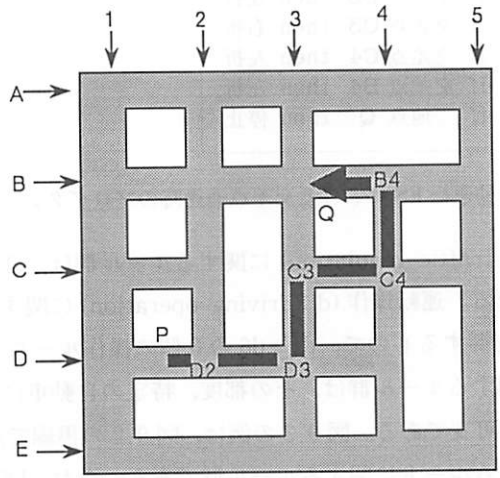


図 9.2　自動運転 ES のルート例

[交通規則に関するルール群]

(t1) if 赤信号 then 停止
(t2) if 黄信号 then 停止
(t3) if 青信号 then 進行
(t4) if 左折 then 左方向指示器点灯
(t5) if 右折 then 右方向指示器点灯
(t6) if 進行 and 直進 then 直進可
(t7) if 進行 and 左折 then 左折可
(t8) if 進行 and 右折 and 直進対向車 then 停止
(t9) if 進行 and 右折 and 左折対向車 then 停止
(t10) if 進行 and 右折 and 右折対向車 then 右折可
(t11) if 進行 and 右折 and 対向車なし then 右折可

[運転操作に関するルール群]

(d1) if 左方向指示器点灯 then 左方向指示器点灯操作
(d2) if 右方向指示器点灯 then 右方向指示器点灯操作
(d3) if 直進可 then アクセル操作
(d4) if 左折可 then アクセル操作 and ハンドル左回転操作
(d5) if 右折可 then アクセル操作 and ハンドル右回転操作
(d6) if 停止 then ブレーキ操作

[ルート(P→Q)に関するルール群]

(r1) if 交差点 D2 then 直進
(r2) if 交差点 D3 then 左折
(r3) if 交差点 C3 then 右折
(r4) if 交差点 C4 then 左折
(r5) if 交差点 B4 then 左折
(r6) if 地点 Q then 停止

図 9.3 自動運転 ES における交差点通過時のプロダクションルールの例

交通規則 (t：traffic regulation) に関するルール群は，t1～t11 の 11 個の判断ルールからなる。運転操作 (d：driving operation) に関するルール群は，自動車の運転を制御するもので，d1～d6 の 6 個の操作ルールからなる。ルート (r：route) に関するルール群は，その都度，特定の自動車に指示を与えるもので，ルール数は可変である。図 9.3 の例は，図 9.2 の黒線で示した P 地点から Q 地点へ移動するルートに関するルール群である。なお，図 9.2 の地図上の交差点の識別のために，左右の位置を表す 1 から 5 の記号と上下の位置を表す A

からEの記号を用いて交差点を「C2」のように表現する。

9.2.3 推論方式

このような知識を用いたプロダクションシステムの推論方式（プログラム実行方式）について説明する。まず，次のような状況例を考える。

状況1

[初期条件]

●自動車の位置：交差点D2

●信号機の状態：青

●対向車の有無：なし

[推論過程の概略]

交差点D2が近づいたが，信号が青であり，かつ，この交差点は直進すればよいため，そのままアクセルを踏み続けた。

このような推論を行うためのプロダクションシステムの基本構造は，**図9.4**のようなものである。図9.3のルールの集合はプロダクションメモリに保持しておく。「青信号」のような外界の状態や「直進」のような推論の中間結果は，ワーキングメモリに書き込んでおく。

推論エンジンは，ワーキングメモリの状態を観察し，プロダクションメモリの中に条件部を満足するルールがあれば，それを実行するとともに，その実行結果をワーキングメモリに反映するという一連の動作を繰り返す。この推論過程は，**図9.5**に示すようなもので，認知－行動サイクルと呼ばれる人間の認知モデルに対応している。すなわち，人間の思考過程を

