内部知識 S-NET とそれを用いた問題解決システム

辻井 潤一
（京大・工学部）

1．はじめに

自然言語理解等の人工知能研究において、対象世界に関する知識をどのように表現しておき、推論・演繹あるいは問題解決の過程でどのようにそれを利用するかは、そのシステムの性質を決める重要な問題である。この問題に対しては、Minsky の Frame 理論、Frame理論を具体的にプログラム言語として定式化した Goldstein の FRL，T. Winograd 等の KRL等の提案が行なわれており、人工知能研究における一つの重要なテーマになっている。日本においても、電総研推論機構研究室の SRL（KRL の変形）や Production System，あるいは阪大田中研究室の Micro-Actor 等が実際に計算機プログラムとして作られ、自然言語理解システムや定理証明プログラムの作成等の種々の実験が行なわれている。

本報告では、我々が自然言語による質問応答システムを作成してゆく過程で、内部知識の表現用言語として開発しているセマンティック・ネットワーク S-Net の持つ意味とその論理的な性質を述べ、これを用いた問題解決システムの構成について報告する。

2．セマンティック・ネットワークの役割

人間の知識の形態には、様々なものが考えられる。'block world' やゲームのように比較的狭い分野を対象とする場合には、その分野の問題解決を行なうのに必要となる知識の量も限定されていると同時に、その形態もある程度均質である。しかしながら、例えば化学の分野といったようなもう少し広い対象分野における問題解決システムを考えると、そこで必要となる知識は単に量的に多くなるだけでなく、質的（形態的）にも多様なものになってくる。ここでは、論理的な推論に関係した知識だけでなく、数式で表現できる知識、プログラムで表現するのが適切であるような知識，データ・ベースアクセスを伴うような量的には多量であるが一様な構造の知識、あるいは化学式を操作するといった分野固有の手法等が必要になってくる。このような人間の持つ様々な多様な知識を1つのシステムとして組みあげ、問題解決システムを構成することは，人工知能研究の応用として医療診断システム・化学分析システムのような Expert System を構成しようとする知識工学（Knowledge Engineering）にとって重要な課題になっている。

この種のシステムを構成する上で特に留意しなければならないのは次の2点である。

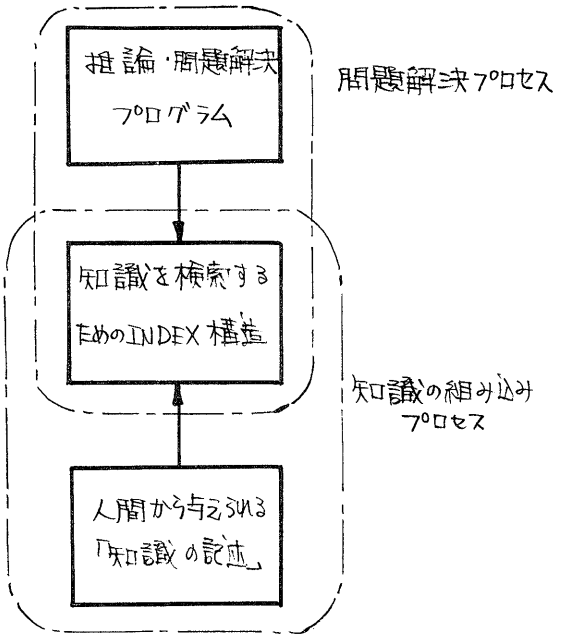
1．多量で多様な形態を持つ知識を系統的に相互の関連まで考慮して記述することは、人間には不可能である。したがって、知識の入力に際しては、直観的に理解し易い形式で入力でき、まとまりを持った知識の単位を入力す

るときには、他の知識との相互関連を考慮することなしにそれ単独で記述することができること。

2. 問題解決システムは、与えられた知識の相互関連を有機的に記憶・管理しておき、問題解決過程で必要に応じて適切な個所で適切な知識を呼び出して起動する能力を持っていること。

