Vol. 23 No. 3 情報処理学会論文誌 May 1982

意味ネットワークの賦活動態を制御するプログラミング言語†

嶋津好生†† 田町常夫†††

意味ネットワークの要素的な賦活動態をルーチンとして定着するために，活性化意味ネットワークを賦活制御する手続きを記述するプログラミング言語の構成が要請された．これを意味ネットワークの賦活制御言語と呼ぶ．この言語の各命令は，節点や有向弧の原子概念からなるデータ集合を SIMD 並列に制御する．意味ネットワークの再帰的拡張節点を識別しながら，柔軟なネットワーク横断を実現するために，いくつかの工夫がなされた．TRANSFER STATE 命令によって節点間の励起状態の移送を制御する．再帰的拡張節点の同定を容易にするため，節点間の励起共有を任意に設定したり解除したりできるようにする．SET SHARING-EXCITATION 命令は，照合条件を示し，それに適合する弧によって励起共有設定する．CLEAR SHARING-EXCITATION 命令はその設定を解除する．励起をもたらす，MATCH, TRANSFER STATE, SET SHARING-EXCITATION の3命令に，励起と同時にその痕跡を残す機能を添えた．この機能を利用し，プログラム中にブロックを指定してその中で指定された励起命令が個別に意味ネットワークを励起するのを集めて，その AND 条件や OR 条件で最終的な励起状態を得ることができるようにした．SET LOGICAL MODE 命令と CLEAR LOGICAL MODE 命令がそのブロックを示す．分岐命令は意味ネットワークの励起状態を分岐条件とする．

1. まえがき

Rumelhart, D. E.[1] らは，記憶の過程を考察するのに「記憶活性」(memory-activation) という有益な枠組みを与えた．我々は，概念水準の知識を意味ネットワークで表現し，「記憶活性」の枠組みを取り入れて，概念記憶の過程に関するモデルを構成した．これを活性化意味ネットワーク (Activated Semantic Network, ASN)[2] のモデルと呼んでいる．ASN の理論は，主として，意味ネットワークの静態構造論と賦活動態論とにより構成される．「活性化」(activated) という言葉は，記憶活性の形式的表現法を与えるため，意味ネットワークの表現力を増強したことを言い表わしている．意味ネットワークの静態構造に関しては文献 3) において詳しく議論している．賦活動態に関しては文献 4) において詳述し，概念記憶過程の範例を分析して要素的な過程を抽出している．これらの議論をふまえて意味ネットワーク賦活動態の形式的な表現法を与えることが本稿の主題である．意味ネットワークを賦活制御する手続きを記述するプログラミング言語を構成し，それを使って具体的に賦活制御プログラムの実例を示す．

† A Programming Language for Controlling Activation-Movement of Semantic Networks, by YOSHIO SHIMAZU (Faculty of Engineering, Kyushu-Sanyo University) and TUNEO TAMATI (Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州産業大学工学部
††† 九州大学総合理工学研究科

2. 準　備

2.1 諸定義

定義 1 節点・弧ラベル付有向グラフは3つ組 ($\mathcal{N}$, $\mathcal{A}$, $\mathcal{L}$) で定義される．

i) $\mathcal{N}$ は節点の有限集合である．それぞれの節点は互いに異なる節点ラベルを持つので，節点ラベル集合でもある．

ii) $\mathcal{L}$ は弧ラベルの有限集合である．

iii) $\mathcal{A}\subset\mathcal{N}^2\times\mathcal{L}$, すなわち, $(x, y; u)\in\mathcal{A}$. $x, y\in\mathcal{N}$, $u\in\mathcal{L}$ のとき，節点 x から節点 y へ有向弧が存在し，弧ラベル u が添付されている．

定義 2 再帰的拡張ネットワークは5つ組 ($\mathcal{N}$, $\mathcal{A}$, $\mathcal{L}$, S, $\mathcal{R}$) で定義される．

i) 再帰的拡張ネットワークは節点・弧ラベル付有向グラフ ($\mathcal{N}$, $\mathcal{A}$, $\mathcal{L}$) を基底構造とし，その上に再帰的拡張節点集合 $\mathcal{R}$ を同定できる．$\mathcal{N}$ は $\mathcal{R}$ と区別するため基底節点集合と呼ばれる．

ii) $\mathcal{N}\subset\mathcal{R}$ である．

iii) 構造認識規則 S に従って $\mathcal{R}$ の部分集合を作る．それら部分集合の集合が再帰的に $\mathcal{R}$ に包含されるとき，$\mathcal{R}$ は基底節点集合 $\mathcal{N}$ の，構造認識規則 S に関する再帰的拡張節点集合と呼ばれる．

