文書処理とヒューマンインタフェース 15－2
（１９８７．１１．５）

発想の計算機支援

西尾信彦　山田尚勇
東京大学理学部情報科学科

人間が何か新しいことを考えるということを、既存のアイディアの断片を新たな構造に埋め込むことによって行なっていると定義し、その過程を計算機によって支援することを考える。その目的で、アイディアの断片と構造を別々に格納できて、断片上のテキストに限らず、その構造自身の編集や検索ができ、アイディアのもつ構造の解析を支援し、人間に新たな発想の生成を促すことのできるような文書処理システムを提案し、アイディア・プロセッサと名付けた。本稿ではその実現のためにホルダ・ネットワークというデータ構造を提出しそれを用いたプロトタイプ・システムについて報告する。

Computer-Assisting Environment for Idea Processing

NISHIO, Nobuhiko YAMADA Hisao
Department of Information Science, Faculty of Science, University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113 Japan

This article proposes an idea processor which assists a person in his idea processing. We assume that, during the idea processing, he constructs a new concept by forming some kinds of structural relationships among the scattered ideas. The idea processor is expected to give us pieces of advice by analyzing the structural relationships among the ideas. To implement the idea processor, we provide a holder-network, a new form of the semantic network. The prototype of this kind of idea processor is introduced.

１．発想の計算機支援

発想ということばに代表される創造的な活動をしているときには、人間の頭の中ではどのようなことが行なわれているのであろうか。またその発想を支援するシステムも古くからさまざまなものが考えられてきている。その中でも特に普及しているのは、川喜田が著したＫＪ法[KAWA67]であろう。ＫＪ法は、カード・システムを用いてボトム・アップ式に考えの整理をつけていくものである。発想時の思考過程をモデル化して、そのときに人間が適用している処理の方略を形式化できるならばその行為を計算機によって支援することができるだろう。本稿では計算機による発想支援について考察する。

川喜田によれば発想とは、「もやもやした情報群の中から、いっそう明確な概念をつかみ出してくる」ことだといっている[KAWA67]。本稿でもその「発想」という言葉の定義を与えるのだが、それもこれに近い。すなわち発想とは、「アイディアの断片の集まりに構造を与えることにより、新しい概念を構成すること」と定義する。この新しく構成された概念はまた新たにアイディアの断片として既存のアイディアに加わり、さらなる発想過程が続けられるわけである。それにのっとると発想行為において人間が行なっていることは基本的に大きく２つのことが挙げられる。それらは、

1. アイディアの断片の集まりに構造を与える。
2. 欠落しているアイディアの断片を生成する。

の２つである。次節以降では発想行為とは具体的にはこの２つにより構成されると仮定して話を進める。

本稿では、創造的な仕事（すなわち発想）の中でも特に文章作成過程に注目している。すなわち、発想行為などという幅の広い、漠然とした対象ではなくしようということである。ここでいう文章作成過程とは、ワード・プロセッサなどを用いて、単に文書を入力・清書するといった過程ではなく、「アイディアの断片を表わす文を連ねて明確な概念を表わす文章を作る」ことをいう。また、本稿で述べる文・文章・文書といった３つは下の図に表わしたとおりに区別される。すなわち先ほど述べたアイディアの断片・新しい概念を表現する一つの形態をそれぞれ文・文章と呼び、全体のフォーマット情報を加えてできあがった清書体を文書と呼んで区別する。

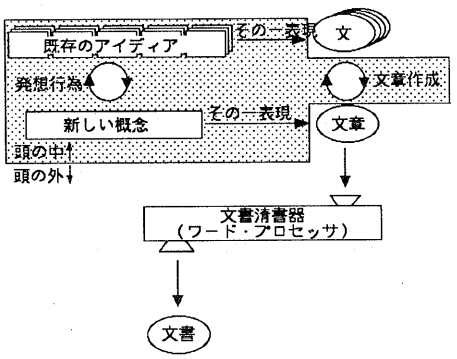

Fig. 1　　文書作成過程

本稿で議論する発想支援とは、ＫＪ法を計算機上で実現することとは異なるし、それだけでは目的は達っせられない。例えばＫＪ法を行なっているときに人間が行なっていると思われることのうちから計算機に行なわせることができ、そのうちから計算機上で実現したほうが効率がよいと思われるものをのせることである。また、いくら発想という言葉を定義したとはいえ、その行為中に人間が行なっていることがすべてわかろうはずはない。どのようなことをやっているのかを考察することと、その中から計算機にできることを探すことが主題である。

