知的情報処理システムに関する調査研究報告書

第２分冊　知識システム構築ツールの評価

平成元年３月

JIPDEC

財団法人　日本情報処理開発協会
ICOT-JIPDEC AI センター

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和６３年度に実施した「知的情報処理システムの導入・活用に関する調査研究」の成果の一部をまとめたものであります。

まえがき

通商産業省情報処理振興審議会は、１９８６年３月の答申において、１９９０年代を目標とする第４期「電子計算機利用高度化計画」を取りまとめましたが、この中で、従来のコンピュータ利用技術の延長線上では捉えきれない新しい概念として「知識情報処理」を取り上げています。また、情報処理振興審議会の答申から遡ること４年、１９８２年４月に通商産業省では、世界に先駆けて、知識情報処理機能を組み込んだ全く新しいコンセプトの「第五世代コンピュータ」の開発をスタートさせました。

欧米諸国もこの日本の第五世代コンピュータプロジェクトに刺激を受け、相次いで知識情報処理技術の研究開発をナショナル・プロジェクトとして取り上げています。

一方、産業界でも、各種の知識システムの開発、導入に積極的に取り組み始めていますが、構築方法論が確立されていないことや知識獲得の問題などがあり、知識システムの多くは、未だプロトタイプの段階に留まっています。

このような現状認識に基づき、ＩＣＯＴ－ＪＩＰＤＥＣ　ＡＩセンターでは、昭和６１年度から「知的情報処理システム調査研究委員会」を設置し、学界、産業界で活躍している方々により、３年間にわたって活発な調査研究活動を行い、ここに、全５分冊に及ぶ研究成果を取りまとめました。

本調査の実施にあたっては、委員会の委員の方々はもとより関係省庁ほか関係各方面から御指導・御協力をいただきました。ここに謝意を表する次第であります。

平成元年３月

財団法人　日本情報処理開発協会

会　長　　影山　　衛司

「知的情報処理システムに関する調査研究」概要

わが国における知識システムの研究開発の数は 2,000を越えたとみられ，産業界のあらゆる分野への応用の裾野が広がりつつある。しかし，そのうちフィールドテストを経て，実用化の段階に達したシステムの数は，多めに見積もっても， 500以下とみられ，多くのシステムはプロトタイプの段階またはデモンストレーションの段階に留まっているとみられる。その理由として，いくつかの要因が考えられるが，その中で，特に，

1) 知識システムを構築していくことを支援する方法論が確立されていないこと，

2) 知識システムを試作・実装するためのツールが十分には整備されていないこと，

3) 知識獲得を支援する方法論やツールの研究開発が立ち遅れていること，

4) 知識システム技術者が不足し，育成の指針が確立されていないこと，

5) 開発事例を蓄積・分析し，結果を有効に利用する体制が確立されていないこと，

などが指摘されている。このような現状認識に基づき，ICOT-JIPDEC AIセンターは昭和61年９月に「知的情報処理システムに関する調査研究委員会」を設置し，つぎのようなテーマに関する調査研究活動を行ってきた。

1) 知識システム構築方法論を確立すること，

2) 知識システム構築ツールの評価体系を確立すること，

3) 知識獲得支援の動向調査を行い，今後あるべき支援のイメージを提案すること，

4) 知識システム技術者を育成するためのガイドラインを作成すること，

5) 知識システム開発事例の収集・分析を行い，今後の開発の資とすること，

調査研究活動の前半（昭和61年９月～昭和62年９月）では，知識システム構築方法論および知識システム構築ツールの評価に重点をおき，その研究成果は昭和62年９月の成果発表会で報告され，その詳細は「知識システム開発方法論」および「知識システム開発ツールの評価」と題する報告書にまとめられている。

調査研究活動の後半（昭和62年９月～平成元年３月）では，計画・設計型問題の構築方法論，構築ツールの性能評価，知識獲得支援の動向分析，知識システム技術者の育成方法および知識システム開発事例の分析に重点をおいて活動を展開してきた。

調査研究の成果は，以下の５分冊からなる本報告書にまとめられている。

第１分冊「計画・設計型知識システムの構築方法論」では，計画・設計型知識システムの特徴分析を行い，計画・設計型問題には探索的側面，意思決定的側面，事例利用的側面および知識獲得的側面の４つがあることを指摘，各側面に対して，問題解決的接近，仮説推論的接近，

事例ベース推論的接近および学習的接近の必要性と枠組みをまとめている。

第2分冊「知識システム構築ツールの評価」では，システム構築ツールを評価するための評価体系を整備，ツールの性能評価を行うための標準問題を作成，ＡＩセンターオープンハウスに導入されているツールに対し性能評価のテストを行い，評価結果を取りまとめるとともに，ツールの評価に関するいくつかの提言を行っている。

第3分冊「知識獲得支援」では，知識獲得支援ツールの位置づけ・分類を行い，文献調査とヒアリングに基づき，各ツールの機能と特徴を客観的にとりまとめ，知識システム開発事例における知識獲得の実際を分析，知識獲得支援のツールのあり方および知識獲得支援ツールに対する要求事項をまとめている。

第4分冊「知識システム技術者育成ガイドライン」では，知識システム技術者の役割を，知識システムの開発手順の段階に対応して，システムアナリスト，プロトタイピングアナリスト，知識プログラマ，保守・管理技術者の4つに類別した上で，修得すべき技術を人工知能技術，知識工学技術，システム技術の3つに分け，その概要をまとめている。

第5分冊「知識システムの事例」では，生体や社会などの解釈／診断問題事例，プラントや人工衛星などの制御問題事例，スケジューリング，レイアウト，ＬＳＩ設計などの計画／設計問題事例など全部で２３の代表的な事例を取り上げ，比較可能な同一の枠組みで紹介し，さらに各システムにおける基本タスクと問題解決機能を明らかにしている。

以上を要するに，本報告書は知識システムの研究開発においてその解決が緊急とされる課題を重点的に取り上げ，産業界と学会で活躍している技術者・研究者からなる横断的な組織によって調査研究を行った成果である。本報告書がわが国における知識システム研究開発の一層の発展に寄与する資として役立つことを願っている。

最後に，本調査研究を取りまとめる上で，数十回におよぶ会合を通じて，終始，熱心にご協力頂いた委員の各氏ならびにこれを支えてくれた事務局の皆様に感謝の意を表する。

平成元年3月

知的情報処理システム調査研究委員会　委員長　小林重信

昭和６３年度　知的情報処理システム調査研究委員会

委　員　名　簿

委員長	小　林　重　信	東京工業大学
幹　事	寺　野　隆　雄	(財)電力中央研究所
委　員	関　根　史　麿	花王(株)　知識情報科学研究所
〃	菊　地　一　成	キヤノン(株)　情報システム研究所
〃	山　崎　知　彦	(株)豊田中央研究所
〃	森　　啓	日本電気(株)　Ｃ＆Ｃシステム研究所
〃	千吉良　英　毅	(株)日立製作所　システム開発研究所
〃	中　島　俊　哉	富士通(株)　科学システム部
〃	小　林　慎　一	(株)三菱総合研究所
〃	福　島　正　俊	三菱電機(株)中央研究所
〃	森　谷　正　晴	新日本製鐵(株)　君津製鐵所　設備部
〃	桃　井　茂　晴	日本電信電話(株)　ＮＴＴ情報通信処理研究所
〃	畝　見　達　夫	長岡技術科学大学
オブザーバ	生　駒　憲　治	(財)新世代コンピュータ技術開発機構
〃	渡　辺　　敏	新日本製鐵(株)　君津製鐵所　設備部
事務局	平　井　吉　光	(財)日本情報処理開発協会　ＡＩ振興センター
〃	茂　呂　知　明	〃　　〃

ツール評価ワーキンググループ

主　査	寺　野　隆　雄	(財)電力中央研究所
委　員	関　根　史　麿	花王(株)　知識情報科学研究所
〃	菊　地　一　成	キヤノン(株)　情報システム研究所
〃	森　谷　正　晴	新日本製鐵(株)　君津製鐵所　設備部
〃	桃　井　茂　晴	日本電信電話(株)　ＮＴＴ情報通信処理研究所
〃	畝　見　達　夫	長岡技術科学大学

評価作業協力者

ヒューマンシステム(株)　ＡＩ技術部

事務局	平　井　吉　光	(財)日本情報処理開発協会　ＡＩ振興センター
〃	茂　呂　知　明	〃　〃

知識システム構築ツールの評価・目次

1．序　論

1．序　論

ＩＣＯＴ－ＪＩＰＤＥＣ　ＡＩセンターでは，昭和６１年９月より知的情報処理システム調査研究委員会を設置して，知識システム方法論の確立とシステム開発ツールの評価とを目的とした調査研究を続けてきた．その成果の一部は，[ICOT-JIPDEC87-1]と [ICOT-JIPDEC87-2] の２冊の報告書としてすでに発行されている．本報告は，エキスパートシステム開発ツールを評価するための一般的な枠組みを提案した２番目の報告書の内容を発展させたものである．

昭和６２年９月に[ICOT-JIPDEC87-2] を発表した時点では，ツール評価については，時間的な制約から，評価用の枠組みを提案し，”標準問題”についてひととおりのインプリメンテーションを示すにとどまった．そして，各評価項目に十分な説明をつけること，この項目にしたがって既存のツールの評価を実施すること，また，性能評価用テスト問題のインプリメンテーション作業も行うことが課題として残されていた．そこで，本報告をとりまとめるにあたっては，ツール評価ワーキンググループが中心となって次の様な調査研究活動を実施した（図１－１）．

①各ツールの概念に依存しないように定めたツールの評価項目リストの内容を再検討し，各項目の記述が容易にできるように説明をつける．
②すでに提案した小規模でwell-definedな性能評価用テスト問題の仕様を見直す．
③ＡＩセンター・オープンハウスに導入済みのツールを使って性能評価用テスト問題のすべてをインプリメントし，各ツールならびに性能評価問題の評価を行う．
④ツール評価結果（評価リストとテスト問題のソースコード）をツールベンダーに示して内容の確認を行う．
⑤ＡＩの基本用語の解説，諸ツールで使われる概念のとりまとめを行う．

[ICOT-JIPDEC87-2] ですでに述べたように，我々は，ツールの評価そのものは，ツールの提供者，あるいは，利用者の問題であり，単純に解が一つに定まるものではないと考えている．しかし，本報告で述べるような大規模なツール評価の試みは，これまでにあまり例がなく，今後のツール調査，評価・選択，設計・開発，さらには知識技術者教育などを実施する場合に非常に有用であると考えている．

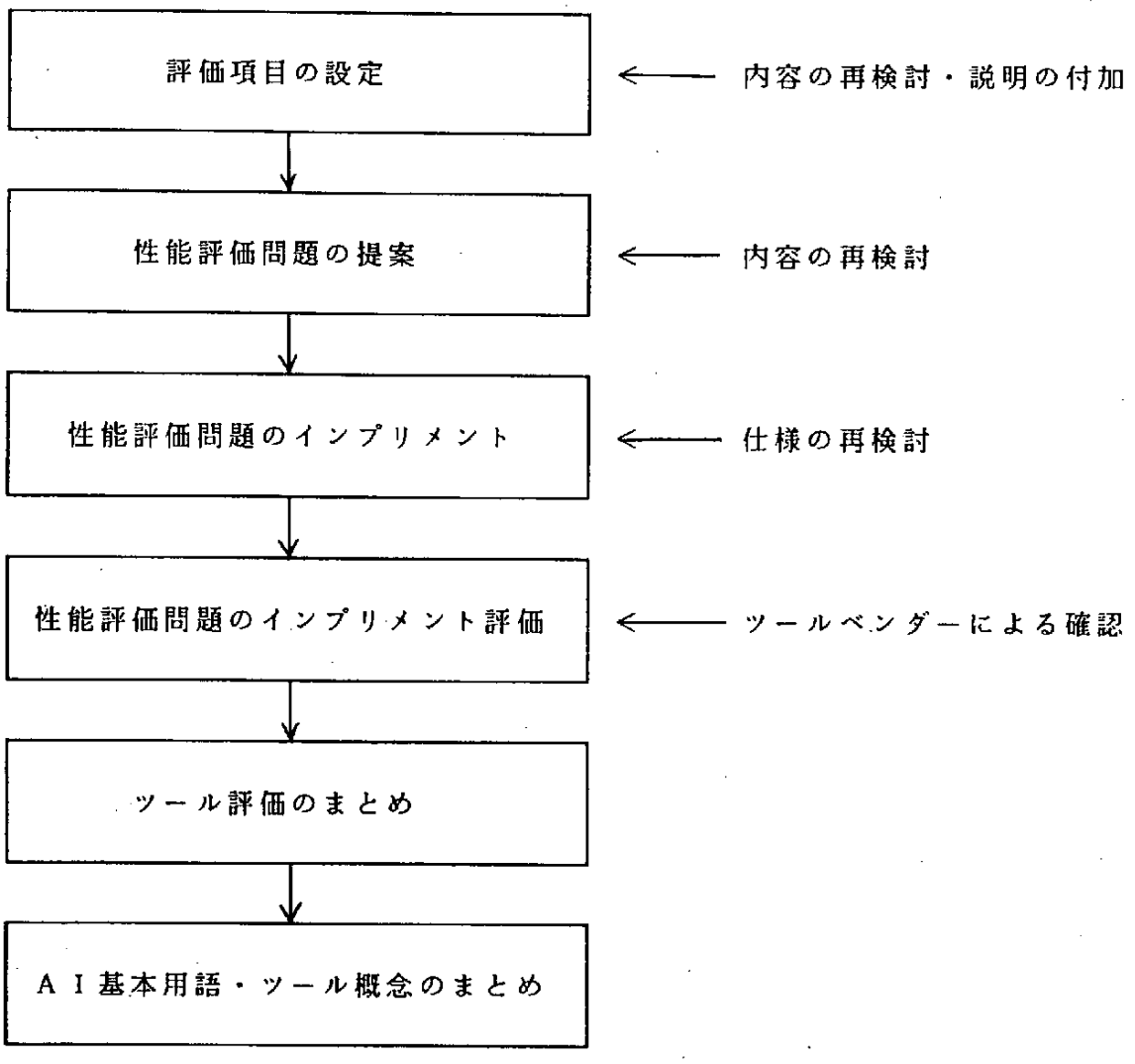


図１－１ ツール評価の方法と本調査研究の概要

本報告書の構成は次のとおりである．２章では知識システム構築ツールの一般的な構造を示し，ツール評価項目の説明の基礎とする．３章は知識システム構築ツールの評価項目とその説明である．これは[ICOT-JIPDEC87-2] で提案した評価項目一覧についてより詳細に解説することを目的として作成したものである．４章では，性能評価用テスト問題の仕様の説明を行う．この内容は[ICOT-JIPDEC87-2] の記述を改善したものである．５章では，ＡＩセンター・オープンハウスに導入済みのツールでインプリメントすることを前提に，テスト問題の基本設計を行い，それに補足的な説明をつけている．６章は性能評価用テスト問題に基づく知識システム構築ツールの比較結果を３章で示した評価項目にしたがって簡単にとりまとめている．７章は，評価に用いた各種のツールで使われる独特の用語を対

照表にまとめている．８章では，３年間におよんだツール評価のための調査研究の結論と今後の課題をとりまとめる．また，付録には，基本的なＡＩ用語の解説をとりまとめる．

参　考　文　献

[ICOT-JIPDEC87-1] ICOT-JIPDEC AIセンター：　"知的情報処理システムに関する調査研究報告書-知識システム開発方法論-". 1987年９月.

[ICOT-JIPDEC 87-2] ICOT-JIPDEC AIセンター：　"知的情報処理システムに関する調査研究報告書-エキスパートシステム開発ツールの比較評価-".1987年９月.

2．知識システム構築ツールの構造

2. 知識システム構築ツールの構造

我々はエキスパートシステム開発ツールの一般的な構成を図２－１のように取りまとめている。ＡＩツールの基本的な構成要素は点線で囲まれた部分である。この中心は、対象分野の知識を格納する知識ベース（Knowledge Base）と、それを利用して推論処理を実行する推論機構（Inference Engine）である。また、推論の実行順序などの制御をおこなう制御機構（Control Mechanism)、推論の途中経過など動的に変化するを保持する作業記憶（ワーキングメモリー；Working Memory)、推論中はあまり変化しない静的な情報を保持するデータベース（Database）などから構成される。本来の人工知能研究の用語では、作業記憶とデータベースは、同じ概念を示し、データベースという言葉で使われることが多い。ここでは、推論中に動的に変化するものを作業記憶、変化しないものをデータベースと呼ぶことにする。

エキスパートシステム開発ツールには、さらに、外部との情報の授受を行うために、システム・インタフェースとユーザインタフェースとをもつ。ユーザインタフェースはさらに知識ベースシステム開発者が主に使う開発者インタフェースと、開発済みのシステムを使うための利用者インタフェースにわけることができる。

システムインタフェースは、他のプログラム言語や他のソフトウェアシステム、他の計算機との結合方式を定める。開発者インタフェースは、開発者、すなわち知識技術者や専門家が、特定の知識ベースシステムを容易に開発しうる環境を提供する。これは、ツールのサポートする人工知能技術（知識表現と推論方式）、ならびに、知識ベースシステムの行うタスクの内容を、開発者が理解しやすいように表現するためのインタフェースである利用者インタフェースは、完成したシステムのエンドユーザが利用するインタフェースであり、エンドユーザにとって真に使いやすいシステムを実現するものである。

以下では、人工知能技術の代表的な概念であるプロダクションルール、フレーム／オブジェクト（Object）、ブラックボードに基づくツールにおいて、それぞれ、図３－２の点線で囲まれた部分の機能がどのように実現されるかを述べる。なお、フレームあるいはオブジェクトはともによく似た概念であり、これらの概念に基づくツールも、たがいによく似た構造をもつ。そこで、以下では、フレームという用語のみを用いて解説する。

ルール型のツールでは、知識ベースにはプロダクションルールが格納される。推論機構は、通常、前向きまたは後ろ向きの推論をサポートする。さらに、推論の高速実行を目的としたルールコンパイラを備えたツールもある。推論途中の状態は、構造を持たないフラ

ットな作業記憶に収められる。推論制御部は、ルールの競合解消のしくみを提供する。データベース機能は純粋のルールベースシステムでは省略される場合が多い。

フレーム型のツール（Frame Based System）では、作業記憶とデータベースの機能をともにフレームの概念で実現する。また、知識そのものもフレームに埋め込まれるので、ルールベースシステムに見られたような明示的な知識ベースをもたない。推論機構は、フレームを操作することで処理を進めていく。そして、推論制御機構は付加手続き（Attachd Procedure)あるいはメソッド（Method）としてフレーム内に記述される。

ブラックボード型のツール（Blackboard Based System)では、知識ベースは、知識源(Knowledge Source)という単位で、複数に分割される。個々の知識源の知識は、通常、プロダクション・ルールの形式で記述される。作業記憶と推論制御機能の役割は黒板がはたす。黒板とは、各知識源が操作することが可能な共通の作業領域で、推論に使う事実や中間的な結論および推論を進めるための制御情報が記述される。そして、推論機構は、黒板の内容を参照し、適切な知識源を起動することで処理を進めていく。データベース機能はもたない。

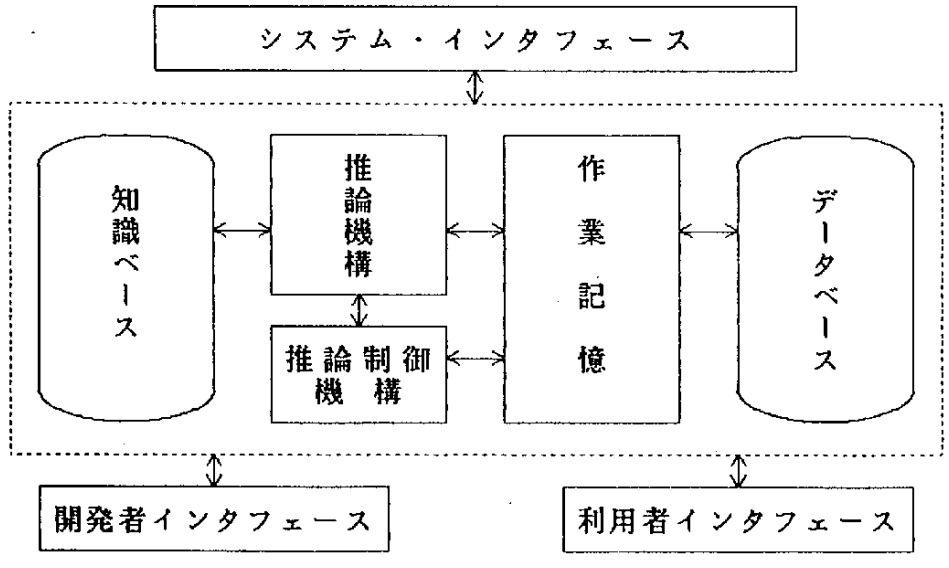


図２－１　ＥＳ開発ツールの基本構成

参考文献

〔Harmon 85〕 Harmon, P., and King, D. : "Expert Systems-Artificial Intelligenc in Business." John-Wiley, 1985.

〔日経 85〕 日経エレクトロニクス："商品化相次ぐエキスパート・システム開発用ツールを比較する." 1985.11.4, pp.153-175.

〔Cilmore 86〕 Gilmore, J. F., Pulaski, K., Howard, C. : A Comprehensive Evaluation of Expert System Tools. Proc. of SPIE, Vol.635 (Applications of Artificial Intelligence III), pp.2-16 (April 1986).

〔Richer 86〕 Richer, M. H. : An Evaluation of Expert System Development Tools. Expert Systems, Vol.3, No.3, pp.166-183 (July 1986).

〔Wall 85〕 Wall, R. S., et al. : An Evaluation of Commercial Expert System Building Tools. Data & Knowledge Engineering, Vol.1, No.4 pp.279-304 (1985).

〔Chandrasekaran 1986〕 Chandrasekaran, B. : Generic Tasks in Knowledge-Based Reasoning : High-Level Building Blocks for Expert System Design. IEEE Expert, Vol.1, No.3, pp.23-30 (Fall 1986).

〔小林 86〕 小林重信："知識工学." 昭晃堂, 1986.

〔ICOT-JIPDEC 1987-1〕 ICOT-JIPDEC AIセンター："知的情報処理システムに関する調査研究報告書－知識システム開発方法論－", 1987年9月.

〔ICOT-JIPDEC 1987-2〕 ICOT-JIPDEC AIセンター："知的情報処理システムに関する調査研究報告書－エキスパートシステム開発ツールの比較評価－", 1987年9月.

３．知識システム構築ツールの評価項目とその説明

3．知識システム構築ツールの評価項目とその説明

現在発表されているツールの多くは、ルールやフレームの概念に基づいて構成されているが、現実には各概念の実現方法が個々のツールによって微妙に異なっている。そのため、ツールを適切に評価するためには、特定のツールに特化された概念をきめ細かく調査することが必要となる。また、ツールを利用するという立場からは、応用分野の特性に関する考察も必要となる。

そこで、我々は、ツールの評価項目の枠組みを、①ツールの概要、②ツールの機能／知識表現、③開発者／利用者インタフェース、④システムインタフェース、⑤ツールの性能、⑥ツールの利用目的、⑦ツールが得意とする分野の特性、の７つに分類して定めることとした（図３－１）。そして、各評価項目は、重複をいとわずにできるだけ詳細なものとし、また、個々のツールで用いられる特有の概念に左右されないように定めた。さらに、現在のツールでは提供されていない機能・項目についても、人工知能研究の立場から今後重要となると考えられるものについては積極的に記述することとした。これらの評価項目は、各種の文献に発表されている項目を完全に含んでいる。したがって、現在入手できるツールで標準的に採用されているルールやフレームなどの項目については非常に詳細なレベルまでの記述が可能となっている。一方、そのような従来型のツールの概念では捉らえるのが難しいような機能をもつツールについては、問題解決レベルや基本タスクの項目から必要な評価が得られるようにした。さらに、ツールと開発方法や基本タスクとの関連性にも留意している。これらの評価項目は、特別の記述がないかぎりは、○、△、×の３レベルで記述することを前提としている。

具体的な項目は、以下に記述したとおり膨大なものであるが、これによって、ツールのベンダーにとってもユーザにとっても、共通の評価の枠組みを設定できるものと考えている。ただし、これらの評価項目は、どのようなツールに対しても適用可能なことをねらって設定したものであるので、ツールの実現技術上、性能が互いに矛盾する項目も含まれている。したがって以下の評価項目を実際に利用する場合には、適当に項目の取捨選択を行なうことも必要と考えられる。たとえば、ツールの機能の豊富さと実行速度との間には必ずある程度のトレードオフが存在する。そのため、必ずしも、すべての項目について○印がつくことがよいツールの条件を表わすわけでない。むしろ、ツールの利用目的が明確であれば、それに関連する項目についてのみ良い評価がなされ、不要な機能が少ないツール

の方が、実行性能という意味から望ましいことになる。

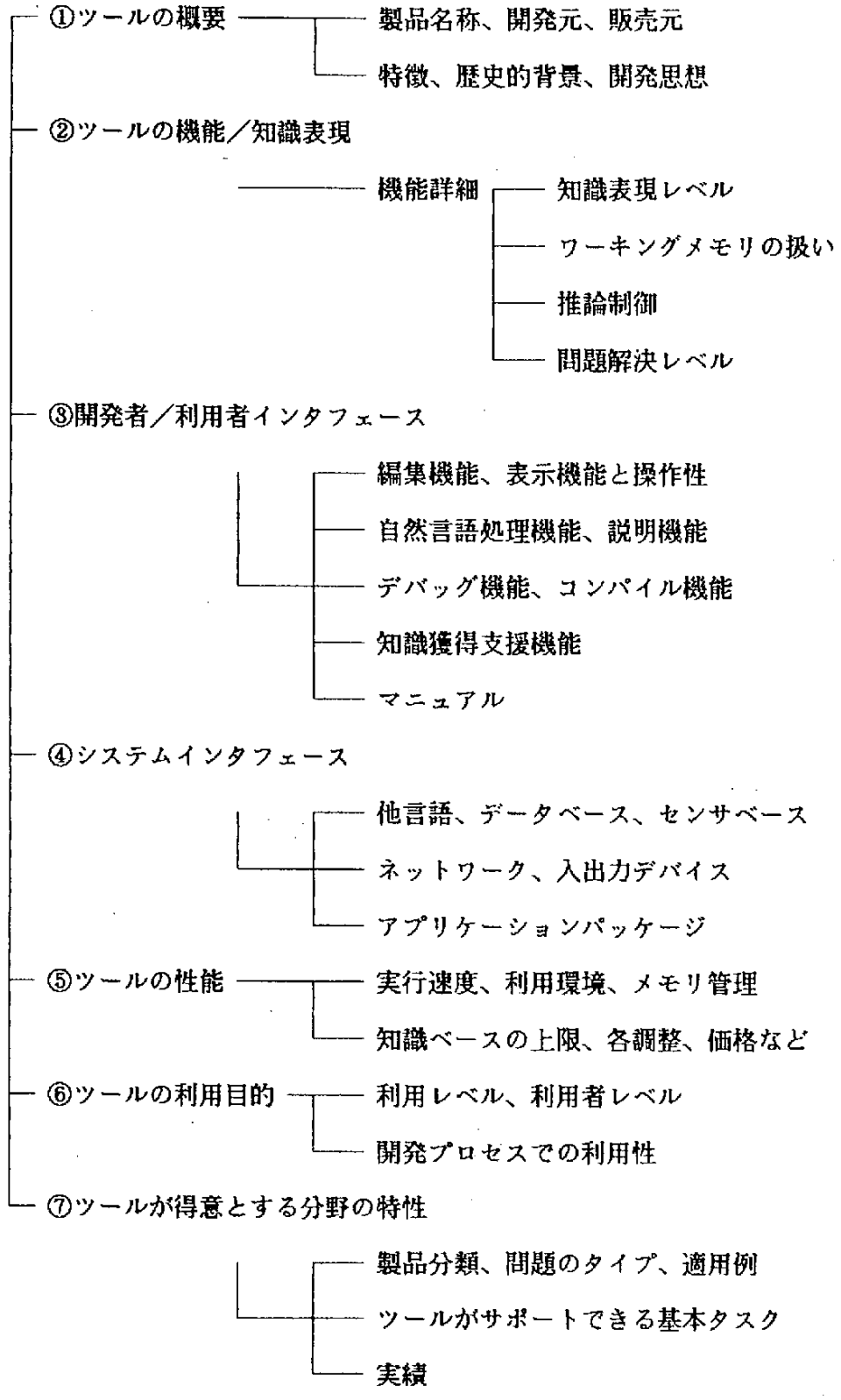


図３－１　ツール評価項目の分類

3.1 概　要

以下に述べる項目は各ツールの概要を一覧できるようにとりまとめるものである。

3.1.1）製品名称（略称）

製品の名称ならびに略称を記述する。

3.1.2）開発元（販売元）

製品としてのツールを実際に開発した大学、研究所、メーカなどの名称を記述する。さらに、開発元と販売元・配布元とが異なる組織である場合はその両社の名称を記述する。

3.1.3）特　徴

製品の特徴・特性をパンフレットレベルの記述量で完結に記述し、項目の利用者が、製品の概要を把握できるようにする。

3.1.4）ツールの歴史的背景と開発思想

エキスパートシステム開発ツールには、製品化される以前に一応の成功も収めた特定のエキスパートシステムが存在している場合が多い。そして、そのエキスパートシステムの特徴はツールに何らかの意味で継承されているのが普通である。したがって、ツールの歴史的な背景と基本的な開発思想を簡潔に記述しておくことで、ツールの特徴を的確に把握することが可能となる。本項目は、この目的で設定したものである。

3.2　ツールの機能／知識表現

以下では、現在のエキスパートシステム開発ツールに採用されている知識表現と機能を中心とした評価項目を示す。まず、機能の概要について記述し、ついで、その詳細について知識表現レベル、推論制御レベル、問題解決レベルの３つの視点から評価項目を取りまとめる。

3.2.1　概　要

機能の豊富さ、バラスンの良さ、記述の統一性、表現の理解容易性についておおざっぱな評価を行なう。機能の豊富さを除くと、残りの項目には主観的な判断がはいりやすいので注意が必要である。

・機能の豊富さ：

基本的な知識表現手法および推論機能を不足なくサポートしていれば○、単一のものしかサポートしていなければ×、その間ならば△とする。

・機能間のバランス：

各機能の間に性能的なばらつきがあるかないかを３段階で記述する。単一の機能のツールの場合にはこの項目の記述は不要である。

・記述の統一性：

ツールのサポートする機能のエキスパートシステム開発者レベルでの記述方法に統一性があるかないかを３段階で記述する。

・表現の理解容易性：

各機能向きに表現された知識が、エキスパートシステム開発者レベル、ならびに、エンドユーザレベルでどの程度容易に理解できるかを３段階で記述する。

3.2.2　機能の詳細項目

3.2.2.1　知識表現レベル

①　ルール

－推論方法

前向き推論[21]のための表現の有・無

→ルールがデータ駆動型に記述出来る。

後向き推論[22]のための表現の有・無

→ルールがゴール駆動型に記述出来る。

前向き・後向き両方向[23]のための表現の有・無

→推論時、データ駆動型にもゴール駆動型にも使える記述が出来る。

－ルールの表現要素

・条件部

結合子

and結合子[24]の有（明示的記述／非明示的記述）・無

→複数の制約条件を論理積で結合する。非明示的記述は結合子を書かない。

or結合子[25]の有・無

→複数の整数条件を論理和で結合出来る。

その他の結合子[26]の有（あればその詳細）・無

→例えば、照合のループ制御[51]を記述する。論理依存性[27]を記述出来る。等の特徴があれば記述する。

論理述語による制約[28]の有・無

→述語の評価結果の真理値によって照合出来る。

論理述語による制約のユーザ拡張性の有（あればその方法）・無

→例えば、ツールの記述言語により述語拡張が出来る等を記述する。

手続きの記述の有（あればその種類）・無

→例えば、パターン照合に用いる値の計算に算術演算が記述出来る等を記述する。

・パターン照合の対象データ型と方法[29]

取り扱えるデータ型

→パターン照合の対象にすることの出来るデータ型の種類

・基本データ型

パターン	制　限	文字型制約	変数型制約	ワイルドカード型制約
整数	最大数値範囲	有・無	有・無	－
実数	最大数値範囲	有・無	有・無	－
シンボル	最大文字数と日本語	有・無	有・無	－
配列	最大次元数	有・無	有・無	