定義 3 活性化意味ネットワークの静態構造は再帰的拡張ネットワークである．

定義 4　活性化意味ネットワークの節点や弧のラベルを原子概念と呼ぶ．また原子概念は次のような概念素項によって構成される．

活性項 AT	励起項	(＊)
	励起痕跡	(＃1, ＃2)
	指数的特定化項	PS
	符標的特定化項	SS
蓄積項 CT	活性度	AV
	あいまいさ	FZ
	調整子	MD
	記述子	DC
	同定子	ID

定義 5　原子概念が活性項に情報を保有していれば，その原子概念は「賦活されている」という．とくに励起項にそのしるし (＊) を持てば，「励起されている」という．

定義 6　活性化意味ネットワークの上で励起領域が活性項の変化を伴い，拡散，集中，転移していく過程を「賦活動態」と呼ぶ．

2.2 活性化意味ネットワークの図示法

3. において意味ネットワークを賦活制御するプログラミング言語を導入する．ここでは，制御言語の効果が意味ネットワークの上にどのように顕われるかを説明するための，意味ネットワークの図示法を決める．意味ネットワークを構成する基本単位は2つの節点とその間を結ぶ1つの有向弧である．**図1**において，AT, CT はそれぞれ活性項，蓄積項を表わす．添字 n と a とは節点と弧とを区別している．弧は方向を持つので，弧の概念素項の中に両端の節点の方向に分極して2つになるものがある．蓄積項の中では，数量的因子である活性項やあいまいさが弧の方向によって異なる値を持つ．ただし，記述子に添えた矢印が弧の方向を表わす．活性項の中では，励起項と励起痕跡とが両端に分極する．特定化項に関しては蓄積項に準じて考えればよい．

弧は弧に固有な励起共有設定というもう1つの活性項を持つ．励起共有設定は**図2**のように太い矢印で表わす．

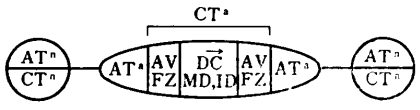

図 1　意味ネットワークの基本構成要素
Fig. 1　The elementary constituent of the semantic network.

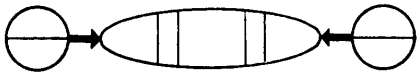

図 2　励起共有設定の表現
Fig. 2　Representation of setting of the sharing-excitation.

励起共有設定された弧によって，その両端の節点の間に励起状態を共有する関係が生じる．図2の場合は左右両方に太い矢印を持つので，左右いずれか一方の節点が励起されたら他方の節点も自動的に励起状態になる．左右のいずれか一方だけに矢印が付けられていれば，その方向にのみ励起共有である．たとえば左の矢印だけならば左の節点が励起されると右の節点も自動的に励起されるが，右の節点が励起されてもそれによって左の節点が励起されることはない．

励起項，励起痕跡，励起共有設定の3つの活性項は賦活制御言語の命令操作の中に組み込まれて，システム固有の制御を受ける．これをシステム活性項と呼ぶ．一方，2つの特定化項はユーザ活性項と呼ぶ．

3. 意味ネットワークの賦活制御言語

賦活動態の要素的類型群を賦活制御の手続きとして定着していくために，活性化意味ネットワークを賦活制御するプログラミング言語を構成する．この言語には次のような機能が要請される．

1)　意味ネットワーク上に潜在する再帰的拡張節点の同定を容易にする機能が必要である．

2)　励起動態を制御するための基本操作として，節点間で励起状態を移送する能力が必要である．

3)　原子概念に対する内容指定の接近 (content access) が可能であること．またこれを手段として，節点集合に対する並列的な照合操作 (matching) や複数の弧において並行的に励起状態を移送する能力を持たせる．

第1項の要請には励起共有設定によって応える．第3項の要請から，この命令系は SIMD (Single Instruction Stream Multiple Data Stream) 並列処理の特徴を持つ．意味ネットワークの基底構造，節点・弧ラベル付有向グラフにおいて，弧集合の要素，

$$(x, y\,;\, u)\in\mathcal{A}$$

が存在するとき，その逆の要素，

$$(y, x\,;\, -u)$$

の存在を想定し，

$$\forall x\forall y\forall u\,\{(x, y\,;\, u)\in\mathcal{A}\rightarrow\{(x, y\,;\, +u)\in\tilde{\mathcal{A}}\wedge(y, x\,;\, -u)\in\tilde{\mathcal{A}}\}\}$$

を真とする集合 $\tilde{\mathcal{A}}$ を考える．このとき合併集合，$\mathcal{N}\cup\tilde{\mathcal{A}}$ を1つのデータ集合とすれば，賦活制御言語の命令系はこのデータ集合に対して SIMD 並列処理を実行する．すなわち，弧に対してはその方向も識別して処理を行う．

命令形式を次に示す．言語記述は AN 記法による．

〈命令形式〉::=〈非選択的操作項〉
[〈選択的操作項〉][〈照合子形式〉]

〈選択的操作項〉::=[(〈励起痕跡選択〉)]
[〈論理様式選択〉,]
[〈節点・弧選択〉]
[.〈照合条件〉.]