1のためには、知識を記述するための枠組が適切であること、2の実現のためには、与えられた知識に対してシステムが適切なindex構造をつけ加えて、問題解決過程においては、不必要な知識を参照することなく、出来る限り容易に関連する知識をとり出す能力を持っていることが必要である。

知識表現のためのシステムの一般的な構成を簡略化して示すと図1のようになる。



この図からわかるように、この種のシステムは2つのプロセスからできている。すなわち

1. 知識の組み込みのプロセス
2. 問題解決のプロセス

そして、この2つのプロセスを結びつけるのが、図1中の「知識を検索するためのINDEX構造」である。上記のシステムに対する要求を満足できるかどうかは、このINDEX構造がいかにうまくこの2つのプロセスに課せられた条件を満たすことができるかどうかにかかっている。

極端な場合として、通常のプログラミング言語（FORTRANやLISP）で知識をcodingしてゆく場合を考える。この場合、システム（プログラム言語）が提供するINDEX構造は、各プログラムに付けられた名前である。したがって、プログラマは新しい知識をcodingする時には、常にそれまでに作られているプログラムの名前と内容とを記憶しておき、必要な時点で適切なプログラムをCALLするように配慮していなければならない。したがって、このようなシステムにおいては、1の条件を全く満たしていないことになる。2の条件については、もしプログラマの配慮が完全で、作られたプログラムが完備なものになっていれば、適切な個所で適切なプログラム（知識）が唯一つ呼び出されることになり、もっとも効率の良い問題解決を行なうことになる（ただし、人工知能分野における問題解決過程は試行錯誤のプロセスを含むはずで、通常のFORTRANやLISPでこれを行うことになると複雑で難解なプログラムになってしまう。またこの場合には、必ずしもプログラムとして固定された呼び出しの順序が、必ずしも最適のものとは限らないので、2の条件も満足されないことになる）。

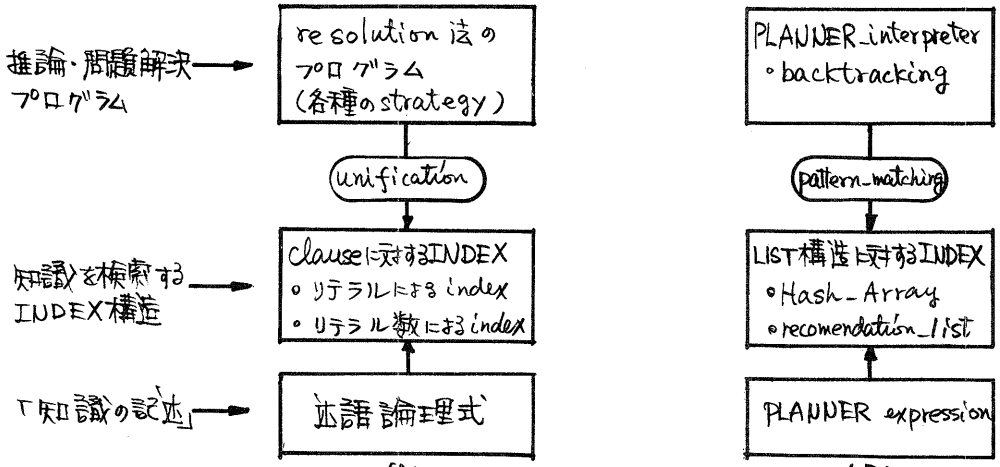


図2. 導出法とPLANNER

図2(A)は、演繹機構として導出法(resolution 法)を使い、知識記述の枠組として一階述語論理式を使うシステムの概念図である。このシステムでは、人間は述語論理式で知識を表現するが、与えられた論理式はclauseにまで分解されて、clauseの単位でINDEXされる。問題解決プログラムは、このINDEX構造を参照しながら、どのclauseとどのclauseとから次に導出を行なうかを決定してゆく。