認知→行動→認知→…→行動→認知→行動

ととらえ，図9.5における照合と実行を繰り返すものである。照合のあとの競合解消は，実行可能なルールが複数あった場合にどれを優先して実行するかを決めるものである。

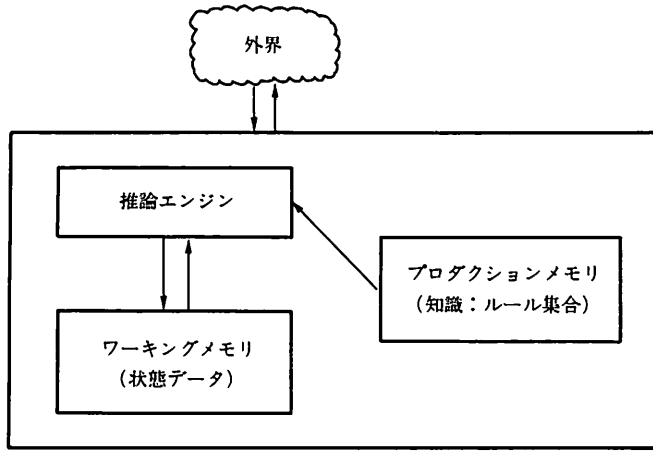


図9.4 プロダクションシステムの基本構造

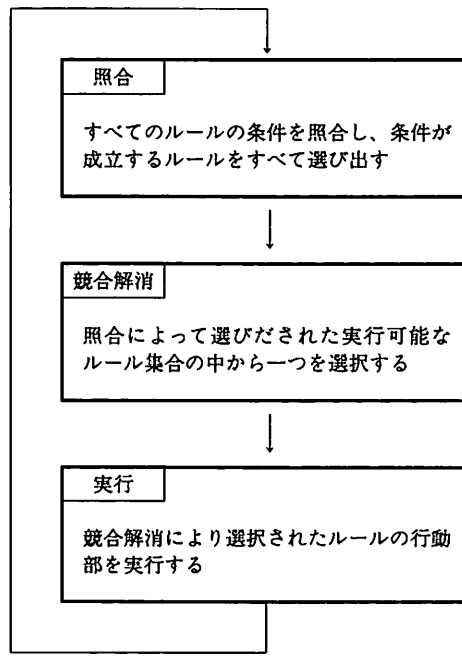


図9.5 プロダクションシステムの推論実行方式

先の状況1の場合の実行過程を示したのが**表9.1**である。まず，初期条件の「交差点D2」と「青信号」と「対向車なし」がワーキングメモリに書き込まれ，図9.5の推論実行が開始される。最初の実行サイクルでは，照合過程で実行可能ルールとしてt3とr1が選び出され，次の競合解消過程でそのうちのt3が選ばれ，最後の実行過程でそのルールの行動部の「進行」がワーキングメモリに書き加えられる。以下同様にして，表9.1に示すように，5回目の実行サイクルで実行可能ルールがなくなるまで推論が繰り返される。なお，この時，実行されたルールは実行可能ルールから削除されるものとする。

表9.1　プロダクションシステムの実行過程(自動運転ESの状況1の場合)

サイクル	ワーキングメモリの内容	実行可能ルール	実行ルール	出力
1	交差点D2 青信号 対向車なし	t3 r1	t3	
2	交差点D2 青信号 対向車なし 進行	r1	r1	
3	交差点D2 青信号 対向車なし 進行 直進	t6	t6	
4	交差点D2 青信号 対向車なし 進行 直進 直進可	d3	d3	アクセル操作
5	交差点D2 青信号 対向車なし 進行 直進 直進可	なし		

9.2.4　競合解消戦略

今の例では，1番目の実行サイクルにおいて，実行可能ルールが複数個あり，ルール番号の小さい方を優先して実行した。このように，実行可能な複数のルールの中から1つを選択することを競合解消(conflict resolution)といい，その方法を競合解消戦略という。これには次のようなものがある。

①記述順優先：ルール記述順の早いものを優先
②最新優先　：より最近にワーキングメモリに書き込まれたデータを照合対象にする条件部を持つルールを優先
③詳細優先　：条件部が複雑なルールを優先
④重要度優先：各ルールに付けられた重要度の重みの大きいルールを優先

この他にもいろいろ考えられるが，適切な競合解消戦略を設定することは，一般には容易ではない。

その例として，自動運転ESの次のような例を考えてみよう。

状況2

[初期条件]

- ●自動車の位置：交差点C3
- ●信号機の状態：赤
- ●対向車の有無：直進対向車

[推論過程の概略]