本稿で提唱するシステムは、ワード・プロセッサやアウトライン・プロセッサという文書処理システムの進化の延長としての位置づけ、アイディア・プロセッサと呼ぶことにする。

ワード・プロセッサ
　テキストの編集機能のみ。
アウトライン・プロセッサ
　テキストに階層構造という構造を与えて、
　その構造の編集機能も与えた。
アイディア・プロセッサ
　テキストをアイディアとその構造で格納し、
　その構造を解析することにより新しいアイ
　ディアを生む助けをする。

２．システムへの要請

本節では、実際にアイディア・プロセッサを構築するにあたって、そのシステムが満たすべき要件について述べる。

２.１ アイディアのこぼれない器

人間は、文章作成過程において、とにかく思いついた発想を次々と文章化していく。その思いつく順番については自分でも制御のしようがない。そこでアイディア・プロセッサは、まず最終的な文書の形態にとらわれずに思いついたものを思いついたときに受け止められるものでなくてはならない。最終的な文書の形態はアイディアの断片とその構造が絡みついているのがよくなくて、その解決のためには、思いついた発想の断片を表現した文章と、それらが埋め込まれている構造を分離して扱うことができなければならない。また、文章構造の解析を行なうわけだから、当然構造についての編集機能や検索機能が要求される。

２.２ 人間の意図の察知

計算機が発想の支援を行なうようにするために、その文章の意味を計算機がどう解釈するかという問題がある。計算機がテキストに現れる地の文章を直接に意味理解するほどには現在の自然言語理解の技術は進んでいない。本稿では、アイディアの断片そのものの意味を捉えることはなく、それが埋め込まれた構造を解析することによりアイディアの断片の意図を推測しようという立場に立っている。

２.３ 発想の過程で人間が行なうこと

人間が発想の過程において行なっていることのなかから、モデル化できてしかもそれを計算機に行なわせることができ、

そのほうが効率がいいものを探す。つぎに挙げる５つの例はそれぞれ実際の文章作成過程において人間が行なっていると思われるものである。

（１）科学論文などのように論理的な展開が必要な文章の作成については、その論理が正しく展開されていることが重要である。すなわち、ある程度論理的な文章を書こうとしたら文章の論旨の展開が明確に演えきにより納得できるようなものになっていなくてはならない。演えき的推論の適用によって、欠落したアイディアの断片を捜し出すこともできる。文章の構成や論旨の展開のさせ方を考える上で演えき法は大きな役割を果している。

（２）いくつかのアイディアがすでに出ていてそれらのつながり具合いを考えているとき、おのおののアイディアのもつ共通の概念をまとめてより抽象的・一般的なレベルに引き上げてまとめ直すことがある。また、それによってどこにそれぞれのアイディアをめ埋め込んで構造化すべきかを判定できたりする。このようなとき帰納法が適用されている。アイディアの断片の集合のいかなる部分集合をひとつの構造に埋め込めばいいのかの判定、すなわちアイディアのもつ概念の類似性を認識するのに貢献している。

（３）類推とは、以前に形成したアイディアの構造にあてはめて、いま考えているアイディアの足りない部分を導き出そうとすることで、必ずしも常に有効であると保証はできないが頻繁に行なわれる。これは、おそらく新しい考えを生むときにもっともよく使われる方略であろう。

（４）あることを考えたときには必ず対として考えておいた方がいいということがある場合がある。例えば、地震が発生したときには必ず津波の心配をするといったことである。これは、もう既に過去の経験によって獲得されたヒューリスティックであるともいえ、対となるものを考え出すのにもう論理は使わなくなっている場合が多い。これは連想行為と呼ばれるものに当てはまると思われる。さらにこれは、過去からの学習により有効なものに淘汰された特別な連想と考えられる。直観的にひらめくとはこのようなことかもしれない。