図2(B)は、人工知能用言語PLANNERを用いる場合のシステムの概念図である。この図でpattern_matchingの機能をunificationの操作に置換えるとPLANNERによる関連知識へのアクセスと導出法によるアクセスとは本質的に同じ操作であることがわかる。PLANNERが、ある場合には導出法に比べて効率的であると考えられる理由は、知識を与える際に、その知識が引き出されるための条件をinvoke patternの形式でプログラマが明示できることによる。例えば、「シカゴ生まれの弁護士は優秀である」(T. Winogradの与えた例)は、次のPLANNER表現で記述することができる。

```
(CONSEQUENT  (CLEVER  ??X)
             (AND  (THGOAL  (CHICAGAON  !X))
                   (THGOAL  (LAWYER  !X)))
```

この表現には、次の2つの部分がある。

1. 知識の呼び出しのためのinvoke_patternを指定する部分(CLEVER ??X)。
2. 知識の本体(（AND … 以下のプログラム部分)。

すなわち、このPLANNER表現でINDEX構造に組み込まれるのは、人間によって明示的にinvoke_patternとして指定された(CLEVER ??X)の部分だけであり、(CHICAGAON !X)，(LAWYER !X)の部分に関しては全くINDEX構造には組み込まれない。このことは、同じ事実を述語論理で表現すると、

$$(\mathrm{CLEVER}\ X) \vee \sim(\mathrm{CHICAGAON}\ X) \vee \sim(\mathrm{LAWYER}\ X)$$

となり、このclauseが要素である3つのリテラルから対等にINDEXされるのと対照的である。したがって、PLANNERでは上記の知識は「ある人間が優秀である」ことを証明する場合に限って呼び出されるのに対して、導出法によるシステムでは、（たとえ原始的な導出法を改良した種々のstrategyを使ったとしても）このような知識の使用法に対する制限を、知識入力時に明示的に指示することはできない。このために、導出法を基礎にした問題解決システムにおいては関連知識の探索においてPLANNERほど有効に探索範囲を狭めることができないことになる。

しかしながら、この利点の代償としてPLANNERにおいては、例えば上記の知識と「Johnはシカゴ生まれで、しかも優秀でない」とから、「Johnは弁護士ではない」を演繹することができない，というように（論理的に完全な）導出法に比べて演繹の能力を制限することになる。このことは、知識入力時に、その知識が問題解決過程でどのように使われるかを常に意識しながら入力しなければならないことを意味しており、'block world'のように比較的使われ方の明確な知識から構成できるシステムの場合にはそれほど大きな支障とはならないが、すこし範囲の広い対象世界を扱う問題解決システムを考える場合には、ほとんど不可能なことになる。また、すこし広い対象世界を考えると、PLANNER的なpattern-matchingの機能のみに依存したindex構造が禁止的な記憶量と処理時間を必要とするようになることも指摘されている（S. Fahlman）。

導出法のように同じ知識をいろいろな目的に多重に利用することと、PLANNERのように関連する知識の検索を効率よく実行することとは、一見相互に矛盾しているように見える。この矛盾は、通常の文献検索システムにおける「呼び出し率」と「正答率」との矛盾とのアナロジーで考えることができる。すなわち、検索システムにおいて、注意深く選ばれたkey wordの集合から、各文献ごとに人間がkey wordを選定した場合には、正答率は高くなるが、初めのkey wordの付け方によって、関連があるにもかかわらず呼び出すことのできない文献が生じて、呼び出し率は低下する。PLANNERにおいては、各知識のinvoke_patternを人間が注意深くくしたがって他の知識との関連を考えながら）選ぶことによって、検索される知識の正答率を高くしていると考えられる。一方、導出原理は、ちょうど文献中に出現する単語すべてについてのinvert fileを作る文献検索システムと同様に、論理式中にあらわれるリテラルすべてにinvert fileを作るために、検索の正答率は低下するが、呼び出し率においてすぐれていることになる。文献検索システムのこの種の矛盾をよく整備されたシソーラスを用いることによって解消することができるのと同様に、問題解決システムにおいてもうまく構造化されたINDEX構造を用いることによって、この矛盾が解消できることが考えられる。この役割を果すものとして、我々はsemantic network S-Netを考えている。