〈照合子形式〉::=〈m〉
::=〈[#(1/2)]
[PS:〈指数的特定化項群〉]
[SS:〈符標的特定化項群〉]
[AV:〈活性度〉,]
[FZ:〈あいまいさ〉,]
[MD:〈調整子群〉]
[DC:[+|−]〈記述子〉,]
[ID:〈同定子〉]〉

ここで，任意の概念素項に対して，

〈概念素項群〉::=〈概念素項〉{;}$_{\circ\circ\circ}$

である．＋，－は弧記述子に付随している方向を識別するため，弧原子概念に対する照合子にのみ使われる．＋の場合は矢印の方向に，－の場合は矢印の逆方向に照合される．照合子形式は励起項がないことを除けば，原子概念の構成と同じである．

さらに細かく規定した命令の記法を次に示す．

〈励起痕跡選択〉::=z::=1/2

〈論理様式選択〉::=w::=AND|OR

〈節点・弧選択〉::=x::=N/A

〈照合条件〉::=y
::=EQT|NEQ|LTH|STH|LEQ|SEQ

〈非選択的操作項〉::=LINK|STORE|PUT|
ADD|READ|MATCH|TRANSFER
STATE|SET SHARING-EXCITATION|
COMPOSE|SET LOGICAL MODE|
CLEAR EXCITATION|CLEAR
SHARING-EXCITATION|
CLEAR LOGICAL MODE

非選択的操作項を見れば，この制御言語の主要な機能が窺い知れる．MATCH は照合操作を実行し，TRANSFER STATE は励起状態を移送し，SET SHARING-EXCITATION は励起共有設定を行う．

次に個々の命令の機能を説明する．各命令は意味ネットワーク（以下 SN と略記する）に変化をもたらす．

1) LINK 〈m〉

この命令の照合子は必ず記述子〈DC:(+|−)〈記述子〉〉を持つ．記述照合子が〈DC：+dc〉であれば，SN 上のすべての励起節点から**図3**(a)に示すように新しい弧が伸びる．

記述照合子が〈DC：−dc〉であれば(b)のように

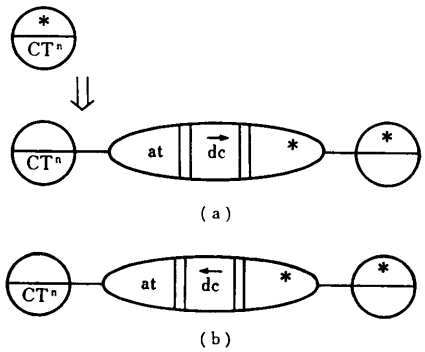

図 3 LINK 命令の機能
Fig. 3 Function of the LINK-instruction.

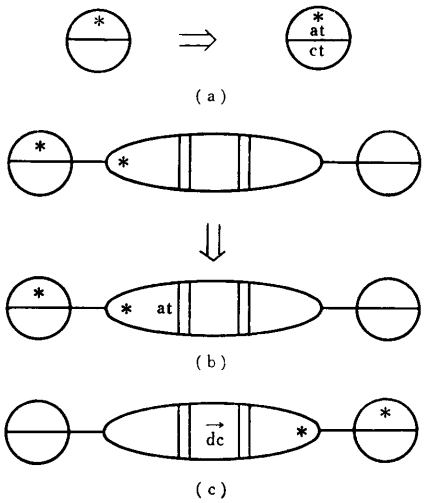

図 4　STORE 命令の機能
Fig. 4　Function of the STORE-instruction.

なる．照合子形式は活性項や，記述子以外の他の蓄積項を含むから，それらを同時に書き込むこともできる．LINK 命令は SN を増殖するのに使用される．

2)　STORE　x　〈m〉

この命令は励起されている節点や弧に照合子の内容を格納するのに使用される．節点選択 (x=N) の場合は**図 4** (a)のようになる．弧選択 (x=A) の場合は，照合子が記述子を含むか含まないかによって異なる変化が起る．照合子が記述子を含まなければ(b)のようになる．(弧の励起状態は隣接する節点の励起状態に従属する．図 4 (b)のように，弧が励起されると，その指定された側の端節点も必ず励起されることに注意すること．) 照合子が記述子を含むならば(c)のようになる．ただし，図 4 (c)は弧記述照合子が＋のときである．もしこれが－ならば図中の矢印は逆方向になる．

3)　PUT　x　〈m〉

この命令は，励起領域に限らず，SN のすべての節点あるいは弧に照合子の内容を格納する．SN 全体にわたり，原子概念の特定のセグメントを解消したいとき，この命令を使用する．たとえば，〈$\overline{\#1}$〉を照合子とする PUT 命令を使えば，#1 励起痕跡を SN 全体から除去することができる．ただし，$\overline{\#1}$ は #1 励起痕跡を解消する照合子である．

