知識工学と人工知能 48－3
（１９８６・９・１８）

プロダクションシステムによる意味ネットワークの探索

石田 亨　　赤埴 淳一
（ＮＴＴ電気通信研究所）

あらまし

ﾃﾞｰﾀ駆動型のﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑを用いて，効率良く意味ﾈｯﾄﾜｰｸの探索を行う方法を述べる。ﾃﾞｰﾀ駆動型のﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑは，① ﾃﾞｰﾀの変更によって，ﾙｰﾙが自動的に起動される，② 以前の探索結果をRETEﾈｯﾄﾜｰｸ内に保存し，次回以降の探索に用いることができる等，意味ﾈｯﾄﾜｰｸの維持や探索に有利な素質を備えている。

しかしながら，現時点のﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑにはﾈｯﾄﾜｰｸの探索を制御する有効な方法が無く，探索効率が著しく悪いという問題点がある。本論文ではﾃﾞｰﾀ駆動型ﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑに，① 前向き推論を制御する機能及び，② 冗長な経路の探索を防止するために事実の非活性化機能を追加することによって，意味ﾈｯﾄﾜｰｸの探索が効率良く実行できることを示す。また，上記機能を導入して開発したﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑ PLANETの概要を報告する。

"Exploration of Semantic Networks using Production Systems" (in Japanese)

by Toru Ishida and Jun-ichi Akahani
(NTT Electrical Communication Laboratories, Yokosuka-shi, 238-03, Japan)

This paper describes an efficient semantic network exploration method using a data driven production system. Production systems have the following advantages for maintaining and accessing semantic networks.

- Rules are automatically triggered by the changes of working memory elements.
- Production systems preserve previous results for future reference.

However, current production systems have no efficient inference control mechanism to explore semantic networks. PLANET (Production system LAnguage for NETwork exploration) introduces a new control mechanism for forward-chaining production firings, and enables it to inactivate unnecessary working memory elements to avoid redundancies.

1 まえがき

最近の意味ネットワークの実現例では，is-a，instance-ofによる継承機能の他に，関係固有の継承機能を用意するものが増えてきている。関係固有の継承規則[1]に基づいて継承値を計算するということは，意味ネットワーク中の各種のリンクを探索することを意味している。

本論文では意味ネットワークの1つの実現方式として，データ駆動型のプロダクションシステム(OPS5[2]，ART[3]等)を取り上げ，機能を拡張することによって効率の良い探索を実現できること示す。

プロダクションシステム(以下、PSと略す)で意味ネットワーク(以下，SNと略す)を実現するには，SNのデータをワーキングメモリに格納し，SNの規則をプロダクションルールで表せばよい。データ駆動型のPSは，SN探索に適した以下の素質を備えている。

① データ駆動型のPSでは，ワーキングメモリの更新が引き金となって，ルールが自動的に起動される。従って，SNに外部から変更が加えられると，SNの規則を表したルールが起動され，影響を受ける継承値が自動的に再計算されることになる。この性質はSNの維持に有効である。

② データ駆動型のPSの内部表現であるRETEネットワーク[4]は，ルールの条件部の判定に必要な計算の途中結果を全て保存するものである。従って，特別なプログラミングテクニックを要することなく，以前のSN探索中に行った計算結果を，次の探索に用いることができる。この性質はSN探索の効率化に有効である。

しかしながら，現状のPSの機能で，SN探索を行おうとすると，以下の問題が生じる。

① 前向き推論では，ルールの発火を制御する方法がないため，SNの全探索が行われてしまう。一方，後向き推論では，探索を制御するために多数のgoalが生成されるが，このgoalが探索効率が著しく低下させる原因となる。また3章で述べるように，現在のPSで採用されている深さ優先の競合解決戦略はSN探索に適さないという問題がある。

② SNでは同一の事実が種々の探索経路で導き出されることがある。特にSN中に冗長な知識が存在すると，冗長な経路を探索するために無駄なルールの発火が多数生じてしまう。

本論文では，上記の問題を解決し，PSが本来備えているSN探索に適した素質を活用するために，① 前向き推論でのルールの発火を制御する機能及び幅優先の競合解決戦略，② 冗長な経路の探索を防ぐために冗長な事実を非活性化する機能，をPSに導入する。また，これらの機能を導入して試作したプロダクションシステム PLANET(Production system LAnguage for NETwork exploration)について報告する。

2 プロダクションシステムの定義

この章では，PSの基本部分を従来のPSの機能の範囲で定義する。SN探索のために導入する機能については，4章で改めて述べる。

2.1 事実データベース

本論文では従来のPSのワーキングメモリを事実データベースと呼ぶ。事実データベース(以下事実DBと略す)は，以下の形式から成る事実の集合である。

(事実ID メタ情報 事実情報)