導出法，PLANNERともに、人間から与えられた知識表現を直接内部の一様な構造を持ったINDEX構造（例えばhash array）に登録することを基本としている。このために、内部のINDEX構造は人間からの記述のsyntacticな構造のみに依存して作られることになる。我々は、与えられた記述から直接に内部のINDEX構造を作るのではなく、その中間に論理的な性質の明確な中間構造を設定し、一様なINDEX構造ではなく、INDEX構造自体に意味のある階層性を持たせる必要があると考える。しかも、Expert Systemのように多様な形態の知識をうま

く統合するためには、その階層性のあるINDEX構造の中に、知識の形態に応じた種類の異なった様々なINDEX構造がうまく分散して存在している必要があると考える。例えば、化学の分野を対象としたシステムを考えると、化学物質の分子量についてのデータのように、大量ではあるが一様なデータが多く存在する。このようなデータを、一般の知識を検索するためのhash arrayのようなINDEX構造に登録しておくことは、いたずらにそのINDEX構造を大きくし、検索効率を低下する原因となる。このようなデータは、一般のデータ・ベース・システムのような比較的単純ではあるが、大量のデータを効率よく蓄積・検索できるシステムにその管理をまかせてしまう方がはるかに現実的である。しかも、このような外部データ・ベースが、問題解決過程の中で適切に参照できるためには、階層的に構造化されたINDEX構造(semantic network)の中に、外部データ・ベースへの参照を指示する記述が自然な形で埋め込まれていなければならないことになる。このことは、外部データ・ベースへのアクセスだけでなく、算術アルゴリズムや、対象分野固有の手続のように種類の異なる知識単位へのアクセス全般についていえることである。

このように、適切に構造化されたINDEX構造として、我々のsemantic network S_Netは次のような性質を持っている。

(1) 関連知識の検索のためのINDEX構造自体が、論理的に意味のある構造を持っており、関連知識の検索のためにS_Net中を探索する過程自体が、論理的な操作を実行することに対応している。

(2) PLANNERや真出法のように、globalなINDEX構造が唯一つ存在するのではなく、INDEX構造自体が分散されており、推論や問題解決過程の各段階でも、activeなINDEX構造はS_Net中に分散されたINDEX構造の一部分に局限されているために、関連知識の検索が効率化される。

(3) 算術アルゴリズム、外部データベースへのアクセス、プログラム等の様々な種類の異なった知識単位がS_Net中に混在することが可能であり、問題解決過程で必要に応じてこれらを起動することが可能である。

(4) S_Net中の各リンクは、概念(述語)間や変数の値の間の依存関係を表現しており、これを大雑把に探索することによって、与えられた問題を解くためのPLANを作成することができる。したがって、PLANNER等の人工知能用言語で提案された goal orientedな手法を含み、より一般的で自然な問題解決のための手法をS_Netを使って実現することが可能である。

(5) S_Netにおいても、従来のsemantic networkで採用されてきた概念の階層構造の考え方を、関連知識の組織化の手段として用いるが、新しいタイプのノードを導入することによって、従来の表現形式よりも論理的な表現力の豊かなものになっている。

3. S-Netの記述と表現

S-Netを構成する基本的なノードのタイプとその図式表現については、すでに発表しているので、ここでは以下の説明に関係する部分を簡単に示すことにする。またS-Netを記述するための言語の記述とそれに対応するS-Net表現とを示すことにする。