ストリング　最大文字数と日本語　有・無　　有・無

リスト　　最大要素数　　有・無　　有・無

→文字型制約とは、表現されたシンボルパターン自体に照合する。

→変数型制約とは、変数として照合・束縛される。

→ワイルドカード型制約とは、任意の値に照合する。

抽象データの記述について

→フレーム等のデータの構造化とデータ間の関係の制約

構造名の変数型制約制約について

関係に基づく構造名の変数型制約有・無

→関係の制約によって、変数としての構造名を束縛することが出来る。

例えば子フレームとis-a関係[30]にある親フレーム名を得る。

属性に基づく構造名の変数型制約の有・無

→属性の制約によって、変数として構造名を束縛することが出来る。

例えば属性による制約を満たす構造名を得る。

属性に対する変数型制約の有・無

→構造名・関係・他の属性による制約を満たす属性値を得る。

疑似自然語表現の有・無

あれば、その種類（日本語・英語）、入出力別（入力・出力）

→ルールの内容が疑似自然言語で表現出来る。

・実行部

四則演算の有・無

→パターン照合で得られたデータに対し四則演算（＊／＋－）が出来る。

関数演算の有（組み込み関数・ユーザ定義関数）・無

→パターン照合で得られたデータに対し関数演算が出来る。

組み込みはシステムサポートのもの、ユーザ定義はユーザによる拡張

手続き構造の有・無

→実行部内で条件分岐、繰り返し等の制御が出来る。

データを仮説[31]として取り扱う機能の有・無

→仮説に基づく推論のための仮説データの言明等が出来る。

－その他の特徴について

・デーモン[32]の有・無

→あるルールの実行に伴って暗黙的に起動される前処理、後処理等を記述出来る。

・実行の保留機能[33]の有・無

→実行部の内の処理がある条件下で動く様に一時的に保留する事が出来る。

・その他（あれば具体的に）

－ルールの制御表現

・ルールのグループ化[34]機能の有・無

ある場合はその使い方（call関係・イベント駆動・グループの優先度・その他）

→関連の高いルールをグループ化する機能を提供している。

→call関係は手続き呼び出しにとしてルールグループを呼ぶ。

→イベントはルールグループ呼び出し専用のフラグによる駆動。

② フレームとオブジェクト

評価項目を取りまとめる上で、我々は、知識表現手法としてのフレームとオブジェクト[1]の概念を同一のものとして扱う。さらに、意味ネットワーク[2]の概念もこの項目で取扱う。これらの概念の特徴は、情報をモジュール化して記述し、その間に階層的な関係をつけること、また、各関係のなかには各情報や属性を上位から下位へ継承する機能をもつものがあること、情報の記述形式のひとつとして手続きが書けること等である。

－ツールでサポートする機能の名称

ツールによってここで扱う同一の概念をいろいろな用語で表現する。概念の混乱が生じないように基本的な用語の対応をとる。

・フレーム[3] ： (Frame, unit, schema, object, semantic-network,そのほか)

・関係[4] ： (relation, slot, そのほか)

・手続き[5] ： (attached procedure, method, そのほか)

－関係の記述：

フレーム機能がもつ関係の取扱いについての項目である。

・非継承関係：

継承機能[6]をもたない関係についての項目である。

非継承関係の有無　　有　無（どちらかに丸をつける）

非継承関係の種類　　(組込みの非継承関係にどのようなものがあるかを記述する)

part-of [7]　　(全体／部分間の関係を示す、他の名称でもよい)

その他[8]　　(具体的に関係名と機能とを記述する)

・継承関係

継承機能をもつ関係についての項目である。

非継承関係の有無　　有　無（どちらかに丸をつける）

非継承関係の種類　　(組込みの継承関係にどのようなものがあるかを記述する)

class-subclass[9]　　(集合間の包含関係を示す、他の名称でもよい)

class-instance[10]　　(集合／要素間の関係を示す、他の名称でもよい)

その他の関係　　(具体的に関係名と機能とを記述する)

多重継承の有無[11]　　有　無

(下位の概念が複数の上位概念から情報を継承してきた場合の取扱いが組込まれているかどうかを記述する)

・継承の制御方法

概要：

(継承の制御方法にはさまざまなものがあるのでツールで採用した方式の概要を具体的に記述する。)

継承単位の指定方法

(継承機能の指定がどの概念のレベルでなされるかを記述する)

フレーム単位に指定可？

スロット[12]単位に指定可？

その他の指定方法（具体的に記述する)

制御方法

(継承した値の扱いについて記述する)

親の値をデフォルトとしてとる

必ず親と同値になるか？

その他の制御方法

多重継承の制御（　　）

(具体的に記述する)

・関係のユーザ定義の可能性　　有　無

継承関係の拡張性　　有　無

非継承関係の拡張性　　有　無

(有の場合にはその方法について具体的に記述する)

－属性／データの記述

フレームに記述できるデータの属性に関する記述である。

・属性値の型と制約

（属性値のデータ型とそれに対する制約を記述する。）

データ型　制約

整　数　最大数値範囲

実　数　最大数値範囲

シンボル　最大文字数　日本語の使用可能性

配　列　最大次元数

ストリング　最大文字数　日本語の使用可能性

リスト　最大要素数

その他（　　）

・スロットの制御（ファセット[15]など）

（スロットの制御方法について記述する）

属性値の数の制限　可　不可

スロットの存在するフレーム／オブジェクトの範囲の制限　有　無

その他の制御方法　（方法：　　）

・情報隠蔽[16]機能

特徴的な情報隠蔽機能があればそれについて具体的に記述する。

－手続きの記述

付加手続きの扱いに関する記述である。

・付加手続き[17]の有無　有　無

機能が存在する場合の種類：

if-needed (get-slot), if-added (put-slot), if-removed[51],

その他（名称と機能とを具体的に記述する）

・メソッド[18]としての扱いの有無　有　無

メソッドの記述言語（　　）

ルールの記述　有　無

メソッドの継承[19]の有無

多重継承の有無　有　無

多重継承の制御（　　）

(具体的に記述する)

・起動操作

付加手続きを起動する倍の方式の記述である。

起動操作[20]の種類（メッセージパッシング、デーモン[32]、その他)

メッセージパッシングの方式・形式（　　）

実行の並列性　　有　無

③ 論理表現

論理型のパラダイムをツールがサポートするかどうかを記述する。

・言語の種類（Prolog，その他)

バックトラック機能　　有　無

ユニフィケーション機能　　有　無

協調推論処理機能　　有　無

(複数の論理表現が可能であれば、各言語ごとに記述する)

④ 他言語インタフェース

他言語インタフェースがある場合対象言語の情報を記述する。なお、詳しい情報についてはシステムインタフェースの項目に記述する。

・言語名（　　）

⑤ 不確実な情報の扱い

不確実な情報の扱い方を記述する。

・確信度

ルールに記述する確信度（CF：Certainty Factor）の扱いを意味する。

確信度のサポートの有無　　有　無

有の場合の計算方法　and

or

combination

その他

利用者による確信度計算方法の指定　有　無

・ファジィ

ファジィ推論のサポートの有無　　有　　無

有の場合

メンバーシップ関数の種類

台形型

三角型

その他

ファジィ演算の種類と型

積演算　論理積（X・Y＝min(X, Y)）

代数積（X・Y＝XY）

限界積（X・Y＝0 v （X＋Y－1））

激烈積（X・Y＝X where　Y＝0,

Y where　X＝0,

0 where　X, Y＞0）

和演算　論理和（X・Y＝max(X, Y)）

代数和（X・Y＝X＋Y－XY）

限界和（X・Y＝0 ^ （X＋Y））

激烈和（X・Y＝X where　Y＝0,

Y where　X＝0,

1 where　X, Y＞0）

否定演算

ファジィ推論の方法

（ファジィ推論の合成規則について記述する）

利用者によるファジィ演算計算方法の指定　　有　　無

有の場合の指定方法

・その他の扱い方法

Dempster-Shafer 理論

そのほか

3.2.2.2 ワーキングメモリーの扱い

ワーキングメモリーは推論処理の途中で動的に変化する情報（仮説、目標、データ）などを保持する。以下の項目ではその扱い方についての情報を記述する.

①ワーキングメモリー上のデータ表現

ワーキングメモリーの構造として次のどの概念をサポートするか○をつけて表す.

ファクト

複数個の引数をとる述語.

ベクタ

任意長の順序集合.

ブラックボード

複数の実行単位（知識源，エージェントなど）から共通にアクセスする場合.

フレーム

フレームをワーキングメモリーとしている場合.

その他

特にあれば記述する．データ型として何が許されるかも記述する.

（シンボル，数値，文字列，ファクト，ベクタ，その他）

②ワーキングメモリーの管理

ワーキングメモリーの更新に伴う管理のうち次の概念をサポートする場合に○をつける.

ＴＭＳ （Dipendency Directed Trutm Maintenance System

従属関係に基づく真実の維持機能．推論の途中経過を保持することで，ワーキングメモリーの情報（仮説群）を効率良く矛盾のない状態に保つ技法．仮説空間を縦型に探索する．矛盾しないひとつの解釈を維持する.

ＡＴＭＳ （Assumption Based Truth Maintenance System)

仮説に基づく真実の維持機能．仮説の組合わせを複数個同時に保持することで整合性の維持を行う技法．ＴＭＳと比較すると，矛盾しない複数の解釈を維持することが特徴である．仮説空間を横型に探索する.

ブラックボード

複数の実行単位（知識源，エージェントなど）から共通にアクセスできる構造をもったワーキングメモリー.

③多重世界の表現

仮説推論などを行う場合に複数の世界（仮説集合）を表現する機能の有無

コンテクスト

仮説群（コンテクスト）の従属関係をツリー状態で保持する機能.

ビューポイント

仮説群（ビューポイント）の従属関係をネットワーク状態で保持し，推論の経過に伴って生ずる変化に対し整合性を自動的に維持する機能．

その他

特にあれば記述する．

3.2.2.3　推論制御

推論制御について，ルール，フレームにわけて機能の有無を記述する．

①ルール

－競合解消

条件部を満足するルールが複数個存在する場合にその中から次に実行すべきルールを選択する方法．

ルールの優先度：　ルール自身に数値などの形で優先度をつけられる．

ルールのあいまいさ：　ルールについた確信度などのあいまいさを現わす尺度を利用できる．

データの新しさ：　条件部と照合したデータの最終更新時刻を利用する．

条件部の詳細さ：　条件部がより厳しいルールを優先することができる．

ルールの記述順序：　記述した順序で実行できる．

その他：　特にあれば記述する．

決定要素の適用順序

決定要素の複数の組合せが可能な場合，その組合せ方を記述する．優先順序が存在する場合はその順序を記述する．

適用順序の変更可能性

前項がユーザの手で変更可能か，また，実行中に動的に変更可能かどうか．

－推論制御の最適化手法

競合ルール集合の抽出，および，競合解消の高速化手法の有無．たとえば，以下の手法などが採用されていれば記述する．

（RETE: あらかじめルール集合をコンパイルし，ワーキングメモリー要素を分類する弁別ネットワークと，変数値の関係をテストするネットワークワークを作成する．そして，しべてのワーキングメモリー要素をこのネットワークに記録しておき，ワーキングメモリーの更新のたびにネットワークの記録を書きかえる．これによって，１つの要素の更新の後，ただちにどのルールの条件部が満足されるかを知ることができる．

改良RETE: 上記のアルゴリズムの欠点，たとえば変数間の関係についての制約条件の組合せが大きいとき生ずるオーバーヘッドの改良など．どう改良されているかもできれば記述する．

部分RETE: 上記のアルゴリズムの一部を実現した場合．どう実現されているかもできれば記述する．）

－メタ制御

ルールの優先度などの実行制御に関する手続の記述がユーザに許されるかどうかを記述する．

－メタルール

前項のメタ制御をルール形式で記述できるかどうか．

②フレーム／オブジェクト

フレームにおける推論制御について記述する．

－継承解決

多重継承機能において，複数の継承方法が存在する場合の競合解消．たとえば，以下の方法など．

（最短パス優先：継承の経路の短いものを優先する
縦型探索：あらかじめ決った順序にしたがって次々と親フレーム／クラスをたどる）

－デモン制御

スロット値の参照あるいは変更に伴って，ユーザの記述した手続（デモン）を実行する機能の有無．機能がある場合に記述できる内容はなにか．

－メッセージパッシングの制御

他フレームとの情報交換，呼出しに関する方式を記述する．たとえば，サブルーチン形式(call - return)，コルーチン形式(broadcast)など．また，並列処理が可能な場合の同期の取りかたを記述する．

③アジェンダ機能

実行可能な単位（ルール，オブジェクト，デモン，プロセス，タスクなど）を優先順位にしたがって並べたリストをもっているか．もしあれば，その要素が何で優先順位を決定する方法はどのようなものか，実行決定の方法は何かを記述する．

3.2.2.4　問題解決のレベル

ツールに陽には表現されない問題解決手法の実現可能性・容易性について記述する．

①探索手法の実現性

次に示す探索手法を容易に記述・実現できるか．○をつける．

－blind search（網羅的探索）：　深さ優先，幅優先

－heuristic search（ヒューリスティック探索）：
　　最適探索（評価関数などが決る場合），
　　最良優先探索（目標との差異を調べるA*など）
　　山登り法（選択肢のうちで一番よいものを選ぶ）
－その他（AND/OR グラフ，ゲーム木，など）

②問題解決のモデル
　次に示す問題解決モデルを容易に記述・実現できるか，○をつける

－推論モデル
　　不確実な推論（データ，ルールなどに不確実な情報が含まれる場合）
　　仮説的な推論（仮説の生成・管理を利用する場合）
　　時制推論（時間・時刻・期間などに関する知識を利用する場合）
　　分散協調推論（複数の問題解決モジュールの協調によって全体として問題解決を行う場合）
　　定性推論（定性モデルを利用した推論を行う場合）
　　その他（具体的に記述する）

－データの扱い
　　非同期入出力（人間あるいは実験装置などからの入力・応答のタイミングについて，実行時での融通性はあるか）
　　リアルタイム性（プラント制御，シミュレーションなど実時間性が要求されるデータの入出力が可能か）

③知識処理手法の実現性
　以下に示す標準的な知識処理手法が容易に記述・実現できるかどうかを記述する．○をつける．

-generate and test　（生成検査法：解候補を生成して，それをテストする方法）
-hierarchical generate and test　（階層的生成検査法：生成検査法を階層的に適用する）
-guessing　（推測：不完全な知識・状況のもとでもっともらしい推論を行う方法）
-topdown refinement　（トップダウン精密化：生成検査法で解の構成の詳細度に着目して，上位から階層的に解を生成する方法）
-least commitment　（拘束最小化：生成検査法で解の構成の詳細度に着目したとき，上位階層での決定が下位階層に伝播する場合その制約条件を最小にする方法．）
-constraint propagation　（制約条件の伝播：制約条件の影響を追跡し，全体の整合性をとることで解を導く方法）

-backtracking （バックトラッキング：問題解決候補の探索中に失敗した場合，後戻りして次の候補をさがす系統的な方法）

-dependency directed backtracking （従属関係にもとづくバックトラッキング：推論中の従属関係情報を保持することでバックトラッキングを効率化する方法）

-opportunistic scheduling （機会駆動型スケジューリング：複数の問題解決機構を用いる場合に，各機構の状況を判断しながら全体を制御する方法）

-classification（分類：解候補をあらかじめ数え上げておき，目的の解がそのうちのどれに分類されるかを調べる問題解決方法）

－その他（具体的に記述する）

3.3 開発者／利用者インタフェース

ツールを利用する観点からツールの操作性、インタフェース機能、等を中心に、システムを開発する側面と、完成したシステムを利用する側面とから、ツールを評価する。

3.3.1 概 要

ツールの各機能を全体として評価する。詳細な個別の項目は3.2）～3.5）に記述する。

(1) 操作性

人手操作についてキー操作やメニュー選択が少ない操作回数で、かつ、覚えやすい構成となっている。

(2) 統合性

各機能が良く分類されていて、任意の機能を必要となった時点で簡単に利用できる。

(3) 一貫性

同種の操作／機能について複数種類の指定方法がない。

(4) 拡張性

ツールが本来は持たない拡張した機能をツール利用者固有の操作や機能として追加、もしくは変更できる。おな、個別の操作や機能を組み合わせてマクロとして追加、もしくは変更することを含む。

(5) 説明機能

推論の動作について何故該当の結論に至ったかを順を追って、もしくは必要に応じて示し、理解させることができる。

(6) ヘルプ機能

各機能についてその概要、呼び出し方法、基本動作、利用例を簡単に示すことができる。また、編集や推論の途中の任意の時点で起動できる。

(7) エラー処理機能

編集や推論動作中で利用起因／システム起因／ツール起因のエラーを生じた場合、

① 他のサブシステムに影響を及ぼさない

② 正常動作に自動復帰、あるいは手順を示し手動復帰できる

③ エラーからの復帰が早い

④ 何故その状態となったかを説明できる

⑤ 今後どうすればエラーが生じないかを示すことができる

3.3.2 編集機能

ルールやフレーム等の知識や、システム全体やサブシステムを操作するためのシステムデータを編集する機能について評価する。

また、「ブラウズ[35]」とは、該情報の参照のみ行い、更新しないことを示す。

(1) 編集対象の種類

編集対象の記述形式（テキスト、図形、音声、イメージ等）を記述する。

（例：知識表現のうちフレームとルールをテキストで記述）

(2) 専用エディタ／ブラウザ

知識ベースや図形の編集に走行マシン付属のスクリーンエディタ、テキストエディタ等以外の専用機能がある。また、図形に関しては指定位置を座標値でなくマウス／タブレット等を利用して完成後の図形を確認しながら編集できる。

(3) 一貫性管理[36]

該情報を編集する際、誤った情報の入力や誤操作等を生じないようなガイド／デフォルト／前回投入のコマンドやデータの再利用／区切り記号のチェック／シンタックスチェック（括弧のレベル他）の機能等がある。

(4) 知識ベースの世代管理[37]

編集済みの知識をその編集契機で世代管理し、最新の知識を保持する、もしくは必要に応じて過去に投入した知識の状態に戻る、また、変更部分や変更差分を表示できる。

(5) 拡張性

編集機能に関し、ツール利用者固有の操作や機能、マクロ操作や機能等を追加、もしくは変更できる。

(6) 操作性、スピード

アイコン、メニュー、ヘルプ機能等のマンマシンインタフェースが拡充していて操作し易い。また、各操作が高速に実現され、応答待ちに時間を要することはない。

(7) 説明機能

編集処理の各機能についてその概要、呼び出し方法、基本動作、利用例を簡単に示すことができる。また、編集の途中の任意の時点で起動できる。

(8) 一括編集機能

各操作の指示（コマンド他）をファイル等に前もって作成しておき、会話処理を介さないで大量の編集処理を一括に処理できる。

(9) コンパイルチェック

知識の記述に関し、編集中あるいは編集直後にそのシンタックス（ＬＩＳＰ記述であれば括弧のレベル）をチェックする。また、コンパイル処理を編集処理の中でも起動できる。

(10) エラー処理

操作誤り、割り込みの発生、無限ループ等の異常動作に対し、①他のサブシステムに影響を及ぼさない。②正常動作に自動復帰、あるいは手順を示し手動復帰できる。③エラーからの復帰が早い。④何故その状態となったかを説明できる。⑤今後どうすればエラーが生じないかを示すことができる。

(11) デバッグ機能

推論、アプリケーションソフトウェアに関して、トレース、ブレーク、ブレーク中での知識の編集、継続処理、等ができる。また、その指示も当該情報を初期に投入した形式（コンパイル後の記述形式でない）可能である。

(12) 複数人による開発サポート

同一知識（ルール、フレーム等）を複数人で共用して操作できる（知識の共用、モジュール管理リスト等）。

(13) プロジェクト管理

プロジェクトの進捗状況（基本検討、プロトタイプ作成、実用システム構築、等）を管理するための機能（モジュール毎の進捗状況、テスト済フラグ、線表管理、等）がある。

3.3.3 表示機能と操作性

マンマシンインタフェースのうち、視覚用出力である表示機能と、入力法等で操作性を評価する。

(1) ウィンドウシステム

知識や図形の初期入力や更新、推論実行や結果出力について、それぞれを同時に見ることのできるマルチウィンドウ表示ができる。

① グラフィック

任意図形の表示が可能である（ビットマップ・ディスプレイ利用等）

② アイコン

該操作や機能の概要を示す図形をポインティングデバイス（キー、マウス、タブレット、等）で選択することで操作の指示ができる。

③ ポップアップメニュー

主操作と副操作、前処理や次処理、等がメニュー（アイコン／テキスト）等で同時に表示され、これを選択することで該操作の指示ができる。

④ ズーミング

図形の一部を拡大／縮小して表示できる。

⑤ アニメーション

静止画でなく簡単な動きを伴って表示できる。

(2) カラー表示

表示がカラーである。また、その同時表示可能な色の数はNである。

(3) シミュレーション

時間の経過を示す時計図形、対象の動きを示す物体の図形の移動や振動を表示できる。

(4) グラフィック（2次元／3次元）

2次元の線画（円、多角形、任意図形、等）、面塗り潰し、3次元のワイヤフレーム図形、ソリッドモデル図形等の表示が簡単にできる。また、自然画の入力（イメージスキャナ等）や出力ができる。

(5) 知識ベースのグラフィック表示／編集（フレーム、ルール、その他）

知識ベースの内容を全体－部分関係、上下関係、呼出側－呼ばれ側の関係、等を図で表示できる。

3.3.4 自然言語処理機能

知識の記述、メニュー、結果の出力、等に予約語以外の自然言語が利用できる。

① 対象言語

利用可能な言語を、出力のみ／入力のみ／入出力、とに分けその可能性を評価する。また、規則知識については、条件部について、もし…ならば（かつ、あるいは）、

実行部について、…である等の自然言語で記述できる。

i）日本語（入力／出力）

ii）英　語（入力／出力）

iii）その他（入力／出力）

i）、ii）以外の可能な自然言語を記述する。

② 表示機能

出力手段としての表示に自然言語が利用できる。

i）テキスト

文章等のテキストのなかに自然言語が表示できる。

ii）アイコン／メニュー

操作を指示するアイコンやメニューに自然言語が表示される。

③ 自然言語理解

入力された自然言語の文章から必要な事項を抽出することができる。

i）字句解析[38]

操作の指示や知識を記述する予約語やキーワード等を単語レベルで抽出する。

ii）構文解析[39]

予約語やキーワード等の単語間の掛り受けを抽出する。

iii）意味解析[40]

単文について主語、述語、目的語を抽出する。

iv）文脈解析[41]

複文について文の前後関係から、暗黙に表現されている事項を抽出する。

④ 音声処理、会話理解

知識の記述、メニュー、結果の出力、等のメディアとして音声が利用できる。

また、主語や目的語の省略、代名詞の多い会話的入力ができる。

3.3.5）説明機能

推論過程に関する説明機能について評価する。

① アジェンダ[42]

実行可能な実行単位（ルール、オブジェクト、デモン、プロセス、タスク等）を優先順位に従って並べたリストを表示する。

② マッチング

ルールの条件照合に関し、各ルールと条件との一致、不一致を表示する。

③ 推論過程のトレース

ルールと条件の照合、選択実行された結果、等の推論過程を連続的に表示できる。

④ ジャスティフィケーション[43]

利用者からの質問に対して深い知識（対象の構造知識等）を用いて説明する。

（例１：車の構造知識をもとに、部品間の接続関係、包含関係等を説明する）

（例２：内蔵知識について、いつ誰が入力したか、何をもって正しいとしたか、等の説明をする）

⑤ ＷＨＹ[44]

利用者からの推論過程に関する質問に対して回答を表示する。該当する知識が無ければその旨を表示する。

⑥ ＨＯＷ[45]

結論に到った過程を表示する。

⑦ ＷＨＡＴ－ＩＦ[46]

結論を導くための仮説を表示する。

⑧ その他

3.3.6 デバッグ機能

知識入力の正常性、知識の充分性の確認のためのデバッグ機能について評価する。

(1) 専用デバッガの有無

走行マシン固有に付属、あるいはツールの記述言語（ＬＩＳＰ、Ｃ等）に付属したシンボリックデバッガ等以外に知識記述言語（ルール、フレーム、等）専用のデバッガがある。

(2) 知識ベースの一貫性管理

知識の内容に関してその追加時に、前もって投入しておいた知識と矛盾が生じた場合の指摘、無限ループの指摘等ができる。

(3) 拡張性

デバッグ機能に関し、ツール利用者固有の操作や機能、マクロ操作や機能等を追加、あるいは変更できる。

(4) 操作性、スピード

①デバッグ対象部分の指示、②ブレークポイントの指示、③ステップ動作の指示、等が簡単にできる。また、デバッグモードでも推論時間等が極端に増加しない。

(5) エラー処理

デバッグ中に利用者起因、ツール起因のエラーが生じた場合、①他のサブシステムに影響を及ぼさない。②正常動作に自動復帰、あるいは手順を示し手動復帰できる。③エラーからの復帰が早い。④何故その状態となったかを説明できる。⑤今後どうすればエラーが生じないかを示すことができる。

(6) テスト機能

知識の正当性、充分性を確認するための機能について評価する。

① 実行過程のセーブ機能

推論の実行過程をファイル等に蓄積し、後刻その追跡、再実行、継続実行等に利用できる。

② テストデータの保存、管理、再実行

テストに使用する入力データを保存・管理し、テストの度に再入力することがない。

また、テストデータのみを編集できる。

③ テストケースの自動生成

知識の正当性、充分性を確認するためのテストケースを自動的に作り出す。

（例：組合せ的に生成、選択は人手、等）

(7) チューニング機能

システムの本格運用に備えて操作性の向上、高速化、メモリ量削減のための機能を評価する。

① パフォーマンス測定

推論時間、ガベッジコレクション（GC）時間、使用メモリ量、使用ファイル量等を出力する。

② ボトルネックの発見のサポート

処理時間／メモリ量を多く費やしている部分を発見するための機能がある。

（例：多数回実行されるルールやルーチン名、多数回参照されるフレーム、等の指摘。また、これらの全体に占める割合の出力。）

③ 知識ベースの変更サポート

知識の整理、体系化に利用できる、常にA⇒Bの順に起動される、等の従属関係、全体－部分関係を出力する。

3.3.7 コンパイル機能

記述された知識は、通常利用者の理解し易い記述レベルであり、これを計算機内部で効率良く（時間／メモリ）実行するためのコンパイル機能を評価する。

① インタプリタ／コンパイラ両モードの併用

編集を優先させインタプリティブに実行、性能を優先させるコンパイラ後の実行、編集とデバッグをインタプリティブに実行し、最終的にはコンパイル後に実行、等を記述する。

② インクリメンタルコンパイル

編集した部分のみをコンパイルすることでコンパイル時間を短くする。

③ シンタックスチェック

知識の記述言語レベルで、括弧のレベルチェック、予約語の誤り、等を指摘する。

④ 基本的なセマンティックチェック

論理的な誤り（記述上文法誤りはないが、X＞0、かつX＜0、等）を指摘する。

3.3.8 知識獲得支援機能

知識ベースへの知識内容の入力を簡単にするための機能を評価する。

① 開発方法論のサポート

エキスパートシステムの開発ステップ（初期の基本検討段階でのモデル作り、知識の拡張等）に応じた検討項目、次ステップへの継承法、評価法等を支援する機能がある。なお、その形式（ドキュメント、ソフトウェア等）を記述する。

② 知識表現（ルールなど）の自動生成

一定のガイドラインに従ってモデルや条件を入力すると、ツールの記述言語に従ったルール記述や条件記述（フレーム等）を自動生成する。

③ モデリング支援機能

環境条件、制約条件、拘束条件、数個の条件とその動作例等から、専門家の思考のモデル、対象業務（診断、設計等）のモデル等を作成するための支援、あるいは

自動生成機能がある。なお、その形式（ドキュメント、ソフトウェア等）を記述する。

④ 学習機能

同一の試行もしくは同一の母集団からの試行を繰り返すとき、次の回の試行は前の回のものよりより良い性能を示すようなシステムの変化をもたらす機能がある。

(参考)

(例：新しい条件に対し、これに対応／対処する新しい知識を自動生成する、例題に対する規則を作る、制御を改善する、経験的知識について、共通部分の指摘による特徴を抽出する)

(参考) Simon, H. A. : Why Should Machine Learn?, in Machine Learning An Artificial Intellegence Approach, Springer-Verlog (1984)

3.3.9 マニュアル・教育

ツールを利用するための必要な専門知識を習得するためのドキュメント、ソフトウェア等について評価する。なお、誤りは極力少ないこと。

(1) マニュアル

① 入門マニュアル

他のツール、知識処理等を全く知らない初級レベルを対象とした概説内容がある。

② レファレンスマニュアル

ツールが持っている全ての機能について、網羅的な内容がある。また、索引が完備している。

③ その他のマニュアル

特徴的なマニュアルがあれば記述する。

(2) 教 育

① ツールに則した教科書の有無

訓練または自習用として知識処理の歴史から、モデルの分類、エキスパートシステム開発方法論までを含んだ、ツール購入者以外でも手に入れられる本がある。

② ツールに則した教科書の名前

上記②に該当する教科書の名前を書く。

③　例題（ディスク、テキスト、ビデオ、その他）

エキスパートシステムが対象とする業務の特徴、問題分析、モデルの作成等を含めたエキスパートシステム開発事例、一般的なモデルのツールによる記述法、等の例題がある。

④　教育カリキュラム

他のツール、知識処理等を全く知らない初級レベルを対象とした概論レベルのセミナ、ツールの使い方や問題分析の演習を含めた訓練コース、等について記述する。

⑤　ＣＡＩ

ツールの機能を習得する、エキスパートシステムを構築するためのＣＡＩソフトウェアがある。

3.4　システムインタフェース

問題ツール自体が持っている機能だけでは、エキスパートシステムの実用化に不十分な場合が多い。またエキスパートシステムが既存の大規模システムの一部分に組み込まれて使われる事がある。そこで既存のシステムとのインタフェースの容易さが重要な評価項目となる。たとえば以下のような事が実用システムでは必要となる。

・数値計算部分はＦＯＲＴＲＡＮで記述されたパッケージで行い、その結果を推論で利用する。

・大量のデータベースにアクセスし、推論に必要なデータを利用する。

・診断システムや制御システムで、各種センサーからデータを自由に取込みたい。

・推論に必要な情報を、ネットワーク経由で収集する。

・利用者インタフェースにタッチパネルや画像情報などを利用したい。

ここでは、これらの評価項目として、以下の６項目にまとめた。

3.4.1　他言語インタフェース

ＬＩＳＰ等ツールの構築言語とのインターフェースではなく、ここではＦＯＲＴＲＡＮ等の手続き型言語とのインターフェースについて記述する。

・有無（手続き型言語とのインターフェースの有無）
・種類（Ｃ等、具体的に記述する）
・入出力パラメタの受け渡し（具体例を記述する）
・デモンの記述　？

3.4.2　データベースインターフェース

リレーショナルデータベース等のデータベース管理システムとのインターフェースがあれば具体的に記述する。

・有無
・種類
・方法

3.4.3 センサベースインターフェース

センサベース分野向けのインターフェースを特別に持っていれば記述する。

- 有無
- 種類
- 方法
- オンライン機能（リアルタイム機能の有無）
- 非同期処理機能（指定したタスクへの割り込み処理機能があるか）

3.4.4 ネットワークインターフェース

ネットワークを通じた情報の入出力機能を持っていれば具体的に記述する。

- 有無
- 種類
- 方法

3.4.5 入出力デバイスインターフェース

マウス、タッチパネル、ジョイスティック等、利用者用の入出力デバイスとのインターフェースをサポートしていれば記述する。

- 有無
- 種類
- 方法

3.4.6 アプリケーションパッケージインターフェース

- 有無
- 種類
- 方法

3.5 ツールの性能

3.5.1 実行速度

① ルール

1 sec あたりの発火回数を測定する。

② フレーム

フレームの発生に要した時間、スロット値の書き換えに要した時間、などを測定する。

③ ＢＢ[47]／ＡＴＭＳ[48]

ＫＳ／仮説空間の数と問題解決にまで要した時間を記述する。

（対象ハードウェアを記述すること。各項目ともインタプリタ、コンパイラの区別を記すこと、２つあれば、どちらも記述する。ルール、フレームの場合では単純なモデル（性能評価問題1.3.5、６などを参照）を採用する。ＢＢ／ＡＴＭＳは使用したモデル（例えば、８クイーンのランクの数とすべての解を求めるまでの所要時間など）を明記する。）

3.5.2 利用環境

マシンの名称と種類（lisp machine, workstation, personal computer, main frame)、ＯＳ、開発言語、必要な容量（ディスク、メモリ）を記述する。

3.5.3 メモリ管理

ＯＳの処理にまかせているか、独自の管理機能を持つかを記述する。特に、ガーベジコレクション[49]はバッチ型（未使用セルがなくなった時点で開始する）か、リアルタイム型（関数がセルを手放すときに随時開始される）かなどの記述をする。

3.5.4 知識ベースの上限

ルール数、フレーム数など知識ベースを構築する上で物理的な上限があれば記述する。

3.5.5　拡張性

ソースプログラムの提供、推論機構の変更、知識表現の拡張性などを記述する。

（スイッチ群によるツールの設定の多様さなどがあれば、それも記述する。）

3.5.6　価格等（ハードウェアで異なる場合はハードウェアごとに記述する。）

① 価　格

1 set 購入の場合、2 set 以上購入の場合に分ける。

② 保守費用

③ トレーニング費用

入門／高度／高度

④ コンサルタント費用／方法

複数あればそれぞれについて記述する。

3.6　ツールの利用目的

3.6.1　利用レベル(どのレベルに適しているか○をつける。特記事項があれば記述する。)