交差点C3が近づいたが，ここは右折しなければいけないので，右方向指示器を点灯した。信号が赤なので，停止し，青信号になるのを待った。信号が青になったが，直進対向車があるので，引き続き停止し，対向車の通過を待って，右折した。

この場合の実行過程を**表9.2**に示す。競合解消戦略は状況1（表9.1）の場合と同様に記述順優先を採用した。なお，説明を簡単にするために外界の状況認識に関しては以下の前提を設けたが，これらについてもルール化は可能である。

①進行中の自動車の位置は，自動的に認識され，ワーキングメモリに書き込まれる。

②交差点での信号の状態および対向車の有無は，自動的に認識され，ワーキングメモリに書き込まれる。

③信号待ちの状態で，信号が赤から青に変わったとき，ワーキングメモリの「赤信号」または「黄信号」と「停止」が削除され，かわりに「青信号」が書き込まれる。

④対向車待ちの状態で，対向車が通過した時，ワーキングメモリの「直進対向車」または「左折対向車」および「停止」が削除され，かわりに「対向車なし」が書き込まれる。

表9.2　プロダクションシステムの実行過程（自動運転ESの状況2の場合）

サイクル	ワーキングメモリの内容	実行可能ルール	実行ルール	出力
1	交差点C3　赤信号　直進対向車	t1 r3	t1	
2	交差点C3　赤信号　直進対向車　停止	d6 r3	d6	ブレーキ操作
3	交差点C3　赤信号　直進対向車　停止	r3	r3	
4	交差点C3　赤信号　直進対向車　停止 右折	t5	t5	
5	交差点C3　赤信号　直進対向車　停止 右折　右方向指示器点灯	d2	d2	右方向指示器点灯操作
6	交差点C3　赤信号　直進対向車　停止 右折　右方向指示器点灯	なし		
（青信号待ち：赤→青）				
7	交差点C3　直進対向車　右折 右方向指示器点灯　青信号	t3	t3	
8	交差点C3　直進対向車　右折 右方向指示器点灯　青信号　進行	t8	t8	
9	交差点C3　直進対向車　右折 右方向指示器点灯　青信号　進行　停止	なし		
（直進対向車通過待ち）				
10	交差点C3　右折　右方向指示器点灯 青信号　進行　対向車なし	t11	t11	
11	交差点C3　右折　右方向指示器点灯 青信号　進行　対向車なし　右折可	d5	d5	アクセル操作 ハンドル右回転操作
12	交差点C3　右折　右方向指示器点灯 青信号　進行　対向車なし　右折可	なし		

(1)　記述順優先の問題

表9.2の実行過程では，競合解消戦略として記述順優先を用いたが，交差点で停止してから右方向指示器を点灯するという不都合が生じている。これは，実行サイクルの1番目と2番目において，交通規則に関するルール群や運転操作に関するルール群がルートに関するルール群よりも先に記述されているためである。この不都合に関しては，ルートに関するルール群を最初に記述するように順序を変更すれば解決する。この場合，ルールの実行順序はr3，t1，t5，d2，d6の順になる。

(2) 最新優先の問題

この例において，最新優先戦略を用いた場合を考えてみよう。実行可能ルールが複数あった場合，より最近にワーキングメモリに書き込まれたデータを照合対象にする条件部を持つルールを優先するので，1番目の実行サイクルでは，ワーキングメモリに書き込まれた順番が「交差点 C3」，「赤信号」，「直進対向車」の順だったとすると，「交差点 C3」を条件部に持つ r3 よりも「赤信号」を条件部に持つ t1 の方が優先して実行される。以下同様の方法でルールの実行順は，t1，d6，r3，t5，d2 となり，やはり，交差点で停止してから右方向指示器を点灯するという不都合が生じている。

これを避けるためには，交差点において，まず信号機の状態認識をした後，交差点の位置認識をするか，あるいは，「交差点 C3」がワーキングメモリに書き込まれた段階ですぐに実行を開始するようにシステムの制御を行う必要がある。その場合はルールの実行順は，r3，t5，d2，t1，d6 となる。

(3) 詳細優先の問題

次に，条件部が複雑なルールを優先する詳細優先戦略をとった場合を考えてみよう。交差点 C3 における信号機の状態が「青信号」で，対向車の有無が「対向車なし」の時，t5 と t11 が競合する場合があり，詳細優先戦略では条件部が複雑な t11 の方が優先される。条件の複雑さが同じときは最新優先を採用することにしておくと，右方向指示機を点灯しないまま，右折してしまうという不都合が生じる。