3.1 基本的なノード・タイプ

S-Netでは、network表現を論理式と対応させて定式化するので、論理式表現で使われる記号のタイプに対応して、次の5種類のノード・タイプを設定する。

(1) 述語ノード　(2) 関数ノード　(3) 変数ノード　(4) 定数ノード　(5) 論理結合子ノード

これらの基本的なノードのタイプは、問題解決過程における役割に応じてさらに細分化される。これらのノードの図式表現を図3に示す（論理結合子に対応するノードについては、階層構造の項で示す。図中のリンク名の意味は自明であろう。

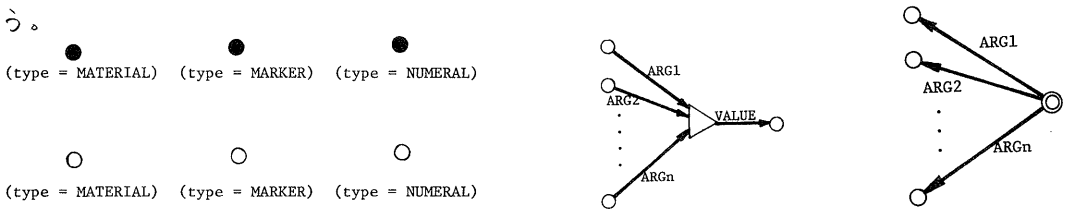


図3　S-Net中のノード

3.2 述語ノードの細分化

述語は、論理式表現の最も重要な概念である。例えば、LOVE(x, y), HUMAN(x), ON(x, y)のように、性質の異った様々な関係が述語を使って表現される。導出法のような一様で機械的な定理証明手法では、このような性質の異なった述語をすべて同じ表記法で表記し（したがって、同じ記憶構造で管理し）、証明過程でも全く同様に扱われる。PLANNERにおけるINDEX構造においても、これらは区別されず、同じように扱われる。

S-Netでは、述語を次のようなタイプに細分化し、それぞれに対して異なったタイプのノードを用意する。

(1) 物-述語（物ノード）：記号論理における定義域(Domain)を細分化するための述語。「化合物」,「電導体」,「ハロゲン化合物」等は、化学の分野における物-述語の代表的なものである。また、「pH 3 以下の液体」のように、単一の「物」-述語で表現できないものも、S-Net中では物ノードとして表現することができる。概念の階層構造は、この物ノードを中心に組み立てられる。

(2) 算術-述語（算術ノード）：引数に入るノードのタイプが数である述語。引

数間に計算可能な関係があることを示す述語。この種の述語を用いた論理表現は、その真理値が意味を持つのではなく、この述語の引数の間に存在する計算可能な関係を利用して、既知の引数の値から未知の引数の値を求めるのに使われる。

(3)関係－述語（関係ノード）：算術ノードは、引数間に計算可能な関係のある場合の述語に対応したノードであるが、関係ノードは引数間の関係がそれ以外の一般的な関係である場合の述語である。一般に外部データベースへのアクセスを含む。図4に「化合物の分子量」，「化学式」，「色」の関係を2つの関係ノードと、外部データ・ベースとによって示した例を挙げる。

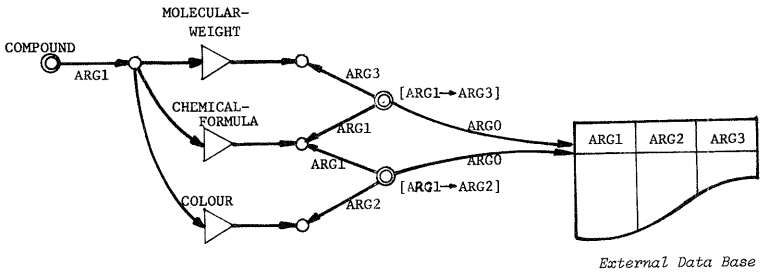


図4．関係ノードの例

(4)論理－述語（論理ノード）：算術ノード・関係ノードは、引数間にある種の依存関係が存在して、これ利用して、既知の引数の値から未知の引数の値を求めるときに使われる。これに対して、論理ノードは主としてすべての引数の値が既知になった時に、その論理ノードに対応する述語が真となるか偽となるかが評価される。GREATER，LESS，EQUAL等はその代表的なものである。