・ＡＩ技術の勉強／教育

ＡＩの概念やエキスパートシステムの感覚を把握したい。例えばルール中心の簡単な機能のＡＩツールを、パソコン上で自ら組んでみたいというレベルに適したもの。

・フィージビリティスタディ

ある分野についてエキスパートシステムの応用の可能性を検討し、できればいくつかの具体的開発テーマを検討したいというレベルに適したツール。

例えば、ワークステーション上で稼働し、ルール・フレーム、前向き・後向き推論等の基本的機能、および教育コースを備えているツール。

・プロトタイプ開発

いくつかの対象問題を選定し、実用化の可能性を検討するためにプロトタイプを開発するレベルに適したツール。例えば大型コンピューター又はワークステーション上で稼働し、高機能、優れたユーザーインターフェースおよびレベルの高い教育・サポート体制を有するツール。

・実用システム開発

特定の問題について実用化を目指すレベルで、実用化のためのプロトタイプ作成を含む。例えば、既存システムとの連携の容易性、処理の高速性、優れた利用環境、豊富な機能、および十分なサポート体制を有するツール。

3.6.2　利用者レベル（一部を除き記述する。）

・トレーニング…ＯＪＴ、教育プログラムの有無・特徴

1）教育プログラムの幅広さ

初級から上級まで、幅広く段階的な教育カリキュラムの有無。

2）教育内容の充実度

理論から実際まで、深く掘り下げた教育内容の有無。

3）教材、教育設備の充実度

ソフトウェア環境、ハードウェア環境、テキスト、問題集、ケース・スタディ、教育ソフト、ＶＴＲの有無。

4）教師の優秀度

経験豊富な技術者の有無。

・使用時のサポート…On－Call、その他

的確なコンサルテーションが速やかに受けられるか。

またドキュメントや提供情報は十分か。

1）コンサルテーション方法の多様性

電話、電子メール、オンサイト等

2）コンサルティングの範囲の広さ

コンサルティングのタイミング（導入時、開発時、実機化時）

ナレッジエンジニアの経験年数、開発システム数等。

3）ドキュメントの充実度

マニュアルの整備状況。ソースコード、バグ情報の公開等。

・ユーザー会の有無、頻度、内容

ユーザー会のメンバー数、主要メンバーの特徴、ユーザーの技術レベル等を含む。

3.6.3　開発プロセスでの利用性

各開発プロセスの説明は、昭和62年度報告書の「知識システム開発方法論　2.1 知識システムの一般的な開発手順」を参照下さい。

3.7　ツールが得意とする分野

3.7.1　製品分類

3.7.2　ツールがサポートできる基本タクス（○をつける）

問題のタイプ（合成型問題、解析型問題）および各基本タクスの内容については、昭和62年度報告書の「知識システム開発方法論　3.　応用分野の特徴とアプローチ」を参照下さい。

4．性能評価用テスト問題の仕様説明

4．性能評価用テスト問題の仕様説明

4.1 概　要

ツールの性能を評価するためには、そのツールで提供される人工知能の概念の特性に即した、人工的、well-definedかつ小規模な問題が必要となる。現在、よく用いられる問題には、Monkey-Banana Problem 、農夫のジレンマなどがあるが、これらは、小規模すぎる、問題のサイズをパラメータで設定できない、などの欠点がある。そこで、我々は、推論機構の実行性能をルールシステム、フレームシステムについて評価できる単純な問題を設定した。さらに、エキスパートシステム化の要請が強い、解釈型、設計型、計画型の各問題について、その特性をよく表現する3つの問題を作成した。各問題の記述は、以下に示すとおり、名称、目的、問題の定義・解説、留意事項、例の項目からなる。

解釈型、診断型、設計型、計画型、シミュレーション型の各問題は、それぞれ人手で解くことができる程度のごく小規模なものであるが、いずれも、具体的な知識表現方法やルールは提示していないので、インプリメンタの能力とツールの性能とによってかなり異なる結果が得られるものと思われる。ただし、これらの問題は、いずれも条件を精密にしていけば実用上の問題にも適用できる概念を含んでおり、その意味では、テスト問題として適切なものといえよう。解釈型問題は特にパターンマッチングの知識をどう実現するかが課題となる。診断型問題は、ごく単純な分類問題となっている。これによって診断型問題のエキスパートシステム構築に必要な基本機能が提供されているかどうかが検討できよう。設計型問題は宣言的な知識の記述方法が課題となる。計画型問題は実行可能解をひとつ生成することを目的としているが、これに種々の制約を加えることで実用的なものとなり、また、解の評価を行う最適型の問題にも拡張することができる。シミュレーション型問題は厳密な仕様が与えられている点、インプリメント例が存在する点において有用と考えられる。

ルールシステムに関する実行性能評価問題は、ルール・条件の個数とルールの条件照合に要する時間との関係を測定することを目的としており、Reteシルゴリズムをコンパイラに採用しているツールの性能を知る場合に有用である。フレームシステムに関する実行性能評価問題は、システムがフレーム生成に必要な時間、ならびにシステムに存在するフレームの個数とそのアクセス時間との関係を継承なし／継承ありで測定できるよ

うに設定した。ただし、フレームシステムについては、Reteのような決定的なアルゴリズムが現在のところ存在しないので、設定した問題はツールのインプリメントに採用したアルゴリズムの差を検出できるほど精密なものではない。

これらの問題は、ツールの性能を比較評価する場合に有用であると同時に、知識処理技術やエキスパートシステムの教育時にも利用できるものである。

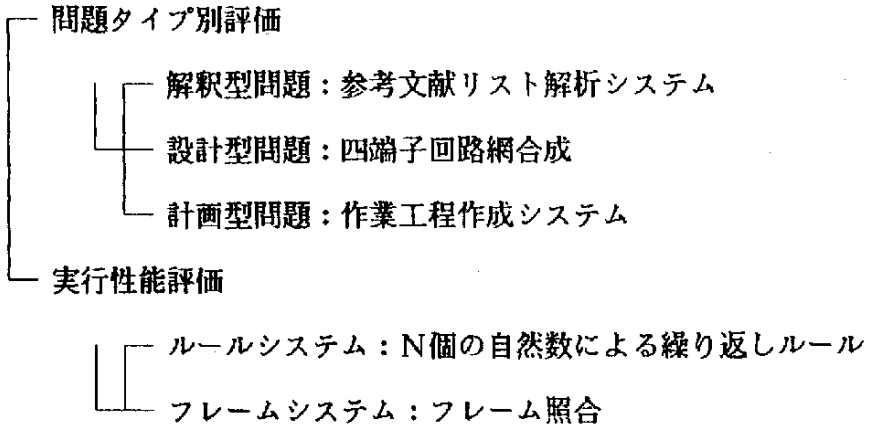


図4.1-1　性能評価用テスト問題の分類

寺野　隆雄（財電力中央研究所）

4.2　解釈型問題

名　称

参考文献リスト解析システム

目　的

文字列領域のデータについて、それに含まれる部分文字列の分類を行ない、全体の構造を抽出する。個々の分類カテゴリーに対応する特徴記述能力、および、それらの統合、競合解消機構の記述能力を試す。

問題の定義

論文の末尾に付されている参考文献リストを入力とし、文献データベースの内容のような、表題、著者名など項目別に整理した構造を出力する。

問題を単純化するために、処理の対象となるデータをつぎのように制限する。

①　1度に1つの文献を表わすデータのみを処理する。

②　単行本の1つ章を指すようなデータは対象外とする。

システムの仕様、および実現上の技術的要点は次のとおり。

(1)　与えられた文字列データに含まれる部分文字列を既知のカテゴリーに分類する。

カテゴリーとして、①著者名、②論文題目、③出版社名、④雑誌名、⑤学位名、⑥会議名、⑦報告書名、⑧機関名、⑨部門名、⑩巻、⑪号、⑫頁、⑬発行年、を考える。その他は無視してもよい。

(2)　1つのデータは、おおむね次の図4.2－1に示す構文に従うと考えてよい。

ただし、発行年は、著者名の後や巻の後などに来ることもある。また、途中に地名や発行月など上記以外の項目が挿入されている場合もある。

(3)　各カテゴリー毎に、データの特徴として、キーワード、少数種類の単語、文法などが使え、各カテゴリーに適した特徴把握手続きをコード化する必要がある。

各カテゴリーの特徴はおおよそ次のようなものである。

①　構文によるもの：著者名、論文題目、巻、号、頁、発行年

例えば、著者名の構文は図4.2－2のようなものと考えてよい。ただし、この構文は、よく使用される著者名の記述法を表現するに過ぎないという点に注意する必要

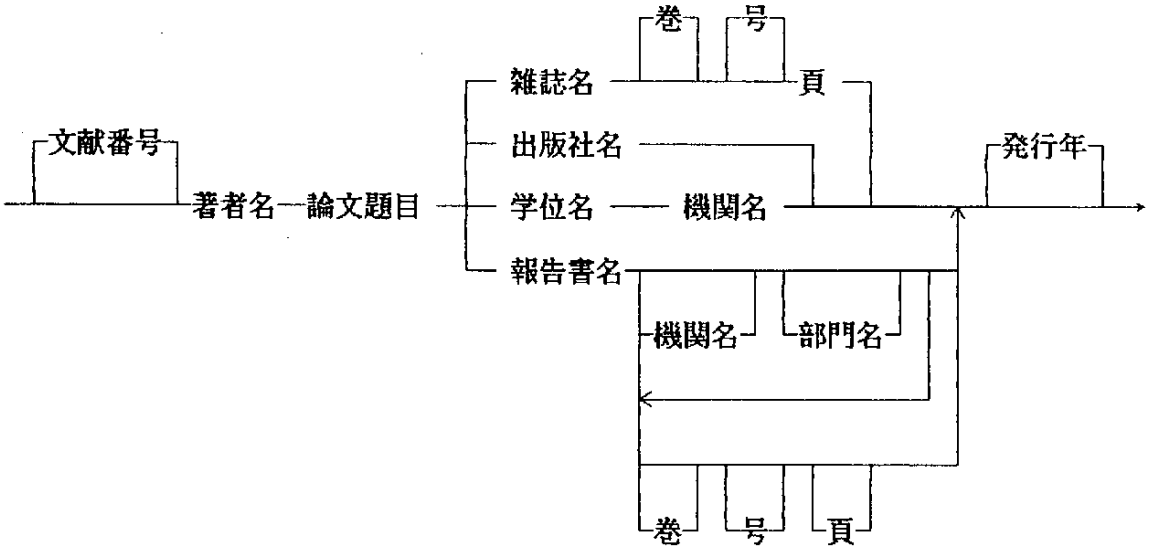


図4.2－1　つのデータの構文

著者名：

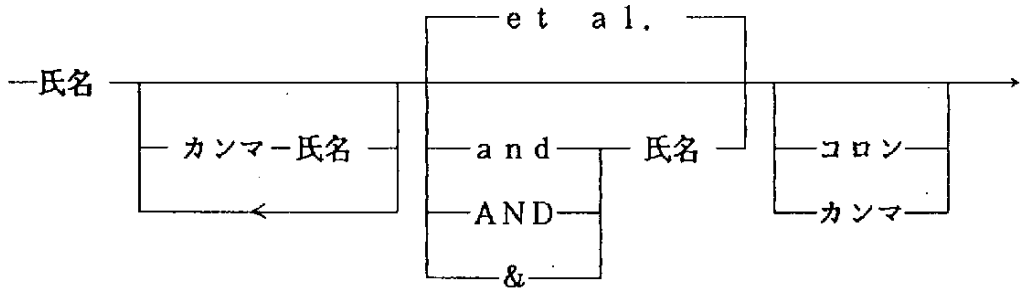


図4.2－2 著者名の構文の例

がある。この構文に従う文字列が必ず著者名になる、あるいは、この構文に従わないものは著者名ではないとは限らない。

② キーワードによるもの：学位名、会議名、報告書名、所属機関名、所属部門名

学位名：Thesis, Ph.D.. Mc.D.. Sc.D..…

会議名：Preceedings of Proc., Conf., Progress,…

報告書名：Report, Rept., TR.…

機関名：University, Univ., School, Institute, Laboratory, Lab., …

部門名：Department, Dept.,…

③ 辞書との照合によるもの：出版社名、雑誌名

データの中に出現する出版社名、雑誌名を辞書に登録しておき、照合する。

(4) 互いに類似の特徴を持つカテゴリー（例：地名と人名、頁と年号など）も存在するが、個々の特徴についての確からしさや全体の文脈で判断がつく場合が多い。

留意事項

解析の対象となる参考文献リストの形式は雑誌などによって様々なものがあるが、できる限り多くの種類の形式を解析可能とするほうが望ましい。

データの性質から考えて、自然言語の構文解析に用いられるような、探索に基づいて構文木を組み上げるといった手法よりは、個々の項目毎に、その特徴を満足する部分文字列を切り出すモジュールを起動し、相互の調整を取るような分散協調型の機構を採用する方が好ましい。

実現例としてＭＡＲＣＳ〔Kobayashi et al. 84〕を参考にするとよい。

例

以下は入出力の対の例である。

入力：〔1〕M.BLUM.R.W.FLOYD.V.R.PRATT.R.L.FIVEST AND R.E.TARIAN. "Time bounds for selection." J.Comp.Sys.Sci., 7(1972), pp.448-461.

⇨

出力：AUTHOR : M.BLUM R.W.FLOYD V.R.PRATT R.L.RIVEST R.E.TARIAN
TITLE : Time bounds for selection
YEAR : 1972
JOURNAL: Comp.Sys.Sci.
VOLUME : 7
PAGE : 448-461

入力：〔1〕Anderson.R.M. (1981) Core Theory with Strongly Convex Preferences. Econometrica 49 1457-1468.

⇨

出力：AUTHOR : R.M.Anderson
TITLE : Core Theory with Strongly Convex Preferences
YEAR : 1981

JOURNAL: Econometrica

VOLUME : 49

PAGE : 1457-1468

入力：〔1〕D.Ohne.Topics in nonlinear filtering theory.Sc.D.Thesis.
Massachusetts linstitute of Technology (1980).

⇨

出力：AUTHOR : D.Ohne

TITLE : Topics in nonlinear filtering theory

YEAR : 1980

TEHSIS : Sc.D.Thesis

ORGANIZATION : Massachusetts Insitute of Technology

入力：〔1〕P.S.Kirshnaprasad and S.I.Marcus.System identification and nonlinear filtering : Lie algebras. Proc. IEEE 20th Conf. on Desicion and Control. San Diego. CA. pp.330-334 (1981).

⇨

出力：AUTHOR : P.S.Kirshnaprasad S.I.Marcus

TITLE : System identification and nonlinear filtering : Lie algebras

YEAR : 1981

CONFERENCE : IEEE 20th Conf. on Decision and Control

PAGE : 330-334

入力：〔1〕J.A.Appels, et al., "Local oxidation of silicon and its application in semiconductor device technology." Phillips Research Reports, Vol.25. no.2, 1970, pp.118-132.

⇨

出力：AUTHOR : J.A.Appels, et al.

TITLE : Local oxidation of silicon and its application in semiconductor device technology

YEAR : 1970

REPORT : Phillips Research Reports

VOLUME : 25

NUMBER : 2

PAGE : 118-132

入力：〔1〕A.V.Fiacco and G.P.McCormick. Nonlinear Programing : Sequential Unconstrained Minimization Techniques. John Wiley and Sons (1968)

⇨

出力：AUTHOR : A.V.Fiacco G.P.McCormick

TITLE : Nonlinear Programing : Sequential Unconstrained Minimization Techniques

YEAR : 1968

PUBLISEHR : John Wiley and Sons

(担当　畝見　達夫)

参考文献

〔Kobayashi et al.84〕小林重信、畝見達夫、望月浩史、参考文献解析エキスパートシステム．情報処理学会．知識工学と人工知能研究会資料34－6 (1984)．

4.3　設計型問題

名　称

四端子回路網合成

目　的

要求仕様として与えたシステムの特性から、これ満足するシステムの構成要素の組み合わせ構造とその各構成要素の仕様の決定を行う簡単な設計型の問題において、ツールの機能・性能等の比較評価をする。

問題の定義

(1)　ゴール規則

要求仕様は、以下のシステムの特性を示す行列Ｆｓとして与えられる。

$$Fs = \begin{vmatrix} As & Bs \\ Cs & Ds \end{vmatrix}$$

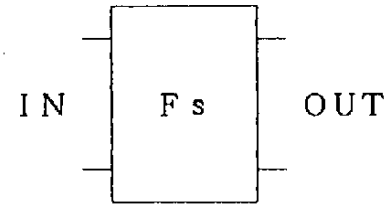


図4.3－1　システム

この行列Ｆｓを満足する、システムの構成とその構成要素を探索する。

ただし、

①　行列Ｆｓの各要素Ａｓ、Ｂｓ、Ｃｓ、Ｄｓの値は、ユーザ入力で与えられる実数とする。

②　要求仕様として与えられた行列Ｆｓ内の各要素Ａｓ、Ｂｓ、Ｃｓ、Ｄｓの値と設計されたシステムのそれらの間には、それぞれ５％以内の範囲で誤差を許容する。

③　求める構成は、構成要素数の最も少ないものとし、同一構成要素数のものが複数存在する場合は全て求め出力する。

④　結果の出力項目は、後述のシステムの各構成要素の仕様の値ｚとその構成法（直列・並列の区別）および構成順である。

(2) 構成要素の使用規則

使用出来る構成の仕様は、パラメータｚのみで定義される（図4.3－2）。このパラメータの値は、以下のいずれかの値をとる必要がある。

$Z = k * 10^n$

ここで、k＝2.7, 3.3, 3.9, 4.7, 5.6,
6.8, 8.2, 10, 12, 15, 18, 22

および、n＝0, 1, 2, 3, 4, 5, 6

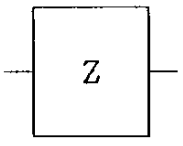


図4.3-2　構成要素

(3) 構成要素による部分システムの生成規則

1つの構成要素から、部分システムを構成出来る。この部分システムは、構成法によって、直列構成と並列構成に分けることが出来る（図4.3－3、図4.3－4）。

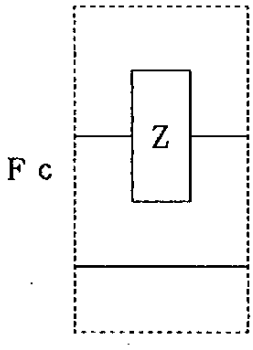


図4.3-3　直列構成による部分システム

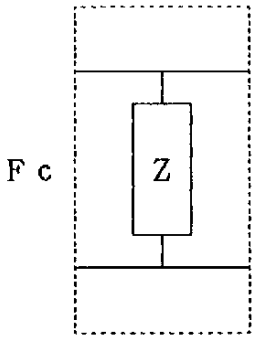


図4.3-4　並列構成による部分システム

それぞれの構成法から出来る部分システムの特性を示す行列Ｆｃは、構成要素の仕様パラメータｚとの間に以下の関係がある。

$$Fc = \begin{vmatrix} Ac & Bc \\ Cc & Dc \end{vmatrix}$$

のとき

① 直列構成の場合、Ａｃ＝1、Ｂｃ＝Ｚ、Ｃｃ＝0、Ｄｃ＝1

② 並列構成の場合、Ａｃ＝1、Ｂｃ＝0、Ｃｃ＝1／Ｚ、Ｄｃ＝1

(4) 部分システム間の合成規則

1列に構成された部分システムにより、システムは構成される。設計されたシステムの特性を示す行列は、部分システムの構成順に全ての部分システムの特性を示す行列の積を行ったものとなる。

したがって、もし部分システムuとv隣接して構成すれば、両者を合成した合成部分システムw（もしくはシステム）を構成できる（図4.3－5）。

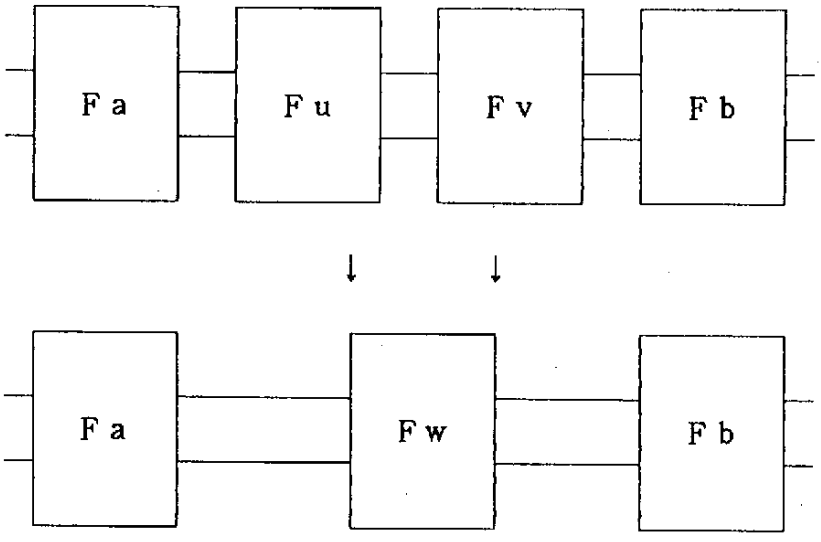


図4.3-5　部分システム間の合成

このとき合成部分システムwの特性を示す行列Fwは、部分システムuとvの特性を示す行列FuとFvの間に以下の関係がある。

$$Fw = Fu * Fv$$

$$Fw = \begin{vmatrix} Aw & Bw \\ Cw & Dw \end{vmatrix} = \begin{vmatrix} Au*Av+Bu*Cv & Au*Bv+Bu*Dv \\ Cu*Av+Du*Cv & Cu*Bv+Du*Dv \end{vmatrix}$$

留意事項

(1) 出来る限り設計知識は、宣言的かつ明示的に記述されなければならない。

例

(1) 要求仕様の入力数値例

実行に先立ち、要求仕様としてシステムの特性を示す行列Ｆｓ内の各パラメータの入力を要求する。このとき、代表値として以下の各パラメータ値の入力する。

Ａｓ＝1.3

Ｂｓ＝２７００

Ｃｓ＝0.0０１

Ｄｓ＝2.8

(2) 出力の例

求められたシステムの構成により、以下の表示例を行う。

構成の解　1

構成要素　1　入力端からの構成要素順　1　構成法　直列　仕様　Ｚの値

構成要素　2　入力端からの構成要素順　2　構成法　並列　仕様　Ｚの値

構成の解　2

構成要素　1　入力端からの構成要素順　1　構成法　直列　仕様　Ｚの値

構成要素　2　入力端からの構成要素順　2　構成法　並列　仕様　Ｚの値

4.4 計画型問題

名 称

作業工程作成システム

目 的

簡単な計画問題であるが条件を付け加える事によって十分実用的であり、解の探索問題としても多様な手法を考えさせる。

問題の定義

異なった五つの作業A、B、C、D、Eを異なった4地点P1、P2、P3、P4で同時に進めなければなりません、ユーザが与えた次の3条件を満たす各地点ごとの1日の作業工程表を作成するシステムを作ってください。

条件1……各地点ごとに、作業可能時間が設定されており、各作業はその範囲内でする。

条件2……各作業はそれぞれ1日に作業しなければならない時間数が指定されている。

条件3……異なった場所でどうじに並行して作業出来るかどうかを表わす作業テーブル表があり、その表でxが与えられている組み合わせは同じ時刻に作業が出来ない。

留意事項

(1) 各作業はできるだけ連続して進められるほうが望ましい。

(2) 与えられた条件で作業工程表が作成出来ない場合はそのようにメッセージを与えること。

(3) 現実の作業を想定し、望ましい解の条件を記述しそれを探索するヒューリスティックな手法をあたえよ。（例えば、全体の作業時間を最小にするなど）

例（入力データ）

(1) 各場所での作業可能時間帯

	0	8	12	13	18	24
P 1		----------		----------		
P 2			----------		------	
P 3		----------		----------		
P 4				----------		

(2) 各作業工程が各場所で必要とする時間

A＝1 h、B＝2 h、C＝1 h、D＝2 h、E＝1 h

(3) 各作業工程が並行して作業できるかどうかを与える作業テーブル

	A	B	C	D	E
A	X		X		
B		X			X
C	X		X		X
D				X	
E		X	X		X

これらの入力データをもとにシステムは（例えば）次の結果を出力する。

例（出力データ）

各地点での作業工程表

	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
P 1	A	C	B	B		D	D	E							
P 2			D	D	B	B		A	C	E					
P 3	D	D	A	C			E		E				B		
P 4							A	D	D		B	B	C	E	

4.5 診断型問題

名 称

ワインアドバイザ

目 的

ごくベーシックで古典的なコンサルテーション問題であるが、その解決を通じて、診断型のエキスパート・システム構築に必要な基本的機能が提供されているかを検討することができよう。

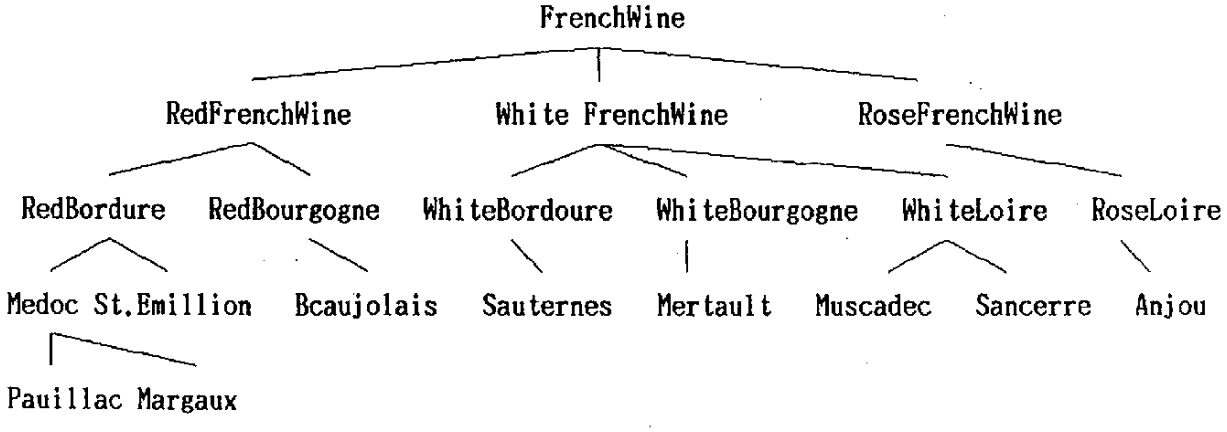
問題の定義

食事の内容に適したワインを選定し、アドバイスするシステムを構築する問題である。選定する上で必要な知識は、ワインの種類と性質、食事の内容および、ワインと食事の間を結び付ける関係である。本問題で与えられるのは、これらの知識とシステムの実行例である。

実行例のようにシステムを動作させるために、知識のデータ化および、ルール表現をどう与えるかを考えて欲しい。

1）ワインに関する知識

(1) ワインの分類



(2) ワインの属性

	Color	Taste	Strength
Pauillac	Red	Dry	Strong
Margaux	Red	Medium	Light
St.Emillion	Red	Medium	Strong
Beaujolais	Red	Dry	Strong
Sauternes	White	Sweet	Strong
Meursault	White	Dry	Light
Muscadet	White	Medium	Light
Sancerre	White	Dry	Light
Anjou	Rose	Medium	Light

2）食事に関する知識

(1) 食事の分類

Fish--halibit

meat-┬-fowl--chiken
　　└beaf-┬-steak
　　　　└veal

sauce--parmesan

(2) 食事の属性

	Color	Taste	Strength
fowl	White		
beaf	Red		
steak	Red		Strong
veal	White		Weak
Parmesan	Red	Spicy	Strong

3）ソースと主食との関係を現わす知識

(1) 好ましさを与える数値の計算例

その食事にあるワインの要素が、色－白、強さ－軽い、あわない要素が色－赤、味

－辛口の場合。

	White	Light	Red	Dry	Total
FrenchWine					0
RedFrenchWine			-3		-3
RedBordeaux			-3		-3
Medoc			-3		-3
Pauillac			-3	-2	-5
Margaux		+1	-3		-2
St.Emillion			-3		-3
RedBourgogone			-3		-3
Beaujolais			-3	-2	-5
WhiteFrenchWine	+3				+3
WhiteBordeau	+3				+3
Sauternes	+3			-2	+1
WhiteBourgogne	+3			-2	+1
Meursault	+3	+1			+4
WhiteLoire	+3	+1			+4
Mauscadet	+3	+1			+4
Sancerre	+3	+1		-2	+2
RoseFrenchWine					0
Roseloire					0
Anjou		+1			+1

(2) 規　則

規則1

ワインは主食とソースの味の強いほうで決定される。

規則2

ワインの色は食事の同じ色が望まれる。

規則3

ワインの色と食事の色が異なるのは望ましくない。

規則4

主食もソースもどちらも指示が強くなければ、ワインの味は強いほうがよい。

規則5

主食かソースかどちらか味の強いほうがあれば、ワインの味が軽いのは好ましくない。

規則6

ソースのスパイスが強い場合は甘いワインは好ましくない。

3）実行例

ユーザの回答はアンダーラインのついている部分であり、他はすべてシステムが与える。

食事は？　veal parmesan

MEAL-1に適したワインの色は赤です。

MEAL-1に適しないワインの色は白です。

MEAL-1に軽い味のワインは望ましくない。

MEAL-1に甘い味のワインは望ましくない。

〔FrenchWine 得点 0〕を考慮中

〔RedFrenchWine 得点 3〕を考慮中

〔RedBordeaux 得点 3〕を考慮中

〔Medoc 得点 3〕を考慮中

〔Pauillac 得点 3〕を考慮中

Pauillacはいかがです？　NO

〔RedBourgone 得点 3〕を考慮中

〔Beaujolais 得点 3〕を考慮中

Beaujolaisはいかがですか？　NO

〔St.Emilion 得点 3〕を考慮中

St.Emilionはいかがてすか？　YES

St.Emilionをお持ちします、しばらくお待ち下さい。

留意事項

問題自体としては極く基本的なものなので、問題が解けることにそれ程の意味や価値があるわけではないが、さらに、知識が増加した場合を考慮し、一般的な診断型問題に対する表現及び解決能力を見るという意味合いで、次のような点を考慮に入れることが必要である。即ち、

(1) 効率的な探索を行なう為には、問題の抽象化・構造化を計ることが必要である。つまり、食事の内容或いは個々の状況とワインとをダイレクトに関係付けるのは意味がない。

(2) 対話機能、特に説明機能を強化すると共に、説明のレベルを幾つか設定し、状況に応じて使い分けるといった工夫をする。通常の推論過程のトレースは情報量が多過ぎるため、却って把握できない。

引用文献

〔1〕HOWARD SCHIRE / DAVID BARSTOW, Tutorial No.14 AI PROGRAMMING IJCAI 1985

(脚注)

引用文献〔1〕に本問題のLispによるインプリメントが記述されている。

4.6　ルールの処理速度測定問題

名　称

N個の自然数による繰り返しルール

目　的

(1) ルールの個数および条件の個数と、ルールの条件照合に要する時間との関係を求める。

(2) 〃　ルールのコンパイル時間との関係を求める。

問題の定義

〔N個の自然数のうち連続昇順のM個の組合せを条件部に持ち、その結論部で条件部に含まれる先頭の自然数を含むデータを更新するN個のルールの組を実行せよ〕

留意事項

(1) ルールの条件部の指定で定数だけでなく、変数や構造化データを指定してもよい。
また、自然数の替わりにＡＡ、ＡＢ、ＡＣ等の連続昇順の文字列で記述してもよい。

(2) データの更新では条件部で照合する部分を更新しないこと。

(3) 処理の終了はルールのファイヤ数で1000、2000、4000を条件とする。
（ルールのファイヤ数＝1000×2^{j}、j＝1，2，3，…）

(4) 競合解決で選択される同時処理数が可変で多重世界を実現している場合、10未満を指定すること。

(5) 一般的な経験の単位を想定し、ルール数50、各ルールの条件記述が4個、としてN＝50、M＝4、同時処理数1、程度を指定する。

記述の例

（条件部に変数を記述しない単純な構造のデータの場合N＝50、M＝4）

```
(ルール1)        if      (1),(2),(3),(4)
                   then  modify (1)

(ルール2)        if      (2),(3),(4),(5)
                   then  modify (2)
    ⋮
(ルールi)        if      (i),(i+1),(i+2),(i+3)
                   then  modify (i)
    ⋮
(ルール50)       if      (50),(1),(2),(3)
                   then  modify (50)

(初期設定ルール) if      null
                   then  create (1)
                         create (2)
                           ⋮
                         create (N)
```