(4) 重要度優先の問題

このような種々の不都合を避けるためには，重要度優先戦略を用いてルールの実行順序を明示的に規定しておく方法がある。たとえば，ルートに関するルール群を重要度最大にし，次に交通規則に関するルール群とし，運転操作に関するルール群を最小の重要度としておく。そして，運転操作に関するルール群の中では方向指示機操作のルールの重要度を高くしておけば，いままで述べた問題は生じない。しかしながら，ルール実行順を明示的に指定する方法は，ユーザの負担が大きく，手続き記述と同じことになるため，簡易プログラミングとしてのプロダクションシステムの利点が失われる。

9.2.5 実用上の課題

簡易プログラミングの観点で，プロダクションシステムを見た場合の実用上の課題について述べておく。

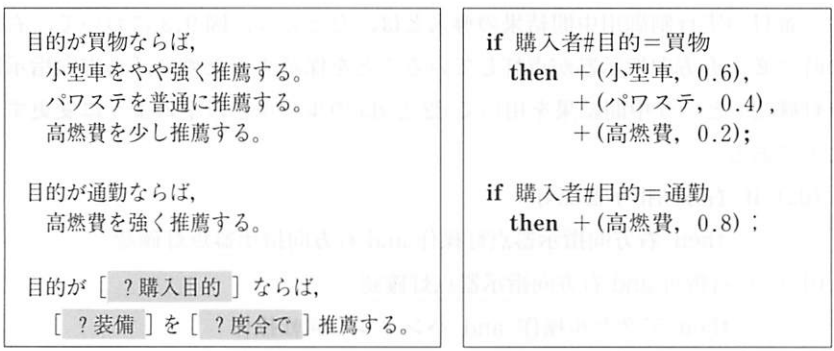
(1) 言語仕様の複雑さ

プロダクションルールの場合は記述形式が「もし～ならば，～せよ」という1つのパターンに限定されていて，従来の手続き型言語に比べれば言語仕様は簡潔であるが，職業的プログラマでない業務専門家が業務知識の記述，追加，修正を行うには，まだまだ複雑である。

この問題は，プロダクションシステムの適用業務を特定したとき，知識表現も限定されることから，特定業務向け簡易知識エディタを導入することで解決する。**図9.6**に報告者らが開発したエキスパートシステム構築ツールES/X90の例を示す。図の(a)は，ES/X90の知識テーラ(専用知識エディタ作成システム)を用いて作成した自動車購入相談エキスパートシステム用の簡易知識エディタである。自動車販売担当者は，

目的が[？購入目的]ならば，
　[？装備]を[？度合で]推薦する。

という文型の[？]の部分にあてはめる用語を選択して，販売ノウハウを入力す



(a) 自動車購入相談エキスパートシステム用簡易知識エディタ　(b) 内部表現（ルール形式）

図9.6 エキスパートシステム構築ツールES/X90における日本語知識表現例

れば，自動的に図(b)のプロダクションルールが生成される。このような方法により，応用分野の専門家自身が知識を入力，管理できる。

(2) プログラム実行制御の複雑さ

簡易プログラミングとしてのプロダクションシステムの利点の1つは，手続き型言語と異なり，ルールの記述順が自由であることである。しかしながら，実際には，先の例でも述べたようにプログラムの実行制御がどのように行われるかをユーザが十分理解して，ルールの記述に工夫をする必要がある。その方法として以下のようなものがある。

①適切な競合解消戦略を選択する。
②ルール群単位の実行順序の制御のためのメタルール（メタ知識）を活用する。
③ルールの実行順序制御のための推論中間結果を導入し，先に実行したいルールの実行部とその後に実行したいルールの条件部に書き加える。

最初の競合解消戦略については9.2.4項ですでに述べた。メタルールとは，たとえば図9.3の例において，3種類のルール群の実行順序を次のように指定するものである。

[ルート（経路）ルール群]→[交通規則ルール群]→[運転操作ルール群]

3番目の実行制御用中間結果の導入とは，たとえば，図9.3において，右折の時に必ず右方向指示器が点灯していることを保証するために，「右方向指示器点灯確認」という中間結果を用いてd2とd5のルールを以下のように変更することである。