3.3 概念の階層構造，SELFノード，DISJOINノード

semantic networkを基礎とする問題解決システムでよく使われる手法に概念の階層構造を基本とする関連知識のINDEX付けがある。これは集合間の包含関係を特別のリンクとして導入し、これを探索することによって適用可能な知識の範囲を限定してゆこうとするものである。S-Netにおいては、この種のリンクをSORT，SUBSORTリンクと呼ぶ。

図5において、変数xと同定可能な物（すなわち、述語Bを満足する物）は、xを支配する「物」ノードBがSORT－リンクで「物」ノードAと結ばれているので、このAに支配されている変数ノードxと自動的に同定できる。あるノードと同定できると、そのノードについた知識が適用可能になる。すなわち、図中yと同定できた物に対しては、yについた知識だけでなく、xについた知識も適用可能となる。一般にSUBSORTリンクは無条件にはたどることができない。これに対して、SORTリンクは無条件に何段でもたどることができ、自分よりも上位のノードについた知識はすべて適用可能になる。

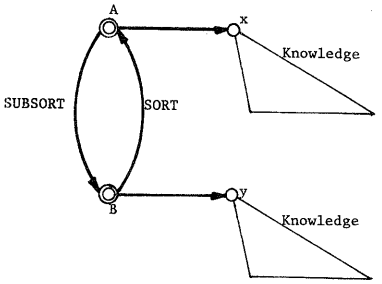


図5．概念の階層構造

通常のsemantic networkにおける概念の階層構造は、上記のような比較的単

純な推論規則を、関連知識検索のためのINDEX構造に反映させたものであった。

しかしながら、集合を規定する条件は必ずしも1つの「物」述語で表現されるとは限らない。例えば、「温度が沸点と凝固点の間の物質」、「pHが3以下の酸」のように、複合的な条件で集合を規定する場合も多い。このような集合を、概念の階層構造の中に組み込むためには、「どのような条件が満足された時に、その集合の元であると見なすことができるか」までもこの概念の階層構造の中に埋め込んで表現しておく必要がある。この条件は、S-Netではsubsor上のSELFノードを使って表現される（図6参照。図中◇はSELFノードを示す）。

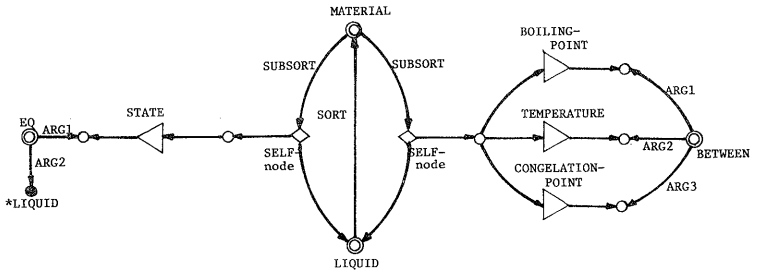


図6. SELFノードの例

（「温度が沸点と凝固点との間にある物質は液体である」，「状態が*LIQUIDである物質は液体である」を示す）。

この図で、「液体」のための2つの同定条件は、どちらか一方が満足されれば、もう一方は自動的に、適用可能な知識とみなされる。

また、S-Netの概念階層構造に豊富な表現力を与えるもう一つのタイプのノードにDISJOINTノードがある。これは、「銅を溶かす酸は、塩酸か硝酸か硫酸である」といったように、上位の集合を互いに排反な集合に分割するためのノードである。図7にその例を示す。