4)　ADD　x　〈m〉

この命令の照合子になれるのは，指数的特定化項 (PS)，活性度 (AV)，あいまいさ (FZ) 等である．これらはすべて数値的情報である．この命令は励起されている節点や弧の数値的概念素項に加算をほどこす．励起節点あるいは励起弧の数値に照合子の数値が加算される．(必要に応じて，加算以外の演算を導入すべきだと考えている．活性度やあいまいさ等が活用されるシステムにおいて，このことを改めて考慮する．)

5)　READ　x

この命令によって，励起節点あるいは励起弧の原子概念を ASN の外へ読み出す．読み出す場所はスタック（$）に構成されている．読み出された情報はこの場で直接，記号処理の実行に移れる．この場所は照合子の格納場所としても使われる．したがって，この READ 命令で読み出したものを直ちに照合子として使用するような手続きが実現できる．

6)　MATCH[(z)]x . y. 〈m〉

この命令は，照合条件 x . y. 〈m〉に適合した原子概念を持つ節点や弧を励起する．x=N のとき節点だけに対して，x=A のとき弧だけに対して，x=NA のときは節点と弧の両方に対して照合操作が実行される．弧の原子概念が適合して励起されると，その指定された側の端節点も励起される．この MATCH 命令によって，励起と同時に，指定された型の励起痕跡を残すこともできる．励起痕跡の型は z で指示され，1, 2, 12 が区別される．12 は #1 と #2 との両型の励起痕跡を残すよう指示するものである．照合条件による基底節点集合の指定は 1 つの再帰的拡張節点構造認識規則とみなしうる．

7)　TRANSFER STATE[(z)] . y. 〈m〉

この命令によって，節点間で励起状態を移送する．この命令には，弧原子概念に対する照合操作が含まれる．照合条件は A . y. 〈m〉である．弧の照合子は方向の指示を含み，＋，－で表現されることに注意すること．SN 上に励起節点が存在し，それに隣接して，上記の照合操作によって指定の方向から適合した弧が存在すれば，その弧を通して励起状態が隣接節点へ移

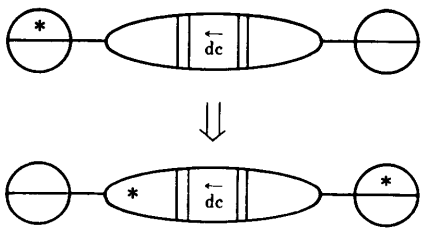

図 5　TRANSFER STATE 命令の機能
Fig. 5　Function of the TRANSFER STATE instruction.

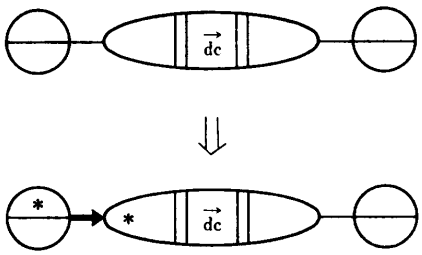

図 6 SET SHARING-EXCITATION 命令の機能
Fig. 6 Function of the SET SHARING-EXCITATION-instruction.

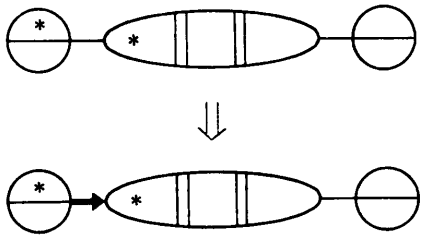

図 7 COMPOSE 命令の機能
Fig. 7 Function of the COMPOSE-instruction.

送される．たとえば照合条件が A. EQT. 〈DC : −dc〉であれば，SN に**図 5** に示すような変化が生じる．

8) SET SHARING-EXCITATION[(z)] .y. 〈m〉

この命令によって，照合条件 A .y. 〈m〉 に適合した原子概念を持つ弧を励起すると同時に，その弧において方向性を含めた励起共有設定を行う．たとえば，照合条件が A .y. 〈DC : +dc〉 であれば，SN に**図 6** に示すような変化が生じる．

記述照合子に符号がなく，〈DC : dc〉 のような場合，当該弧の両方向に励起共有設定される．一般に弧の記述照合子は＋符号を持つ場合，−符号を持つ場合，符号なしの場合という，3つの場合がある．＋符号によって有向弧の矢印の方向を，−符号によって矢印の逆方向を指示する．符号なしの場合は両方向を指示する．

9) CLEAR SHARING-EXCITATION

この命令は SN 上からすべての励起共有設定を解消する．賦活制御プログラムの中で，上記の2命令，**8)** と **9)** とで挟まれたブロックにおいて，指定の励起共有設定が効力を持つ．