50個のルールの組

N個

但し、上記の例は以下の規則で記述される。また、初期設定ルールは独立に処理する。

① (i)、(j) はワーキング・メモリ内に整数iを値として持つデータと、整数jを値として持つデータとが共に存在する条件

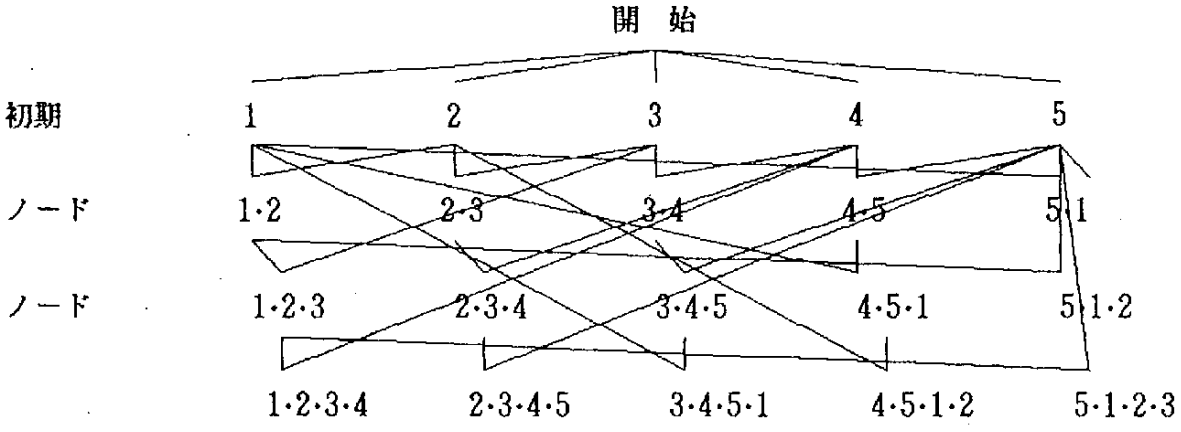
② create (i)は整数iを値として持つデータをワーキング・メモリ内に生成すること

③ modify (j)とは整数jを値として持つデータをワーキング・メモリ内で更新すること（なお、照合対象データは構造を持つ、更新では照合用の整数jは変更しない）

補足説明

(1) ルールの条件照合については、全ての条件とルールの条件部との照合が必要であり処理に時間がかかる。これを解決するためにReteネットワークが考案されているが、その実現法にはツールにより差があり処理時間が異なる。ここでは、〔目的〕で述べた関係をもとに、具体化の差を評価する。

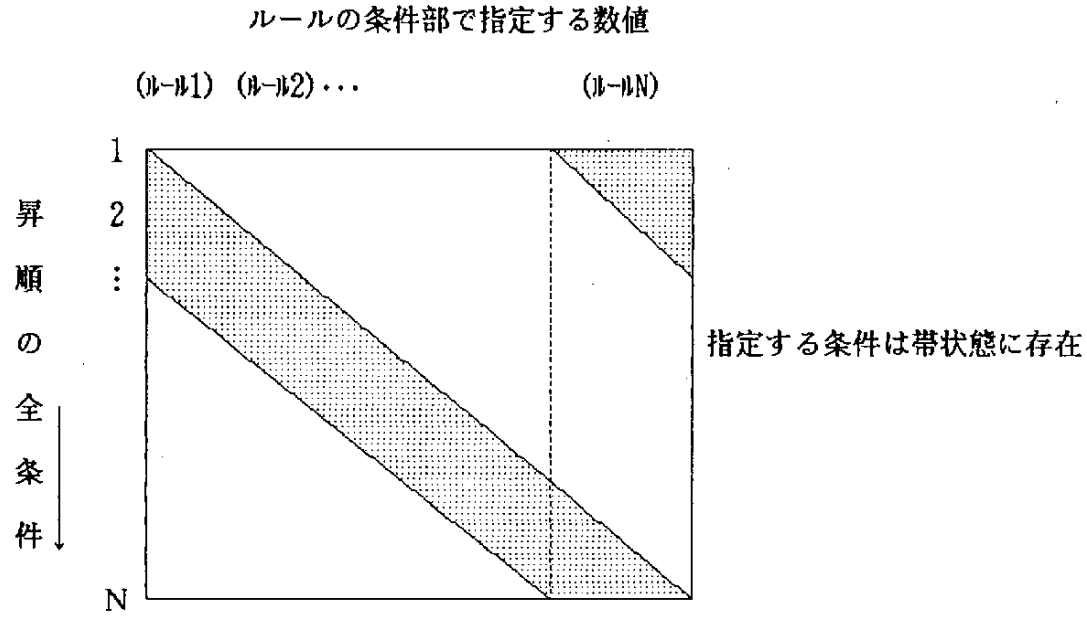
(2) Rete-networkの構成の例（N＝5、M＝4の場合）



(3) N個の自然数のうち任意のM個の組合せを条件部に持つ場合も想定できるが、この場合のルール数は ${}_{N}C_{M}$ 個と非常に多いため注意が必要である。

(4) ハイブリッド型のツールでは、各自然数に対応するデータがフレームとして構造を持ち、かつ、それぞれがL個ずつ存在する場合をも実現できるが、通常はRete-networkは変わらない。

(5) 全条件とルールで指定する条件との関係は、以下の様な関係にある。



(6) ルールとフレーム（照合対象はインスタンス）を利用し、条件部が変数となる具体的な記述例を示す。なお、"(frame"はフレーム条件を、"-->"は先が実行部を表す。

```
(初期設定ルール)
   --> (create 枠 (定数 1)  (変数 0)
       (create 枠 (定数 2)  (変数 0)
                  ⋮
       (create 枠 (定数 N)  (変数 0)

(ルール1)
   (frame (枠 ?name (定数 1)  (変数 ?x&(< stoper)))
   (frame (枠 ?     (定数 2)
   (frame (枠 ?     (定数 3)
   (frame (枠 ?     (定数 4)
   -->
   (modify ?name  (変数 !(+ 1 ?x)))
  ⋮
(ルールi)
   (frame (枠 ?name (定数 i)  (変数 ?x&(< stoper)))
   (frame (枠 ?     (定数 i+1)
   (frame (枠 ?     (定数 i+2)
   (frame (枠 ?     (定数 i+3)
   -->
   (modify ?name  (変数 !(+ 1 ?x)))
```

N個のインスタンスの生成

N個のルール

但し、

$$stoper = \frac{\text{ルールのファイヤ回数}}{N}$$

フレームの構成

枠	
定数	1～N
変数	初期0

〔担当　桜井〕

4.7　フレームの処理速度測定問題

名　称

フレーム照合

目　的

(1)　システムに存在するフレームの個数と、フレーム照合に要する時間との関係を調べる。

(2)　インヘリタンスを含んだ場合の照合に要する時間を測定し、(1)と比較する。

問題の定義

(1)　問題1

①　0－99の整数を値としてとりうるスロット（スロット名をRand-numとする）を持つフレームを定義する。（レベル　0）

②　①で定義したフレームを親とする、0－99の整数乱数をRand-numスロットの値としてもつフレームをN個発生させる。フレームの発生に要した時間を測定する。（レベル　1）

③　②で生成したフレームについて、与えられた整数をRand-numスロットの値としてもつようなフレームを全て検索し、検索に要した時間を測定する。

Nの値は500 、1000、2000、4000とする。

（注　$N=500\times 2^{M}$，M＝1，2，3，…）

(2)　問題2

①　(1)－②で発生させた各々のフレームの下に、Rand-numスロットの値を継承するフレームを発生させ、その時間を測定し(1)と比較する。（レベル　2）

②　①で発生させたフレーム（レベル2のフレーム）について、(1)－③と同様に与えられた整数をRand-numスロットの値としてもつようなフレームを全て検索し、検索に要した時間を測定し、(1)と比較する。

(3)　問題3

①　(2)－①で発生させた各々のフレームの下に、Rand-numスロットの値を継承するフレームを発生させ、その時間を測定し(1)(2)と比較する。（レベル　3）

②　①で発生させたフレーム（レベル3のフレーム）について、(1)－③と同様にフレー

ムを検索し、検索に要した時間を測定して、(1)・(2)と比較する。

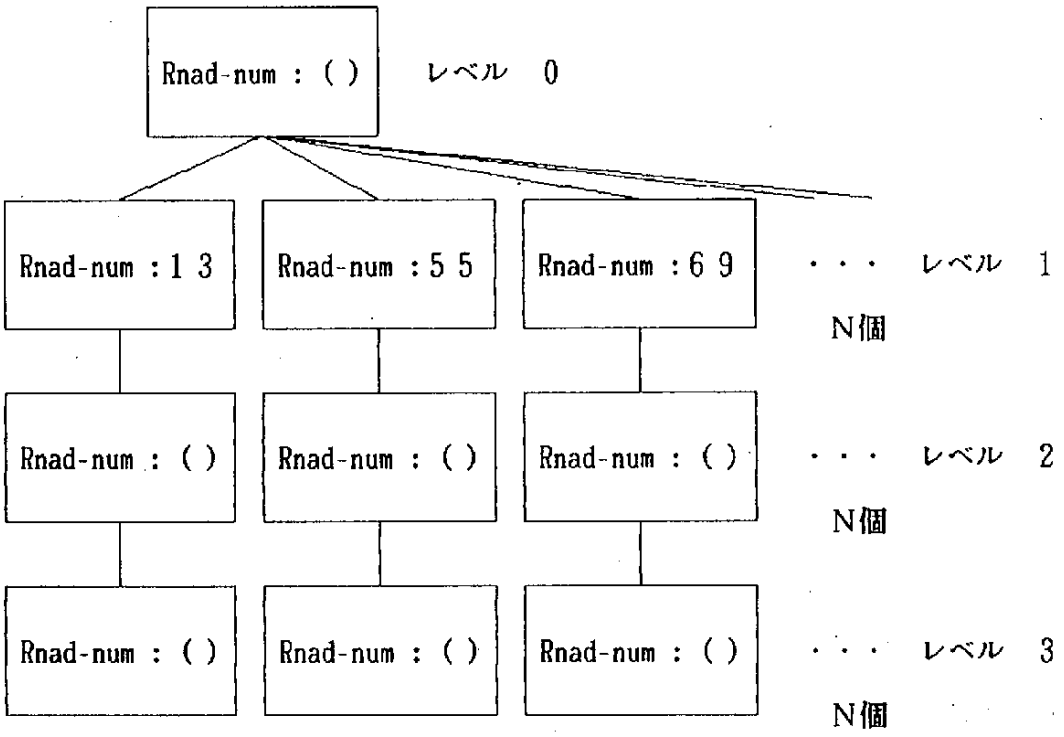
留意事項

(1) フレームの検索にはパターン・マッチ機能を用いること。

(2) 性能評価に用いたツール／ハードウェアの組合せにより、Nの値は適当に増減させ、目的とする比較評価ができるようにする。

例

問題のフレーム構造



5．基本設計と補足説明

5．基本設計と補足説明

5．1　基本設計方針

（1）各テスト問題には明確な出題意図があり、その内容に従ったものとする。

（2）推論などのシステムの機能部分は、本書にて定義するルール、およびフレームによる実現を基本とする。

（3）ツールに依存した知識表現を使用することで効率のよいシステムが実現できる場合は積極的に利用する。

（4）ユーザーインターフェースなどのシステムの副作用部分は、処理系のシステム記述言語（ＬＩＳＰ、ＰＲＯＬＯＧ、Ｃ言語など）によって実現する。

5．2　テスト問題と出題意図

本インプリメンテーションの対象とするテスト問題は、実行性能評価問題と問題タイプ別評価問題の２つに分けられる。前者については問題自体がインプリメントの基本構成を定めているので、ここでは言及しない。後者はさらに、解釈型、設計型、計画型、診断型の４つの問題を含んでいる。これらは、既存のエキスパートシステムの構成を参照することができ、典型的な構成方法によって構築するものとする。その概要を次に示す。

（1）解釈型問題：参考文献リスト解析システム

音声、画像などの信号レベルの解析から段階を経て意味的な内容を取り出すものがある。本問題においても、文字列→単語→項目→カテゴリーといった段階を認識するルール群と推論制御によって同様な方法をとる。

（2）設計型問題：四端子網合成

推論システム単独で設計問題を解くよりも、従来型のプログラムによって構築された設計システムの補完をする場合が多い。

多くの要素とその組合せにおいて、網羅的判定を行う中で、組合せの爆発の生ずるのを防ぐためのルールの適用を行う場合がある。あるいは、数値計算などの適用方法にルール

を使用したり、計算結果の補正などにルールを使う場合もある。本問題においては、離散的な数値を網羅的に適用してゆく場合に、適用する数値の枝切り、精度補正などにルールを使用する。

（３）計画型問題：作業工程作成システム

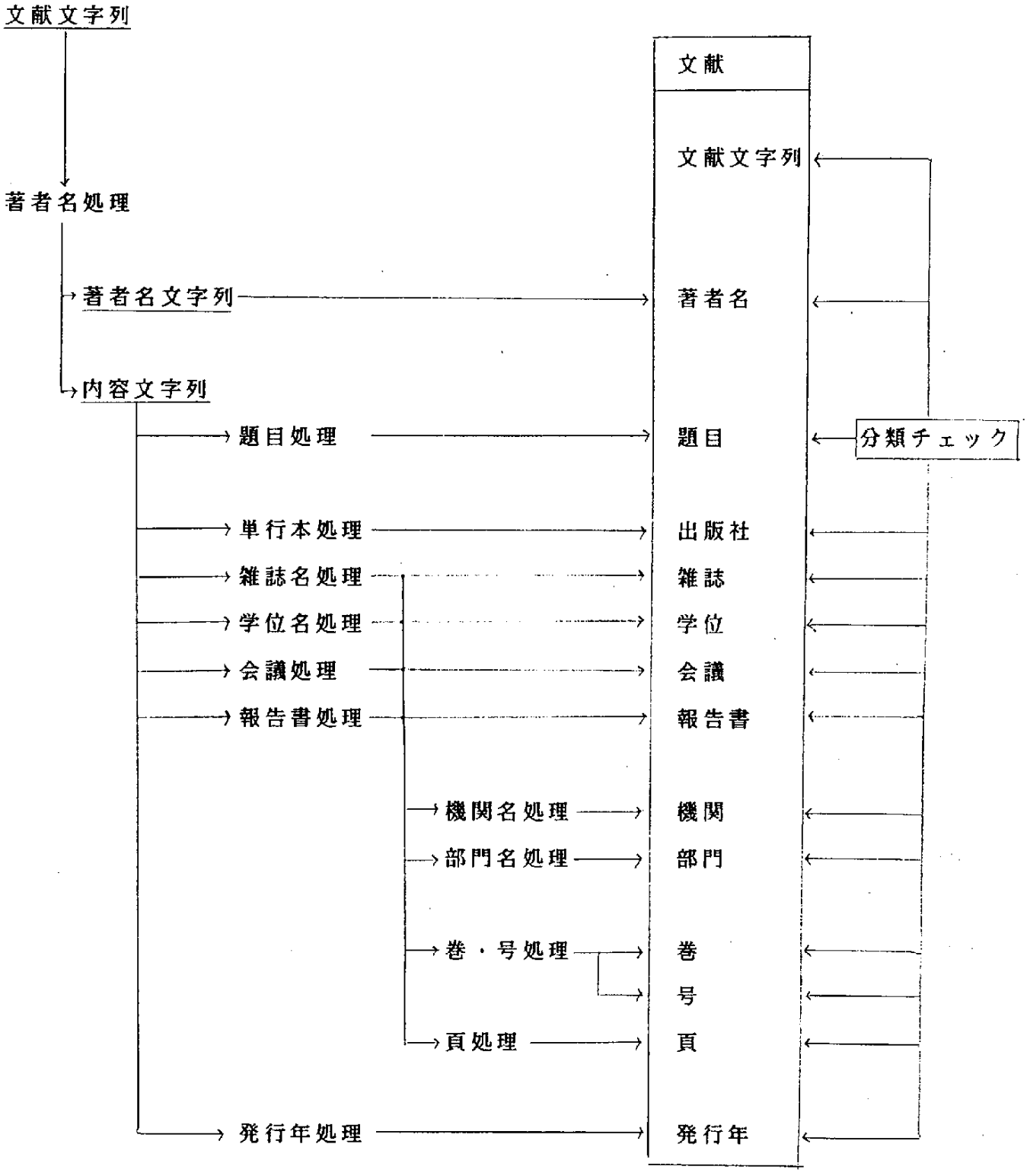
設計型問題と類似する部分が多いが、構成要素が日程および時間（期間）などのデータに限定されている。本問題においては、制約条件の固定的なものから優先度の高いもの、低いものへという組合せの決定順序をルール化する。

（４）診断型問題：ワイン・アドバイザ

数多くの診断システムの構築例があるが、推論形態として大ざっぱに分けると症状から原因の追求を行うもの、仮説－検証型の２通りがある。本問題では、予め症状（入力データ）から中間仮説を導出し、診断結果候補の評価を行う。

５．３　各テスト問題における基本構成

５．３．１　解釈型問題：参考文献リスト解析システム



（１）ルール

<<制御ルール>>

<<文字処理>>

参考文献の文字列を１文字づつ読み、文字ブロック（単語）に分ける

文字ブロックは、次に示す '単語クラス' のインスタンスである。

フレーム

名称：単語

スロット：

文字列

前単語（前の単語がないとき

－単語が文字列の最初の単語のとき－には、START）

次単語（次の単語がないとき

－単語が文字列の最後の単語のとき－には、END）

前区切り文字（前の単語がないとき

－単語が文字列の最初の単語のとき－には、START）

次区切り文字（次の単語がないとき

－単語が文字列の最後の単語のとき－には、END）

例：文字列が、"this is a test." のとき

FRAME start-word

instance : 単語

文字列 : "this"

前単語 : start

次単語 : word1

前区切り文字 : start

次区切り文字 : #¥space

FRAME word1

instance : 単語

文字列 : "is"

前単語 : start-word

次単語 : word2

前区切り文字 : #¥space

次区切り文字 : #¥space

FRAME word2

instance : 単語

文字列 : "a"

前単語 : word1

次単語 : word3

前区切り文字 : #¥space

次区切り文字 : #¥space

FRAME word4

instance : 単語

文字列 : "test"

前単語 : word3

次単語 : end

前区切り文字 : #¥space

次区切り文字 : end

① 文字読み処理終了

最後の文字を読んでしまっていた。

IF SYSTEM の POINTER が ?N

?N = SYSTEM の STRING.LENGTH

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

THEN ?CW の次単語が END

?CW の次区切りが END

② 開き記号がなければ、区切り文字を読み取ったところで単語化

IF　SYSTEM の POINTER が ?N

?N が SYSTEM の STRING.LENGTH より小さい

開き記号（SYSTEM の OPEN.MARK）がない

?N 番目の文字が ?C

?C が開き記号ではない

?C が 区切り文字（CHAR.PROCESS の DELIMITER）

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

?M = ?N + 1

THEN 新単語 ?NEW を生成

?CW の次単語が ?NEW

?CW の次区切り文字が ?C

SYSTEM の CURRENT-WORD が ?NEW

SYSTEM の POINTER が ?M

③　ただし、単語にすべき文字リストがなければ、読み飛ばす

IF　SYSTEM の POINTER が ?N

?N が SYSTEM の STRING.LENGTH より小さい

開き記号（SYSTEM の OPEN.MARK）がない

?N 番目の文字が ?C

?C が開き記号ではない

?C が 区切り文字（CHAR.PROCESS の DELIMITER）

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

?CW の 文字列 がない(NIL)

?M = ?N + 1

THEN SYSTEM の POINTER が ?M

④　開き記号があれば閉じ記号（')'　'"'）で単語化

```
IF   SYSTEM の POINTER が ?N
     ?N が SYSTEM の STRING.LENGTH より小さい
     開き記号（SYSTEM の OPEN.MARK） が ?MARK
     ?N 番目の文字が ?C
     ?C = ?MARK に対応する CLOSE-MARK
     SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW
     ?W = (?CWの文字列)+?C
     ?M = ?N + 1
THEN 新単語 ?NEW を生成
     ?CW の文字列が ?W
     ?CW の次単語が ?NEW
     ?CW の次区切り文字が #¥,
     SYSTEM の CURRENT-WORD が ?NEW
     開き記号（SYSTEM の OPEN.MARK）の値を削除
     SYSTEM の POINTER が ?M
```

⑤ 開き記号（'(' '"'）があれば、閉じ記号まで読み続ける

```
IF   SYSTEM の POINTER が ?N
     ?N が SYSTEM の STRING.LENGTH より小さい
     開き記号（SYSTEM の OPEN.MARK）が ?MARK
     ?N 番目の文字が ?C
     ?C != ?MARK に対応する CLOSE-MARK
     SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW
     ?M = ?N + 1
THEN ?CW の文字列に ?C を加える
     SYSTEM の POINTER が ?M
```

⑥ 開き記号（'(' '"'）がなければ、区切り文字まで読み続ける

```
IF   SYSTEM の POINTER が ?N
     ?N が SYSTEM の STRING.LENGTH より小さい
```

開き記号（SYSTEM の OPEN.MARK）がない

?N 番目の文字が ?C

?C が開き記号ではない

?C が 区切り文字（CHAR.PROCESS の DELIMITER）でない

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

?M = ?N + 1

THEN ?CW の文字列に ?C を加える

SYSTEM の POINTER が ?M

⑦ 開き記号が現れたら、そこまでの文字リストを単語化

IF SYSTEM の POINTER が ?N

?N が SYSTEM の STRING.LENGTH より小さい

開き記号（SYSTEM の OPEN.MARK） がない

?N 番目の文字が ?C

?C が 開き記号 である

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

?CW の文字列がある（NIL でない）

?M = ?N + 1

THEN 新単語 ?NEW を生成

?NEW の文字列が ?C

?CW の次単語が ?NEW

?CW の次区切り文字が #¥,

SYSTEM の CURRENT-WORD が ?NEW

開き記号（SYSTEM の OPEN.MARK）は ?C

SYSTEM の POINTER が ?M

⑧ 開き記号が現れたが、前の区切りからそこまでの間に単語がない

IF SYSTEM の POINTER が ?N

?N が SYSTEM の STRING.LENGTH より小さい

開き記号（SYSTEM の OPEN.MARK）がない

?N 番目の文字が ?C

?C が 開き記号 である

SYSTEM の CURRENT-WORD（現在処理中の単語）が ?CW

?CW の 文字列 がない (NIL)

?M = ?N + 1

THEN ?CW の文字列は ?C

開き記号（SYSTEM の OPEN.MARK） は ?C

SYSTEM の POINTER が ?M

<<文献番号>>

参考文献の文字列は、著者名から始まっている。ただし、はじめに文献番号がつくことがある。

① 頭に数字があれば文献番号

IF START-WORD の 文字列 が ?W

?W は数字文字列または開き記号と閉じ記号に囲まれた数字文字列

START-WORD の 次単語 が ?NW

THEN 文献 の 文献番号 は ?W である

?NW に著者名ラベルを付ける

② 数字でなければ著者名

IF START-WORD の 文字列 が ?W

NOT (?W は数字文字列または開き記号と閉じ記号に囲まれた数字文字列)

THEN START-WORD に著者名ラベルを付ける

<<著者名>>

① 氏名は ','(カンマ)で区切られる。

IF 著者名ラベルが ?W に付いている

?W の文字列が ?X

?W の次区切り文字が ' '

?W の次単語が ?NEXT

?X は氏名文字列

THEN 著者名は ?X

著者名ラベルを ?NEXT に付けかえる

② <氏名１> AND <氏名２>

著者名の文字ブロックは<氏名２>で終わり

IF 著者名ラベルが ?W に付いている

?W の文字列が ?X

?W の前区切り文字が ','

?W の次区切り文字が ' '

?W の次単語が ?NEXT

?X は氏名文字列

?NEXT の文字列が (AND または and または &)

?NEXT の前区切り文字が ' '

?NEXT の次区切り文字が ' '

?NEXT の次単語が ?NEXT2

?NEXT2 の文字列が ?Y

?NEXT2 の前区切り文字が ' '

?NEXT2 の次単語が ?NEXT3

?Y は氏名文字列

THEN 著者名は ?X

著者名は ?Y

著者名ラベルを ?NEXT3 に付けかえる

③ 著者名を <姓>,<名のイニシャル> と書く場合

IF 著者名ラベルが ?W に付いている

?W の文字列が ?X

?W の前区切り文字が ','

?W の次区切り文字が ','

?W の次単語が ?NEXT

?X は姓文字列

?NEXT の文字列が ?Y

?NEXT の前区切り文字が ' '

?NEXT の次区切り文字が ' '

?NEXT の次単語が ?NEXT2

?Y はイニシャル文字列

THEN 著者名は ?Y+?X

著者名ラベルを ?NEXT2 に付けかえる

④

IF 著者名ラベルが ?W に付いている

?W の文字列が (AND または and または &)

?W の前区切り文字が ' '

?W の次区切り文字が ' '

?W の次単語が ?NEXT

?NEXT の文字列が ?X

?NEXT の前区切り文字が ' '

?NEXT の次区切り文字が ','

?NEXT の次単語が ?NEXT2

?X は姓文字列

?NEXT2 の文字列が ?Y

?NEXT2 の前区切り文字が ','

?NEXT2 の次単語が ?NEXT3

?Y はイニシャル文字列

THEN 著者名は ?Y+?X

題目ラベルを ?NEXT3 に付ける

⑤ ':'(コロン)で著者名の文字ブロックは終わり

IF　著者名ラベルが ?W に付いている

　　?W の前区切り文字が ':'

THEN 題目ラベルを ?W に付ける

⑥ "et al." は最後の著者名

IF　著者名ラベルが ?W に付いている

　　?W の文字列が "et"

　　?W の前区切り文字が ','

　　?W の次区切り文字が ' '

　　?W の次単語が ?NEXT

　　?NEXT の文字列が "al."

　　?NEXT の前区切り文字が ' '

　　?NEXT の次区切り文字が (',' または ':')

　　?NEXT の次単語が ?NEXT2

THEN 著者名は "et al."

　　題目ラベルを ?W に付ける

⑦ 著者名の終わり

IF　著者名ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は氏名文字列でも姓文字列でもイニシャル文字列でも"et"でもない

　　?W の前区切り文字が ','

THEN 題目ラベルを ?W に付ける

⑧ 著者名の終わり

IF　著者名ラベルが ?W に付いている

　　?W の文字列が ?X

　　?W の次区切り文字が ','

　　?W の次単語が ?NEXT

　　?X は姓である

?NEXT の文字列が ?Y

NOT (?Y はイニシャルである)

THEN 題目ラベルを ?W に付ける

<<題目処理>>

著者名の次は題目。ただし、著者名と題目の間に発行年がくることがある。

①

IF 題目ラベルが ?W に付いている

?W の文字列が ?X

?X が数字文字列または開き記号と閉じ記号に囲まれた数字文字列

?W の次単語が ?NEXT

THEN 文献の発行年は ?X(SCORE : 8)

題目ラベルを ?NEXT に付けかえる

②

IF 題目ラベルが ?W に付いている

?W の文字列が ?X

?W の前区切り文字が ?D

?W の次区切り文字が ','

?W の次単語が ?NEXT

THEN 文献の題目に ?D+?X を付け加える

?NEXT にその他ラベルを付ける

③

IF 題目ラベルが ?W に付いている

?W の文字列が ?X

?W の次区切り文字が ?D

?D != ','

?W の次単語が ?NEXT

THEN 文献の題目に ?D+?X を付け加える

題目ラベルを ?NEXT に付けかえる

<<その他処理規則>>

文献番号，著者名，題目を切り出した後の残りの単語にその他ラベルを付ける

①

IF　その他ラベルが ?W に付いている
　?W の次単語が "end"
THEN <<その他処理>>

②

IF　その他ラベルが ?W に付いている
　?W の次単語が ?NEXT
THEN その他ラベルを ?NEXT に付ける

<<雑誌処理規則>>

①

IF　その他ラベルが ?W に付いている
　?W の文字列が ?X
　?X が雑誌のインスタンス
THEN 文献の雑誌名が ?X
　?X に雑誌ラベルを付ける
　<<巻処理>>

②

IF　その他ラベルが ?W に付いている
　雑誌のインスタンスが ?MG
　雑誌のキーワードが ?KEY
　?W の文字列が ?X
　?X = ?KEY+?MG
THEN 文献の雑誌名が ?X

?X に雑誌ラベルを付ける

<<巻処理>>

<<単行本処理規則>>

IF　その他ラベルが ?W に付いている

?W の文字列が ?X

?X が出版社のインスタンス

THEN 文献の出版社名が ?X

?X に出版社ラベルを付ける

<<学位論文処理規則>>

IF　その他ラベルが ?W に付いている

学位のキーワードが ?KEY

?W の文字列が ?X

?X は ?KEY で始まる

THEN 文献の学位名が ?X

?X に学位ラベルを付ける

<<報告書処理規則>>

報告書のキーワードがあれば、その前後の単語を（カンマで区切られたところまで）結合したものが報告書名である。

IF　その他ラベルが ?WORD に付いている

?WORD の文字列は ?STRING

報告書のキーワードは ?KEY

?STRING = ?KEY

THEN 前後の単語を（カンマまで）結合したものが報告書名である

?X に報告書ラベルを付ける

<<巻処理>>

<<会議処理規則>>

会議のキーワードがあれば、その前後の単語を（カンマで区切られたところまで）結合したものが会議名である。

IF　その他ラベルが ?WORD に付いている
　　?WORD の文字列は ?STRING
　　会議 の キーワード は ?KEY
　　?STRING = ?KEY
THEN 前後の単語を（カンマまで）結合したものが会議名である
　　?X に会議ラベルを付ける
　　<<巻処理>>

<<機関名処理規則>>

機関のキーワードがあれば、その前後の単語を（カンマで区切られたところまで）結合したものが機関名である。

IF　その他ラベルが ?WORD に付いている
　　?WORD の文字列は ?STRING
　　機関 の キーワード は ?KEY
　　?STRING = ?KEY
THEN 前後の単語を（カンマまで）結合したものが機関名である

<<部門名処理規則>>

部門のキーワードがあれば、その前後の単語を（カンマで区切られたところまで）結合したものが部門名である。

IF　その他ラベルが ?WORD に付いている
　　?WORD の文字列は ?STRING
　　部門 の キーワード は ?KEY
　　?STRING = ?KEY
THEN 前後の単語を（カンマまで）結合したものが部門名である

<<巻処理規則>>

①

IF　その他ラベルが ?W に付いている

　　巻のキーワードが ?KEY

　　?W の文字列が ?X

　　?X は ?KEY で始まる数字文字列

THEN 文献の巻が ?X(8)

　　?X に巻ラベルを付ける

　　<<号処理>>

②

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 数字文字列

　　?W の前単語が ?PRE

　　?PRE に雑誌／学位／報告書／会議のラベルが付いている

THEN 文献の巻が ?X(5)

　　?X に巻ラベルを付ける

　　<<号処理>>

③

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 数字文字列

　　?W の前単語が ?PRE

　　NOT(?PRE に雑誌／学位／報告書／会議のラベルが付いている)

THEN 文献の巻が ?X(2)

　　?X に巻ラベルを付ける

　　<<号処理>>

<<号処理規則>>

①

IF　その他ラベルが ?W に付いている
　　号のキーワードが ?KEY
　　?W の文字列が ?X
　　?X は ?KEY で始まる数字文字列
THEN 文献の号が ?X(8)

②

IF　その他ラベルが ?W に付いている
　　?W の文字列が ?X
　　?X は数字文字列
　　?W の前単語が ?PRE
　　?PRE に巻のラベルが付いている
THEN 文献の号が ?X(5)

③

IF　その他ラベルが ?W に付いている
　　?W の文字列が ?X
　　?X は数字文字列
　　?W の前単語が ?PRE
　　NOT(?PRE に巻のラベルが付いている)
THEN 文献の号が ?X(2)

<<頁処理規則>>

①

IF　その他ラベルが ?W に付いている
　　ページのキーワードが ?KEY
　　?W の文字列が ?X
　　?X は ?KEY で始まるページ文字列
THEN 文献のページが ?X(8)

②

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 数字文字列でないページ文字列

THEN 文献のページが ?X(5)

③

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 数字文字列

THEN 文献のページが ?X(2)

＊ ページ文字列とは

(1) 数字だけの文字列（数字文字列）

(2) ２つの数字文字列をハイフン(-)で結び付けた文字列

(3) (1),(2)の文字列の最後にピリオド(.)を付けた文字列

<<発行年処理規則>>

①

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 開き記号と閉じ記号で囲まれた数字文字列

THEN 文献の発行年が ?X(5)

②

IF　その他ラベルが ?W に付いている

　　?W の文字列が ?X

　　?X は 数字文字列

THEN 文献の発行年が ?X(2)

（２）フレーム

名称 ： SYSTEM

スロット ： STRING（文献文字列）

: STRING.LENGTH

: POINTER（初期値 ： 0）

: OPEN.MARK

: current-word

名称 ： CHAR.PROCESS

スロット ： DELIMITER（区切り文字 ： 初期値 ： #¥, #¥SPACE #¥:）

BRACKETS(初期値 ： (#¥(#¥)),(#¥" #¥"),(#¥[#¥]))