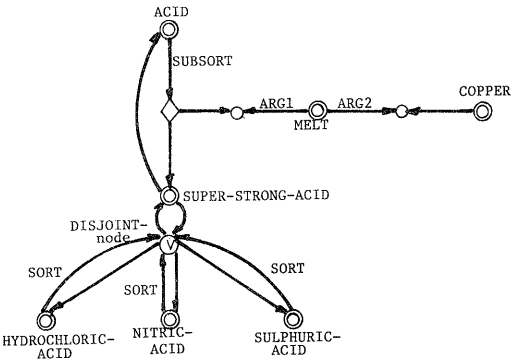


図7 DISJOINTノードの使用例

このタイプのノード（SELFノードとDISJOINTノード）は、階層構造に論理的に豊富な表現力を与えるだけでなく、問題解決過程においても、無駄な探索を禁止する重要な役割を果すことになる。

4. S-Net記述用言語

2節でも述べたように、化学のExpert Systemのように多量で多様な知識を必要とする問題解決システムにおいては、持っている知識を人間が直観的に理解が容易な表現で入力すれば、それまでの内部知識と自動的に統合されて組み込まれてゆくことが必要である。これまでに述べたS-Netの構成は、統合された結果のS-Netがどのような構造を持っているかについて述べたものである。この節では、人間がどのような形式で知識を記述することになるかを、いくつかの例を示すことによって説明する。

＜物ノードの記述＞ 人間はある概念を説明するのに、説明しようとする概念以外の概念との対比を行なって説明することが多い。このような説明の仕方を、

β-Net記述用言語では次のように行なうことができる。

<例1>

```
(CONCEPT        NAME= LIQUID
    SORT=  [(CONCEPT  NAME= MATERIAL)
                (BETWEEN    (!  BOILING-POINT)
                            (!  TEMPERATURE)
                            (!  CONGELATION-POINT)]   )
```

この記述に対応するβ-Net上での図式表現は、図6に示した。

例1の記述では、上位の概念との相異点をBETWEENという論理述語で表現しているが、次の例はこの条件の位置に再び「物」ノードを使った場合の例である。この表現は、「水溶液とは、溶媒が水であるような溶液である」という知識を示している。

<例2>

```
(CONCEPT        NAME= AQUEOUS-SOLUTION
    SORT=  [(CONCEPT  NAME= SOLUTION)
                ( (CONCEPT  NAME= WATER) (!  SOLVENT))]   )
```

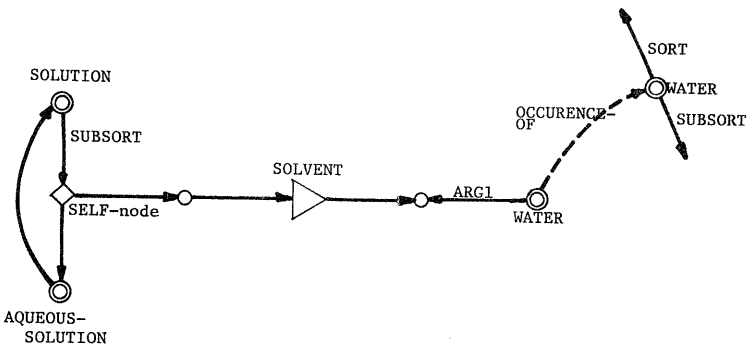


図8. 例2の図式表現

この例2を図式表現したものを図8に示す。図中、'WATER'に対応するノードが2つ存在するが、1つはWATERという概念をβ-Net中で定義しているノード、もう一つは、このWATERという物一ノードのinstanceとしてのノードである。

この2つのノードを分離して表現しておくことによって、「水溶液の溶媒」に関して宣言される知識と「水」一般に関して宣言される知識とを分離して貯えておくことができる。

この他、「物」ノードの定義の中に、DISJOINTな集合を指定すること、また物質がその「物」ノードの定義を満足しているかどうかを直接プログラムでチェックできる場合には、そのプログラムを指定することもできる。

<算術ノードの使用> 上記のようにしてβ-Net中に定義された「物」ノードに関して、様々な知識を宣言しておくことができる。次の例は「算術」ノードを使った場合の知識の宣言の例である（例3）。