（５）ある種の条件または関係が成立している状況に特に名前が付いている場合など、以後その名前でその状況を扱ったほうが話を理解しやすくするのに向いていたり、その言い換えをしたおかげで新たな関係を見つけることができることがある。これが記号化の効用である。これは前提の条件または関係が成り立っていても気がつかなかったりして、それを見つけ出すことそのものに意味があったりする。これは要はパタン・マッチを行なっているわけで認知行為の一種ともとれる。

これらの方略をなんらかの方法で適用できるシステムの構築を目指す。

３．システム構築の方策

第２節で述べたこのシステムへの要請を満たすためにどのような方策を採用することにしたかについて述べる。

まず第一に、人間の頭にうかぶアイディアの断片をことごとく受け止めてくれる器が必要である。すなわち、それらを線形に蓄えていく従来の方法よりも、それらの発生する順序と関係なしに蓄えていくハイパーテキスト[TRIG86]を採用し、ネットワーク構造に文章を格納することにした。

第二に、アイディアの断片を文章で表現したものをハイパーテキストのノードにあて、ノード間リンクに文章間の関係の意味をもたせる。計算機にはアイディアの断片自身の意味解析よりも、その断片間に存在する関係から人間の意図をくみ取らせることとした。そのためノード間リンクには単純な矢印以上の多種多様な意味を持たせる必要がある。この方策の目的は二つの意味を持つ。その一つめは、より大きなアイディアの断片はより小さなアイディアの断片群のネットワークに還元することにより、計算機が扱うアイディア間のリンクが増えて、よりその意味についての情報が増やせるようになる点である。

二つめの意味としては、同じ情報を複数の異なった視点から見ることを可能にする点である。ここで情報が同じであるといった意味は、同じアイディアの断片を用いることである。現状では、同じ情報を単に並べ方を異ならせただけでさまざまに活用する例が多く見受けられる。われわれは、これを、同じアイディアの断片間に異なった関係をつなぐことと解釈する。その実現の支援のためには第二の方策は有効である。

しかし、さまざまな意味を持たせるためにリンクを多種類許すことにすると今度はリンク相互間の意味のとりあつかいが煩雑になってしまう。そこで基本的原始的なリンクを用いてより複雑なリンクを構成できるようにする。これにより少し異なるがほとんど同じといったリンクの量産を回避でき、計算機もリンク自身の意味を捉えやすくなる。

第三に、ハイパーテキストのもつネットワーク構造は非常に自由度の高い表現方法であるためにかえって人間にとっての処理を困難にする。そこでネットワーク構造のなかに階層性を導入できるようにして、ある程度表現の制約を与えることができるようにする。すなわち、いくつかのノードの成すサブネットワークをノードとして扱うことを可能にする。アイディアの断片は互いに異なった抽象度のレベルで発生してくる。例えば、前に述べたことを引用して注釈するようなメタな視点からのものや、さきに述べたことをよりアトミックな構造に分解して述べることがあるが、この方策により表現可能になる。いいかえると、複数のアイディアの断片をまとめていいかえる機能と、大きな断片をより小さな複数の断片に分けていいかえる機能を実現するということになる。

第四に、実際に発想支援のために計算機がアイディアの構造に適用する４つの方略をつぎに挙げる。

（１）ネットワークのバランスを見る。例えばあるノードから同じリンクが２つ以上出ていたらその先の展開に類似性が期待され、それぞれの先のネットワーク構造を比較し、互いの欠損部分の補填を示唆する。

（２）サブネットワークの文章としてのまとまり具合いをチェックする。このためには、ある程度まとまった文章の構成

のひな型を表す構造化リンクを用いてそれにどのくらい当てはまっているかにより推し量る。また、このような文章構成のひな型に照らし合わせることによって人間の演えき行為を実現しているとみることもできる。

（３）ノードの抱えるテキストの大きさの平均より大きなノードは還元化（サブネットワーク化）を勧める。またその逆に散らばったノードの内包する共通部分を探してそれらをひとつのノードにまとめることを示唆する。これは帰納法の実現に関係がある。

