1Q-1

ＪＳＩ　ＡＩワークステーション（１）
―　設計方針と概要　―

藤崎哲之助、戸沢義夫、黒川利明、福永光一
日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

ＩＢＭサイエンス・インスティチュートの知識ベースシステムグループでは知識ベースシステム構築用のワークステーションを、IBM 6150 (RT-PC) [1] ワークステーション上に構築している。これは近年の専門化システムや、知識ベースシステム構築のブームが、新しい種類の、従来のものより、フレキシブルで、ヒューマン・インターフェースのよい構造のソフトウェアあるいはソフトウェア構築技法への漠然とした期待であると考え、その期待に答えるソフトウェア構築のための環境としてのワークステーションを実現すること、また、この漠然とした期待の実体をワークステーションのプロトタイプ開発を通じて具象化し、ハードウェア、ソフトウェアに求められる機能を明らかにすること。また我々自身の研究活動にも使用できるプロトタイプとしてのワークステーションを実現することを目指している。

２．ワークステーション内部階層

このような知識ベースシステム構築用ワークステーションの内部階層として図１に示すような４つの層を設定した。

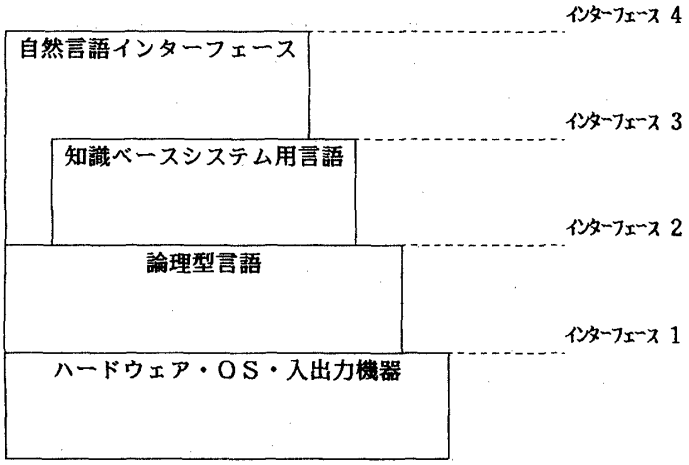

図１　ワークステーション内部階層

これらの４つの層、（１）ハードウェア・ＯＳ・入出力機器、（２）論理型言語、（２）知識ベースシステム用言語、（２）自然言語のインターフェース層から４つの外界とのインーフェースが設定される。第一のインターフェースはＯＳとの接点であるが、開放型で、２の層上に他のソフトウェア・システム（例えばＬＩＳＰ系システム、従来型手続き型言語によるもの、既存のデータ・ベース・システム等々）との共存、さらにはそれらとの間の通信が許されねばならない。ＲＴ－ＰＣおよびそれの上のオペレーティング・システムがここでのインターフェースを提供している。

第２のインターフェースはシステムプログラマーあるいは人工知能研究者のためのインターフェースである。このインターフェースを通じて、ユーザーはこのワークステーションの核言語である論理型言語[2,3]を用いることができる。

第３のインターフェースは知識ベースアプリケーション構築のためのインターフェースで、このインターフェースを通じて推論機能を持つ、ヒューマンインターフェースのよい、フレキシブルなアプリケーションの構築あるいはプロトタイピングが容易に行なえると考えている。論理型とオブジェクト指向型の融合した言語ＳＰＯＯＬ[4,5]、それ上のグラフィックインターフェース構築用キットＩＮＫ[6]、データ駆動型推論用ＰＯＰＳ２[7]、ＳＰＯＯＬのプログラム環境などがこのインターフェースを提供している。

第４のインターフェースは下層の機能を組合せて構築された知識ベースアプリケーションに対して自然言語による入出力インターフェースを与えると同時に、知識ベースアプリケーションの構築自体も自然言語で行なうことを可能とすることを目指すものである。

３．ワークステーション構築要素の選定

本ワークステーション構築に際して各層に何を選定するかは、それらの機能とともに、それらがいかに豊富な研究テーマを内在するかどうかが研究プロジェクトとして本ワークステーション構築を行なう際の重要な要素であった。

３．１　ハードウェア・ＯＳの選択

ハードウェアとしては、IBM 6150システム（ＲＴ－

JSI AI Workstation (1) - Overview and design strategy
Tetsunosuke Fujisaki, Yoshio Tozawa, Toshiaki Kurokawa, Koichi Fukunaga
Science Institute, IBM Japan, Ltd.