① 事実ID

事実を同定するためのものである。IDは事実の生成順にふられる整数である。

② メタ情報

事実の生成日時，理由や，確信度等，事実そのものに関する様々な情報である。ここでは，以下のメタ情報のみ導入しておく。

in: 事実が存在するための"根拠"のリスト(or)である。"根拠"は事実IDのリスト(and)である。無条件に存在する事実の"根拠"はnil とする。

③ 事実情報

事実の内容を表わすリストである(従来のPSのワーキ

ﾝｸﾞﾒﾓﾘｴﾚﾒﾝﾄに相当する)。条件照合の対象はこの事実情報であり，照合が成功するとIDが返却される。事実情報の変更は，旧事実の除去と新事実の生成を意味する。(ﾒﾀ情報の変更は事実情報を変更することにはならない。)また，同一の事実情報を持つ事実は，事実DB中に1個しか存在しない。

2.2 ﾌﾟﾛﾀﾞｸｼｮﾝﾙｰﾙ

ﾌﾟﾛﾀﾞｸｼｮﾝﾙｰﾙの形式を以下に示す。

```
(defrule ﾙｰﾙ名
        { <ﾊﾟﾀｰﾝ> |
         (bind <ﾊﾟﾀｰﾝ変数> <ﾊﾟﾀｰﾝ> ) |
         (not <ﾊﾟﾀｰﾝ> ) |
         (test <Lisp式> )}+
        -->
         {(make <事実情報> {:in <根拠> }) |
          (remove <事実ID> )}+)
```

① ﾊﾟﾀｰﾝは事実情報と同じ形式のﾘｽﾄである。但し，ﾊﾟﾀｰﾝ変数を任意の場所に含んでよい。ﾊﾟﾀｰﾝ変数は?x，?y...と表わす。?は無名変数である。

② 条件部(LHS: Left Hand Side)に記述されたﾊﾟﾀｰﾝは事実DBと照合される。bindは，ﾊﾟﾀｰﾝと照合した事実のIDをﾊﾟﾀｰﾝ変数に束縛する。notについては後述する。testはlisp式を評価し，その返却値を条件判定結果とする。

③ 実行部(RHS: Right Hand Side)には，事実の作成を行うmake，事実の除去を行うremoveが記述できる。makeにinﾘｽﾄが指定されている場合には，ﾒﾀ情報として登録される。removeによって除去される事実をinﾘｽﾄ中に持つ全ての事実は連鎖的に除去される。

3 ﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑによる意味ﾈｯﾄﾜｰｸ探索の問題点

3.1 後向き推論による探索制御の問題点

(1) 探索制御の必要性

図1(1)に，関係を辿り新たな関係を導く，SNの規則を表した前向きのﾌﾟﾛﾀﾞｸｼｮﾝﾙｰﾙを示す。このﾙｰﾙを図1(3)の事実DBに適用すると，発火を繰り返し，(older-than Taro Hanako)を含む全ての組合せが生成される。生成される事実の総数は151であるが，ﾙｰﾙの発火回数はさらに多く678回となる。必要とされる事実が(older-than Taro Hanako)だけであれば，多くのﾙｰﾙの発火は無駄であったことになる。

(2) 後向き推論での問題点

図1(2)に，後向き推論のﾙｰﾙを示す。このﾙｰﾙに(goal older-than Taro Hanako)が与えられると，(older-than Taro Hanako)を生成すべく探索を開始する。ﾙｰﾙ中の(a)(b)の条件で，その時点の変数束縛を用いて自動的にsub-goalが生成されるとしよう。この後向き推論は，一見効率的に働くかにみえるが，実は以下に示す問題が存在する。

① goalが多数生成される。

図1(4)に生成されるgoalの一部を示す。この場合には34個のgoalが生成されている。 PSでは，goalも他の事実と同様にRETEﾈｯﾄﾜｰｸに保持されるので，goalが 多数生成されることは，記憶効率上も実行効率上も好ましくない。

② 排他的でないgoalが生成される。

図1(4)に示すように,(goal older-than Taro ?)と(goal older-than Taro Hanako)等，排他的でないgoalが生成される。これによって，同一の事実を生成するために，ﾙｰﾙが異なるgoalに基づいて複数回発火する。

③ 目的達成後もﾙｰﾙが発火しつづける。

当初の目的(goal older-than Taro Hanako)を満足する事実が得られた後にも，途中で生成されたsub-goalを満足させるためにﾙｰﾙが発火しつづける。結局，この例では，59個の事実が生成される。ﾙｰﾙの発火総数は201回である。

この他，深さ優先探索がSN探索に適さないとい

う問題がある。ﾈｯﾄﾜｰｸの探索には幅優先の方が探索範囲が狭く効率的である。しかし，例えばTaroとHanakoの2点を中心とした幅優先探索を実現するには，幅優先の競合解決戦略を導入するだけでは十分でない。図1(2)の後向き推論ﾙｰﾙの例では，goalを先頭に指定することによって，?n1側から探索を行うことが方向付けられいるからである。 goalによらない探索制御方法が必要である。