（４）サブネットワーク同士の比較を行なう。これは類推の一般的な適用であるといえる。この比較により

1. 似ている部分を捜し出す、
2. 似ている部分の異なった部分を捜し出す。

この中に含めていない「有益な連想」や「パタン認知」はシステムの学習機能を必要とし、ここでは将来の拡張とする。

最後に、このアイディアの断片群の成す構造の編集・検索に対して柔軟なアクセスを可能にする。そのために、ある関係を表わす（定義する）リンクの実体はひとつとする。その関係を実例化した環境の実現を、そのひとつしかないリンクのつながる先を実際のアイディアの断片とユニファイする環境で表すことにする。例えば、異なった２対のアイディアが同じ関係を有していた場合、それぞれの関係はただ一つの関係を表わすものが違った環境に出現しているように表現する。これにより同一の関係を用いた環境を数え上げることが容易になっている。

以上をまとめると次のようになる。

1.ネットワーク構造をサポートしたハイパーテキストを採用する。
2.リンクに意味をもたせ、構造化を企てる。
3.ネットワーク構造に階層性をもたせ野放図な表現を避ける。
4.構造を解析することにより発想の計算機支援を期待する。
5.扱う構造に対する柔軟なアクセスを提供する。

以上のことをふまえて、われわれは次節に述べるホルダ・ネットワークをデータ構造としたアイディア・プロセッサを設計した。

４．アイディア・プロセッサの設計

本節では第２節で挙げた要請を満たし、前節で述べた方策を実現するアイディア・プロセッサの設計のために、アイディアの断片とその構造を格納するハイパーテキストを実現するためのデータ構造であるホルダ・ネットワークについて詳述する。

４.１　ホルダ・ネットワーク

前節で示唆したような思考支援指向のハイパーテキストに向いたデータ構造を提出する。本データ構造の特徴をまとめると、

1.文章単位をノード、その間の関係をリンクとするハイパーテキストを実現する。
2.ハイパーテキストが実現しているグラフのサブグラフをノードとすることもできる。
3.複雑な関係を複数の基本的なリンクから構成できる。
4.リンク間の関係も記述可能である。

といったものが挙げられる。次節では、まずホルダ・ネットワークの構成を説明する。その後、そのホルダ・ネットワークがもつセマンティックについて述べ、続いてそのデータ構造への柔軟なアクセスを可能にするオブジェクト主導的メソッドの定義について述べる。

４.２　ホルダ・ネットワークの構成

ホルダ・ネットワークをその実現法に沿って説明する。ただしホルダ・ネットワークは今のところは、ふつうのテキスト・データにのみ対応するようなものとして考えている。ホルダ・ネットワークを構成する基本オブジェクトは以下のとおり。

1.文字
2.スロット
3.スロット及びホルダへのポインタ

文字とは日本語文字、数字、英字、特殊文字などのことをいう。

スロットは特定の特殊文字により囲まれた文字の列より表される。

文字またはスロットの有限回の並びをテキストという。

ホルダは以下のものにより構成する。

1.テキスト
2.ホルダへのポインタのテーブル×２
（それぞれをスーパー・ホルダ・テーブル、サブ・ホルダ・テーブルと呼ぶ）
3.単一化環境表

ホルダの構成の概念図を下に示した。

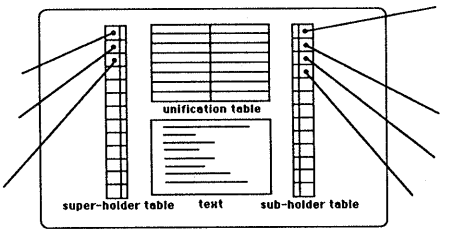

Fig. 2　ホルダの構成

ホルダの保持しているテキストをパラフレーズ・テキストと呼ぶ。これは空でもよい。

スーパー・ホルダ・テーブルやサブ・ホルダ・テーブルは同じホルダを複数回指すことができるようにする。ホルダはそのサブ・ホルダ・テーブルの指している各ホルダをホルドし