ＰＣ）を用いたが、これはIBM 6150がＲＯＭＰと呼ばれるＲＩＳＣ型マイクロプロセッサーを用いたものであり、メモリーアクセスの多い記号処理、特に論理型言語がＲＩＳＣになじむか。このＲＩＳＣ型計算機用にＩＢＭ内で開発されてきた手続き型言語での最適化技術・コンパイラ技術（ＰＬ．８コンパイラ）がやはりＡＩ用の言語に利用可能であるかどうかを見極めることが採用の１つの要素であった。またこれは、記号処理言語による知識ベースアプリケーションの構築・実行を汎用機で十分効率的に行なうことができること。またむしろ汎用機上でこれらのアプリケーションを構築し、同一マシン上での他のソフトウェアシステム、既存のアプリケーション、データ・ベースシステム、あるいは回線を通じてのホストシステム・セッション等とウィンド・システム上で共存するに至るまでの密な交信を許すとともに、そのような環境こそが実用的な知識ベースアプリケーションの構築に不可欠であることを示すことが目的であった。

オペレーティング・システムとしてはＩＢＭワトソン研究所で開発されたＣＰＲを当初採用した。これが前述の様々なソフトウェア間の交信を支援する強力なプロセス間通信機能、また主記憶、ファイルシステムに対する強力な共有を許し、かつ保全性に優れていることによる。またこのＣＰＲは単一レベル記憶を実現しており、この機能を大規模知識ベースシステムにおいていかに利用できるかも大きな興味であった。

３．２　核言語としての論理型言語

論理型言語の一つであるプロローグの使用経験を通じて、この論理型言語が強力であり、そこでの宣言型ルールに基づく知識表現がフレキシブルなプログラム構築に有効であることは明らかであった。ただ逆に、現状のプロローグにおいては言語仕様としても、プログラム環境という面でも、また翻訳実行技術としてもまだ未熟であり、多くの重要な興味深い研究テーマを残していることが選択の大きな要因であった。現在の仕様のプロローグは知識ベース・アプリケーション・プログラムを書くアプリケーションプログラマーには自由度が多過ぎると同時に難し過ぎる。むしろ、システムプログラミング用言語として有効であると考える。ただ、そのような用途に用いるには実行速度、デバッグ環境などの点で大幅な改善が行なわれねばならない。また現在の言語仕様では大規模なプログラムを複数のプログラマーの共同作業として行なうことは困難である。例えば分割コンパイル、名前の局所性等はそれらの１例である。ＬＩＳＰなどと比べた場合、プログラム環境・翻訳技術などに雲泥の差があり、会話型言語としてもまだ実現されていない。このような未完の大器を研究対象として捕え、次に示すような質問に対する答えを見つけだすことは重要な研究テーマであると考えている。

－論理型言語をシステム・プログラミング言語として捕え、その上に多層のソフトウェアを構築しても十分実用となるほどの実行速度を得ることができるか。汎用機を前提としたとき、アーキテクチャの変更、アクセレーターの開発などが必要なのであろうか？

－論理型言語で十分に実用となるほど大きなソフトウェア開発が行なえるか？　分割／追加コンパイル、モジュール化／情報隠匿、会話型環境の実現、デバッガを含むプログラム環境の整備などがどのような形で実現可能なのであろうか？

－大規模アプリケーションのための知識表現、知識操作のため、また、高速実行を実現するためにどのような言語仕様の拡張が有効なのであろうか？　型の導入、オブジェクト指向言語との融合などが有効な手段となり得るのだろうか？　推論メカニズムとしてデータ駆動、黒板型などの導入が可能なのであろうか？

４．終りに

上記に示したような動機に基づき知識ベースアプリケーション構築用ワークステーションの構築を行なっている。

参考文献

［１］ "IBM RT-Personal Computer Technology", IBM マニュアル SA23-1057

［２］ Komatsu, Tamura, Asakawa, Kurokawa, "An Optimizing Prolog Compiler", Proc. of the Logic Programming Conference '86, Tokyo, 1986.

［３］ Kurokawa, Tamura, Asakawa, Komatsu, "A Very Fast Prolog Compiler on Multiple Architectures", Proc. of FJCC '86. (to appear)

［４］ 福永、広瀬："Prolog上のオブジェクト指向言語SPOOLとその実用例"，日本ソフトウェア科学会第２回大会論文集　4B-6, pp. 137-140, 1985.

［５］ k. Fukunaga and S. Hirose, "An Experience with a Prolog-based Object-Oriented Language", Proc. of OOPSLA 86, ACM, 1986 (to appear).

［６］ 横井伸司　"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，pp. 457-458

［７］ 広瀬紳一　"POPS2", Computer Today, Vol.3, No.3, 1986, pp. 63-67