一般に，励起状態や励起共有設定のようなシステム活性項はそれぞれ特定の CLEAR 命令を必要とする．指数的および符標的特定化項のようなユーザ活性項や励起痕跡等は PUT 命令によって解消される．励起痕跡がシステム活性項であるにもかかわらず，PUT 命令によって解消されるのは，これが制御情報のみならず，蓄積情報としての性質を有することを反映している．

10) COMPOSE

これは，弧に対する照合操作を含まない励起共有設定命令である．SN に**図 7** のような変化をもたらす．

11) SET LOGICAL MODE w, x

12) CLEAR LOGICAL MODE

6), **7)**, **8)** の各命令はその動作中に照合操作を含み，z 項を使って励起痕跡を残すことができる．(励起痕跡を残す必要がなければ，命令から z 項を除いておけばよい．) このことを利用して，AND 条件や OR 条件で論理探索を実行する機構を実現する．賦活制御プログラム中，あるブロックを選び，そのブロックに属する上記3種の命令のうち目的に適ったものを選んで，それらが個別に SN を励起するのを集めて，それらの AND 条件や OR 条件をとって SN の最終的な励起状態を得たいことがある．これを実現するため，励起痕跡の機構に，AND か OR かの論理条件の指定を行い，その論理条件に従って励起痕跡が累積されていくような機能を追加した．プログラムブロックは命令 **11)** と **12)** とで挟んで設定される．論理条件の指定は w 項で行う．

13) CLEAR EXCITATION x

この命令は SN 上からすべての励起状態を解消する．

以上，ASN 内部の制御を行う命令を示してきた．この命令系によってプログラムされるグラフ探索は横型探索 (breadth first) を基本とする．またこの言語はグラフ表現やグラフ探索のアルゴリズムをいちじるしく視覚化する．プログラムを書くとき，各命令は必ず行番号（あるいは文番号）を持つものとする．行番号の使用法は BASIC 言語などと同じである．

以下，ASN とその賦活制御部とにまたがる部分を制御する命令について考える．とくに賦活制御プログラムの中に制御の分岐をもたらす命令を導入することが一般のプログラミング言語と変りなく重要なことである．照合操作の結果である適合状態を制御の分岐条件とする．賦活制御部は，MMR (Multiple Match Resolver) と呼ばれる多重適合状態の処理機構を持つものとする．MMR の機能は次の2つである．

i) ASN 上，励起節点の数をかぞえる．

ii) 複数の励起節点の中から，1つだけ優先して選

び出す.

さてi）の機能を利用して，次のような分岐命令を導入する.

14) BRANCH (A).y.s

ここで，Aは行番号，yは大小比較条件，sは励起節点の数を表わす. .y.s は励起節点数に関する条件を示す. ASN の状態がこの条件に適合するとき，行番号Aの命令へ飛ぶ. 適合しなければ次の行の命令へ進む.

MMR のii）の機能を利用して，次のような読み出し命令を導入する.

15) PREFERENCE x

この命令によって，節点か弧かをx項によって選択し，励起されている複数の節点あるいは弧のうちから1つを優先して選び出し，それを読み出す. この命令が実行された後は，はじめに励起されていた節点あるいは弧のうち，いま読み出された節点あるいは弧を除いた残りのものが再び励起された状態で残留している.

以上ですべての命令の説明を終る. 賦活制御部はMMR やスタックを具備すべきことが明らかにされた. スタックの機能についてまとめれば次のようになる. スタックは命令実行に必要な照合子を格納する. また ASN から読み出された原子概念もここに格納される. いま読み出されたばかりの原子概念を照合子として使いたいとき，命令の照合子形式の中に記号$を使って指示する. たとえば，照合子〈DC：$〉を持つ命令がくれば，そのときスタックに格納されている記述子が照合子として使われる. 照合子や読み出し情報をスタックに格納するようにしたのは，言語理解システムを実現するとき，概念推論と前後して実行すべき，文章と概念表現との相互変換のための記号処理に即応できるようにとの配慮からである. 記号処理には複数のスタックが必要であるが，そのうちの1つが，ASN への接続点に位置していると考えればよい.

4. 賦活制御プログラムの実例

例 1　特定の動詞の概念化構造を励起する

SN 上で，**図8**の CD ダイアグラム[3]で表現されるような概念化構造に注目する. これを ASN 簡便図示法で表現すると，**図9**のようになる. ただし，弧の記述子はすべて略記号で表示した.

AN=ACTION, AR=ACTOR, OJ=OBJECT,
DF=DIRECTIVE-FROM, DT=DIRECTIVE-TO, SJ=SUBJECT, AT=ATTRIBUTE,
VL=VALUE, RT=RESULT

SN上のこの部分は，たとえば入力文,

John gave Mary a book.

が与えられたとき，動詞 give に触発されて，**図8**の前件［A］の部分が励起される.