ていると言い、スーパー・ホルダ・テーブルが指している各ホルダにホルドされていると言う。

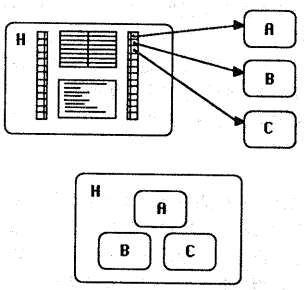

Fig. 3 ホルダ群のホルドの概念図

単一化環境表はスロットへのポインタとスロットへのポインタの対、またはスロットへのポインタとホルダへのポインタの対を連ねたテーブルである。この対として現れるスロットは、

1. そのホルダのパラフレーズ・テキスト。
2. そのホルダのホルドするホルダの各パラフレーズ・テキスト

のいづれかに埋め込まれているものでなくてはならない。またホルダは、

そのホルダがホルドするホルダ

でなくてはならない。

4.3 ホルダ・ネットワークのセマンティックス

ホルダ・ネットワークにおいては、通常のハイパーテキストがいう文章ノードをホルダとそのパラフレーズ・テキストで表現する。

ハイパーテキストにおける文章ノードを結ぶリンクは、スロットを２個埋め込んだパラフレーズ・テキストをもつホルダを用いて、そのスロットが実例化される環境で表現する。

例として「AはBの例になっている。」という記述をホルダ・ネットワークで表してみよう。（ここでは‘%’がテキストのなかでスロットのはじめとおわりを表す特殊文字となっている。）

（１）まずAとBの実体であるホルダAとホルダBと「%スロット１%は%スロット２%の例になっている。」というパラフレーズ・テキストをもつホルダRを生成する。
（２）上記３つのホルダをホルドするホルダHを生成する
（３）ホルダHの単一化環境表にホルダRの%スロット１%とホルダA、ホルダRの%スロット２%とホルダBの２つの対を記録する。

以上の手続きにより、ハイパーテキストにおける文章単位ノード間のリンク付けを実現する。その概念図を下に示した。以後ホルダ・ネットワークでリンクと呼ぶときは上のようにして実現されたものをさすこととする。

このとき単一化環境表に連ねられる各対はそれぞれの単一化の情報を保持するものと考える。ここでは同じ単一化でもスロットにホルダが単一化したときには、それを実例化と呼び区別することにする。さてホルダHとホルダRの意味をここで考えてみると、ホルダRは「クラスとしてのリンク」を、ホルダHは「インスタンスとしてのリンク」をあらわしているととることができよう。ここでいうクラス、インスタンスとはSmalltalk-80[GOLD83]でいうところの意味である。

続いてホルダ・ネットワークのサブ・ネットワークをノードとして扱うことについて説明する。先ほどの例のようにして平面的にネットワークを構成していくとき、ホルダHはその全体のネットワークの中ではサブ・ネットワークを表しており、一方ホルダHはホルダであるのでホルダAなどと同様にホルダ・ネットワークのノードとなり得ることがわかる。

ところで、このようなサブ・ネットワークをあらわすホルダのパラフレーズ・テキストにはどのような意味があるのであろうか。このようなホルダの表わすサブ・ネットワークは、その名のとおりホルダ・ネットワークであらわされているわけであるが、単にスロットをホルダで再帰的に置き換えてできあがるテキストでは自然な文章にならないことが多くなると思われる。そこでそのホルダがあらわす情報をもう一度パラフレーズしたテキストをパラフレーズ・テキストとして格納することにした。そのときそのホルダが実例化していないスロットを持つときパラフレーズ・テキストもそれに単一化したスロットを埋め込むことにする。ただしここではすべての実例化されていないスロットが埋め込まれる必要はない。それはどのようなパラフレーズがされるかによる。

さて以上述べたことから、リンクをあらわすホルダを複数個ホルドするホルダを生成して新たなリンクにパラフレーズすることが可能なことがわかる。これにより、リンクに潜在する介在者を隠ぺいすることもできるし、必要なときにはその隠ぺいされたものの他との関係も得ることができるようになった。またこのように構造化されたホルダが文章のまとまり具合いのテンプレートとしてはたらくことができるのはいうまでもないであろう。例を下に示した。