<例3>

```
(CONCEPT      NAME= SOLUTION
    COMPUTATION ;
        (! MASS)=(! SOLUTE MASS) + (! SOLVENT MASS)
        (! DENSITY)=(! SOLUTE MASS) / (! MASS)
```

<例4>

```
(CONCEPT     NAME= COMPOUND
    RELATION ;
        (! MOLECULAR-WEIGHT)= [TABLE= TAB]
                              (/    (!  CHEMICAL-NAME)   *)
```

<関係ノードの使用> 関係ノードを使用した知識の記述も算術ノードの場合とほぼ同じであるが、外部データ・ベース名の指定と、引数間の依存関係を明示的に指定しなければならない。例4に関係ノードを使用した場合の例を示す。

＜論理ノードの使用＞　論理ノードは、物ノードの定義の際にSELFノードにつけられて、ある物が特定の物ノードと同定可能である条件を示すのに使われる。この使い方以外に、次のような知識の宣言にも使われる。

＜例5＞

```
(CONCEPT      NAME= ALCOHOL
    DECLARATION ;
        (LESS  (! BOILING-POINT) ((CONCEPT NAME= WATER)
                                  BOILING-POINT))  )
```

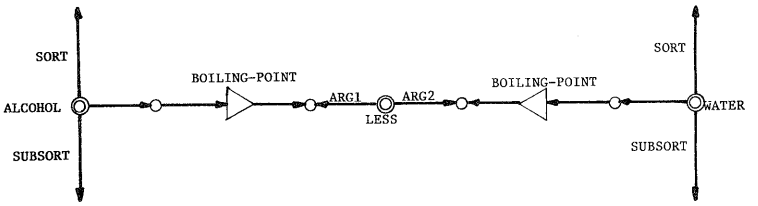


図9　例5の図式表現

以上に示したように、S-Net記述用言語を使うことによって、人間は比較的容易に自分が対象世界に関して持っている知識を直接的に表現することができる。しかしながら、default valueの指定、when-filled, to-fill等を使ってプログラムを付加する手法等を、このS-Net記述用言語の中にどのように組み込むか、またそれをS-Net上でどう表現しておけば良いか等が、今後の問題点となって残されている。

5. S-Netを用いた問題解決プログラム

単純なdepth-first探索を基本とした、問題置換型の問題解決プログラムが、現在このS-Net上に実現されている。関係ノードによってアクセスされる外部のデータ・ベースとしては、一応relational typeのデータ・モデルによって実現されていると考え、relational algebraの基本的な各演算が実行できるようになっている。ただし、これはすべてLISP言語の中で実行しているため、記憶効率、検索効率とも著しく悪い。より大規模なシステムを構成する場合には、この部分は当然他の処理系によって実現すべきものである。また、問題解決プログラムの本体は、LISPで約1000行足らず（S-Netへの組み込みルーティン、S-Netのpretty printer部分は除く）で、京大大型計算機センターM190のLISP処理系を使っている。これまでの実験の結果、S-Netの改良すべき点として、

1. 完全なdepth-first探索を行うため、探索範囲が拡がりすぎる。S-Netの特徴をはかしたPLANNING等の手法を開発する必要がある。
2. S-Netはlocalな論理的関係を表現するのには適しているが、globalな環境を考慮したstrategy的なものを表現できない。
3. 述語ノードを、物ノード・関係ノードetcに細分化したが、実際にはどのcategoryにも属さない述語が多く存在する（あるいは、2つ以上のcategoryに属するものも多い）。より一般的な取扱いをする必要があるが、これにはActorやSMALLTALK等のformalismが役立つように思われる。

＜参考文献＞(1) 長尾，辻井，寺田「セマンティックネットワークの表現力とそれを用いた推論過程」，AL77-43，1977，(2) S Fahlman，MIT Lab Working Paper 57，1974