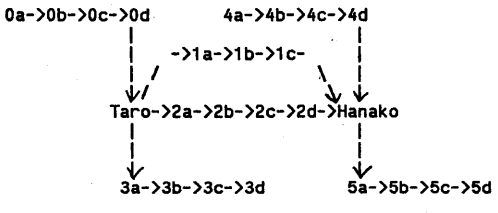
```
(defrule relation-chain
  (bind ?1 (older-than ?n1 ?n2))
  (bind ?2 (older-than ?n2 ?n3))
  -->
  (make (older-than ?n1 ?n3) :in (?1 ?2)))
```

(1) A Forward Chaining Rule

```
(defrule relation-chain
  (bind ?1 (goal older-than ?n1 ?n3))
  (bind ?2 (older-than ?n1 ?n2))  ------ (a)
  (bind ?3 (older-than ?n2 ?n3))  ------ (b)
  -->
  (make (older-than ?n1 ?n3) :in (?1 ?2 ?3)))
```

(2) A Backward Chaining Rule

```
(older-than 0a 0b)
(older-than 0b 0c)
(older-than 0c 0d)
(older-than 0d Taro)
        :
```



(3) A Fact Data Base

```
(goal older-than Taro Hanako)
(goal older-than Taro ?)
(goal older-than 1a Hanako)
(goal older-than 1a ?)
        :
```

(4) Goals

Fig.1 Problems on SN Exploration

3.2 多重経路探索の問題点

SN中に同一の事実を導く経路が多数存在することがあるが，その原因としては以下のものが考えられる。

(1) 事実には冗長性はないが，経路が多数存在する場合。図1(3)のTaro,Hanako間の経路がこれにあたる。

(2) 事実が冗長性であるため，経路が多数生じる場合。図2に例を示す。この例では，sister, sibling, family, relative の4階層の関係間のis-aが定義されている。従って，経路の数は4×4＝16通りである。探索に要する経路が長くなれば経路の数は爆発するので，冗長な事実をRETEﾈｯﾄﾜｰｸから除去し，多重経路の探索を防ぐﾒｶﾆｽﾞﾑの導入が必要である。

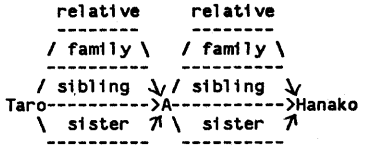


Fig.2 Redundant Paths in SN

4 意味ﾈｯﾄﾜｰｸ探索に必要な機能

3章で述べた問題点を解決するために，以下の機能をPSに導入する。

4.1 前向き推論の探索制御

(1) gate

前向き推論を制御するための機能として，goa1と対象的なgateを導入する。以下，goalとgateの機能を対比して述べる。

goal:

ﾙｰﾙの条件部に記述されたものをgoalﾊﾟﾀｰﾝ，事実DB中のものを単にgoalと呼ぶ。goalﾊﾟﾀｰﾝは，事実DB中のよりspecificなgoalとのみﾏｯﾁする。goalﾊﾟﾀｰﾝは，ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｽﾀｲﾙとしてﾙｰﾙの条件部

の先頭に置かれることが多い。

gate:

一方，gateﾊﾟﾀｰﾝは，事実DB中のよりgeneralなgateとのみﾏｯﾁする。gateﾊﾟﾀｰﾝはｽﾀｲﾙとしては，ﾙｰﾙの条件部の最後に置かれることが多い。図3にgateの例を示す。

```
goal pattern:      (goal relative Taro ?x)

matched goals:     (goal relative Taro Hanako)
                   (goal relative Taro ?)
unmatched goals:   (goal relative ? Hanako)
                   (goal relative ? ?)

gate pattern:      (gate relatove Taro ?x)

matched gates:     (gate relative Taro ?)
                   (gate relative ? ?)
unmatched gates:   (gate relative Taro Hanako)
                   (gate relative ? Hanako)
```

Fig.3 Goals and Gates

(2) any

gateの効果を，より高めるためにnotと対象的なanyを導入する。

not: not は条件ﾊﾟﾀｰﾝと照合する事実が全く存在しなければ満足される。

any: any は条件ﾊﾟﾀｰﾝと照合が成立する事実が1つでも存在すれば満足される。

not とany は，単にﾁｪｯｸに用いられるので，照合により生じた変数束縛は以降の条件照合に影響しない。(any はprologやPOPS2[5]の2重否定と同様にふるまう。)