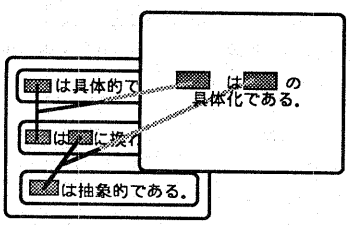

Fig. 4 リンクの構造化

この他にテキスト・データ以外のものを扱いたくなったときには、そのようなデータを文字オブジェクトを導入したレベルであたえておき、パラフレーズ・テキストとしてそれらを文字やスロットと混在できるようにしておくことになる。

4.4 ホルダを個性化するメソッド辞書

元来ホルダ・ネットワークとはデータ構造であり、それに対する手続きについてはまだ述べていない。本節はその手続きに関するものである。これはアイディア・プロセッサのユーザ・インタフェースにも関わってくる部分でありここでは原則的なことについて触れておく。

ホルダは文章ノードを表わしていたり、リンクを表わしていたり、ハイパーテキストのサブネットワークを表していたりでそれぞれに個性をもっているものである。そのためホルダに対する手続きを統一的に決めてしまっては有用なユーザ・インタフェースは供給できない。そこでオブジェクト主導的な考え方を導入してそれぞれのカテゴリに属する手続きに継承関係を与え、ホルダ各々が自分用のメソッド辞書を持つことにした。

そこに用意されるカテゴリの例を挙げると、
自分を計算機のモニタ上で視覚化する手続き、
自分をアイコン化する手続き、
自分を編集するときの手続き、
自分のスロットを単一化するときの手続き、
自分の出現する環境の情報に関する手続き、
などとユーザ・インタフェースを充実させたり、構造に解析のために必要なものが豊富に考えられる。

このようにメソッド辞書を持っておくと、例えば視覚化せよというメッセージを受け取った各ホルダは自分が知っているメソッドまたは親クラスから継承したメソッドを用いて指示に従う。これにより、リンクをあらわしているホルダは例えば矢印などとして、全体として表をあらわしているホルダは表として視覚化することが可能になる。このようなカテゴリ別メソッドの継承はSmalltalk-80の提供する単一継承に比べて柔軟な多重継承が可能である。

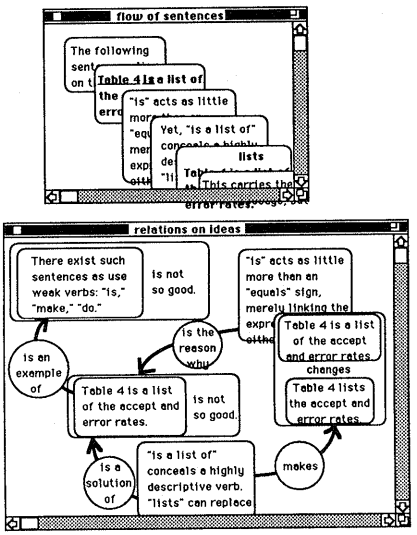

Fig. 5 多様に視覚化されたホルダ・ネットワーク

5．ホルダネットワークの考え方

発想支援をするのに生まれてきた順序に関係なくそれを格納できるハイパーテキストを採用することと、そのハイパーテキストのもつリンクに意味を持たせて、システムがテキストの意味に入ってこれるようにしたことはすでに説明した。ここでは、そのリンクの張り方さえもすべてホルダにホルドすることにより実現するといったホルダ・ネットワークの特異性について考察する。

人間の問題解決は、その問題をより細かくより単純な問題に分けてそれらを順に解決していくことにより全体の解決に向かうという方法を取る。すなわち、一つの問題を他のよく判っている事柄に帰着させることにより解決をはかっていることが多い。よく判っているとは以前にその同じような問題を解決したことを記憶しているという意味である。さてそこでいう同じとは何が同じなのであろうか。それは、二つに分けて考えられる。すなわち、扱っている事柄が同じか、その事柄同士のなす関係が同じかということである。

とりあえず、ここで大まかに問題解決において人間が行なっていることをまとめると、それらには二つの段階がある。その一つめは、一つの事柄をより細かいものに還元していくという段階、そしてその二つめはそれらの中から長期記憶に貯えられているものと同じ事柄もしくは関係を検索する段階である。これら二つの局面において、効果的なデータ構造をということでホルダネットワークは考え出された。はじめの段階においては、サブネットワークのノード化の必要性を感じさせ、リンクをすべてホルダをホルドすることにより張るという手法により実現可能になる。