OPEN.BRACKETS.P(METHOD)

CLOSE.BRACKETS(METHOD)

名称：文献

スロット：

文献番号

著者名

題目

出版社名

雑誌名

学位名

会議名

報告書名

機関

部門

巻

号

頁

発行年

名称：出版社

名称：John_Wiley_and_Sons
instance of 出版社

名称：雑誌
スロット：
　キーワード（J.）

名称：Comp.Sys.Sci.
instance of 雑誌

名称：Econometrica
instance of 雑誌

名称：学位論文
スロット：
　キーワード（Thesis Ph.D. Mc.D. Sc.D.）

名称：会議
スロット：
　キーワード（Preceedings of, Proc., Conf., Progress）

名称：報告書名
スロット：
　キーワード（Report, Rept., TR.）

名称：機関名
スロット：

キーワード（University, Univ., School, Institute, Laboratory, Lab.）

名称：部門名

スロット：

キーワード（Department, Dept.）

名称：巻

スロット：

キーワード（Vol.）

名称：号

スロット：

キーワード（no.,No.）

名称：頁

スロット：

キーワード（pp.,PP.,p.,P.）

５．３．２　設計型問題：四端子網合成

（１）ルール

```
rule1
      if  (hypothese の size が 0)
      then(hypotheses のインスタンス ?H0 を生成)
          (?H0 の size は 0)
          (?H0 の param_a は 1.0)
          (?H0 の param_b は 0.0)
          (?H0 の param_c は 0.0)
          (?H0 の param_d は 1.0)
          (hypotheses の size は 1)

rule2
      if  (hypotheses の インスタンス が ?H)
          (hypotheses の size が ?N)
          (?H の size が (- ?N 1))
          (p_element のサブクラスが ?P)
          (?P の param_a が ?A)
          (?P の param_b が ?B)
          (?P の param_c が ?C)
          (?P の param_d が ?D)
          (?H の param_a が ?HA)
          (?H の param_b が ?HB)
          (?H の param_c が ?HC)
          (?H の param_d が ?HD)
          (goal の param_a が ?GA)
          (<= ?HA ?GA*1.05)
          (goal の param_b が ?GB)
```

```
        (<= ?HB ?GB*1.05)
        (>= ?HA*?B ?GB*0.05)
        (<= ?HA*?B ?GB*1.05-?HB)
        (goal の param_c が ?GC)
        (<= ?HC ?GC*1.05)
        (>= ?HD*?C ?GC*0.05)
        (<= ?HD*?C ?GC*1.05-?HC)
        (goal の param_d が ?GD)
        (< ?HD ?GD*1.05)
    then(hypothese のインスタンス ?HH を生成)
        (?HH の size は ?N)
        (?HH の param_a は (?HA * ?A + ?HB * ?C))
        (?HH の param_b は (?HA * ?B + ?HB * ?D))
        (?HH の param_c は (?HC * ?A + ?HD * ?C))
        (?HH の param_d は (?HC * ?B + ?HD * ?D))
        (?HH の p_element は ?P)
        (?HH の previous は ?H)
        (?HH を次候補とする)

rule3
    if  (?H を次候補とする)
        (?H の param_a が ?A)
        (goal の param_a が ?GA)
        (?A >= ?GA * 0.95)
        (?A <= ?GA * 1.05)
        (?H の param_b が ?B)
        (goal の param_b が ?GB)
        (?B >= ?GB * 0.95)
        (?B <= ?GB * 1.05)
        (?H の param_c が ?C)
```

```
        (goal の param_c が ?GC)
        (?C >= ?GC * 0.95)
        (?C <= ?GC * 1.05)
        (?H の param_d が ?D)
        (goal の param_d が ?GD)
        (?D >= ?GD * 0.95)
        (?D <= ?GD * 1.05)
    then(goal の result は ?H)

rule4
    if  (goal の result は ?H)
        (hypotheses の status は unsuccess)
    then(hypotheses の status は success)

rule5
    if  (hypothese の size が ?N)
        (>= ?N 1)
        (hypothese の status が unsuccess)
    then(hypothese の size は ?N+1)

rule6
    if  (hypotheses の size が ?N)
        (>= ?N 5)
    then(goal の status は finished)
```

（２）フレーム

① 名称：networks

スロット：param_a

param_b

param_c

param_d

② 名称：goal

上位フレーム：networks

③ 名称：hypotheses

スロット：size（初期値：0）

status（初期値：unsuccess）

p_element

previous

上位フレーム：networks

④ 名称：elements

スロット：k,n,connection

⑤ 名称：k_elements

上位フレーム：elements

⑥ 名称：k1～k12

スロット：k（値：2.7,3.3,3.9,4.7,5.6,6.8,8.2,10,12,15,18,22）

上位フレーム：k_elements

⑦ 名称：n_elements

上位フレーム：elements

⑧ 名称：n0～n6

スロット：n（値：0,1,2,3,4,5,6）

上位フレーム：n_elements

⑨ 名称：c_elements

上位フレーム：elements

⑩ 名称：c1,c2

スロット：connection（値：直列、並列）

上位フレーム：c_elements

⑪ 名称：p_elements

上位フレーム：elements,networks

⑫ 名称：p1～p168

スロット：param_a（初期値：1.0）

param_b（初期値：connectionが直列の時:(* k (power 10.0 n))

並列の時: 0.0]）

param_c（初期値：connectionが直列の時: 0.0

並列の時:

(/ 1.0 (* k (power 10.0 n)))]）

param_d（初期値：1.0）

上位フレーム：k1～k12,n0～n6,c1～c2,p_elements

５．３．３　計画型問題：作業工程作成システム

（１）フレーム

①　名称：時間割

②　名称：時間割１

スロット：T0～T23

③　名称：時間割２

スロット：T0～T23

④　名称：時間割３

スロット：T0～T23

⑤　名称：時間割４

スロット：T0～T23

⑥　名称：場所

スロット：割当可能時間帯、時間割

④　名称：Ｐ１

INSTANCE OF 場所

スロット：割当可能時間帯 T8,T9,T10,T11,T13,T14,T15,T16,T17

時間割（初期値：時間割１）

⑤　名称：Ｐ２

INSTANCE OF 場所

スロット：割当可能時間帯 T10,T11,T12,T13,T15,T16,T17,T18

時間割（初期値：時間割２）

⑥　名称：Ｐ３

INSTANCE OF 場所

スロット：割当可能時間帯 T8,T9,T10,T11,T13,T14,T15,T16,T17

時間割（初期値：時間割３）

⑦ 名称：Ｐ４

INSTANCE OF 場所

スロット：割当可能時間帯 T14,T15,T16,T17,T18,T19,T20,T21

時間割（初期値：時間割４）

⑧ 名称：作業

スロット：作業時間、並行不可能作業

⑨ 名称：Ａ

INSTANCE OF 作業

スロット：作業時間（初期値：1）、並行不可能作業（初期値：Ａ，Ｃ）

⑩ 名称：Ｂ

INSTANCE OF 作業

スロット：作業時間（初期値：2）、並行不可能作業（初期値：Ｂ，Ｅ）

⑪ 名称：Ｃ

INSTANCE OF 作業

スロット：作業時間（初期値：1）、並行不可能作業（初期値：Ａ，Ｃ，Ｅ）

⑫ 名称：Ｄ

INSTANCE OF 作業

スロット：作業時間（初期値：2）、並行不可能作業（初期値：Ｄ）

⑬ 名称：Ｅ

INSTANCE OF 作業

スロット：作業時間（初期値：1）、並行不可能作業（初期値：Ｂ，Ｃ，Ｅ）

（２）ルール

rule1

IF　何も割り当てられていない

THEN　Ｐ１のパターンを生成する

rule2

IF　Ｐ１のパターンを生成する

Ｐ１のパターン生成に失敗する

THEN　終了

rule3

IF　Ｐ１のパターンを生成する

Ｐ１のパターン生成に成功する

THEN　Ｐ２のパターンを生成する

rule4

IF　Ｐ２のパターンを生成する

Ｐ２のパターン生成に成功する

THEN　Ｐ２の時間割をチェックする

rule5

IF　Ｐ２の時間割をチェックする

Ｐ２の割当可能時間帯は？Ｔである

Ｐ１の時間割は？Ｗである

？Ｗの？Ｔには？Ｓ１が割り当てられている

Ｐ２の時間割は？Ｘである

？Ｘの？Ｔには？Ｓ２が割り当てられている

？Ｓ１の並行不可能作業が？Ｓ２である

THEN　Ｐ２の時間割では平行不可能
　　Ｐ２のパターンを生成する

rule6
IF　Ｐ２のパターンを生成する
　　Ｐ２のパターン生成に失敗する
THEN　Ｐ１のパターンを生成する

rule7
IF　Ｐ２の時間割をチェックする
　　Ｐ２の時間割で平行可能
THEN　Ｐ３のパターンを生成する

rule8
IF　Ｐ３のパターンを生成する
　　Ｐ３のパターン生成に成功する
THEN　Ｐ３の時間割をチェックする

rule9
IF　Ｐ３の時間割をチェックする
　　Ｐ３の割当可能時間帯は？Ｔである
　　Ｐ１の時間割は？Ｗである
　　？Ｗの？Ｔには？Ｓ１が割り当てられている
　　Ｐ３の時間割は？Ｘである
　　？Ｘの？Ｔには？Ｓ２が割り当てられている
　　？Ｓ１の並行不可能作業が？Ｓ２である
THEN　Ｐ３の時間割では平行不可能
　　Ｐ３のパターンを生成する

rule10

IF　Ｐ３の時間割をチェックする
　　Ｐ３の割当可能時間帯は？Ｔである
　　Ｐ２の時間割は？Ｗである
　　？Ｗの？Ｔには？Ｓ１が割り当てられている
　　Ｐ３の時間割は？Ｘである
　　？Ｘの？Ｔには？Ｓ２が割り当てられている
　　？Ｓ１の並行不可能作業が？Ｓ２である
THEN　Ｐ３の時間割では平行不可能
　　Ｐ３のパターンを生成する

rule11
IF　Ｐ３のパターンを生成する
　　Ｐ３のパターン生成に失敗する
THEN　Ｐ２のパターンを生成する

rule12
IF　Ｐ３の時間割をチェックする
　　Ｐ３の時間割で平行可能
THEN　Ｐ４のパターンを生成する

rule13
IF　Ｐ４のパターンを生成する
　　Ｐ４のパターン生成に成功する
THEN　Ｐ４の時間割をチェックする

rule14
IF　Ｐ４の時間割をチェックする
　　Ｐ４の割当可能時間帯は？Ｔである
　　Ｐ１の時間割は？Ｗである
　　？Ｗの？Ｔには？Ｓ１が割り当てられている

Ｐ４の時間割は？Ｘである
？Ｘの？Ｔには？Ｓ２が割り当てられている
？Ｓ１の並行不可能作業が？Ｓ２である
THEN　Ｐ４の時間割では平行不可能
Ｐ４のパターンを生成する

rule15
IF　Ｐ４の時間割をチェックする
Ｐ４の割当可能時間帯は？Ｔである
Ｐ２の時間割は？Ｗである
？Ｗの？Ｔには？Ｓ１が割り当てられている
Ｐ４の時間割は？Ｘである
？Ｘの？Ｔには？Ｓ２が割り当てられている
？Ｓ１の並行不可能作業が？Ｓ２である
THEN　Ｐ４の時間割では平行不可能
Ｐ４のパターンを生成する

rule16
IF　Ｐ４の時間割をチェックする
Ｐ４の割当可能時間帯は？Ｔである
Ｐ３の時間割は？Ｗである
？Ｗの？Ｔには？Ｓ１が割り当てられている
Ｐ４の時間割は？Ｘである
？Ｘの？Ｔには？Ｓ２が割り当てられている
？Ｓ１の並行不可能作業が？Ｓ２である
THEN　Ｐ４の時間割では平行不可能
Ｐ４のパターンを生成する

rule17
IF　Ｐ４のパターンを生成する

Ｐ４のパターン生成に失敗する

THEN　Ｐ３のパターンを生成する

rule18

IF　　Ｐ４の時間割をチェックする

Ｐ４の時間割で平行可能

THEN　終了

５．３．４　診断型システム：ワイン・アドバイザ

（１）ルール

①　規則１

if　(?A の strength が strong)
　　(?B の strength が weak)
then　(?A の strength は ?B の strengthより強い)

if　(食事 の 主食 が ?A)
　　(食事 の ソース が ?B)
　　(?A の strength は ?B の strengthより強い)
　　(?A の color は ?X)
then(食事 の color は ?X)

if　(食事 の 主食 が ?A)
　　(食事 の ソース が ?B)
　　(?B の strength は ?A の strengthより強い)
　　(?B の color は ?X)
then(食事 の color は ?X)

②　規則２

if　(食事 の color が ?X)
then(ワイン の 望ましい色 は ?X)

ここで、ルールの結論部に複数の命題を記述できる場合は、規則１の結論部に統合してよい。

③　規則３

if　(食事 の color が ?X)
　　(?X の 対抗色 が ?Y)
then(ワイン の 望ましくない色 は ?Yである)

④ 規則４

if （食事 の 主食 が ?A）

（not(?A の strength が strong)）

（食事 の ソース が ?B）

（not(?B の strength が strong)）

then(ワイン の 望ましい強さ は strong)

⑤ 規則５

if （?A の strength は ?B の strengthより強い）

then(ワイン の 望ましくない強さ は light)

if （?B の strength は ?A の strengthより強い）

then(ワイン の 望ましくない強さ は light)

⑥ 規則６

if （食事 の ソース が ?A）

（?A の taste が spicy）

then(ワイン の 望ましくない味 は sweet)

⑦ 判断基準

```
if  (ワイン の 色に対する最高点 が ?A)
    (ワイン の 強さに対する最高点 が ?B)
    (ワイン の 味に対する最高点 が ?C)
    (?S = ?A + ?B + ?C)
then(ワイン の 最高点 は ?S)

if  (ワイン の 望ましい色 が ?X)
    (ワイン の 色の重み が ?W)
then(ワイン の 色に対する最高点 が ?W)

if  (ワイン の 望ましい強さ が ?X)
    (ワイン の 強さの重み が ?W)
then(ワイン の 強さに対する最高点 が ?W)

if  (ワイン の 望ましい味 が ?X)
    (ワイン の 味の重み が ?W)
then(ワイン の 味に対する最高点 が ?W)
```

⑧ 採点

```
if  (?X が wine のサブクラス)
    (?X の color が ワイン の 望ましい色 と等しい)
    (ワイン の 色の重み が ?W)
then(?X の 色の点数 は ?W)

if  (?X が wine のサブクラス)
    (?X の color が ワイン の 望ましくない色 と等しい)
    (ワイン の 色の重み が ?W)
    (?S = ?W * -1)
then(?X の 色の点数 は ?S)

if  (?X が wine のサブクラス)
    (?X の strength が ワイン の 望ましい強さ と等しい)
    (ワイン の 強さの重み が ?W)
then(?X の 強さの点数 は ?W)

if  (?X が wine のサブクラス)
    (?X の strength が ワイン の 望ましくない強さ と等しい)
    (ワイン の 強さの重み が ?W)
    (?S = ?W * -1)
then(?X の 強さの点数 は ?S)

if  (?X が wine のサブクラス)
    (?X の taste が ワイン の 望ましい味 と等しい)
    (ワイン の 味の重み が ?W)
then(?X の 味の点数 は ?W)

if  (?X が wine のサブクラス)
    (?X の taste が ワイン の 望ましくない味 と等しい)
```

```
    (ワイン の 味の重み が ?W)
    (?S = ?W * -1)
then(?X の 味の点数 は ?S)

if  (?X が wine のサブクラス)
    (?X の color が no.value)
then(?X の 色の点数 は 0)

if  (?X が wine のサブクラス)
    (?X の strength が no.value)
then(?X の 強さの点数 は 0)

if  (?X が wine のサブクラス)
    (?X の taste が no.value)
then(?X の 味の点数 は 0)

if  (?X が wine のサブクラス)
    (?X の 色の点数 は ?A)
    (?X の 強さの点数 は ?B)
    (?X の 味の点数 は ?C)
    (?S = ?A + ?B + ?C)
then(?X の 点数 は ?S)
```

⑨ 判断

```
if  (ワイン の 最高点 は ?S)
    (?X が wine のサブクラス)
    (?X の 点数 が ?S)
    (?X を薦めた結果がyes)
then(ワイン の 銘柄 は ?X)
```

（２）フレーム

名称	上位フレーム	スロット（初期値）
wine		点数 color（no.value） 色の点数 strength（no.value） 強さの点数 taste（no.value） 味の点数
FrenchWine	wine	
RedFrenchWine	FrenchWine	color（red）
RedBordeaux	RedFrenchWine	
Medoc	RedBordeaux	
Pauillac	Medoc	strength（strong） taste（dry）
Margaux	Medoc	strength（light） taste（medium）
St.Emillion	RedBordeaux	strength（strong） taste（medium）

名称	上位フレーム	スロット（初期値）
RedBourgogne	RedFrenchWine	
Beaujolais	RedBourgogne	strength (strong) taste (dry)
WhiteFrenchWine	FrenchWine	color (white)
WhiteBordeaux	WhiteFrenchWine	
Sauternes	WhiteBordeaux	strength (strong) taste (sweet)
WhiteBourgogne	WhiteFrenchWine	
Meursault	WhiteBourgogne	strength (light) taste (dry)
WhiteLoire	WhiteFrenchWine	
Muscadet	WhiteLoire	strength (light) taste (medium)
Sancerre	WhiteLoire	strength (light) taste (dry)

名称	上位フレーム	スロット（初期値）
RoseFrebchWine	FrenchWine	color（rose）
RoseLoire	RoseFrebchWine	
Anjou	RoseLoire	strength（light） taste（medium）
ワイン		銘柄 望ましい色 望ましくない色 望ましい強さ 望ましくない強さ 望ましい味 望ましくない味 色の重み（初期値:3） 強さの重み（〃:1） 味の重み （〃:2） 色に対する最高点 強に対する最高点 味に対する最高点 最高点

名称	上位フレーム	スロット（初期値）
食事		color 主食 ソース
食料		color taste strength
fish	食料	
halibit	fish	
meat	食料	
fowl	meat	color（white）
chiken	fowl	
beaf	meat	color（red）
steak	beaf	strength（strong）
veal	beaf	color（white） strength（weak）

名称	上位フレーム	スロット（初期値）
sauce	食料	
parmesan	sauce	color（red） strength（strong） taste（spicy）
色		対抗色
red	色	対抗色（white）
white	色	対抗色（red）

５．３．５　ルールシステム実行性能評価：Ｎ個の自然数による繰り返しルール

（１）ルール

①　rule1

```
if  (instance1 の count が ?X)
    (system の count が ?N)
    (?X < ?N)
    (instance1 の const が 1)
    (instance2 の const が 2)
    (instance3 の const が 3)
    (instance4 の const が 4)
then(instance1 の count が (?X + 1))
```

②　rule2～50

```
if  (instance"i" の count が ?X)
    (system の count が ?N)
    (?X < ?N)
    (instance"i" の const が "i")
    (instance"i+1" の const が "i+1")
    (instance"i+2" の const が "i+2")
    (instance"i+3" の const が "i+3")
then(instance"i" の count が (?X + 1))
```

ここで、"i"は2～50の数字を表す。また、"i+1","i+2","i+3"はそれぞれの"i"の数字に1,2,3を加えたものとし、50より大きい数字は50を減じた数字とする。

（２）フレーム

①　名称：frame1

スロット：count（初期値：0）,const

定義：class

②　名称：instance1～50

スロット：count、const（値：1～50）

定義：instance of frame1

③　名称：system

スロット：count（値：実行時に20,40,80を設定）

定義：instance of frame1

５．３．６　フレームシステム実行性能評価：フレーム照合

（１）ルール

①　フレーム生成（レベル０）

```
if  (system の root が no.value)
then(スロット Rand-num 初期値 0 を持つフレーム frame1 を生成する)
    (system の root は frame1)
```

②　フレーム生成（レベル１）

```
if  (system の maxcount が ?M)
    (system の count が ?X)
    (?X < ?M)
    (?Y = ?X + 1)
    (発生した乱数が ?N)
then(frame1 のサブクラスとしてフレーム ?F を生成する)
    (?F の Rand-num は ?N)
    (system の count は ?Y)
```

③　フレーム生成（レベル２）

```
if  (frame1 のサブクラスは ?F)
then(?F のサブクラスとしてフレーム ?G を生成する)
```

④　フレーム生成（レベル３）

```
if  (frame1 のサブクラスは ?F)
    (?F のサブクラスは ?G)
then(?G のサブクラスとしてフレーム ?H を生成する)
```

⑤ 検索（レベル１）

```
if  (system の key は ?K)
    (frame1 のサブクラスは ?F)
    (?F の Rand-num は ?K)
then(検索結果は ?F)
```

⑥ 検索（レベル２）

```
if  (system の key は ?K)
    (frame1 のサブクラスは ?F)
    (?F のサブクラスは ?G)
    (?G の Rand-num は ?K)
then(検索結果は ?G)
```

⑦ 検索（レベル３）

```
if  (system の key は ?K)
    (frame1 のサブクラスは ?F)
    (?F のサブクラスは ?G)
    (?G のサブクラスは ?H)
    (?H の Rand-num は ?K)
then(検索結果は ?H)
```

（２）フレーム

① 名称：system

スロット：root（初期値：no.value）

count（初期値：0）

maxcount（値：実行時に500,1000,2000,4000を設定）

key（値：実行時に 0～99 の値を設定）

５．４　各問題に関する補足説明

（１）解釈型問題：参考文献リスト解析システム

・文字列の切り離し・比較時に手続きを用いる。

・実行時のデータは、予め用意したものを選択するようにした。

（２）設計型問題：四端子網合成

・離散的な数値を網羅的に適用してゆく際に、その組合せの作業記憶のためにインスタンスを用いる。

・このような構造を実現しにくいツール（ＫＢＭＳ）については、後向き推論によって行列式を解いていく方法を試みた。この場合、段数の最小条件とすべての解の導出はおこなってはいない。

（３）計画型問題：作業工程作成システム

・工程におけるパターン生成手続きを用いる。このことにより、（２）のような作業記憶的なメモリ消費を避ける。

（４）診断型問題：ワイン・アドバイザ

・入力の無いときの処理にどうすればよいか不明のため固定にしている。

（５）フレームの処理速度測定問題

・ツールによっては動的にフレームを生成できないものや、インスタンスの生成しかしないものなどがある。その場合には一段のみの測定としている。

・検索時間の測定について、キーとなる数値が乱数で与えられたものなので、検索時間の平均をとる必要がある。

6．性能評価用テスト問題に基づく知識システム構築ツール比較表

6．性能評価用テスト問題に基づく知識システム構築ツール比較表

6.1 概要

項目 ＼ ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS 注1	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
開発元（販売元）	ＮＴＴ（伊藤忠テクノサイエンス）	カーネギーグループ社（センチュリリサーチセンタ）	東洋情報システム	三菱総合研究所（日本データゼネラル）	Intelli Corp 社（日本ユニシス）	日立製作所	富士通	三菱電機	富士ゼロックス
特徴	ハイブリット型ツールで知識コンパイラにより高速化を実現	ＣＲＬ－ＯＰＳとＣＲＬ－Prolog、そしてＣＲＬによるスキーマ表現を統合。	豊富な開発環境。	対話処理を基本として、オンラインヘルプのみでほとんど操作可能なルールベースと、コンパイル方式で前向き推論と時制推論を実現するフレームベースの混成。	汎用、高機能（多重世界機構、ATMS）ユーザフレンドリーな開発、利用環境	プロダクションルールとフレームを知識表現として持つ。手続き処理をC言語で記述する多階層協調型推論ファジイ推論 注２ ﾕｰｻﾞｲﾝﾀﾌｪｰｽ構築、他ｼｽﾃﾑ(RDB,OA,DSS)接続が豊富	多機能（黒板、ファジイ）。ライブラリが多い（図形、データベース）	Prolog(ESP)をベースとしてプロダクションシステムを中心とするESPの持つオブジェクト指向やSIMPOSの提供するクラス群を利用できる	Smalltalk-80 環境上に構築された、MYCIN 型のエキスパートシステムマルチウィンドウを利用したユーザーフレンドリーなシステム。ST80との親和性が極めて良い。
シェルの歴史的背景と開発思想	RETEアルゴリズムをフレームまで適用させハイブリット型ツールでの高速化を実現した	カーネギーメロン大学で研究・開発の積み重ねの結果の多くを反映	ＥＭＹＣＩＮなどの分類型の表現、フレーム表現と、ＯＰＳ５と同様の純粋プロダクションシステムを統合	ZEUSのオリジナル部分（ルール・ベース）に加え、フレーム・ベースを統合化	DENDRAL-UNITSを拡張。知識表現の汎用性に重点	改良RETEアルゴリズムによる高速化多階層協調型推論による大規模化	HEARSAY, AGEで使われた黒板と、プロダクションシステムとの統合	プロダクションシステムを実現しながら開発環境を重視しているESP/SIMPOSと合わせて高度な開発環境を提供する	MYCIN の機能を基本として確信度も扱えるように設計されている。全て、ビューと呼ばれるウィンドウ上で操作できるようになっている。

（備考） 評価記号の内容について

○、△、×の違いは、概ね次の内容のものを示しているが、項目によってはこの通りではない。そのような場合については可能な限り注を添えるものとした。

○ －－－ 機能・条件を十分に満足するものを備えている。
△ －－－ 機能を備えているが、実用上不十分であるもの。
× －－－ 機能・条件に合うものが備えられてない。または、実用に供さない。
無印－－－・未調査・不明など

注１ Super BRAINS に関する比較表の以下の記述は、すべて Symbolics版 Super BRAINS のものである。
注２ ファジイ推論は、1989年3月サポート予定

6.2　ツールの機能／知識表現

項目 ＼ ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
概要									
機能の豊富さ	○	○	△	○	○	○	○	○ 注8	△
機能間のバランス	△	△	△	△	○	○	○	○	○
記述の統一性	○	△	△	△	○	○	○	○	○
表現の理解容易性	△	△	△	△	○	○	△ 注1	△	○
機能の詳細項目									
推論方向									
前向き	○	○	○	○	○	○	○	○	△注3
後向き	○	○	○	△ 注2	○	○	○	△ 注3	○
両方向混在	○	×	○ 注4	○	○	○	○	○	△注3
ルールの表現要素									
条件部 結合子									
条件部 and, or, not	○	and(Implicit)not	○	○	○	○ 注10	and(Implicit)or	and(Implicit)not	and(Implicit)ornot
条件部 その他	×	×	×	×	FOR.ALWAYS	×	○注12	×	ifAllof,ifAnyofなど
条件部 論理述語のユーザによる拡張性	○	○	×	○	△ 注5	×	○	○	×
条件部 手続き									
条件部 算術演算式	○	○	○	○	○	×	○	○	○
条件部 その他の手続き	○	○ 注9	○ 注6	○	○	○ 注11	○	×	○注8
条件部 その他の記述	×	×	×	×	○ TEXT	×	×	○ 注7	×

注1　ルールにおける知識表現がプログラムレベル。ルールの書式はユーザカスタマイズ可能　注2、3　いずれも使い方が限定されていて不便　注4　知識ユニット単位で指定可能
注5　可能であるが難しい　注6　ユーティリティ、LISPによる記述　注7 ESPオブジェクトへのメッセージ送信　注8. Smalltalkへのメッセージを送信（ただし、ルールの展開形を知ることが必要）ST80を直接書く事ができる。つまりルールの中でST80を呼び出す事が可能
注9　LISPによる記述　注10　not は論理的には not existsts を意味する　注11　後向き推論でのみ使用可能
注12 LHS-Evaluatorにより任意の評価法が定義できる。

ツール名		KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
実行部	手続き									
	言語仕様として	×	○	×	condition	○	○	○	○	○
	他言語の利用可能性	LISP	LISP, CRL-PROLOG	LISP	C, Fortran LISP	LISP	C 注5	○	ESP	Smalltalk 注4
	その他の記述	×	注4	×	×	命題表現	注6	種々のアクション	注1	×
パタン照合の対象データ型と方法	非構造データ									
	データ型	整数、実数、ｼﾝﾎﾞﾙ、ﾘｽﾄ、ｽﾄﾘﾝｸﾞ	整数、実数、ｼﾝﾎﾞﾙ、ﾘｽﾄ	整数、実数、ｽﾄﾘﾝｸﾞ、ｼﾝﾎﾞﾙ、ﾘｽﾄ	整数、実数、ｼﾝﾎﾞﾙ、ｽﾄﾘﾝｸﾞ、ﾘｽﾄ	整数、実数、ｼﾝﾎﾞﾙ、ｽﾄﾘﾝｸﾞ、ﾘｽﾄ/ｼｰｹﾝｽ	整数、実数、ｼﾝﾎﾞﾙ、ｽﾄﾘﾝｸﾞ	× 注2	整数、実数、ｼﾝﾎﾞﾙ、ｽﾄﾘﾝｸﾞ、ﾘｽﾄ/ｼｰｹﾝｽ	整数、実数、ｽﾄﾘﾝｸﾞ、配列、ｼﾝﾎﾞﾙ
	方法	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	変数	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	×	変数、ﾘﾃﾗﾙﾊﾟﾀﾝ	変数
	構造化データ									
	構造の指示、参照	○	○	○	○	○	○	×	○	×
	関係の制約	○	○	○	○	○	○	×	×	×
	フレームの親子関係による制約	×	○	○	○	○	○ 注3	×	×	×
	構造名									
	変数として使用可能	×	○	○	○	○	○	×	×	×
	指定複数フレーム内の照合	○	○	○	○	○	○	×	○	×
	属性名の指定	○	○	○	○	○	○	×	○	○
	属性値の指定	○	○	○	○	○	○	×	○	○

注1　ESPオブジェクトへのメッセージ送信　　注2　パタン照合機能はない　　注3　直接の親しか参照できない　　注4　前ページ　（注8 参照）
注5　Ｃ言語のモジュールを起動できる言語(FORTRAN,LISP,PROLOG,COBOL,PL/Iなど)の中にサブルーチンとしてＥＳを組み込める
　　また、Ｃ言語から起動できる言語(FORTRAN,COBOL,PL/Iなど)で作ったサブシステムをＥＳから利用できる
注6　意思決定支援システムとの連携

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
その他の特徴									
デモンの記述	○	×	○	○	○	○	○	○	×
ルールの before/after	×	×	×	○	△	×	×	○	×
実行の保留機能	×	×	×	○	×	×	○	○	×
ルールの制御表現									
ルールのグループ化	○	×	○	○	○	○	○	○	○
呼び出し方法	関数呼び出し	×	注1	グループ名	起動時に指定	メタルール	起動時に指定	推論起動時に指定	後向き推論なので仮説駆動
イベント起動機能	×	×	×	×	×	○	○	×	×
ルールの実行制御	○	○	○	記述順、優先度	○	○	手続き、ｲﾍﾞﾝﾄｷｭｰ	ﾒﾀﾙｰﾙによる制御	○注4
フレームとオブジェクト									
継承関係の有無	○	○	○	○	○	○	○	○	△注3
継承関係の種類									
class-subclass	○	○	○	○	○	○	○	○	△注3
class-instnace	○	○	○	○	○	○	○	○ 注5	△注3
その他	○ 注2	ユーザ定義	○	○	×	×	ユーザ定義	○ 注5	×
多重継承の有無	○	○	○	○	○	○	○	○	×
継承の制御方法	フレーム属性に記述	継承の道筋の指定	ﾌﾚｰﾑ定義に記述	inheritanceのon/offを指定する	インヘリタンスロールの設定	フレーム定義に記述	オブジェクト定義に記述	スキーマ定義に記述	×
フレーム単位	○	○	○	×	○	○	○	○	×
スロット単位	○	○	×	○	○	×	×	×	×

注1　制御手続きによる記述、ルールによる起動　　注2　GROUP,REATION などフレーム間のPART-OF,RELATIONの継承

注3 Smalltalk のクラスを利用すれば,(やりづらいが)
不可能ではない〔ただし Humbleの核にはこのような機能はない〕

注4ア)メインパラメータを指定するとルールが起動される。
ロ)ディデューシングルールの記述順。ただし、後ろ向きなので、
ルールの実行順は結果に影響しない。

注5　ＥＳＰを用いることによる

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
制御方法									
親の値	○	○	○	○	○	×	○	○	×
親と同値	○	○	×	×	○	○	×	×	×
その他	×	○	×	×	○	×	×	×	×
多重継承の制御	指定順のユニオン	継承範囲の切り換 継承の道筋の指定	△	inheritanceのon/off	override, unique union, same等	ユニオンのみ	ユニオンのみ	ユニオンのみ	×
ユーザ定義の可能性									
継承関係の拡張性	○	○	○	×	○	×	×	×	△
非継承関係の拡張性	○	○	○	×	×	×	○	×	△
属性／データの記述									
属性値の型									
シングルバリュー	○	○	○	○	○	○	○	○	○
マルチバリュー	×	○	○	○	○	○	○	○	△
その他	ﾘｽﾌﾟ型による制約	×	×	×	×	ｷｭｰ,ｽﾀｯｸ	×	×	×
スロットの制御(ﾌｧｾｯﾄ)									
属性値の型の制限	×	○	△	○	○	○	○	△ 注2	○
属性値の値の制限	×	○	○	×	○	○	○	△ 注2	○
属性値の数の制限	×	○	×	○	○	○	○ 注3	△ 注2	×
フレーム／オブジェクトの範囲制限	×	○	×	○	×	×	×	△ 注2	×
その他	×	ユーザ定義	ユーザ定義	システムとユーザ	ユーザ定義	×	ユーザ定義	ユーザ定義	×
情報隠ぺい機能	○	○	○	× 注1	○	○	○	○	×

