1Q-9

JSI AIワークステーション(9) —知識表現言語SPOOLとそのプログラミング環境—

横井 伸司　広瀬 紳一　福永 光一
日本アイ・ビー・エム(株) サイエンス・インスティチュート

1. 知識表現言語の必要性

AIワークステーションの機能としては、その上で知識ベース・システムが容易に作成できることが必要であり、そのためには、知識表現言語が整備されている必要がある。知識表現言語においては、対象分野において必要とされる知識を十分に記述する能力と、記述された知識に対して柔軟で効率的に操作する能力が重要である。

さらに、知識ベース・システム作成には、通常ラピッド・プロトタイピングや部分的な開発実行、設計変更などが伴うため、これらを容易に行えることが必要であり、そのためには、知識ベース管理機能や、開発中のシステムをテストするための実行環境、デバッグ機能などを備えたプログラミング環境が必要となる。

2. 知識表現言語SPOOL

知識ベース・システムで用いられる知識は、多くの場合、数値的な知識というよりは、記号表現された知識である。このように、記号表現された知識を管理、操作する機能を提供する知識表現言語は、それ自身記号処理機能を備えている必要がある。

本ワークステーションの核言語として採用したPrologは、強力な記号処理機能を備えた言語であり、ユニフィケーションやバックトラックなどの特徴は、実際の知識の操作を考えた場合、非常に有益であると思われる。

このように考えると、Prologはそれ自体知識表現言語として要求される多くの機能をあらかじめ備えたプログラミング言語となっているので、知識表現言語の作成に際してはベースとしてPrologを採用することにした。

しかし、Prologは知識表現言語としてみた場合、知識ベースの構造が平坦で、概念としてのまとまりを表現する能力に劣るという問題がある。これを解決するために、我々は知識表現モデルとしてオブジェクト指向言語を取り入れた。オブジェクト指向言語は、知識の概念構造を記述するための表現方法として、内部状態と手続き(メソッド)を備えたオブジェクトと呼ばれる概念要素を導入し、概念間の階層関係とそれに基づく属性の遺伝や、抽象データ型などの考えを実現することによってモジュール化された知識表現方法を提供し、上記の問題を解決している。

SPOOL[1]では、メソッドはPrologのクローズの形で表現され、オブジェクトへのメッセージ送信は、Prologのゴールを起動することに対応している。このようにオブジェクト指向言語とPrologとを融合した考え方には、以下のような利点がある。

(1) 推論機能を持つオブジェクトを作成できること
Prologでは、問題解決のプログラムを作成する場合、事実の記述のみを行えば、汎用の推論メカニズムが推論を行って問題が解決される。従って、SPOOLでは、推論能力を持ったオブジェクトの記述が容易に行える。

(2) メソッドの引数を入出力の区別なく利用できること
Prologでは、引数に入出力の区別がなく、実行時に状況に応じて、引数を入力、出力どちらにも利用できる。この特徴を利用すれば宣言的なメソッドの記述が可能になり、宣言的知識が記述できる。

(3) 柔軟なメッセージ送信が行えること
あらかじめ受け取りオブジェクトを指定せずにメッセージ送信を行うことや、いわゆる放送機能を実現することができる。

3. SPOOLのプログラミング環境

3.1 必要性と特徴

知識ベース・システムの構築は、通常プロトタイプの作成とその検証という試行錯誤的な過程を繰り返すことによって行われる。この過程を効率良く行うためには、対話型のプログラミング環境が必要となる。また、AIでは、複雑な推論や知識表現を行うが、これらのグラフィクスによる視覚化は、文字による視覚化と比較して、より簡潔でより多くの情報を提供することができ、ユーザの理解を助ける。また、より良いヒューマン・インターフェース実現への基礎となるものでもある。このような考えのもとに、我々はプログラミング環境INK[2]を作成した。INKの特徴としては、

(1) 対話型の環境であること
(2) グラフィクス機能の提供
(3) マルチ・ウインドウ機能の実現
(4) 統合化されたツール群

などが挙げられる。

JSI AI Workstation(9) - Knowledge Representation Language SPOOL and its Programming Environment
Shinji Yokoi, Shin-ichi Hirose, Koichi Fukunaga
Science Institute, IBM Japan, Ltd.