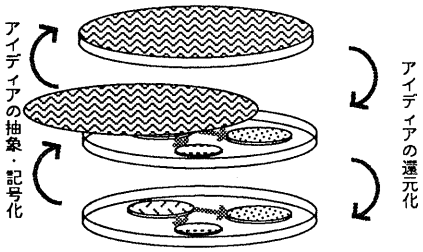

Fig. 6 パラフレーズ・テキストの発想支援効果

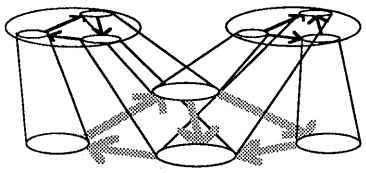

Fig. 7 同じものを違う局面から見る

またその次の段階においては、リンクを表わす概念が一つずつしか存在せず、同一のリンクを有する関係はすべて同じリンクを表わすホルダを用いて個々の環境を作るという手法で実現することにより検索が楽になっている。

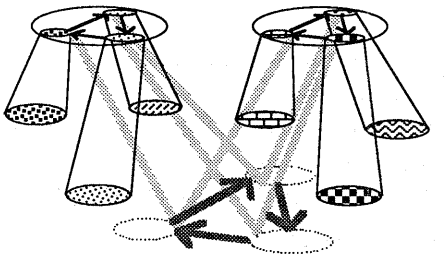

Fig. 8　同じ関係を有する異なった状況

またリンクの構造化という考えは、必ずしも同一の関係を探し出すことができない状況においても対応できるように考案されたともいえる。すなわち、より原子的なレベルでの概念の共通性がより抽象的なレベルにおいての記憶検索に、いい意味でも悪い意味でも影響できることになっている。

6．ホルダネットワークを用いた発想支援

本節では、実際にこれまでのような道具立てにしたがって発想を行なうところを追跡してみることにする。

以下にあげた例は、ホルダ・ネットワークを考案するときに参考になった論理型言語prologとの類推を行ないそこからどのようなアイディアが得られるかを追跡したものである。