(d2) if 右方向指示器点灯
　　then 右方向指示器点灯操作 and 右方向指示器点灯確認
(d5) if 右折可 and 右方向指示器点灯確認
　　then アクセル操作 and ハンドル右回転操作

(3) あいまい知識の扱い

ここでは詳しく述べなかったが，知識表現では，真偽が明確でない曖昧な知

識を表現する機能が必須である。このために，プロダクションシステムでは，ルールに確信度を付加できるようにしている。たとえば，図9.6(b)におけるルール実行部の数値（0～1.0）は確信度を意味し，自動車購入者の購入目的が買物ならば小型車を0.6くらいの確信度で薦めることを示している。

しかしながら，実際には，この確信度を数値で正確に決定するような，曖昧さの数量化が難しい。この問題は，ルールの条件部や結論部にファジー表現を用いることで緩和できる。この場合でも，図9.6の例で「やや強く推薦する」というファジー表現の「やや強く」に対応するメンバーシップ関数の設定が難しい。

さらにニューラルネットワークの学習機能を用いれば，ルールの確信度やファジーのメンバーシップ関数の設定が不要になる。ただし，この場合はうまく学習させることがプログラミングに代わる課題となる。" Learning is programming" といわれる所以である。

このように人工知能技術を簡易プログラミング技術として用いるためには，手続き的記述が不要となる代償として，知識の調整が必要であり，このような調整容易性（adjustability）の向上が必須である。

9.3 フレーム表現

★ *階層化された知識に基づいて推論するプログラムは，わかりやすい。*

9.3.1 フレームの特徴

フレームの概念は，1974年にMITのM. Minskyが人間の記憶構造や推論過程の説明のために提案したものである。フレームは次のような特徴を持ち，述語論理やルールによる知識表現では不十分であった知識の構造化や知識間の関係の記述が容易に行える。

①実世界の概念に対応する枠組みをフレームとして自然な形で表現できる。その概念が持っている種々の属性はフレーム内にスロットとして記述で

きる。

②フレームの階層化と上位フレームから下位フレームへの属性の継承を可能とすることにより，知識を構造化できる。

③この属性には手続きを付加することもできる。

これらの特徴について，例を用いて説明する。

(1) フレームの基本構造

人間の思考過程では，過去の記憶が大まかな枠組みごとに想起されている，という観点から，この枠組みをフレームに対応付ける。たとえば，「会議」という言葉から，

「会議室に何人かの人たちが集まり，テーブルを囲んで議論をしている。中央に議長がいて，黒板には議題が書かれている。」

という情景が浮かんでくる。そこで，会議のフレームを図9.7のように表現することができる。これは，会議というフレームが「会議名」，「場所」，「出席者」，「議題」，「議長」などの属性を持っていることを示している。

このようなフレームの基本構造を図9.8に示す。属性を保持する場所をスロットと呼ぶ。

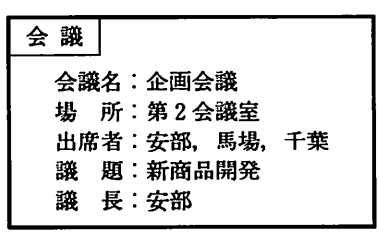


図9.7 「会議」のフレーム

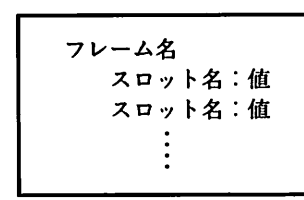


図9.8 フレームの基本構造

(2) 階層構造と属性の継承

フレーム表現では，より抽象的な概念を表現するフレームとより具体的な概念を表現するフレームの間で，図9.9に示すような階層構造を作ることができ

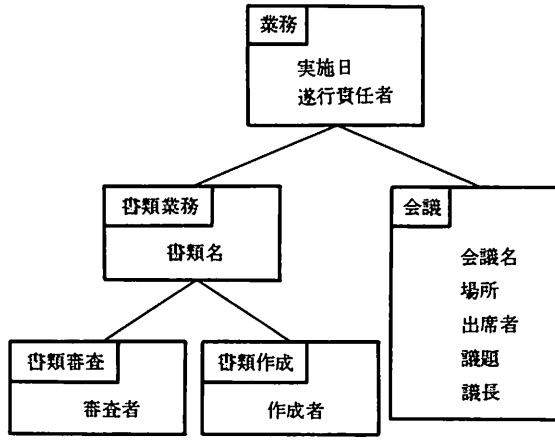


図9.9　フレームの階層構造

る。下位フレームは上位フレームをより具体化したものである。このような上下間の関係をis-aリンクまたはa-kind-ofリンクと呼ぶ。これは，図のような上下関係が「会議is a業務」や「書類審査is a kind of書類業務」を意味することに由来する。