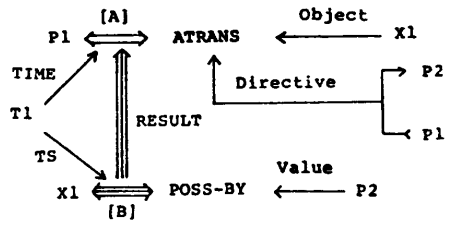

図 8　文章「P1がP2にX1を与えたので，X1はP2のものである.」の概念依存性表現
Fig. 1　The conceptual dependency representation of the sentence "X1 be possessed by P2, because P1 give X1 to P2.".

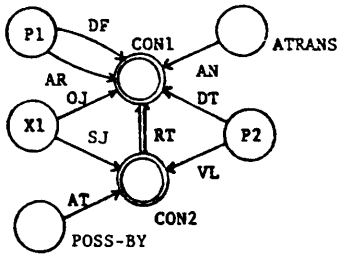

図 9　文章「P1がP2にX1を与えたので，X1はP2のものである.」の ASN 簡便図示法による表現
Fig. 9　Representation by the ASN convenient diagram of the sentence "X1 be possessed by P2, because P1 give X1 to P2.".

```
 1  SET LOGICAL MODE AND, N
 2  MATCH N .EQT. 〈DC : ATRANS〉
 3  TRANSFER STATE (2).EQT. 〈DC : +AN〉
 4  CLEAR EXCITATION NA
 5  MATCH N .EQT. 〈DC : PHYSICAL
    OBJECT〉
 6  TRANSFER STATE
    (2) .EQT. 〈DC : +OJ〉
 7  CLEAR EXCITATION NA
 8  MATCH N .EQT. 〈DC : PERSON〉
 9  TRANSFER STATE
    (2) .EQT. 〈DC : +DT〉
10  CLEAR EXCITATION NA
11  MATCH N .EQT. 〈#1〉
12  TRANSFER STATE
    (1) .EQT. 〈DC : −AR〉
13  CLEAR EXCITATION NA
```

```
14  MATCH (1) N .EQT. ⟨DC: PERSON⟩
15  CLEAR EXCITATION NA
16  MATCH N .EQT. ⟨#1⟩
17  TRANSFER STATE
      (2) .EQT. ⟨DC: +DF⟩
18  CLEAR EXCITATION NA
19  SET SHARING-EXCITATION
      .EQT. ⟨DC: −STRUCTURAL LINK⟩
20  CLEAR EXCITATION NA
21  MATCH N .EQT. ⟨#2⟩
22  PUT N ⟨#1, #2⟩
23  CLEAR SHARING-EXCITATION
24  CLEAR LOGICAL MODE
```

プログラムの説明

動詞 give が表現する概念は，代表節点 CON 1 に率いられた再帰的拡張節点によって内部化されている．このプログラムで ASN を賦活制御すると，SN のこの部分だけが励起される．

行 1 の命令により，励起痕跡の論理条件が AND に設定され，行 24 の解消命令に至るブロックを形成する．このブロック中，行 3, 6, 9, 17 が #2 の励起痕跡を残すように指定した命令である．行 17 の命令が実行された時点に，それぞれの励起命令の AND 条件で励起痕跡 #2 が残される．また，行 12 と行 14 とが #1 の励起痕跡を残すように指定した命令であるから，行 14 の命令が実行された時点にはそれぞれの励起命令の AND 条件で励起痕跡 #1 が残されている．行 2〜4 は節点 ATRANS を励起し，その励起状態を構造的リンク ACTION を通じて代表節点に移送する．行 5〜7 と行 8〜10 とにおいて，他の構造的リンクによって同様の手続きが実行され，#2 痕跡の選択域を挟める．これらの構造的リンクを部分的に持つ代表節点があっても，それらは途中で消えていく．行 11〜18 が #2 痕跡のもう 1 つの選択条件を加える．CON 1 は ACTOR と DIRECTIVE-FROM との 2 つの格に対して同一の人物を取らなければならない．この制約条件を加える過程に #1 励起痕跡の AND 条件を利用した．行 17 の命令が実行された時点で，CON 1 のみが励起痕跡 #2 を残している．行 19 はすべての概念化構造の代表節点から構成要素節点へ向けて励起共有設定する．この行から行 23 の解除命令に至るまで，励起共有のブロックが形成される．このブロックの中で，行 21 が代表節点 CON 1 を励起する．そのとき CON 1 を中心とする概念化構造全体が一度に励起状態に入る．

例 2　概念化構造同士の間で励起状態を移送する

図 8 の CD ダイアグラムにおいて，前件 [A] が励起され，次のように特定化されたとする．

P 1→P 1 (JOHN 1)　P 2→P 2 (MARY 1)

X 1→X 1 (BOOK)