まずそれぞれのキーとなるようなことばをあげる。

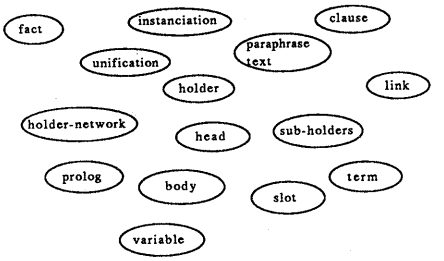

Fig. 9　Scattered ideas

それらを大まかにホルダネットワークに関するものとprologに関するものに分けて、

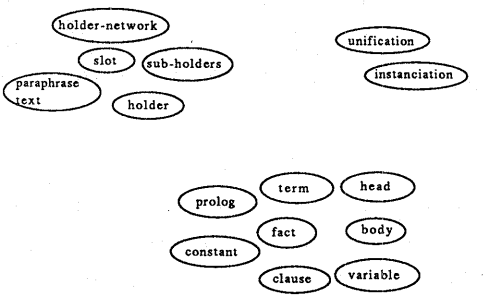

Fig. 10　Roughly gathered ideas

それぞれの存在に対応すると思われるものとのマッチングをとってみる。

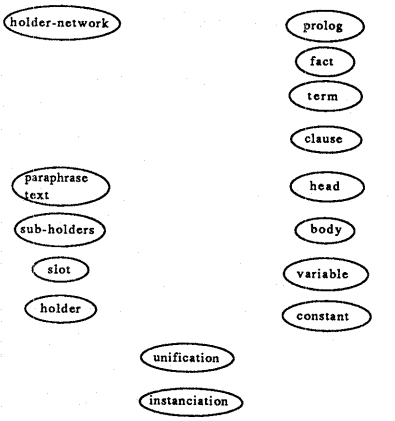

Fig. 11　Mapping the ideas

そして、それらにあわせて関係を記述し構造化をはかる。

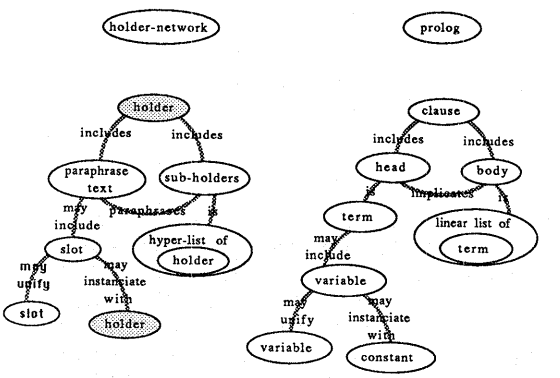

Fig. 12　Structured ideas

これによりわかることは、prologにおいてはその変数が定数としかユニファイできないが、ホルダネットワークにおいてはスロットがまたホルダとユニファイできるために、prologでは扱えない高階述語論理の表現が可能になっていることがわかる。

7. アイディア・プロセッサの実現

本稿で提案したシステムの開発は現在米アップル社のパーソナル・コンピュータMacintosh Plusの上で行なっている。本機種を選択した理由は本機種の持つ優れたtoolboxを使用してユーザ・インタフェースの部分を実現したかったからである。本機種は表示画面が狭く（512*342ピクセル）、外部記憶装置も貧弱であるという批判もある。ただしtoolboxの用意したシステム・コールは上位互換性に優れており、ここで開発されたソフトウェアはほとんど何の手も加えることなしにその上位機種であるMacintoshIIにおいて走行可能であり、もしそうなれば上記の短所も克服される。

最近発表されたソフトウェアHyperCardはMacintoshの優れたユーザ・インタフェースを手軽にユーザが盛り込むことのできるカード型ハイパーテキスト・ツールである。これにはユーザが自分用にカードをオーダ・メイドすることができるように簡易なプログラミング言語HyperTalkを備えている。これを用いた本アイディア・プロセッサのプロトタイプはほぼ完成している。しかし、HyperTalkはユーザ・インタフェース部分の非常に簡便な問い合わせ言語（ハイパーテキスト間移動記述）としては優れているものの、アプリケーション作成のためのプログラミング言語としては非常に使いにくい。そのためにプロトタイプ・システムの実行はかなり遅いものになっている。またHyperCard自身が一度に一枚のカードしか表示できないというネックをもっているためにネットワーク状態の表示には困難を有した。

プロトタイプシステムではテキスト情報のハイパーテキスト化をホルダ・ネットワークを用いて行なっている。実現された機能としては、

1. サブネットワークのホルド
2. スロットの単一化によるリンクの構造化
3. 単純なサブネットワークの比較・検索

といった単純なものばかりである。

8. むすび

本稿では、発想行為を計算機で支援するために、アイディアを表わした文章の成す構造を多種の方略によって提示しその解析を支援することを考え、それを実現するための発想支援システムであるアイディア・プロセッサを提唱した。また、アイディアの断片とその構造をハイパーテキストによって格納するためにホルダ・ネットワークというデータ構造も提唱し、それを用いたアイディア・プロセッサのプロトタイプを作成した。将来はこのプロトタイプによる実験に基づいてフル・システムのアイディア・プロセッサの設計を行ない開発し、発想過程における人間の適用する方略の自動化をはかっていくつもりである。

謝辞

本研究にあたり多くの有益な助言をしていただいた助手の小野芳彦氏、また岡留剛氏をはじめとする大学院生の諸氏に感謝いたします。

参考文献

[BUSH45] Bush, V. :As we may think. Atlantic Monthly, 176, 1 (July 1945), pp.101-108.

[GOLD83] Goldberg, A. and Robson, D. :Smalltalk-80 - The Language and its Implementation, Addison Wesley, 1983.

[KAWA67] 川喜田二郎：発想法－創造性開発のために 中公新書136, 中央公論社 1967.

[TRIG86] Trigg, R. H. and Weiser, M. :TEXTNET:A Network-Based Approach to Text Handling, ACM Transactions on Office Information Systems, Vol.4, No.1, Jan. 1986, pp.1-23.