注1　継承の記述において下位のフレームにおいてもスロット定義を記述しなければならない。　注2　ESPを用いることによる　注3 予め定義したsingle/multiple

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
手続きの記述									
付加手続きの有無	○	○	○	○	○	○	○	○	○
メソッドの有無	○	○	×	○	○	○	○	○ 注3	○
メソッドの記述言語	LISP	LISP	×	LISP	LISP	C, EXCEED2 注4	LISP	ESP	ST80
ルールの記述	×	×	×	×	×	×	○	×	○
メソッドの継承	○	○	×	○	○	○	○	○	×
多重継承の有無	○	○	×	○	○	○	○	○	×
多重継承の制御	ｽｰﾊﾟｰｸﾗｽの指定順	ユーザ指定	×	○	ユーザ指定	ユニオン	ｽｰﾊﾟｰｸﾗｽの指定順	ｽｰﾊﾟｰｸﾗｽの指定順	×
起動操作	メッセージパッシング	メッセージパッシング	デーモン	ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ	ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ、ﾃﾞｰﾓﾝ	ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ、ﾃﾞｰﾓﾝ	ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ、ﾃﾞｰﾓﾝ	デーモン ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ	ﾒｯｾｰｼﾞﾊﾟｯｼﾝｸﾞ
メッセージパッシングの方法、形式	LISP関数	LISP関数	×	ｵﾌﾞｼﾞｪｸﾄ、ﾒｯｾｰｼﾞ、ﾊﾟﾗﾒｰﾀ	LISP関数	C関数	LISP関数	ESPのﾒｿｯﾄﾞｺｰﾙ	ｵﾌﾞｼﾞｪｸﾄﾒｯｾｰｼﾞ
実行の並列性	×	×	×	×	×	○ 注5	×	×	×
論理表現の有無									
言語の種類	×	CRL, Prolog	その他	LISP	その他 注2	×	△ 注7	その他	ST80
バックトラック機能	×	○	△	○	○	×	×	○	(後向き推論)
ユニフィケーション機能	×	○	△	○	○	×	△ 注7	○	×
他言語とのインターフェース	○	○	○	○	○	○	○	○	○
言語名	機種に依存	LISP	機種に依存	C, Fortran	LISP	C	C,FORTRAN	ESP	Smalltalk
方式	LISPにより可能	関数呼び出し	機種に依存	関数呼び出し	関数呼び出し	注6	関数呼び出し	注1	注3

注1　ルール条件部、結論部、デーモンにおける直接呼び出し　　注2　TellAndAskという名称がついている。　　注3 特別な識別子《 》に囲まれたものはST80語だと解釈される。
注4　意思決定支援システムのコマンド言語　　注5　多階層協調推論でサポート
注6　サブルーチン化、関数呼び出し（C呼び出し、Cからの呼び出しも可能）　　注7 Lispオブジェクトとして可能

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
不確実な情報の扱い									
確信度	×	×	○	○	○	○	○	×	○
計算方法	×	×	MYCIN型と同じ	MYCIN型と同じ	MYCIN型と同じ	MYCIN型と同じ	MYCIN型と同じ	×	○
ユーザ定義可能性	×	×	×	×	注1	○	○	×	MYCIN 型と同じ
ファジィ	×	×	×	×	×	○ 注3	○	×	×
メンバーシップ関数	×	×	×	×	×	7種類	4種類	×	×
演算の種類	×	×	×	×	×	MIN,MAX,代数積	MAX	×	×
ユーザ定義可能性	×	×	×	×	×	○	○	×	×
Dempster/Shafer 理論	×	×	×	×	×	×	×	×	×
その他	×	×	×	×	ユーザ定義 注1	×	×	×	×
ワーキングメモリの扱い									
データ表現									
ファクト	○	○	○	×	○	○	×	○	×
ブラックボード	×	×	×	×	×	○	○	△ 注2	×
ベクタ	×	×	×	×	×	×	×	×	×
フレーム	○	○	○	○	○	○	○	○	×
その他	×	×	×	○	×	×	×	×	×
管理									
ATMS	×	×	×	×	○	×	×	×	×
TMS	×	×	×	×	○	×	×	×	×
ブラックボード	×	×	×	×	×	○	○	注2	×

注1　可能ではあるが難しい　　注2　複数の知識源からの参照が可能　　注3　1989年3月サポート予定

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
多重世界の表現	×	○	×	○	○	○	×	×	×
コンテクスト	×	○	×	○	×	×	×	×	×
ビューポイント	×	×	×	×	×	×	×	×	×
その他	×	×	×	×	ワールド	○	×	×	×
推論制御									
競合解消									
決定要素									
ルールの優先度	○	○	○	△ 注1	○	○ 注6	×	×	△注8
ルールのあい昧さ	×	×	×	○ 注1	×	○	×	×	×
データの新しさ	○	○	○	○	○	○	×	○	×
条件部の詳細さ	○	○	×	×	○	○	×	△ 注2	×
ルールの記述順序	○	×	○	○	○	○	○	○	○
その他	×	×	○	×	○	×	×	○	×
決定要素の適用順序		任意		画一	複数			FIFO/FILO 注3	
適用順序の変更可能性	○	○	○	×	○	○	×	×	
推論制御の最適化手法	改良rete	rete	不明	×	不明	改良rete	×	その他	×
メタ制御	×	○	×	×	ﾙｰﾙｼｽﾃﾑで定義	○ 注7	ｲﾍﾞﾝﾄ実行順指定	○	×
メタルール	×	×	○	×	ユーザ定義可	○	△ 注4	○	×
フレーム／オブジェクト									
継承解決	○	×	×	○	○	×	○	○ 注5	×
デモン制御	○	×	○	○	○	×	○	○ 注5	×
メッセージパッシングの制御	○	×	×	×	×	×	×	○ 注5	×

注1　ルールベースにおいて、まずルールの確信度の高いものを優先し、確信度が同一の場合は、優先度で決める
注2　第2ランクの決定要素である
注3　ＦＩＬＯはＡ１０版以降
注4　イベントセレクションでルール実行順序を指定できる
注5　ESPを用いることによる
注6　ルール群の優先度としてサポートしている
注7　多階層協調形推論で複数のＥＳの実行を制御できる

注8 あらかじめ、はじめに推論すべきパラメータを指定を指定しておくことによって、その値を決めるためのルールを先に適用させることもできる。

ツール名	KBMS/SUN	Knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
アジェンダ機能	○	○	×	○	○	○ 注1	○ 注3	△	×
探索手法の実現性									
blind search		深さ、幅優先	深さ、幅優先	深さ優先	深さ、幅優先	深さ、幅優先	深さ、幅優先	深さ、幅優先	深さ優先
heuristic search	×	×	×	×	最良優先	×	最良優先	×	×
その他	×	×	×	×	And/Or graph 他	×	○ 注4	×	×(前ページ 注8)
問題解決のモデル									
推論モデル			不確実、仮説的、モデルベース推論	不確実、仮説的、時制推論	仮説的、時制、モデルベース推論	不確実、分散協調推論 注2	分散協調、ファジイ推論	仮説、時制推論	
データの扱い	非同期入力	非同期入力	非同期入力	非同期入力	非同期入力	非同期入力	非同期入力	非同期入力	
知識処理手法の実現性									○
generate-and-test	×	○	×	○	○	○	○	○	
hierarchical-generate-and-test	×	○	×	○	○	○	○	○	
guessing	○	○	○	○	○	○	○	○	
topdown-refinement	○	○	○	×	○	○	○	○	
least-commitment	×	○	○	○	○	○	×	○	
constraint-propagation	○	○	×	○	○	○	×	○	
backtracking	○	○	○	○	○	○	○	○	
dependency-directed-backtracking	×	×	×	×	○	×	×	○	
opportunistic-schedule	○	○	×	○	○	○	○	○	
classification	○	○	○	×	○	○	○	○	
その他									

注1　ビューノートのイベントキューで実現
注2　1989年3月よりファジイ推論をサポートする予定
注3　イベントキューによる制御
注4　FOCUSオブジェクトの概念によって種々の探索手法を実現する

6.3 開発者／利用者インタフェース

ツール名	KBMS/SUN		Knowledge Craft		Super BRAINS		DG/ZEUS II		KEE		ES/KERNEL		ESHELL/X		EXTKERNEL II		HUMBLE	
概要	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者	開発者	利用者
操作性	△	△	△	△	△	×	△	△	○	○	○	○	△	○	△	△	○	○
統合性	△	○	△	△	○	○	△	△	○	○	○	○	○	○	○	○	○	○
一貫性	○	○	△	△	○	○	△	△	○	○	○	○	○	○	○	○	○	○
拡張性	○	○	○	○	×	○	△	○	○	○	○	○	○	○	○	○	×	×
説明機能	△	×	△	×	×	×	×	△	○	○	△	△	×	×	△	△	△	△
ヘルプ機能	△	×	△	×	×	×	×	△	×	○	×	×	×	×	×	×	×	×注1
エラー処理機能	△	△	△	△	△	×	×	×	△	△	△	△	×	×	△	△	△	×注1
編集機能	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ	編集	ブラウズ
編集対象の種類	テキスト、図形		テキスト、図形		テキスト		テキスト		テキスト、図形		テキスト、図形		テキスト		テキスト		テキスト	
専用エディタ／ブラウザの有無	○	△	○	○	×	○	×	×	○	○	○	○	×	△	△	△	○	○
シンタックスチェックなどの一貫性管理	○	△	○	×	×	×	×	×	○	○	○	×	×	○	○	×	○	○
知識ベースの世代管理	○	△	○	×	○	○	○	○	○	×	○	×	○	○	○	△	×	×
拡張性	×	×	×	×	×	×	○	○	○	○	○	×	×	×	×	×	○	△
操作性、スピード	○	○	○	○	○	○	△	×	○	○	○	○	△	○	○	×	△	△
説明機能	△	×	△	△	×	○	○	×	△	△	○	×	×	×	○	×	×	×
一括編集機能	○	−	○	−	○	−	○	−	×	−	×	−	○	−	×	−	×	—
コンパイルチェック	○	−	×	−	×	−	○	−	×	−	△	−	×	−	△	−	×	—
エラー処理	△	−	△	−	△	−	×	−	△	−	△	−	△	−	△	−	×	—
デバッグ機能	○	−	△	−	×	−	○	−	○	−	○	−	○	−	△	−	×	—

注1 Smalltalk のエラーメッセージを理解している必要あり。

ツール名	KBMS/SUN		Knowledge Craft		Super BRAINS		DG/ZEUS II		KEE		ES/KERNEL		ESHELL/X		EXTKERNEL II		HUMBLE	
複数人による開発サポート	編集 ○	ブラウズ ○	編集 ×	ブラウズ ×	編集 ×	ブラウズ ×	編集 ○	ブラウズ ×	編集 ×	ブラウズ ×	編集 ×	ブラウズ ×	編集 △	ブラウズ ×	編集 ○	ブラウズ ×	編集 ○	ブラウズ ×
プロジェクト管理	○	○	×	×	×	×	×	×	×	×	×	×	△	×	○	×	○	×
表示機能と操作性	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時
ウィンドウシステム	△	×	○	○	○	△	×	×	○	○	○	○	○	○	○	○	○	○
グラフィック	×	○	○ 注1	○	△	△	×	×	○	○	○	○	○	○	○	○	×	×
アイコン	×	×	○ 注1	○	×	×	×	×	○	○	○	○	○	○	○	○	○	○
ポップアップメニュー	○	×	○ 注1	○	○	△	×	×	○	○	○	○	○	○	○	○	○	○
ズーミング	×	×	○	×	×	×	×	×	○	○	×	×	○	○	×	×	×	×
アニメーション	×	×	○	×	×	×	×	×	○	○	×	×	×	×	×	×	×	×
カラー表示	×	×	×	×	×	×	○	○	×	×	○	○	○	○	△	△	×	×
シミュレーション機能	×	×	×	×	×	×	○	○	○	○	×	×	×	×	×	×	×	×
グラフィック（2次元／3次元）	×	×	○	○	○ 2次元	×	×	×	○ 注2 2次元	○ 2次元	○	×	○	○	×	×	△ 注3	△
知識ベースのグラフィック表示／編集	○	○	○	○	○	○	○	○	○	○	○	×	△	×	△	△	△ 注3	△
自然言語処理機能																		
対象言語																		
日本語	○	○	×	×	○	○	○	○	○	○	○	○	○	○	○	○	○	○
英語	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
その他	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

注1　コマンドシステムとして用意されている

注2　KS-300のグラフィック機能で実現可能

注3 ST80言語で記述可能

HUMBLE自身にはこの機能は用意されていない。

ツール名	KBMS/SUN		Knowledge Craft		Super BRAINS		DG/ZEUS II		KEE		ES/KERNEL		ESHELL/X		EXTKERNEL II		HUMBLE	
表示機能	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時	開発時	利用時
テキスト	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
アイコン／メニュー	×	×	×	×	○	○	×	×	○	○	○	○	○	○	○	○	○	○
自然言語理解																		
字句解析	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
構文解析	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
意味解析	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
文脈解析	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
音声処理、会話理解	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
デバッグ機能																		
専用デバッガの有無	○		○		○		○		○		○		○		○		○	○
知識ベースの一貫性管理	○		×		△		×		○		○		×		○		△	△
拡張性	×		×		×		×		△		×		×		×		△注1	△注1
操作性、スピード	△		○		○		△		○		○		△		△		○	△
説明機能																		
アジェンダ	○		×		×		○		○		○		×		×		×	
マッチング	×		○		△		○		○		○		×		×		×	
推論過程のトレース	○		○		△		○		○		○		○		○		○	
ジャスティフィケーション	×		×		×		○		○		×		×		×		×	

注1 ソースファイルで提供されるためST80言語を知っていれば拡張可能。ST80のひとつのクラスとして提供される。

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
WHY	×	×	×	○	○	×	×	×	○
HOW	×	×	×	○	○	○	×	○	○
what-if	×	×	△	×	○	×	×	×	×
その他	×	×	×	△	△	×	×	×	△
エラー処理	△	△	△	×	△	△	△	△	×
テスト機能									
実行過程のセーブ機能	○	×	×	×	×	○	○	×	△注1
テストデータの保存管理、再実行	○	△	○	△	△	△	△	×	×
テストケースの自動生成	×	×	×	×	×	×	×	×	×
チューニング機能									
パフォーマンス測定	×	×	×	×	×	×	×	×	×
ボトルネックの発見のサポート	×	×	×	×	○	×	×	×	×
知識ベースの変更サポート	×	×	×	×	×	○ 知識テーラ使用	×	×	×

注1 Smalltalk のイメージとして

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
コンパイル機能									
インタプリタ／コンパイラ両モードの併用	○	×	×	コンパイラ	○	×	○	×	×
インクリメンタルコンパイル	○	△	×	×	○	×	○	△	△ 注2
シンタックスチェック	○	△	○	○	○	○	○	○	△ 注2
基本的なセマンティックチェック	×	×	×	×	○	○	×	○	×
知識獲得支援機能									
開発方法論のサポート	×	×	×	×	×	×	×	×	×
知識表現の自動生成	×	×	×	×	×	○ 注1	×	×	×
モデリング支援機能	×	×	×	×	×	×	×	×	×
学習機能	×	×	×	×	×	×	×	×	×
マニュアル									
入門マニュアル	○	×	△	×	○	○	×	○	×
リファレンスマニュアル	○	○	×	○	○	○	○	○	
その他のマニュアル	×	×	○	×	○	○	×	○	×
ツールに即した教科書の有無	×	○	○	×	×	○	○	○	○
ツールに即した教科書の名前	×	セミナーテキスト	セミナーテキスト	×	×	ﾋﾞｼﾞﾈｽﾏﾝのためのAI入門(ｵｰﾑ社)	教育テキスト	セミナーテキスト	× 注3
例題	×	○	テキスト	×	○	○	○	○	○ ○

注1　知識テーラで作成したカスタマイズドエディタで実現

注2 Smalltalk のコードへコンパイル

注3 HUMBLEオペレーション編及びHUMBLEプログラム編

6.4　システムインタフェース

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
他言語インタフェース									
有無	○	○	○	○	○	○	○	機種に依存	○
種類	LISPによる	LISPによる	C, FORTRAN	C, Fortran,LISP	PROLOG	注1	C, Fortran	同上	Smalltalk のみ
入出力パラメタの受渡し	×	○	○	○	LISP経由	C経由	LISP経由	同上	○
デモンの記述	×	○	×	×	同上	○	○	同上	×
データベースインタフェース									
有無	×	×	×	×	○	○	○	×	×
種類	×	×	×	×	SQLシステム	RDB	RDM	×	×
方法	×	×	×	×	パッケージ	システムメソッド	専用ｵﾌﾞｼﾞｪｸﾄ経由	×	×
センサベースインタフェース									
有無	×	×	×	×	×	×	×	×	×
種類	×	×	×	×	×	×	×	×	×
方法	×	×	×	×	×	×	×	×	×
オンライン機能	×	×	×	×	×	×	×	×	×
非同期処理機能	×	×	×	×	×	×	×	×	×

注1　LISP,PROLOG（当該言語のアプリケーションからＥＳを起動できる）　COBOL,PL/I（ホスト(VOS3)版のみ）　FORTRAN

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
ネットワークインタフェース									
有無	×	×	×	×	○	○	○	○	△ (ST80経由)
種類	×	×	×	×	LAN	LAN	リモートディスク	LAN	LAN
方法	×	×	×	×	パッケージ	C経由	ファイルアクセス	パッケージ	ファイルアクセス
入出力デバイスインタフェース									
有無	×	×	×	×	×	○	×	○	○(ST80経由)
種類	×	×	×	×	×	注2	×	注1 GPIB/RS232C	マウス
方法	×	×	×	×	×	フレーム,C経由	×	システムメソッド	メッセージセンディング
アプリケーションパッケージ・インタフェース									
有無	×	×	×	×	×	○	○	×	×
種類	×	×	×	×	×		×	×	×
方法	×	×	×	×	×	C経由	×	×	×
その他	×	×	グラフィックツール	×	ACTIVE IMAGE	グラフィックインターフェース	グラフィックインターフェース	×	×

注1　GPIB は General Purpose Interface Bus の略
注2　マウス入力、ＬＡＮのリモートプリンタ

6.5 ツールの性能

AＩオープンハウスに設置してあるツールの性能比較

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
測定環境									
ツール	KBMS/SUN	knowledge Craft	Super BRAINS Symbolics版	DG/ZEUS II	KEE 3.1	ES/KERNEL/W	ESHELL/X	EXTKERNEL II	
対象ハードウェア	SUN3/260	Explorer II	Symbolics 3650	DS/7500-II	KS301(Explorer)	2050/32	FACOM-G150A	PSI-II	
主メモリ	8MB	8MB	8MB	8MB	6MB	8MB	8MB	80MB	
OS	SUN-OS		Genera	AOS/VS		HI-UX	SX/G	SIMPOS II B00版	

実行速度 注1	KBMS/SUN インタプリタ	KBMS/SUN コンパイラ	knowledge Craft インタプリタ	knowledge Craft コンパイラ	Super BRAINS インタプリタ	Super BRAINS コンパイラ	DG/ZEUS II インタプリタ	DG/ZEUS II コンパイラ	KEE インタプリタ	KEE コンパイラ	ES/KERNEL インタプリタ	ES/KERNEL コンパイラ	ESHELL/X インタプリタ	ESHELL/X コンパイラ	EXTKERNEL II インタプリタ	EXTKERNEL II コンパイラ	HUMBLE インタプリタ	HUMBLE コンパイラ
ルール	37.38		60.05		0.607		0.55		2.590	5.090	48.99		37.06		123.6			
フレーム生成																		
レベル1 注2	212.1		32.99		6.83 注5		2.808		2.976	6.431	12.92		1.535		193.7			
レベル2 注3	254.2		22.67		1.16		6.537		5.080	7.150	− 注7		− 注7		− 注8			
レベル3 注4	181.1		2.492		1.08		3.639		3.430 注6	5.383	−		−		−			
フレーム検索																		
レベル1 注9	36.1		3211.		62.5		27.05		7.574	151.902	1000 注13		254.2		6.77			
レベル2 注10	35.9		4.36		26.7		19.23		5.283	108.278	−		− 注7		− 注8			
レベル3 注11	− 注12		1.3		17.9		5.392		4.312	85.101	−		−		−			
BB/ATMS																		
その他 注14	897	97050	1173	214420	1296	103961	124	47731	363	49381	−	81684	1525	13045	− − 注15 − 注16	167072 333.9 270.7		

注1 単位：logical instruction/sec　　注2 サブクラス（またはインスタンス）のフレーム

注3 レベル1のサブクラスのフレーム　　注4 レベル2のサブクラスのフレーム

注5 ES／KERNEL、Super BRAINSは、フレーム数の制限を変更することができるが、デフォルトの設定で使用した。このため、1000個までのフレーム生成しか測定できていない。

注6 フレームの数が4000個の場合などで、実行時間がかかりすぎて実質的に測定不能の場合を除いたもの。

注7 ES／KERNEL、ESHELL／Xは動的にクラスを生成する機能をサポートしていないので、レベル2、レベル3のテストができなかった。

注8 EXTKERNELは階層的なフレームを動的に生成する機能をサポートしていないので、レベル2、レベル3のテストができなかった。

注9 フレーム自身の値に対する検索　　注10 直接の継承値に対する検索　　注11 間接の継承値に対する検索

注12 KBMS／SUNは、関係を指定して検索することができないので、レベル3の検索テストができない。

注13 フレームが500個の検索時間は 0 秒で、1000個の時 1 秒であった。

注14 (TARAI 10 5 0) の関数呼び出し回数／秒

注15 30要素のappend (単位:KLips)

注16 30要素のappend (単位:KLips)

（参考）異なる測定環境での性能

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
測定環境									
ツール					KEE 3.1	BS/KERNEL/W (ES/KERNEL/H)	ESHELL		
対象ハードウェア					KS303(Explorer2)	2050/32 (M-680H)	G-150		
主メモリ					64MB	16MB	8MB		
OS						H1-UX/Wリリース2.0 (VOS 3)	SX/G		
実行速度　注1									
	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ	インタプリタ　コンパイラ
ルール					5.553　8.730	48.99　(602.6)	157.50		
フレーム生成									
レベル1					10.426　21.100	8.93　(102.7)	24.90		
レベル2					17.085　24.997	−	−		
レベル3					8.791　19.071	−	−		
フレーム検索									
レベル1					20.664　600.07	3770　(44.486)	595.32		
レベル2					15.241　425.50	−	−		
レベル3					17.965　366.51	−	−		
BB/ATMS									
その他　注2					1154　214420	−　81684	1525　13045		

注1　単位：logical instruction/sec
注2　(TARAI 10 5 0) の関数呼び出し回数／秒

（　）内のデータは、Mシリーズによる測定結果。

利用環境： KBMS

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine	ELIS	TAO	LISP		
workstation	SUN Apollo	UNIX AEGIS	LISP LISP		8M
personal computer	PC9801	MSDOS	LISP		8M
main frame	DIPS IBM	BEAM MVS	LISP LISP		
その他					
	VAXｼﾘｰｽﾞ	VMS	LISP		
	MVｼﾘｰｽﾞ	AOS/VS	LISP		

利用環境： Knowledge Craft

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine	Explorer II		CommonLisp		
workstation					
personal computer					
main frame					
その他					

利用環境： Super BRAINS Symbolics版

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine	Symbolics	Genera	CommonLisp	1MB	1MB
workstation					
personal computer					
main frame					
その他					

利用環境： DG/ZEUS II

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine					
workstation	DS/7500-II	AOS/VS	LISP	70MB	4MB
personal computer					
main frame					
その他	MVｼﾘｰｽﾞ	AOS/VS	LISP	70MB	4MB

利用環境： Humble

	名　称	ＯＳ	開発言語	最　小　容　量 ディスク	メモリ
lisp mashine					
workstaion	Xerox 1161 sun3,HP Tektronics Apollo,Mac	UNIX small talk	Smalltalk		
personal computer					
その他					

利用環境： KEE

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine	注1		LISP		
workstation	注2		LISP		
personal computer					
main frame					
その他					

注1 KS301(EXPLORER),KS303(EXPLORERⅡ), SYMBOLICS, XEROX D-MACHINE
注2 SUN, APOLLO, HP

利用環境： ESHELL/X

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine					
workstation	FACOM-G	SX/G	UTILISP	75MB	8MB
personal computer	FMR	MSDOS	GCLISP		4MB
main frame	Mｼﾘｰｽﾞ	OSIV/F4 MSP	UTILISP		注1
その他	注2	SX/AR	LISPV10		12MB

注1 ﾘｰｼﾞｮﾝｻｲｽﾞ 4MB 注2 FACOM-Aｼﾘｰｽﾞ

利用環境： HUMBLE

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine					
workstation	Xerox 1161	UNIX	Smalltalk		
personal computer					
main frame					
その他					

利用環境： ES/KERNEL

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine					
workstation	2050/32	HI-UX	C	20MB	8MB
personal computer					
main frame	Mｼﾘｰｽﾞ	VOS3	C	20MB	3MB
その他	T-860/30	DPOS	C	20MB	1.5MB

利用環境： EXTKERNEL Ⅱ

	名称	ｏｓ	開発言語	最小容量 ディスク	メモリ
lisp machine					
workstation	M3300	MSDOS	C	620KB	400kB
personal computer	MULTI 16	MSDOS	C	620KB	400kB
main frame					
その他	PSI Ⅱ	SIMPOS	ESP	1MB	40MB
	MELCOM70MX	OS60/UMX	Prolog	10MB	4MB

注 機種間で若干差異あり
workstation および personal computer 用は、μ-EXTKERNEL
MELCOM70MX 用は、EXTKERNEL-R/U

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
メモリ管理：OS／独自				OS		独自			
ガーベッジコレクションの方式	環境に依存	環境に依存	環境に依存	環境に依存	環境に依存	−	環境に依存 注1	環境に依存	環境の依存
知識ベースの上限（入門／教育用の場合）									
ルール			環境に依存	9999		環境に依存	制限なし	制限なし	
フレーム			環境に依存	制限なし		環境に依存	制限なし	制限なし	
その他				制限なし			制限なし	制限なし	
合計				制限なし			制限なし	制限なし	
拡張性									
ソースプログラムの提供	×	×	○	○ (600万円)	×	×	×	×	○
推論機構の変更	×	×	△	×	○	×	×	×	△ 注8
知識表現の拡張性	×	○	△	×	○	×	×	×	△ 注8
価格等									
価格：1set	380万円		400万円	198万円	1200万円	注5	注3	380万円 注4	75万円
2set 以上	同上		280万円／SET	198万円／SET	200万～ 400万 注6	同上	同上	380万円/SET 注4	—
保守費用	なし		60万円	20万円 注2	30万～90万/年セット 注7		なし	都度相談	—

注1　UTILISP のインプリメントに依存。また、FACOM-G の場合は仮想記憶をサポートしていない。

注2　SSS（ソフトウェアのメンテナンス）契約。REVISION UP がされる（1年間）

注3　FACOM-G の ESHELL/X　70万円（一括）
　　Mｼﾘｰｽﾞ の ESHELL/X　17万円／月
　　ｽｰﾊﾟｰA の ESHELL/X　17万円／月，510万円（一括）
　　ｺﾝﾊﾟｸﾄA の ESHELL/X 100万円（一括）

注4　ＰＳＩＩの場合

注5　2050ｼﾘｰｽﾞの開発環境：100万円（一括），実行環境：25万円（一括），Mｼﾘｰｽﾞ：17万円／月

注6　2セット目以降の価格。セット数により異なる。

注7　1セット目の初年度保守費用は，KEE価格に含まれる。

注8　ST80のソースリストで提供されている為

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
トレーニング費用									
入門	都度相談		100,000円		無料	−	61,000円(４日間)	都度相談	
高度				50,000円	注１	−		都度相談	
その他			個別サポート			−		都度相談	
コンサルタント 費用／方法				別途見積		−	別途見積	都度相談	

注１　１セット目のＫＥＥ価格に含まれる。

6.6　ツールの利用目的

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
利用レベル									
AI 技術の勉強／教育	○	○	△	△	○	△	○	○	○
フィージビリティスタデェイ	○	○	○	○	○	○	△	○	△
プロトタイプ開発	○	○	△	○	○	○	○	○	△
実用システム開発	△	○	△	×	○	○	○	○	△ST80とのリンクが不可欠
利用者レベル									
記号処理言語の経験の必要性	LISP	LISP	LISP	LISP	LISP	(C)	LISP	Prolog	Smalltalk
使用者レベル									
エンドユーザ	○	×	○	△	○	○	△	△	○
知識技術者	○	○	○	○	○	○	○	○	○
人工知能研究者	○	○	△	△	○	△	△	○	△
その他							SE		
開発の委託の可能性／容易さ	○	△	○	△	○	○	○	○	△
トレーニング									
OJT				×				都度相談	
教育プログラムの有無・特徴				○	○	○	○	○	○
使用時のサポート									
On-Call	○	○	○	○	○	○	○	○	○
その他				×					

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
Version-up、バグ情報の種類、頻度				○		随時	○		
ユーザ会の有無、頻度、内容			ユーザ会 1回／年	×		×	×		
開発プロセスでの利用性									
問題の設定	×	×	×	×	×	×	×	×	×
既存技術の評価	×	×	×	×	×	×	×	×	×
知識源の同定	×	×	×	×	×	×	×	×	×
専門家モデルの同定	○	○	○	○	○	○	○	○	○
ユーザモデルの同定	○	○	○	○	○	○	○	○	○
知識表現の選択	○	○	○	○	○	○	○	○	○
知識の移植	○	○	○	○	○	○	○	○	○
知識ベース管理	○	○	○	○	○	○	○	○	○
性能評価	○	○	○	○	○	○	○	○	○
システムの拡張	○	○	○	○	○	○	○	○	○

6.7　ツールが得意とする分野

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
製品分類									
汎用ツール	○	○	○	△	○	○	○	○	×
専用ツール／アプリケーションパッケージ	×	×	×	△	×	×	×	×	×
問題のタイプ									
合成型問題：計画型（適用例）	杭工法選択	熱処理工程設計	薄板在庫ｽﾗﾌﾞ運用最適ﾎﾟｰﾄﾌｫﾘｵ選定	生産システム工程シミュレーション	製造ｽｹｼﾞｭｰﾘﾝｸﾞ、化学物質分析計画	配車計画、スーパーマーケット管理	生産計画立案、船舶出荷作業計画	組立加工工程計画、教育ｶﾘｷｭﾗﾑ作成	
設計型（適用例）	化学物質合成LSI の設計 FMSｽｹｼﾞｭｰﾘﾝｸﾞ	機械設計の材料選択	化学ﾌﾟﾗﾝﾄﾌﾟﾛｾｽ設計支援	製品材料選定	山留工法選定	山留工法選定、計算機室ﾚｲｱｳﾄ	通信制御装置設計、山留工法選定	土木基礎工法選定、プレス加工設計	
解析型問題：解釈型（適用例）									
診断型（適用例）	住宅相談、データベース性能診断、知的教育支援		ﾌﾟﾗﾝﾄ法令検索、ﾌﾟﾗﾝﾄ運転支援、ｺﾝｸﾘｰﾄひび割れ診断	電子機器の故障診断、照明法コンサルテーション	人工衛星・原子炉・交換機の故障診断	年金相談、予算査定、自動車故障診断、設備故障診断	FORTRANトラブルシューティング、異常炉況診断	自家発電設備故障診断、製造ﾌﾟﾗﾝﾄ運転保障・故障診断	○注1
制御型（適用例）					冷暖房コントロール		高炉制御		
その他									
ツールがサポートできる基本タスク									
合成型問題：計画型									
計画過程の階層化		○	○	○	○	○	○		×
組合せ的探索		○	○	○			○	○	
制約条件の相互作用	○	○	○	○	○		○	○	
環境の予測	○	○	○			○	○		
知的バックトラッキング	○	○	○	○	○	○	○	○	