３．２　ツール

ＳＰＯＯＬの特徴として、(ｱ)クラス間の継承関係、(ｲ)内部状態を持つオブジェクト、(ｳ)メッセージ送信による実行、が挙げられる。プログラミング環境としては、これらの特徴に対応して次のようなツールが用意されている。

(1) クラス階層構造ブラウザ

知識ベースが大きくなると必要なクラスの定義を多数のファイルの中から検索したり、クラスの階層構造を全体的に把握することが困難になってくる。これらの不便を解消するために、クラス階層構造ブラウザを作成した。このブラウザでは、指定されたクラスの下位クラス全体（一般に、ラティス構造）を表示する。これにより、クラスの全体構造を把握しやすくしている。

(2) インスペクター

ＳＰＯＯＬのオブジェクトは状態を持っており、それはインスタンス変数の値によって表現される。実行時にインスタンス変数の値を容易に調べられることはシステムの実行過程を理解する上で有益である。このために、ツールとして、インスペクターを用意した。インスペクターに対して、オブジェクトの名前を指定するとそのオブジェクトの属するクラス名とそのオブジェクトが参照できるインスタンス変数、及びクラス変数の名前と値の組を表示する。ここで、ユーザは、必要に応じて変数の値を変更することができる。

(3) トレーサー

プログラムの実行は、オブジェクトからオブジェクトへのメッセージ送信によって行われる。従って、オブジェクト間のメッセージの流れをトレースすることは実行制御の流れを理解するのに役立つ。そのために、ツールとしてトレーサーを用意した。しかし、すべてのメッセージ・トレースを表示するのでは、そのなかから本当に得たい情報を検索するのに不便である。そこで、ユーザの要求に合うメッセージのみを表示させるため、表示するメッセージに、例えば、特定の［オブジェクト／クラス］［の集合］［へ／から］のメッセージ、などの条件を付けられるようにしている。

３．３　統合的環境

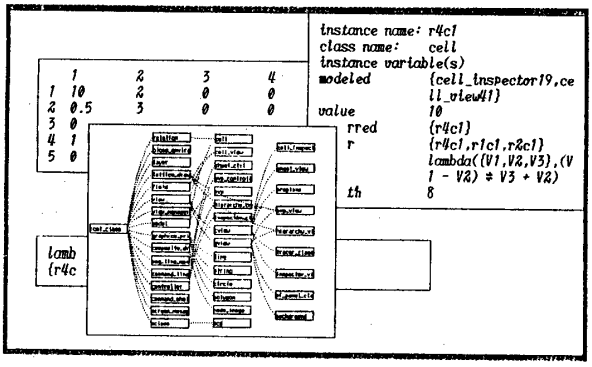
さまざまなツールが用意されており、それぞれの機能が充実していることは大切であるが、プログラミング環境としてさらに重要なことは、これらのツールが互いに独立したものでなく統合されていることである。

ＩＮＫにおいては、それぞれのツールが単体として使用できるだけでなく、実行時にすべてのツールを同時に使用することができる一体型のプログラミング環境を提供しており、しかも、それらのツールがお互いに関連しあった統合的なプログラミング環境を構成している。

例えば、インスペクターのなかである変数の値を変更すると、その変更はすぐに反映され、画面上の表示で影響を受けるものがあれば、正しく再表示される。また、クラス階層構造ブラウザで全体的な階層構造を表示しているときに、クラスの定義が見たければそのクラスの表示のところにカーソルを移動してエディターを呼びだせば自動的にそのクラスの含まれるファイルを選んでクラス定義の表示ができ、その場で編集も行える。

ツールはいくつかの基本的な機能を持った部品から構築されており、両者は共にＳＰＯＯＬ自身で記述されている。従って、ユーザは必要な部品を利用してさらに高度なツールを構築することができる。さらに、このツールはシステムの用意したツールと同様にＩＮＫから使用することができる。この意味で、プログラミング環境は開かれたシステムであるということができる。

次に、実際の使用例を示す。この例では、スプレッド・シート、インスペクター、クラス階層構造ブラウザが表示されている。



４．　おわりに

以上述べてきた知識表現言語ＳＰＯＯＬ及びそのプログラミング環境ＩＮＫは、すべて Prolog により実現されている。従来、Prolog はシステム・プログラミングや大規模なソフトウェアには不向きであると考えられていたが、我々は Prolog をベースにしたワークステーション上で実用に耐え得る知識表現言語やプログラミング環境を実現できることを示した。

参考文献

[1] Fukunaga, K.,Hirose, S.:An Experience with a Prolog-Based Object-Oriented Language, Proc. of the ACM Conf. on OOPSLA, Oct. 1986

[2] 横井、対話型システム作成用の道具：ＩＮＫ、情報処理学会第３２回全国大会、6F-9、1986