したがって，このような階層関係は属性継承機能を伴っている。すなわち，下位フレームは上位フレームの属性を継承することができる。図の例では，会議フレームは自分のフレームに固有の5種類の属性のほかに，上位の業務フレームから継承した「実施日」と「遂行責任者」という属性も持っていることになる。このような階層構造と属性継承に関する基本的な性質は第8章で述べたオブジェクト指向概念とほぼ同じである。

9.3.2　フレームのプログラミング機能

(1)　属性の定義機能

フレーム内の各々の属性スロットには，次のようないろいろな性質を表現するためのファセットを記述できる。

①値ファセット：スロットの値
②タイプファセット：スロットのデータ型

③レンジファセット：スロットの値の範囲
④デフォルトファセット：スロットの値が未定の時に用いるデフォルト値
⑤デモンファセット：スロットがアクセスされた時に呼びだされる手続き

これらのファセットを利用した会議フレームの例を図9.10に示す。たとえば，出席者数スロットのタイプファセットとレンジファセットは，出席者数の値が整数値であり，その値の範囲が1から8の間であることを示している。議長スロットのデフォルトファセットは，議長が未定のときは会議フレームの上位フレームである業務フレームから継承した遂行責任者スロットの値を議長スロットの値とすることを示している。なお，通常，すべてのスロットは値を持ちうるため，特に値ファセットを明示するためのキーワードは用意していない。

```
フレーム名：会議
    親フレーム名：業務
    会　議　名：type(string)
    時　　間：type(list of integer)
    出　席　者：type(list of string), if-removed (出席者数変更)
    欠　席　者：type(list of string), if-added (出席者リスト削除)
    出 席 者 数：type(integer), range(1..8), if-needed (出席者
                リストカウント)
    議　　題：type(string)
    議　　長：type(string), default (業務.遂行責任者)
```

図9.10 会議フレームの定義例

デモンファセットで指定される手続きは，それが指定されているスロットがアクセスされた時に呼びだされる。一般に手続き型言語では，プログラム内の手続きを呼び出したいところに明示的に手続き呼び出し文を記述しておくが，デモンファセットの手続きは，このような記述はしない。必要に応じて自動的に呼びだされるためにデモンと呼ばれる。デモンの代表的な例を以下に示す。

①if-neededファセット：スロット値が必要とされたときにその手続きが呼び出され，スロット値が設定される。

② if-added ファセット：スロットに値が書き込まれたときにその手続きが呼び出される。

③ if-removed ファセット：スロットの値が削除されたときにその手続きが呼び出される。

具体的な利用方法を図 9.10 の例で説明する。たとえば，出席者数を知る必要が生じたときに出席者スロットにアクセスすると，if-needed ファセットで指定された出席者リストカウントの手続きが呼び出され，出席者スロットに書き込まれている人の数を数えて出席者数スロットの値とする。次に，欠席者スロットに欠席者名を追加したときには，if-added ファセットで指定された出席者リスト削除の手続きが呼び出され，出席者スロットから欠席者名が削除される。この時，出席者スロットの if-removed ファセットで指定された出席者数変更の手続きが呼び出され，先ほど設定した出席者数スロットの値を 1 だけ少なくする。

(2)　推論方式

フレームで表現された知識を利用した推論方式は，すでに述べた機能のうち，以下のようなものが用いられる。いずれもプログラミング技法と見ることができる。

①属性の継承：会議フレームには会議の開催日の属性が定義されていないが，上位フレームの業務フレームから継承してきた実施日スロットを会議開催日とする。

②デフォルトファセット：議長が未定のときに業務フレームの遂行責任者を議長とする。

③デモンファセット：出席者数が未知のときは出席者リストを数える。欠席者リストに追加した人は出席者リストから削除する。出席者リストを変更したら出席者数も変更する。