この状態からはじまり，1 サイクルの結果の推論が実行され，前件 [A] から後件 [B] が推論される過程をプログラムする．いま前件 [A] の部分が励起痕跡 #2 を残しているものとする．

```
100  MATCH N .EQT. ⟨#2⟩
101  STORE N ⟨SS: m 21⟩
102  PUT NA ⟨#2⟩
103  CLEAR EXCITATION NA
104  MATCH N .EQT. ⟨DC: CON⟩
105  PUT N ⟨SS: i⟩
106  CLEAR EXCITATION N
107  MATCH N .EQT. ⟨SS: m 21, i⟩
108  PUT N ⟨SS: i⟩
109  TRANSFER STATE
       (2) .EQT. ⟨DC: −RT⟩
110  CLEAR EXCITATION NA
111  SET SHARING-EXCITATION
       .EQT. ⟨DC: −STRUCTURAL LINK⟩
112  CLEAR EXCITATION NA
113  MATCH N .EQT. ⟨#2⟩
114  CLEAR SHARING-EXCITATION
```

プログラムの説明

行 100 で #2 励起痕跡を励起状態に戻す．この励起領域は複数の基底概念化構造から成る．これから結果の推論を実行する．そのために，この複合概念化構造を 1 つの全体と見ることはもとより，これを構成する基底概念化構造の 1 つ 1 つを識別して，そのそれぞれから CAUSE リンクを逆行して励起状態を移送し，新しく励起された概念化構造群を求める．行 101 は結果の推論のマーキングを行っている．行 104〜108 において，初期励起領域に含まれていた，すべての代表節点を取り出し励起する．行 109 でこれらの代表節点から CAUSE リンクを逆行して励起状態を移送し，その結果に励起痕跡 #2 を残す．そのあと，行 110 において励起状態をいったん解消し，行 111 によって代表節点から構成要素節点へ向けて共有設定したうえ，行 113 において ⟨#2⟩ で照合励起している．

例 3　論理条件が設定されたブロック内で，それまでの痕跡を解消する方法について

図 10 の CD ダイアグラムは “kill oneself” (自殺

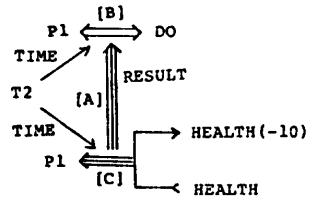

図 10 文章「P 1 が自殺する.」の概念依存性表現
Fig. 10 The conceptual dependency representation of the sentence "P 1 kill oneself.".

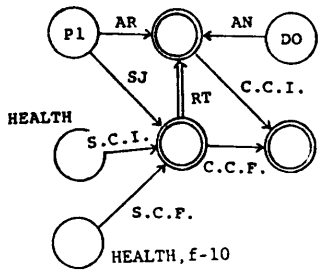

図 11 文章「P 1 が自殺する.」の ASN 簡便図示法による表現
Fig. 11 Representation by the ASN convenient diagram of the sentence "P 1 kill oneself.".

する）の概念化構造である．ASN 簡便図示法では**図 11** のようになる．ただし，弧の記述子はすべて略記号で表示した．

S. C. I.=STATE CHANGE INITIAL
S. C. F.=STATE CHANGE FINAL
C. C. I.=CAUSAL CHAIN INITIAL
C. C. F.=CAUSAL CHAIN FINAL

次に示すプログラムは，入力文

Mary killes herself.

が与えられて，SN上，図 11 に示された部分が励起・特定化される手続きを示している．このプログラムを例にして論理条件が設定されたブロック内で，それまでの励起痕跡を解消し，改めて論理条件を適用して行く手続きについて説明する．

```
201 SET LOGICAL MODE AND, N
202 MATCH N .EQT. 〈DC: DO〉
203 TRANSFER STATE .EQT. 〈DC: +AN〉
204 TRANSFER STATE
      (2) .EQT. 〈DC: −RT〉
205 CLEAR EXCITATION NA
206 MATCH N .EQT. 〈DC: HEALTH〉
207 TRANSFER STATE
      (2) .EQT. 〈DC: +S. C. I.〉
208 CLEAR EXCITATION NA
209 MATCH N .EQT. 〈FZ: f−10, DC:
      HEALTH〉
210 TRANSFER STATE
      (2) .EQT. 〈DC: +S. C. F.〉
211 CLEAR EXCITATION NA
212 MATCH N .EQT. 〈#2〉
213 TRANSFER STATE
      (1) .EQT. 〈DC: +RT〉
214 CLEAR EXCITATION NA
215 MATCH N .EQT. 〈#2〉
216 TRANSFER STATE .EQT. 〈DC: −SJ〉
217 TRANSFER STATE
      (1) .EQT. 〈DC: +AR〉
218 CLEAR EXCITATION NA
219 MATCH N .EQT. 〈#1〉
220 TRANSFER STATF
      (2) .EQT. 〈DC: −RT〉
221 CLEAR EXCITATION NA
222 MATCH N .EQT. 〈#2〉
223 STORE N 〈SS: CON 02〉
224 TRANSFER STATE
      (1) .EQT. 〈DC: +RT〉
225 CLEAR EXCITATION NA
226 MATCH N .EQT. 〈#1〉
227 STORE N 〈SS: CON 01〉
228 PUT NA 〈#1〉
229 TRANSFER STATE
      (1) .EQT. 〈DC: +C. C. I.〉
230 CLEAR EXCITATION NA
231 MATCH N .EQT. 〈#2〉
232 TRANSFER STATE
      (1) .EQT. 〈DC: +C. C. F.〉
233 CLEAR EXCITATION NA
234 MATCH N .EQT. 〈#1〉
235 STORE N 〈SS: CON 12〉
236 CLEAR EXCITATION NA
237 MATCH N .EQT. 〈SS: CON 01〉
238 TRANSFER STATE .EQT. 〈DC: −AR〉
239 STORE N 〈SS: MARY 1〉
240 CLEAR EXCITATION NA
241 PUT NA 〈#1, #2〉
242 CLEAR LOGICAL MODE
243 SET SHARING-EXCITATION
      .EQT. 〈DC: −STRUCTURAL LINK〉
244 MATCH N .EQT. 〈SS: CON 12〉
```