注1 ほぼ、診断型だけしか扱えない

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
計画事例の検索／利用			○		○	○	○	○	
計画の評価		○	○		○	○	○	△	
その他									
設計型									×
構成要素とその関連の表現	○	○	○	○	○	○	○	○	
設計問題の階層的表現		○	○	○	○	○	○	○	
代替案の生成	○	○	○		○		○	△	
部分システムの評価	○	○	○		○	○	○	○	
知的バックトラッキング	○	○	○	○	○	○	○	○	
設計事例の検索／利用		○	○		○	○	○	○	
並列的問題解決							○	△	
その他									
解析型問題：解釈型									
特徴抽出	○	○	○		○	○	○	△	
モデルとの照合	○	○	○		○	○	○	○	
システム構造の同定	○	○	○		○	○	○	△	
システム状態の推定	○	○	○		○	○	○	○	
あい昧さの処理						○	○	△	○
不完全性の処理					○	○	○	△	
解釈の多様性の処理	○	○						△	
その他									

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
診断型									
事象の分類／階層的表現		○	○	○	○	○	○	○	○
システム構造の階層的表現	○	○	○	○	○	○	○	○	○
計測点の選択／決定	○	○	○		○	○	○	○	○
異常原因の同定	○	○	○	○	○	○	○	○	○
浅いモデル	○	○	○	○	○	○	○	○	○
深いモデル			○					△	
その他									
制御型									×
システム構造の表現	○	○	○	○	○	○	○	○	
システム動特性の表現	○	○	○	○	○	○	○	△	
状態の予測	○	○	○	○	○	○	○	△	
安定化操作			○			○	○	○	
低コスト操作		○	○			○	○	○	
ガイダンス	○	○	○	○	○	○	○	○	
その他									

ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS II	KEE	ES/KERNEL	ESHELL/X	EXTKERNEL II	HUMBLE
実績									
販売実績：国内			15ｾｯﾄ (120) 注1	３０セット		1,500セット		５０セット	
全体				－		－		－	
合計			15ｾｯﾄ (120) 注1	３０セット		1,500セット		５０セット	
実用システム：国内			0 (25) 注1	１５システム		－		１０システム	
全体				－		－		－	
合計			0 (25) 注1	１５システム		－		１０システム	
代表的システムとその特徴				METIS（機械材料疲労設計ＥＳ）		日立評論1988年11月号に特集として掲載した。		注２	
主要ユーザ				ソニーなど		（同上）		注２	

注１　（）内はSymbolics版以外の機種の実績
注２　三菱電機技報　1989年7月号に特集として掲載する

7．知識システム構築ツールで使われる用語対照表

7．知識システム構築で使われる用語対照表

用語 ＼ ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS	KEE	ES/KERNEL	ESHELL/X	EXT-KERNEL	HUMBLE
知識ベース	知識ベース	知識ベース	知識ベース	知識ベース	KB（知識ベース）	知識ベース	知識ベース	知識ベース	
フレーム（オブジェクト） クラス インスタンス	フレーム クラス インスタンス	スキーマ ﾌﾟﾛﾄﾀｲﾌﾟｽｷｰﾏ インスタンス	フレーム インスタンス	フレーム クラス インスタンス	ユニット クラス メンバ	フレーム クラス インスタンス	知識オブジェクト クラス インスタンス	スキーマ － －	
関係（意味ネットワーク） superclass of subclass of instance of part of	 super is-a instance-of part-of	リレーション is-a + inverse is-a instance	リレーション is-a instance-of component-of	関係 subclass-of instance-of －	 superclass of subclass of member of －	 上位クラス 上位クラスを持つ 上位クラスを持つ	関係 上位 下位(sub-class) 下層(sub-level) PART-OF	 直接上位スキーマ －	
インヘリタンス	遺伝	インヘリタンス	インヘリタンス	継承	継承	継承	継承(ｲﾝﾍﾘﾀﾝｽ)	継承	
スロット	フレーム変数	スロット	スロット	属性	スロット	スロット	スロット	属性	
ファセット	－	ファセット	－	ファセット	ファセット	ファセット	ファシット	－	
メソッド	メソッド	メソッド	－	メソッド	メソッド	メソッド	メソッド	－	
メッセージ	－	メッセージ	－	メッセージ	メッセージ	メッセージ	－	－	
デーモン if added if updated if needed if removed	トリガ ＰＵＴ関数 ＧＥＴ関数	デーモン Put-val Get-val Delete-val	付加手続き(ﾌｧｼｯﾄ if_added if_needed if_removed	デーモン after-added if-needed before-added	ｱｸﾃｨﾌﾞ ﾊﾞﾘｭｰ ＡＶＡＤＤ ＡＶＰＵＴ ＡＶＧＥＴ ＡＶＲＥＭ	デモン when-changed when-asked	デモン 副作用デモン 欠測値デモン 副作用デモン	デーモン 生成手続き 参照手続き 削除手続き	
ファクト（ワークエリア） ワーキングメモリの形態	ワーキングメモリ	ワーキングメモリ メモリエレメント	事象	－	非構造化事実	ビューノート プライベートメモ	－	ワーキングエリア エレメント	
ルールのグループ	ルールセット	－	知識ユニット	複合ルール	ルールクラス	ルール群	ルール集合	ルール・セット	
競合解消戦略	競合解決	 LEX MEA	知識ﾕﾆｯﾄ推論戦略 LEX MEA	適用する知識の選択	制御戦略	競合解消戦略 R戦略　RC戦略 RO戦略　RSC戦略 OR戦略	－	衝突ルールの制御 最新ＷＡＥ 優先順	
ＢＢ（Black Board）	－	－	－	－	－	黒板	黒板、ＢＢ	－	
確信度	－	－	確信度、信頼度	確信因子	－	確信度	確信度	－	
仮説	－	－	仮説	仮説	ゴール	ゴール	ゴール	仮説	
推論過程の木構造の表現	－	－	決定木	推論木	ORﾄﾚｰｽ、ANDﾄﾚｰｽ	ルール関連図	－	推論過程木	

用語 ＼ ツール名	KBMS/SUN	knowledge Craft	Super BRAINS	DG/ZEUS	KEE	ES/KERNEL	ESHELL/X	EXT-KERNEL	HUMBLE
アクション	アクション	アクション	−	−	アクション		アクション	−	
タイムタグ	タイムタグ	タイムタグ		−	−	時間属性	−	生成時刻	
アジェンダ	−	アジェンダ	−	−	アジェンダ	イベントキュー	−	−	
（独自の用語）	知識ﾍﾞｰｽｸﾞﾙｰﾌﾟ ｺﾝﾌﾘｸﾄｾｯﾄ エレメント グループ 関連 ロール 知識コンパイラ ヘッド部 エミュレータ フレームトレース クローズ状態 オープン状態 ユース状態 アンユース状態 カレント状態	ﾌﾟﾛﾀﾞｸｼｮﾝﾒﾓﾘ ｺﾝﾌﾘｸﾄｾｯﾄ ｶｰﾃﾞｨﾅﾘﾃｨ コンテキスト インバース ドメイン レンジ メタスキーマ メタスロット メタバリュー イベント メンバー セット サブセット	知識定義 データ定義 知識ベース定義 処理定義 処理手続 事象条件 仮説条件 フレーム条件 ＬＩＳＰ条件 関係木 影響木	ルールベース フレームベース 文脈クラス 文脈インスタンス 状態フレーム シミュレータ	継承方法 ﾊﾞﾘｭｰ・ｸﾗｽ ｶｰﾃﾞｨﾅﾘﾃｨ ｱｸﾃｨﾌﾞ・ｲﾒｰｼﾞ ｗｆｆ 推論ルール REINDEX	スカラ型 集合型 ﾎｰﾑﾙｰﾙ群 ｱｸﾃｨﾌﾞﾙｰﾙ群 イベント メタルール フレーム関連図 多階層強調型推論 ファジイ　注１ デモンルール群	ﾌｫｰｶｽｵﾌﾞｼﾞｪｸﾄ 属性スロット 正規化デモン 制約デモン イベント ファジィ 選択子 関係図	事実データベース 左(右)辺肯定要素 左(右)辺否定要素 外部関数 前向きルール 後向きルール ルール拡張	

注１　サポート予定

8．結論と今後の課題

8．結論と今後の課題

本報告では，さきに提案した知識システム開発ツールの評価体系を改良し，さらに，提案した性能評価問題のインプリメンテーションを通じて，ＡＩセンター・オープンハウスに導入済みのツール（KBMS/SUN, Knowledge Craft, Super BRAINS, DG/ZEUS, KEE, ES/KERNEL, ESHELL/X, EXT-KERNELの８種類）の本体系による評価を実施した．６題の性能評価問題はすべてのツールでインプリメントされ，合計４８種類のプログラムが試作された．これらの詳細な情報は，ＩＣＯＴ－ＪＩＰＤＥＣ　ＡＩセンターから入手可能であり，また，実現したプログラムはＡＩセンター・オープンハウスにおいてデモンストレーションできる状況にある．なお，富士ゼロックス社のHumbleについては，ワインドアドバイザのみをインプリメントした。

現在までのところ，このような大規模なツール評価の調査研究はあまり実施されておらず，この結果は，今後，知識システム開発ツールの調査，評価・選択，設計・開発，さらには知識技術者教育などを実施する場合の参考として役立つものと考えられる．

本報告の結論はつぎの３点にまとめられる．

（１）ツール評価項目について

６２年度報告で提案した評価項目に各ツールのデータを記述することで，これらの項目の有用性を確認することができた．評価項目の記入にあたっては対象とするツールや知識処理技術の概念に対する十分な理解が必要である．評価項目リストを記入するには，本報告７章と付録の情報が有用であろう．それには，性能評価問題の試作が非常に有用である．

（２）性能評価問題について

本報告で示した問題は，小規模ながら，解釈，設計，計画，診断という典型的な分野の問題を取扱っている点，厳密に仕様が確定している点でこの種の評価を実施するための問題としては適切である．実行性能を評価するためのルール，フレーム処理の問題は，典型的なアルゴリズムが確立していない現状では，多少不確実な要素を含むものとならざるを

えない．処理速度を測定する上で，たとえば，ルールコンパイル方法の差などを詳しく分析するには，さらに推論処理のアルゴリズムにまで立入った問題を設定する必要がある．

これらの性能評価問題は，ツールの機能／知識表現，性能といった項目を評価する上では有用なものであるが，利用者／システムインタフェースを評価する上では適切なものではない．この点については別の立場からのアプローチが必要である．

（３）評価を行った知識システム開発ツール全般について

本報告で評価の対象とした８種類のツールはいずれも大規模かつ本格的なものであるため，その使用にあたっては，記号処理言語ならびにマシンへの習熟が前提条件として必要であった．そのため，ＡＩ技術を心得た知識技術者が各問題のインプリメンテーションにたずさわっている．ところが，各ツールで用いられる概念が多いこと，さらにそれらがツールによって微妙に異なること，また，マニュアルが不完全なツールが多かったことなどが作業を進める上で大きな障害となった．このような状況は，ツールそのものの問題とも言えるが，一方では，現在の知識処理技術の未熟さを示していると考えることもできる．

性能評価問題のうち，解釈・診断問題については，どのツールを利用しても，比較的容易に，ほぼ同じ機能のプログラムが実現された．それに対し，計画・設計問題については，いずれのツールでもすなおなインプリメントが困難であり，しかもその方法は大きく異なるものとなった．これは，現在の知識処理技術で強調されている宣言的な知識の表現方法が，計画・設計問題にはほとんど無力であることを意味する．これら計算の複雑さの壁に直面しなければならない問題については，明確な仕様が与えられ，機能の豊富なツールが利用可能であっても，適切な時間内に解を導くようなプログラムを実現することは困難である．

性能評価問題を利用したツール評価の方法は，知識システムの仕様が厳密に定まっていることを前提としている．しかし，実際のシステム開発においては，知識ベースのインプリメント方法が開発のボトルネックになる場合は少ない．知識システムの仕様が明確になる以前の知識獲得問題，あるいは，システムの運用が開始された後の知識ベース保守の問題が重要である．これらについては，昭和６２年度にまとめた知識システム開発方法論の報告，ならびに，本報告と同時にまとめられた計画・設計型問題に対する知識システム開

発方法論の報告，知識獲得に関する調査研究報告を参照されることをお薦めする．適切な開発方法論と組み合せて利用することで，知識システム開発ツールの諸機能がより有用なものとなる．ただし，このような知識システム開発方法論の立場からツールを考察する研究はまだ緒についたばかりであり，今後さまざまな面で，調査研究開発を進めていくことが重要である．

謝　辞

本調査研究の実施にあたって，性能評価問題の実現にあたったヒューマンシステム（株）の方々，ならびに，御協力いただいた各ツールベンダーの関係者の方々に感謝の意を表する．

付録　ＡＩ用語解説

付録　ＡＩ用語解説

［１］　オブジェクト

内部状態と、メッセージを受けたときに行うべきこと返答すべきことについての知識、演算を持つ抽象データ型のモジュールである。

［２］　意味ネットワーク

知識表現形式の一つで、この表現形式は節点（点あるいは丸印や四角で書かれる）と、この節点を互いに結ぶ弧（または枝といい、矢印で書かれる）とからなる表記法が使用される。節点も弧もラベルを持つことができ、節点は通常問題領域の対象あるいは概念や状態を表し、弧はそれらの間の関係を表す。例えば、チータ、トラ、キリン、シマウマ、ペンギン、アホウドリ、ダチョウの７種類の動物に関する知識として、次のような規則が与えられているとする。

(1)　体毛があるかまたは授乳をするならば、それはほ乳動物である。

(2)　羽毛があるかまたは飛びかつ卵を生むならば、それは鳥である。

(3)　ほ乳動物であり、肉を食べるかまたは鋭い目と歯を持つならば、
それは肉食動物である。

(4)　ほ乳動物であり、蹄を持つかまたは反芻するならば、それは蹄足動物である。

(5)　肉食動物であり、茶色で黒い斑点があれば、それはチータである。

(6)　肉食動物であり、茶色で黒い縞があれば、それはトラである。

(7)　蹄足動物であり、茶色で黒い斑点があり、首や足が長ければ、
それはキリンである。

(8)　蹄足動物であり、白色で、黒い縞があれば、それはシマウマである。

(9)　鳥であり、色が黒と白で、飛べないが、泳げるならば、
それはペンギンである。

(10)　鳥であり、色が黒と白で、首や足が長く、飛べなければ、
それはダチョウである。

(11)　鳥であり、白色で、飛ぶのが上手であれば、その動物はアホウドリである。

これを意味ネットワークで表現すると、次のような二項関係の集合として表現される。

is-a（ほ乳動物）
is-a（鳥、動物）
is-a（肉食動物、ほ乳動物）
is-a（蹄足動物、ほ乳動物）
is-a（チータ、肉食動物）
is-a（トラ、肉食動物）
is-a（キリン、蹄足動物）
is-a（シマウマ、蹄足動物）
is-a（ダチョウ、鳥）
is-a（ペンギン、鳥）
is-a（アホウドリ、鳥）
do（ほ乳動物、授乳）
has（ほ乳動物、体毛）
do（鳥、産卵）
can（鳥、飛ぶ）
has（鳥、羽毛）
do（肉食動物、肉食）
has（肉食動物、鋭い目と歯）

do（蹄足動物、反趨）

has（蹄足動物、蹄）

color（チータ、茶色）

texture（チータ、黒い斑点）

color（トラ、茶色）

texture（トラ、黒い縞）

has（キリン、長い首と足）

color（キリン、茶色）

color（シマウマ、白色）

texture（シマウマ、黒い縞）

cannot（ダチョウ、泳ぐ）

has（ダチョウ、長い首と足）

color（ダチョウ、黒と白）

cannot（ペンギン、飛ぶ）

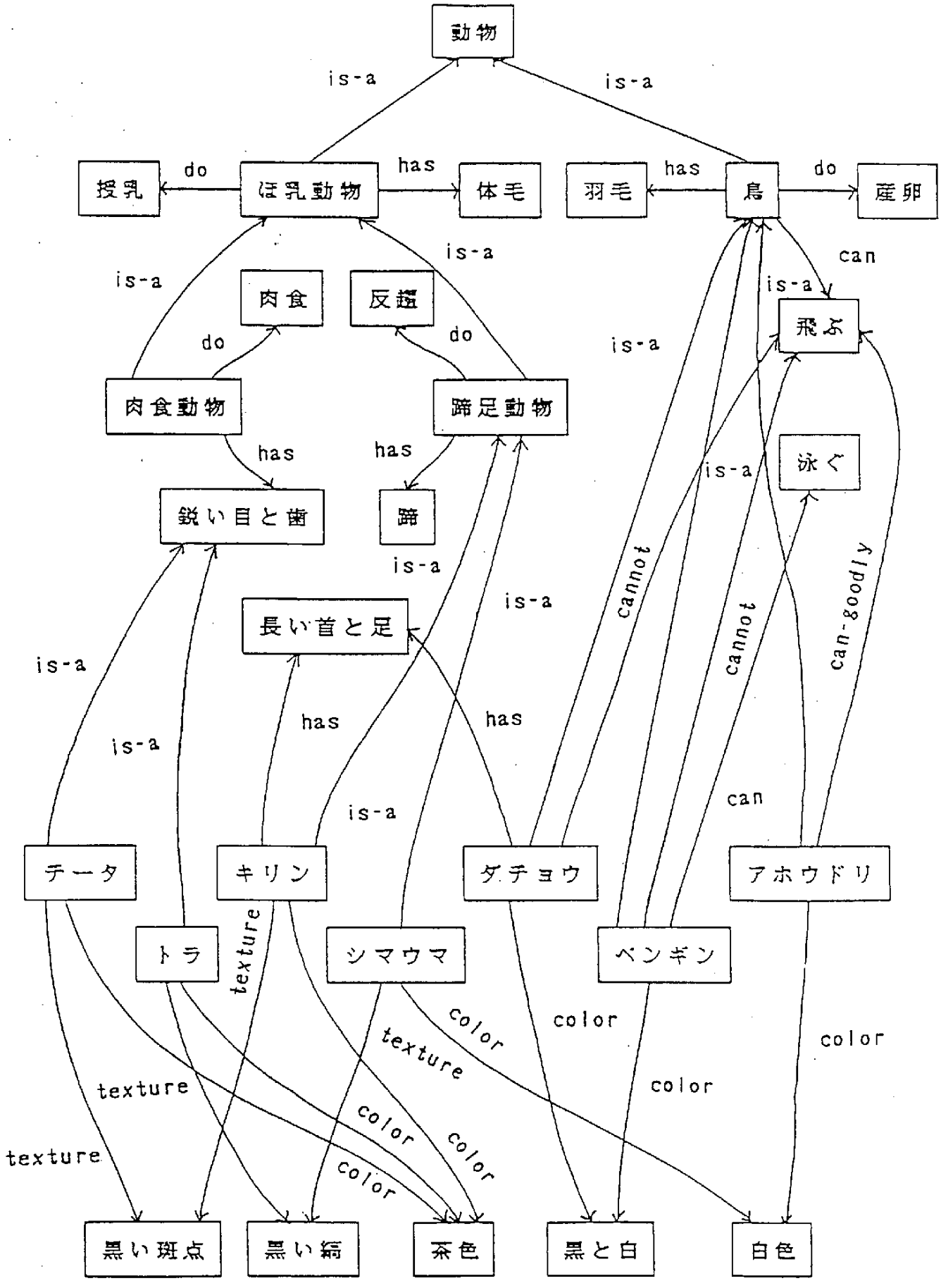
can（ペンギン、泳ぐ）

color（ペンギン、黒と白）

can-goodly（アホウドリ、飛ぶ）

color（アホウドリ、白色）

これらを図で表現すると次の図のようになる。この図から明かになるように、意味ネットワークではリンクを順にたどることにより、次々と関連する情報を抽出することが容易に行える。


動物
is-a
is-a
授乳
do
ほ乳動物
has
体毛
羽毛
has
鳥
do
産卵
is-a
is-a
can
肉食
反芻
do
do
飛ぶ
is-a
肉食動物
蹄足動物
has
has
泳ぐ
is-a
鋭い目と歯
蹄
is-a
長い首と足
is-a
cannot
cannot
can-goodly
is-a
has
has
is-a
can
テータ
キリン
ダチョウ
アホウドリ
is-a
トラ
texture
シマウマ
ペンギン
color
color
texture
color
texture
color
color
color
color
color
texture
黒い斑点
黒い縞
茶色
黒と白
白色

［３］　フレーム

ミンスキーによって最初に提案され、視覚や自然言語による対話、あるいはその他の複雑な行動を理解するための基礎としたものである。人間の記憶はフレームと呼ばれる枠組みを単位として構造化されており、ある場面に出会ったとき、人は記憶の中からその場面によって得られた情報を手がかりに適当なフレームを選び出し細部を一致させるように推論を行うという理論に基づく。しかし、もし十分な一致が得られなければこのフレームが持っている情報に基づいて、他の適切なフレームを選択して照合を繰り返す、などの情報処理をフレームが持っている手続き的知識を使って実行するようにもなっている。

［４］　関係

対象間の関係に関する知識とは、二つまたはそれ以上の対象の間に成立している関係の記述である。例えば、「太郎と花子は夫婦である」という言明は、対象”太郎”と対象”花子”の間に”夫婦である”という関係が存在することを意味している。

［５］　手続き（的知識）

車の運転の仕方や料理の作り方のように、ある目的を遂行するための一連の行為を記述するには、個別知識の起動条件や順序関係など手順に関する知識を表現する必要がある。これを手続き（的知識）と呼び、その表現されたもの自体を指す。フレームにおいては、静的な事実の表現を実現する方法としての宣言的構造の基礎に加えて、フレームに基づくシステムの動的な、つまり手続き的な知識を含む重要な側面がある。一般に「手続き」と「宣言」は対比して用いられる。普通の計算機プログラムも手続きである。

［６］　継承機能

知識ベースシステムが対象とする分野で用いられる、様々な概念の間には、様々な関係の一つとして、概念間の階層関係がある。例えば「人間」という概念と「ほ乳動物」という概念を考えるならば、集合としてみた場合に前者はすべて後者に含まれるという意味で、後者は前者より一般的な、または階層の上位の概念であるということができる。概念間にこのような階層関係が成立しているとき、上位の概念の持つ属性は、そのまま下位の概念にも引き継がれていることが多い。例えばほ乳動物一般に成立する「乳で子を育てる」という属性は、当然人間においても成立している。このように上位の概念で成立する属性がそのまま（あるいは多少形を変えて）下位の概念でも成立する場合、その属性は概念の階層関係を通じて下位の概念に継承するといわれる。

［７］　ｐａｒｔ－ｏｆ

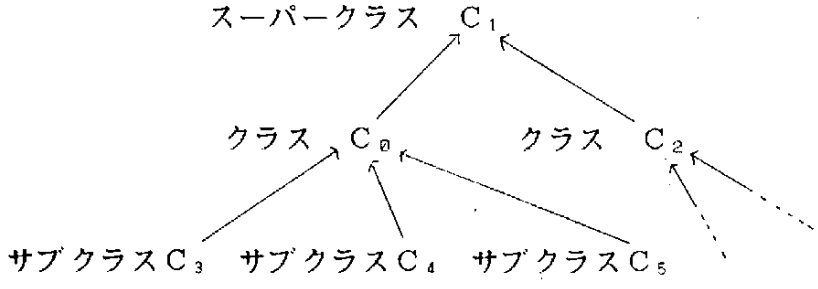
全体－部分関係。階層的知識表現において、より全体を表す対象を上位にその部分を表す対象を下位におく関係である。例えば「指は人間の一部か？」という質問に対し、「指は手の一部である」と「手は人の一部である」という二つの蓄積された知識と節点間の part-of（～の一部）という枝をたどる専用推論手続きとを使用することにより答えることができる。つまり part-of は節点と節点（ここでは HAND と FINGER）を互いに結ぶ枝であり、枝は節点の関係を表す。

［８］　その他

has-part のような全体－部分関係であるもの。抽象－具体関係（例えば is-a または a-kind-of）においては、上下のフレームは共通の性質を持っているが、全体－部分関係においては下位フレームは上位フレームの部分構造であるため、性格が全く異なり、継承機能を持たない。

［9］ class-subclass

継承関係において、あるクラスの上位のクラスをスーパークラス、下位のクラスをサブクラスと呼ぶ。サブクラスは、あらゆる点でクラスと同一で、つまり自分のサブクラスを持つこともできる。各クラスは一つのスーパークラスを持ち（多くのクラスが同一のスーパークラスを共有することもある）、したがってクラスはピラミッド型の構造を形成している。この前後関係は、当該クラスのスーパークラスから始まり、さらにそのスーパークラスへと続いていく。クラス間のスーパー・サブの関係を図に表すと、下図のような木構造になる。



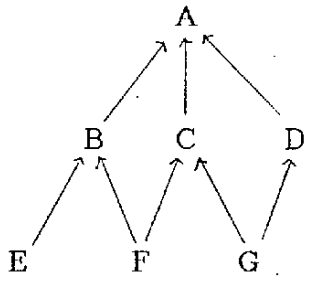
クラスC_0のスーパークラスは、C_1であり（C_1のみがC_0のスーパークラスである）、またC_0自身はC_3、C_4、C_5という三つのクラスのそれぞれにとっての唯一のスーパークラスである。

［10］ class-instance

クラスによって記述されたオブジェクトの一つで、メモリーを持ち、メッセージに応答する。遊園地を例にとるとクラス Ride が作れ、Ride のインスタンスは、メリーゴーランドや、ローラーコースターになる。それぞれの Ride は何らかのサービスを求めて並ぶ各々の待ち行列を持ち、これはクラスのインスタンス（適切な種類のデータ構造を記述する）であるべきものである（このデータ構造は命じられた客の並びに配列する）。全てのものはオブジェクトで、各オブジェクトはクラスのインスタンスなので、客はクラスのインスタンスに相違なく、そのクラスは Park Visitor となる。

［１１］　多重継承

意味ネットワークや、フレーム表現の最大の特長は、is-a リンクによる性質の継承であるが、図のように一つのノードが複数の親ノードを持ち、それから性質を継承することを多重継承という。この場合に複数の親ノードから同一の属性について継承を行う場合に、値をどういう方法で決定するかが問題となる。一般に、すべての値を継承する（マルチバリューとなる）ことや、さらに位置の近い親から順に値を持つことが多い。この他にも逆順で持ったり、あるいはどれか一つの親（近い位置の）の値を継承する方法などがある。



［１２］　スロット

フレームなどの知識表現の枠組みを形成する要素であり、属性とも言われる。すなわち、一つのフレームは複数個のスロットにより構成される。各スロットは、そのスロットが属するフレーム内でユニークな名前を持つ。また、各スロットは値を持つが、数値、テキスト、式、テーブルなどに加えてフレーム名（すなわち他のフレームへのポインタ）や手続き名（すなわち付加手続き）などが値として許されるので、色々な情報の表現や操作を可能とする。最近はスロット値として、PROLOG のホーン節の集合やプロダクション・ルールの集合などを許すものも現れている。またデフォルトやデモンがスロットに付加できることも多く、高度な推論を比較的容易に実現できるように工夫されている。またスロット自身も、普通はより小さな部分フレームとなっている。

［１３］ シングルバリュー

各々のものはいくつかの属性によって特徴づけられているが、その属性の中でとる値が一つしかないものをいう。例えば、太郎の誕生日は、1981年8月3日というもので、これはこの値しかとれないものである。

［１４］ マルチバリュー

シングルバリューとは逆に、いくつかの属性値がある場合をいう。例えば、乗用車の色は、「白、赤、青、黄、緑、黒」などである。

［１５］ ファセット

フレームでは、スロットの値が未知のとき上位フレームの性質を継承することを基本とするが、それが不可能な場合には、スロットの値を決定するための代替的な方法つまり、そのことを表現するための記述子が要請される。さらにスロットの値についてもデータのタイプ、データが取り得る範囲及び値が未知なとき、矛盾がなければ暗黙のうちに仮定してもよいデフォルト値などを表現するための記述子を必要とする。このような要請に応えるために、多くのフレームシステムでは、サブスロットを導入しており、これがファセットと呼ばれるものである。ファセットには、多くの種類が考えられているが、代表的なファセットとして次のものがある。

1) value ファセット：

スロットの値を表す。

2) data-type ファセット：

スロットのデータタイプを表す。

3) range ファセット：

スロットの値が取り得る範囲を表す。

4) default ファセット：

スロットのデフォルト値を表す。

ファセットの使用例として次のようなものがある。

```
frame:   乗用車
 self:   value:   (a-kind-of  自動車)
maker:   value:
         data-type:   文字列
         range:   [トヨタ  日産  ホンダ  三菱  いすず  ダイハツ]
         default:   トヨタ
color:   value:
         data-type:   文字列
         range:   [白  赤  青  黄  緑  黒]
         default:   白
```

［１６］ 情報隠ぺい

オブジェクトに対してメッセージを送る際、オブジェクトの内部動作をいちいち気にせずオブジェクトに対するメッセージ伝達法（とそれに対する反応）だけを知っていればよい。このように相手のオブジェクトを完全にブラックボックス化して、内部状態や内部構造を見えなくすることを、情報隠ぺいと呼ぶ。オブジェクトをブラックボックスとして見たとき、外から見えるのはそのオブジェクトに送ることができるメッセージの形とメッセージに対する反応だけである。

［１７］ 付加手続き

フレームやユニットに付加された手続き的知識。LISP プログラム、プロダクションルール、デーモンなどによって実現される。例えば太郎の誕生日のユニットのスロット data を次のように書くものとする。

```
data:   DAY 125
```

この DAY125 は別のユニットの名前である。それは

```
DAY 125
    self:  (ELEMENT-OF DAY)
    year:  1981
    month:  8
    day:  3
    day of week:  MONDAY
```

のように表される。一般に DAY ユニットのスロット値が全て与えられるとは限らない。また文章から情報を抽出してスロットを埋めていく場合には、誤った値が入ることを防ぐ必要がある。そこで、次に示す DAY ユニットを用意しておくことにする。

```
x | DAY

    year:  (y | INTEGER)
    month:  (when-filled(check MONTH))
    day:  (when-filled(check-day))
    day-of-week:  (to-fill(get-day-of-week))
```

スロット year の値は整数であることを要求する。スロット month には、もしその値が指定されたなら、確かに月を表す数（１から１２まで）であるかどうかを調べることが示してある。 day には年と月がわかっている場合に、指定された日付が矛盾しないか（例えば1981年2月29日）を調べる手続きを呼ぶことが書かれている。最後のスロットは、もしその値が必要ならば年月日から計算して求める手続き get-day-of-week を呼ぶことを示している。もちろん、スロット値が与えられればそれを採用する。このように、まとまった一つの手続きをユニットの中に埋め込むことができる。ユニットを参照する他のユニット、あるいはプログラムは、手続きが埋め込まれているかどうかを知らなくてもよい。このような手続きの埋め込みを付加手続きと呼ぶ。

［１８］ メソッド

オブジェクトがメッセージを受け取ったときにどういう動作をすべきであるかを記述したものをメソッドと呼ぶ。メソッドは、そのクラスに属するオブジェクトに共通である。したがってメソッドは、個々のオブジェクトにではなく、そのオブジェクトの属するクラスに登録されている。

［１９］ メソッドの継承

各クラスは、そのスーパークラスの持っているメソッドをスロット同様に受け継ぐ。さらにそれらは下位のクラスにも受け継がれる。

［２０］ 起動操作

例えば、フレーム型知識表現言語 FRL では付加手続きが、スロット値に対するデーモンとして結びつけられているので、スロット値に関する操作が行われたとき自動的に必要な手続きが起動される。この例として、下図に示されるようなフレームに対し、年齢が参照されたとする。この例ではまだ値がセットされていないので if-neededに対する付加手続き ask が自動的に起動され、数値を入力するとこれがスロット値としてセットされるとともに、参照に対する結果の値として返される。さらに、値がセットされたとき、if-added に対する付加手続き check が起動されて、妥当な値か否かのチェックを行う。

スロット名	スロット値	if-needed	if-added	if-removed
年齢	NIL	ASK	CHECK	
...				