プログラムの説明

行 201 は論理条件 AND を設定し，行 242 の解除命令に至るブロックを形成する．SN 上，図 10 に示された概念化構造の［C］を同定するために，行 202〜

205, 行 206～208, 行 209～211, 行 212～221 がそれぞれの条件で代表節点を探り，その AND 条件で励起痕跡 #2を残す．行 220 の命令が実行された時点で指定の代表節点が同定される．行 222～236 において3つの代表節点に絶対同定子を与える．DO-Action の概念化構造へ CON 01, HEALTH の状態変化へ CON 02, そして，この2つによって構成される因果関係連鎖の代表節点に CON 12 という絶対同定子を格納する．行 237～240 において DO-Action の Actor を MARY 1 で特定化する．行 243, 244 において入力文によって指摘された部分が励起状態になる．

一般に，論理条件 AND が設定されたブロックにおいて，たとえば命令，

PUT NA 〈#1〉

が現われると，この命令の前にある #1 痕跡に対するすべての AND 条件が解消されて初期状態に戻る．上記のプログラム中，行 228 がその例である．行 213, 217, 224 の3つの AND 条件が行 228 で解消され，そのあと改めて行 229 と 232 とで AND 条件が取られている．AND 条件設定の原理は，設定の初期時点ですべての節点あるいは弧に痕跡を付け，条件が重なるにつれて，適合しない節点あるいは弧から痕跡を消去して行く過程によって与えられる．したがって上記の命令によって，その過程がふりだしに戻る．

同様に，OR 条件設定のブロックでは，たとえば命令

PUT NA 〈$\overline{\#1}$〉

によって，それ以前の OR 条件が解消される．初期状態で，すべての節点あるいは弧が励起痕跡を持たず，OR 条件が重なるにつれて，痕跡を持つものが増えてくるのが，OR 条件設定の原理である．したがってこの命令によって #1 痕跡がすべて解消されると，OR 条件はふりだしに戻る．

5. むすび

制御対象がネットワークであるから，意味ネットワーク賦活制御言語も一種のグラフ処理言語だと考えてよいが，従来のグラフ処理言語とは原理的な違いがある．従来のグラフ処理言語[5),6)]は一様に，連糸操作用言語 SNOBOL をグラフ操作用に拡張したものである．一方，賦活制御言語はネットワーク横断を柔軟に実現することに主眼を置いた言語であり，概念記憶システムの設計者が記憶の賦活を制御するプログラムを作成するための言語として考えられた．意味ネットワークの賦活動態には限られた数の類型しかない[4)]．本稿はそれらの類型を表現する言語を与え，概念記憶に関する活性化意味ネットワークモデルの中核となる部分を示した．

参考文献

1) Rumelhart, D. E.: Introduction to Human Information Processing, John Wiley & Sons, Inc. (1977).
2) 嶋津，田町：概念記憶の意味ネットワークモデル，九州大学総合理工学研究科報告，第3巻，第2号 (1981).
3) 嶋津，田町：意味ネットワークの静態構造論，情報処理学会論文誌，Vol. 23, No. 1 (1982).
4) 嶋津好生：概念記憶システムの研究――概念記憶の意味ネットワークモデルと連想プロセッサによる実現法――，九州大学出版会 (1982).
5) Savitt, D. A., Love, H. H. and Troop, R. E.: Association-Storing Processor, AD 818529, AD 818530 (1967).
6) 真野，杉藤，鳥居：グラフ処理2次元言語 GML とその機能，信学論誌，Vol J 59-D, No. 9 (1976).

(昭和56年7月3日受付)
(昭和56年11月18日採録)