フレーム型知識表現言語 FMS では、付加手続きはスロット値の一種であるので、このフレームに対して外部からのメッセージが送られて付加手続き ask が起動される。FMS における推論の進め方は、このメッセージ送受信とそれに対する応答の繰り返し、もしくは連

［２１］　前向き推論

得られたデータは証拠から出発し、そこから導かれる結論、ゴール状態を知識を適用しながら探す推論法。データ駆動型推論、ボトムアップ推論ともいう。言語の構文解析では、単語から出発し、文法規則に従って名詞節とか動詞節とかの大きな概念に変換していき、最終的に文を構成するかどうかを調べるのが相当する。画像理解では画像中の簡単な要素から出発し、知識に従って次第に大きな単位にまとめ上げていき、目標とする対象のモデルに到達するかどうか、調べるのが相当する。

［２２］　後向き推論

ゴール状態から出発し、関連知識に従ってこれを達成するのに必要な仮説またはサブゴールを作り、これを繰り返して最終的に仮説を満たす状態にあるか否か検証する推論法。ゴール指向型推論、トップダウン推論ともいう。言語の構文解析では、文を仮定し、文法規則に従って次第に小さい単位へと分割していき、最終的に文法に従う単語の並びになっているかどうか検証することに相当する。画像理解では目標物のモデルを仮定し、知識に従って小さい単位に分解していき、それを構成する要素が画像中に存在するか否か調べるのが相当する。ゴールの数が1または少数のときは、通常前向き推論より速く答えが出る。

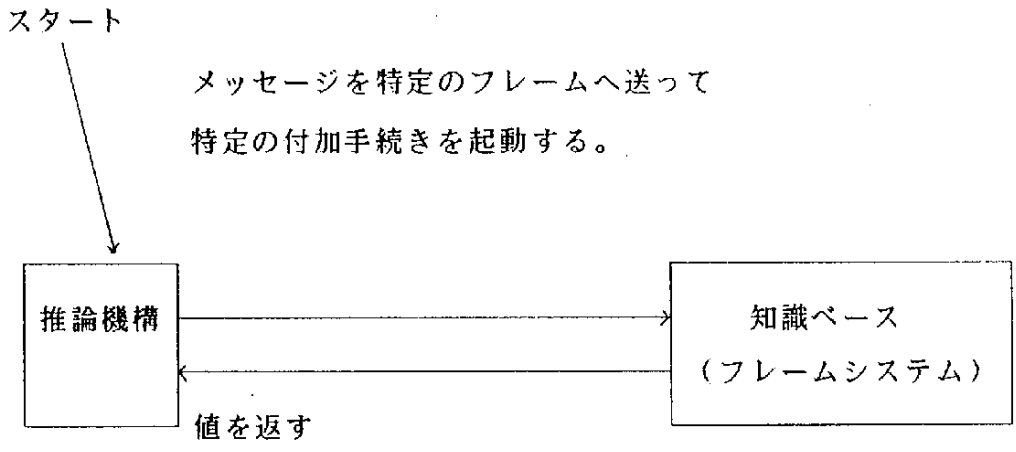
［２３］　前向き、後向き両方向

前向き推論と後向き推論を組み合わせた推論法。得られたデータから前向き推論により仮説を形成し、これを後向き推論で検証するといった混合形の推論法や、ゴール状態と現状態から同時に探索を進め、中間状態で照合をとるといった推論法。

［２４］　ａｎｄ結合子

話したり考えたりすることの多くは単純な命題を組み合わせた命題で表現できる。各々の単純な命題を組み合わせるのに用いられるのが結合子である。and 結合子は記号で△を意味している。例えば、X△Y（X と Y は命題）は、「X が真でかつ、Y が真ならば真、それ以外の場合は偽である」ということを示すものである。

鎖である。すなわち、基本的には下図のようにして実行される。つまり推論機構（に相当する付加手続き）から特定のフレームにメッセージが送られると、そのフレームの指定された付加手続きが起動される。そして実行の結果としての値が返されると、推論機構ではその値を評価して、次に起動すべき付加手続きを決定し、それを持つフレームへ引数とともにメッセージを送る。このようにしてゴール状態へと移行していく。ただし、メッセージを受け取ったフレームでは、そのフレーム内の情報だけでは処理に応じられないときは、さらに別のフレームへメッセージを送ることとなる。このときは、一時的にこのフレームの付加手続きが推論機構として働く。



FMS においてはメッセージを送ることによって付加手続きが起動される。また、推論機構も付加手続きとして実現されるので、スタートとは特定のフレームへのメッセージの送信である。

［２５］　ｏｒ結合子

and 結合子と同様、命題を組み合わせるのに用いられる。or 結合子は▽を意味し、X▽Y は、「X が真、または Y が真、または X と Y の両方が真ならば真」ということを示している。

［２６］　その他の結合子

結合子には、上記の and と or の他に not（¬）、implies（→または）、equivalent（≡）がある。これはそれぞれ以下のことを示している。

¬ X　　X が真なら偽、X が偽なら真

X→Y　　「X が真だと仮定するなら、Y は真であるに違いない」という考え方を命題論理で表したもの。つまり X が真であることは、Y が真であることを包含（imply）する。われわれは日常の会話でも「もし Jenny が生まれて９カ月だとすれば、その子には計算は無理だ」というようにこの考え方を使っている。「X→Y」の真理値の定義は、Y が真または X が偽ならば真。

X≡Y　　X と Y の両方が真、または X と Y の両方が偽ならば真。
X と Y が互いに異なる真理値を持つ場合は偽。

［２７］　論理依存性

論理プログラムにおける実行過程は、同一の節に対する単一化であっても、結果は異なることがある。すなわち、論理の適用はその導出経路に依存して決定される。また論理プログラムの構造において依存性が強いほど並列性が低い。

［２８］　論理述語による制約

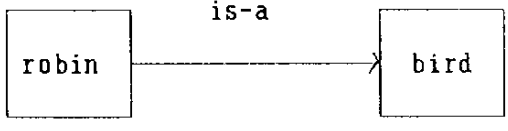
例えば、Prolog における節は各々の結果に対する制約ということができる。単一化は、その制約を解決することになる。

［２９］　パターン照合の対象データ型と方法

変数に値を代入する操作に対して、構造データ同士の比較を行い、その中に含まれる変数に値が代入される機構をパターン照合と呼ぶ。例えば、（１２３）と（ｘ２ｙ）という二つのリストに対してパターン照合が行われるとｘには1が、ｙには3が代入されこのパターン照合は成功する。PROLOG のプログラムの実行機構はパターン照合によっているが、他のパターン照合と比較して能力が強力である。例えば、パターン照合により、まずｘが（１ｙ２）という値を代入されたとする。この次に（１（３）２）というリストとパターン照合されると、ｙに（３）が代入されその結果ｘの値が（１（３）２）となる。このように、パターン照合の結果に未定義のものが含まれてもよいし、未定義のものが後で値を持つようになってよい。PROLOG が自然言語処理やデータベースに向いている1つの理由はこの機能による。

［３０］　ｉｓ－ａ

抽象－具体関係。階層的知識表現において抽象性の高い対象を上位に具体性の高い対象を下位に置く関係。例えば「コマドリはすべて鳥である」という事実は、「コマドリ（robin）」と「鳥（bird）」を表す二つの節点とそれを結ぶ弧で下面のように書いて表される。



［３１］　仮説

分析型意思決定問題における結論の候補となる項目あるいは命題。例えば医学診断支援システムでの病名がこれに当たる。診断手続きの中には、仮説の失敗回数や測定器の信頼性に関する仮定を用いるものが多く、このため推論を進める途中で、そこで用いた仮定のいくつかが不当であったことが判明することがある。このような場合には、仮説に基づいた効果を打ち消してもとの状態に戻さなければならない。さらに患者の監視、診断作業は、時間がたつにつれて変化する状況を扱わなければならない。ある疾患が自然にまかせておけば回復するのか、または何らかの処置が必要であるのかなどの判断には時間のファクタが重要である。そして測定器はしばしばノイズのあるデータをそのまま集めてしまう。そのため、データの解釈や診断、監視のように測定結果に基づいて推論しなければならない分野では、ノイズのあるデータの扱い方が重要となってくる。

［３２］　デーモン

特定の条件が生じたときに自動的にそれに対応する手続きを起動するしかけをデーモンという。典型的なデーモンの例としては、フレームの構成要素であるスロットに付加された IF-NEEDED, IF-ADDED, IF-REMOVED などのデーモン機構がある。これらは付加手続きの一種であり、スロット値に対する操作に対応して自動的に起動される。例えば、AGE という名前を持つスロットに IF-NEEDED デーモンとして ASK-TO-USER という手続きが準備されているとする。もし AGE の値が参照されたときまだ未定義であれば、ASK-TO-USER という手続きが起動されて、”年齢は何歳ですか？”などの質問が表示され、入力値がこのスロットの値としてセットされるとともに、参照先へその値が返される。また、パターン照合によって実行部を起動するという意味では、プロダクション・ルールもデーモンの一種である。

［３３］　実行の保留機能

処理がある条件下で働くよう、実行を一時的に保留すること。

［３４］　ルールのグループ化

仮説を階層化し、仮説レベルあるいはレベル間に作用するように準備されるもの。

［３５］　ブラウズ

これまでのソフトウェアシステムでは、ユーザーがいま必要とする機能がどういう位置づけでシステムのどこにあるのか見つけることは大変であったが、ブラウザはウィンドウを用いたシステムでクラスや、メソッドの定義を見たり追加したり、修正したりすることができる。ブラウズとは、ブラウズが階層構造で索引可能な情報を、順次アクセスできるビューであるのに対し、ブラウズとはそのうち当該情報の参照のみを行うことである。

［３６］　一貫性管理

知識ベースの完全性および無矛盾性が確保されるように、知識ベースの管理を行う。

［３７］　知識ベースの世代管理

編集済みの知識を保持するだけでなく、過去に投入した知識の状態に戻ったり、また変更部分や変更差分を表示できるというものである。

［３８］　字句解析

日本語の活用規則や単語相互の連結の規則は伝統的な国文法で整理されており、これらは辞書の中に記述されているものである。これらは次の2段階で行われる。

(1)　文字列からの辞書引き

（この段階では、図1のようにいくつもの可能な単語分割の仕方がある）

(2)　単語相互の連結規則を使って、解釈の可能性を絞る

(2)の連結規則は、「助動詞『ない』の前の語は用言の未然形でなければならない」といったような、前と後にどのような品詞、活用形の単語が連続し得るかに付いて各品詞ごとの拘束条件として定式化できる。しかしこの拘束条件は、解釈を一つに決定するには十分でない。例えば、

「彼は教師で学校へ行っている」

「彼はバスで学校へ行っている」

では、上の「で」は助動詞「だ」の連用形、下の「で」は格助詞の「で」と解釈されるのが正しいが、連結規則という局所的な拘束条件だけでは一意に決定することはできない。また、ひらがな書きが多用されると、一つの文に対していくつもの解釈が生じる（図2）。これらの解釈はいずれもあり得る解釈であるが、このように多数の解釈をこれ以後の文法

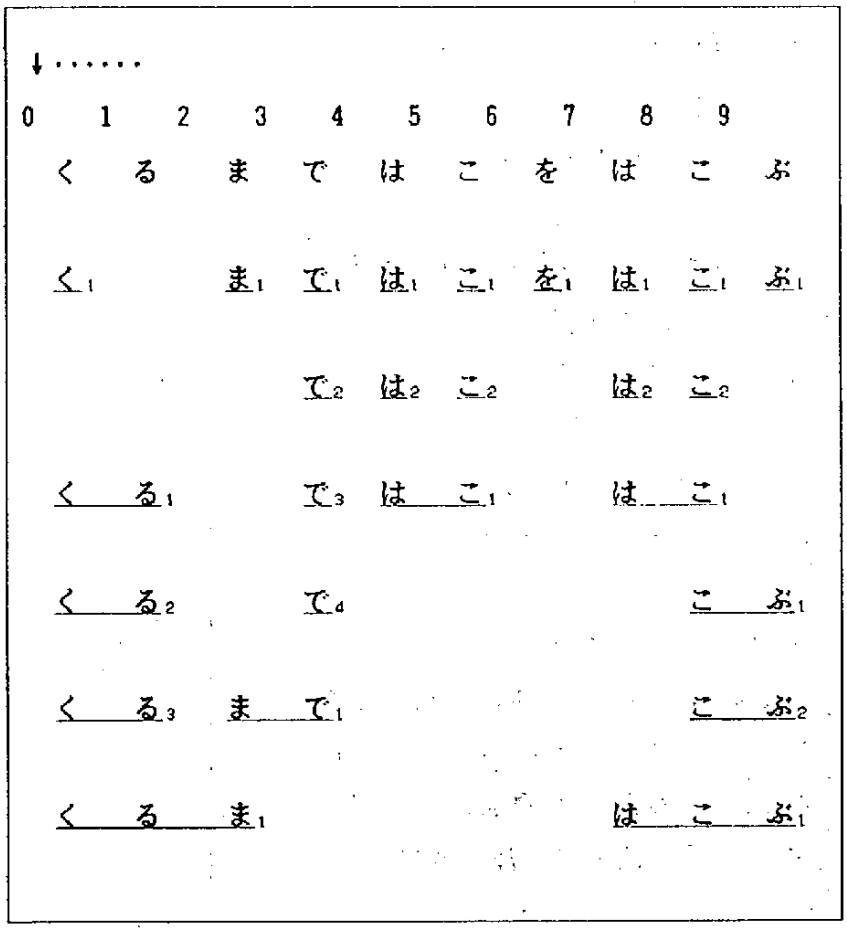
処理に渡すことは、処理効率上不可能であるので、種々の発見的基準で正解の可能性の高い解釈のみを残して、以後の処理に引き渡すのが普通である。このような発見的基準として代表的なものに、

a, 最長一致法

b, 最小文節優先法

がある。a が最も一般的な手法であるが、b はこれを一般化したもので処理速度は遅くなるが、正解の精度は高いといわれている。

(a) 入力文から単語を先頭から切り出してゆく。



(b) 切り出された文字列に対しても、いくつもの単語の解釈がある。

く$_1$：（句・苦・区、名詞、・・・）
くる$_1$：（来る、カ変、終止・連体）
くる$_2$：（繰る、5段、終止・連体）
くる$_3$：（くる、名詞、・・・）
くるま$_1$：（車、名詞、・・・）
ま$_1$：（間・真、名詞、・・・）
まで$_1$：（まで、副助詞、・・・）

で$_1$：（出、名詞、・・・）
で$_2$：（出る、下一、連用）
で$_3$：（で、格助詞、・・・）
で$_4$：（で、助動詞、連用）

は$_1$：（葉・刃・歯・端・羽、名詞、・・・）
は$_2$：（は、係助詞、・・・）
はこ$_1$：（箱、名詞、・・・）

こ$_1$：（子・粉・蚕、名詞、・・・）
こ$_2$：（来る、カ変、未然）

を$_1$：（を、格助詞、・・・）
はこぶ$_1$：（運ぶ、5段、終止・連体）
こぶ$_1$：（こぶ・昆布、名詞、・・・）
こぶ$_2$：（鼓舞、サ変動詞、・・・）
ぶ$_1$：（分・武・歩・部、名詞、・・・）

図1

入力ひらがな列：くるまではこをはこぶ。

1	車　で	箱を	運ぶ
2	車　で　は	子を	運ぶ
3	来るまで	箱を	運ぶ
4	来るまでは	子を	運ぶ
5	繰るまで	箱を	運ぶ
6	繰るまでは	子を	運ぶ
7	くる　間で	箱を	運ぶ
8	くる　間では	子を	運ぶ
9	来る　間で	箱を	運ぶ
10	来る　間では	子を	運ぶ
11	くるまで	菓子を	運ぶ
12	来るまで	菓子を	運ぶ
13	車で	菓子を	運ぶ
・			
・			
・			
85	来る　間　出	菓子を	運ぶ

図2　解析の結果

［３９］　構文解析

自然言語の文がどのような部分要素から構成されるか、あるいは、その構成要素がより下位のどのような部分要素から構成されるかを表現するのに、文脈自由形文法の書換え規則がよく使われる。例えば、英語の文構造は次のような規則の集合で規定できる。

1. S → NP VP
2. S → Wh VP?
3. S → VP
4. NP → N
5. NP → ADJ NP
6. VP → V NP
7. VP → V
8. NP → NP PP
9. PP → P NP

ここで、S. NP. VP. PP などの記号は、それぞれ文、名詞句、動詞句、前置詞句という複合的な対象物を示している。上のような書換え規則は、この複合的な対象物のそれぞれについて、部分要素の配列が異なる複数個のモデルを定義していると考えることができる。上記の規則は、例えば次のように読むことができる。

i)　文は、名詞句と動詞句という2つの部分要素とピリオドがこの順序で配列されているか（1）、Wh 語（Who, which など）と動詞句と疑問符がこの順序で配列されているか（2）、動詞句とピリオドがこの順序で配列されているか　（3）、である。

ii)　動詞句という部分要素は、動詞と呼ぶ部分要素と名詞句と呼ぶ部分要素とがこの順序で配列されているか（6）、あるいは、動詞と呼ぶ部分要素だけからできている（7）。

入力文がこのような配列規則の中のどの配列規則に従って、全体として文にまとまっているかを明らかにすることを、構文解析という。

［４０］　意味解析

自然言語理解において文法規則や他の知識を用いて入力文（単語の一次元的列）中の単語の機能を決定することである。この知識には次のようなものがある。

1)　語の活用形などに関する知識

例えば、「動詞には五段活用、下一段活用、上一段活用があってそれぞれ未然形、連用形、終止形、仮定形、命令形の活用形を持つ。助動詞や助詞は、特定の品詞の特定の活用形にしか付かない」といった中学校の文法で習うような知識がある。この種の知識は、形態素に関する知識といわれ、単語の切れ目がスペース等によって区切られていない日本語を処理するためには、重要な知識となる。

2)　文法に関する知識

我々は、どのような単語の並び方が日本語の文として認められるかを知っている。例えば、1つ1つの単語は日本語の単語であっても、

「教師で行く学校へ彼は」

「花子が太郎を醜い愛している美しい」

（cf.　美しい花子が醜い太郎を愛している）

といった文は、日本語の文として破格であると判断できる。このような知識（統語的知識）は、「単語の並びが文であるかどうか」を判断するだけではない。例えば、

「賢い太郎が愚かな花子を愛している」

「愚かな太郎が賢い花子を愛している」

といった2つの文では、同じ単語が使われているが、「誰が賢くって、誰が愚かであるか」の解釈は明かに異なる。

3)　単語が支持する現実世界の事物に関する知識

例えば、

「太郎は川で泳いでいる花子を見た」

「太郎は橋で泳いでいる花子を見た」

という文では、普通の人間であれば、上の文の「川で」は「花子が泳いでいる場所」を表現しているのに対して、下の文の「橋で」は「太郎が見ている場所」を表現していることが極く自然に判る。単語の「意味」を考えないと、「川で」も「橋で」も、ともに「名詞＋格助詞デ」の形をしており、統語構造の差を認識することはできない。人間がこの差を認識して、２つの文に対して正しい解釈ができるのは、「川で泳ぐ」ことはできても、「橋で泳ぐ」ことはできないと言った「現実世界での物の事との関係」の知識を持っているためである。

4) ユーザについての知識

ある質問を発するときには、その質問が意味を持つための前提となる条件が満たされていなければならない。例えば、データベース・システムに対して、

「昨年、吉田先生の自然言語処理で落第した学生は、何人いますか？」

という質問を人間がした場合には、その人は昨年度に吉田先生の自然言語処理の授業が開講されたことを前提にしている。したがって、もしこの前提条件が満足されていなければ、その人は誤解をしていることになる。授業そのものが存在しなかった場合には、当然だれも落第していないので、普通にデータベースの検索結果を表示すると、

「０人です」

となるが、これでは質問者の誤解はそのまま放置されてしまう。システムはこの場合、

「昨年度は、吉田先生の自然言語の授業は開講されませんでした」

と答えることが望ましい。このように User-friendly な自然言語フロント・エンドを実現するためには、「使用者（User）が何を知っていて、何を知りたがっているか」など使用者に関する知識を持っていなければならない。

［４１］　文脈解析

ある表現がどのような場で使われたか、また、発話の場面と自然言語の解釈との間にどのような規則性があるか、などをを明らかにするもの。

［４２］　アジェンダ

処理が中断されている活動（普通、様々な知識片の適用）の優先度つきリスト。黒板において実行待ちの潜在アクションを記録しているもので、それは一般に先に黒板に書かれたいくつかの決定に適用できそうな知識ベース上の規則に対応している。

［４３］　ジャスティフィケーション

節点の組合せから節点への推論段階を表す（より正確には、節点の組合せの意味するものより、他の節点の意味するものへ）ものである。最も単純なジャスティフィケーションは、その先行節点のリストである。例えば、プログラムか知識ベースに次の項目を持つとする。

Node-1　患者は風邪を引いている

Node-2　患者は風邪を引くとくしゃみをする

Node-3　もしAでかつAがBを意味するなら、Bである

ここで Node-1 と Node-2 は事実であり、Node-3 は推論規則である。このプログラムはそうすると、Node-4 をラベルされた新たなステートメント「患者はくしゃみをする」を推論し、Node-4 のジャスティフィケーションとして（Node-1, Node-2, Node-3）というリストを記録する。

［４４］　ＷＨＹ

例えば、次のような MYCIN のコンサルテーションの過程で臨床医が MYCIN の質問の背景となる推論過程の説明を受けたいと考えたとする。

1)　患者名：　　（医師による応答の先頭に**が付けられている）

**　PT538

2)　年齢

**　34ｻｲ

3)　性別

** ｵﾄｺ

4) 肌色

** ｺｰｶｼｱﾝ

.

.

.

14) 患者538は最近持続性の頭痛あるは他の異常な神経性症状（目まい、昏睡症状など）を感じたことがありますか。

** ﾊｲ

15) 患者538は最近異常な神経症の徴候（首筋の剛直、昏睡、発作など）を検査あるいは調査で示しましたか。

** ﾊｲ

16) 脳膜炎の臨床徴候（所見、うったえ、あるいは医学検査）が最初に発見された日時を入力して下さい。

** 77-1-29 22:15

.

.

.

27) 患者538は異常な神経症状が何日続きましたか。

** 7.2 ｼﾞｶﾝ

28) 患者538のCBC測定値を知っていますか。

** ﾊｲ

.

.

.

35) 脳膜炎は病院で感染しましたか。

** WHY(ﾅｾﾞﾃﾞｽｶ)　{ユーザは MYCIN が行った推論の筋道を提示するように "WHY" と聞いた。システムはまずこの質問を文章化し、次に現在評価中のルールの条件部を示す。}

[i.e. この患者が病院で感染したかどうかを決めることがなぜ（WHY）重要なのですか]

[2.0] これは（培養菌やしみに見られるものではなく）感染の原因となる細菌を決定する助けになろう。

次のことがらは既に判明している。

[2.1] 治療に必要な感染は脳膜炎である、そして

[2.2] 細胞はこのCSF培養菌の染色剤に関しては見られない、そして

[2.3] この感染のタイプはバクテリア型である、そして

[2.4] 患者538の病歴からは神経外科が関係していない、そして

[2.5] 患者538の年齢は10日を超えている

したがって、もし

[2.6] この患者が入院中に感染したとする

ならば

次の疑いがある、すなわち、（培養菌やしみに関して見られるものではなく）感染の原因となる可能性のあるこの細菌は、

E. coli (.75), staphylococcus-coag-pos (.3),

pseudomonas-aeruginosa (.3), klebsiella-pneumoniae (.5)

である

{各結論の後のかっこ内の数値はその結論の重要度を示す。この数値は頻度と考慮の有効性を合わせたものである。}

このようにWHY（ナゼデスカ）は推論の道筋つまり上位のノードを分析して説明するものである。

［４５］ HOW

上記の MYCIN のコンサルテーションの過程において WHY が上位のノードを分析して説明するのと対象的に、HOW はノードの下位に展開済みのまたはこれから展開される可能性のあるノードについて説明する。

** HOW 2.3 (2.3ハイカニシテカ)

{この時点でユーザは上に示されたルール中の2.3の条件がなぜすでに満たされているのかを重ねて聞いている。MYCIN はこの結論を得るために用いられたルールのすべてを提示し、要求があればプリントする}

[i.e. この感染のタイプがバクテリア型であるということをいかに（HOW）決定したのか]

次のルールがこの結論を出した：

この感染のタイプはバクテリア型であるようだ

	ルール	ハイ	イイエ
[2.7]	RULE148	(.19)	
[2.8]	RULE500	(.51)	
[2.9]	RULE501	(.95)	
[3.0]	RULE502	(.97)	
[3.1]	RULE526	(.98)	
[3.2]	RULE504	(.97)	
[3.3]	RULE524	(.98)	

このようにして、この感染のタイプがバクテリア型であるということが決定された

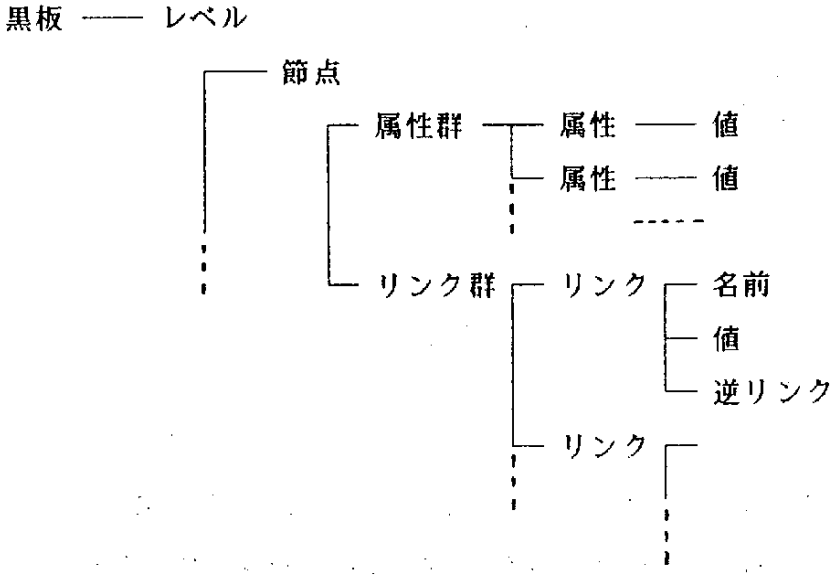
このように結論をどのようにして導いたかということを、HOW（イカニシテ）と質問することができ、その結論へ至った推論の道筋を利用されたルールの系列をさかのぼって提示するものである。

[４６]　WHAT－IF

結論を導くための仮説を表示するものである。

[４７]　BB（Black Board）

複雑な対象を取り扱う場合には、複数の知識源により推論を行い、そして各知識源はルールやプログラム実行に必要なデータを読み、評価結果を記録しておく必要がある。そのためには、各知識源が読み書きする独立した作業領域を必要とする。しかし、それらの作業領域が完全に独立していたのでは、各知識源がどのような結果を得ているか、またどのような仮説を処理しているかを、他の知識源が知ることはできない。したがって、各作業領域間でそこに書かれたデータにリンクを用いて関連をもたせ、それを利用して他領域のデータにアクセクできなければならない。このような要求を満たすものとして、BB（黒板）が考えられた。黒板には、あらかじめ解くべき問題を表現するために必要となる部分構造を定義しておき、推論プログラムがこれらを補強するために新しい仮説や結論を付加したり、誤った仮説を消去しながら推論を行うのが普通である。黒板の一般形式を示すと下図のようになる。

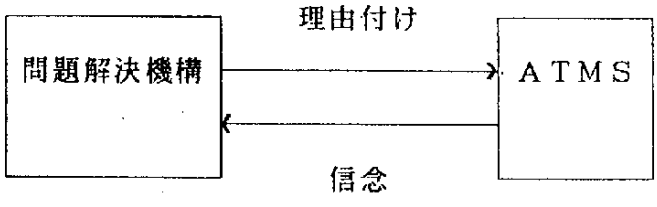


上述したように、各分割および分割内に書かれたデータ間にリンクを用いて適当な関連を与えることができるため黒板にフレームと同じ構造を構築することもできる。黒板の各レベルをそれぞれある階層関係の1つのクラスに対応させ、1つのレベル内で特別なリンク

を用いて関連付けられるものをそのインスタンスとして、クラスやインスタンスが持つ特徴を、属性－属性値対で表現すればよいわけである。フレーム・システムに特徴的な手続き付加は、例えばクラスの属性－属性値対を利用して表現することができる。黒板に新しいレベルや節点、属性を加えたり、逆に古いものを消去したりすることも自由にできるので、黒板は様々な構造を表現しうる一般的枠組みを提供するものと考えられる。しかし、黒板が十分に自由で柔軟な知識表現の道具であるためには、黒板を参照する推論機構も同様に自由で柔軟でいなければならない。

［４８］ ＡＴＭＳ（Assumption-based Truth Maintenance System）

日常生活において情報のほとんどは、あい昧なものであるにもかかわらず、人間はそのような環境下でも正しく問題解決を行っている。知識のあい昧さには、不確実な知識の基づく問題解決をＡＩで行うためには、曖昧な知識を表現し、利用するための枠組みが必要であり、その一つとして仮説推論がわかった場合、仮説を棄却するとともに仮説から導き出された信念も同時に取り除くことが必要である。このように、特定の状況（ATMS でコンテキストと呼ばれる）においてどの知識が成り立つかを決めること、知識ベース全体の一貫性、無矛盾性を保つことが ATMS の基本的な機能となる。



このような真理性維持を行う機構を備えた推論システムは上図に示すように問題解決機構と ATMS からなる。問題解決機構は、推論の実行ごとに推論の結果を理由付けとして、ATMS に与える。ATMS は与えられた理由付けを記録するとともに、これらの理由付けから信念の状態を決定し、問題解決機構に返す。ATMS では、複数のコンテキストを取り扱うため、データが信じられているかどうかではなく、データがどのコンテキストで信じられているかがラベルとして計算される。

［４９］　ガーベジコレクション

LISP に代表されるリスト処理では、2つのデータを格納するリスト・セル（あるいは単にセル）が単位になっており、これらをさまざまにつなぎ合わせることによって処理が行われている。リストというのはただ、その要素を並べただけのものであり、例えば、3つの要素を持ったリスト（a. b. c）を考えてみる。もし第一の要素を積分記号、第二の要素を積分される式、第三の要素を積分変数とすれば、このリストによって不定積分を表すことができる。このようにリスト処理の計算は、リストを次々と変換しながら進んでいく。この長所の一つは古いリストを保存したまま、新しいリストを効率よく作れる点にある。新しいセルをいくつか用意して、それらを使って古いリストの要素をつなぎ直せば古いリストはそのままで、要素の順序が入れ替わったり要素の一部が置き変わった新しいリストが構成できる。しかし個々のセルは実際には計算機の記憶装置に割り当てられているので個数が有限である。一方、リスト処理では二度と使わない使用済みのセル（これを「ゴミ」と呼ぶ）もどんどん出てくることから、こうしたセルを回収して再利用することが必要になる。この作業をガーベジコレクションと呼ぶ。

［５０］　照合のループ制御

複数個のルールが適用可能なとき、どのルールから実行すべきかを決定することをいう。その方法としては次のようなものが考えられる。

(1)　ルールを番号順に調べて、適用可能と最初にわかったルールを使う。

(2)　ルールの適用条件部が最も長いルールを使う。

(3)　(2)とは逆に、ルールの適用条件が最も短いルールを使う。

(4)　高い優先順位を持つルールを用いる。

(5)　新しく作業領域に加えられた事柄に関与したルールを用いる。

(5)'　(5)の変形としてルールの第一条件が、新しく加えられた事柄とマッチするルールを用いる。

(6)　まだ使われていないルールを使う。

(7)　すべての適用可能なルールを適用する。

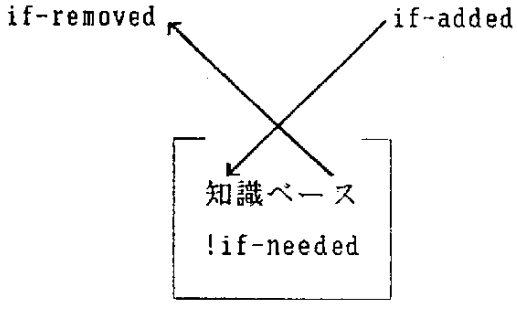
(8)　メタルールを満足するルールを適用する。

[５１] if-needed(get-slot),
if-added(put-slot),
if-removed

if-needed　知識ベースの情報を参照された場合に機能する。

if-added　知識ベースに情報が付加された場合に機能する。

if-removed　知識ベースから情報が削除された場合に機能する。



参考文献

[１] 小林重信：知識工学，昭晃堂（１９８６）．

[２] 上野晴樹：知識工学入門，オーム社（１９８５）．

[３] 科学技術庁編：知識ベース・システム，大蔵省印刷局（１９８５）．

[４] 白井良明・辻井潤一：人工知能，岩波書店（１９８２）．

[５] 淵一博監修 古川康一・溝口文雄共編：自然言語の基礎理論，共立出版（１９８６）

[６] Goldberg, A.:Smalltalk-80,The interactive programming environment,Xerox Palo Alto Research Center(1984).［相磯秀夫監訳：ＳＭＡＬＬＴＡＬＫ－８０－対話形プログラミング環境－，オーム社（１９８６）］．

[７] Goldberg, A. and Robinson, D.:Smalltalk-80:The Language,Xerox Palo Alto Research Center(1987).［相磯秀夫監訳：ＳＭＡＬＬＴＡＬＫ－８０　－言語詳解－，オーム社（１９８７）］．

[８] Cohen, P.R. and Feigenbaum, E.A. (Eds.): The Handbook of Artificial Intelligence, Vol.3, William Kaufmann(1982).［田中幸吉，淵一博監訳：人工知能ハンドブックⅢ，共立出版（１９８４）］．

[９] 湯浅太一：汎用コンピューターでの実時間"ごみ集め"，サイエンス日本版，Vol.18, No.9, pp.56-71.

平成元年３月発行

発行所　　財団法人　　日本情報処理開発協会

東京都港区芝公園３丁目５番８号

機械振興会館内

電話（０３）４３２－９３９０

６３－Ａ００２