(3) 探索制御の例

図4(1)にgateとany を用いたSN探索のためのﾙｰﾙを示す。このﾙｰﾙに，(gate older-than Taro ?)が与えられると，Taroと関係する事実だけが生成される。図4(2)はgoalからgateを生成するﾙｰﾙを表している。今，(goal older-than Taro Hanako)が与えられると，このﾙｰﾙが発火し，(gate older-than Taro ?)と(gate older-than ? Hanako)が生成される。競合解決戦略を幅優先(LEX[2]の逆順)としておくと，TaroとHanakoの2点をを中心として同心円を描くように探索が進められる。しかし，この例では目的(goal)とした事実，(older-than Taro Hanako)が生成されても探索は停止しない。探索を停止させるには，次節で述べる事実の非活性化機能を用いる必要がある。

```
(defrule relation-chain
  (bind ?1 (older-than ?n1 ?n2))
  (bind ?2 (older-than ?n2 ?n3))
  (any (gate older-than ?n1 ?n3))
  -->
  (make (older-than ?n1 ?n3) :in (?1 ?2)))
```

(1) A Forward Chaining Rule with Gate

```
(defrule Gate-1
  (bind ?1 (goal older-than ?n1 ?n2))
  (test (instanciated-p ?n1))
  -->
  (make (gate older-than ?n1 ?) :in (?1)))

(defrule Gate-2
  (bind ?1 (goal older-than ?n1 ?n2))
  (test (instanciated-p ?n2))
  -->
  (make (gate older-than ? ?n2) :in (?1)))
```

(2) Rules for Creating Gates

(instanciated-p ?x) returns t when ?x is fully instanciated; no ? is included in the value of ?x.

Fig.4 SN Exploration under Gate Control

4.2 事実の非活性化

3.2で述べた多重経路の探索を防ぐ手段として，事実を非活性化する方法を述べる。まず，事実の状態と根拠について述べ，次にmake，remove等の事実を操作するﾌﾟﾘﾐﾃｨﾌﾞを拡張する。

(1) 事実の状態

事実の活性化のﾚﾍﾞﾙを以下の3状態で表す。

active:

活性化された事実である。探索に用いられる。(即ち，RETEﾈｯﾄﾜｰｸ内に置かれる。)

sleep:

事実は論理的な根拠を持つが，冗長であるため非活性化されている状態である。事実DBには存在するが探索には用いられない。(即ち，RETEﾈｯﾄﾜｰｸ中には存在しない。)

killed:

事実の存在は論理的に根拠がなく，除去されるべき事実である。しかし，何らかの理由で復活する可能性があるため事実DBに残されている。例えば，矛盾を解消するために，除去される事実等がこれにあたる。探索には用いられない。(即ち，RETEﾈｯﾄﾜｰｸ中には存在しない。)

(2) 事実の根拠

事実の根拠の表現は，Doyle のTMS[6]を踏襲している。

in: 事実が導かれる原因となった事実を示す。

out: 事実がsleep，killed 状態となった原因の事実を示す。

事実の状態及び根拠は事実のﾒﾀ情報として登録される。

(3) 事実操作ﾌﾟﾘﾐﾃｨﾌﾞ

事実の状態と根拠を導入したので，ここで改めて事実操作ﾌﾟﾘﾐﾃｨﾌﾞを定義する。

make: 形式は(make 〈事実情報〉 :in 〈根拠〉)。

"根拠"はこの事実生成の前提となった事実IDのﾘｽﾄである。makeでは，まず同一の事実情報を持つ事実が既に存在しているかどうかをﾁｪｯｸする。存在しなければ事実を生成しactive状態とする。次に"根拠"を，生成した事実のinﾘｽﾄに追加する。特に"根拠"が指定されていない場合には，nil をinﾘｽﾄに追加する。

remove: 形式は(remove 〈事実ID〉)。

指定された事実を除去する。事実DB中の全ての事実のinﾘｽﾄ及び outﾘｽﾄから，除去された事実を含む根拠を除去する。この結果，もしinﾘｽﾄが空になる事実があれば，その事実をremoveする。outﾘｽﾄが空になる事実があれば，その事実をactive状態とする。

sleep: 形式は(sleep 〈事実ID〉 :out 〈根拠〉)。

"根拠"はこの事実がsleep させられる原因となった事実IDのﾘｽﾄである。sleep では，まず指定された事実をsleep 状態とする。次に"根拠"をoutﾘｽﾄに追加する。もし"根拠"が指定されていなければnil を追加する。

kill: 形式は(kill 〈事実ID〉 : out 〈根拠〉)。

指定された事実をkilled状態とする。次に"根拠"を outﾘｽﾄに追加する。もし"根拠"が指定されていなければnil を追加する。事実DB中の全ての事実のinﾘｽﾄ，及び outﾘｽﾄからkillされた事実を含む"根拠"を除去する。この結果，もしinﾘｽﾄが空になる事実があれば，その事実をremoveする。outﾘｽﾄが空になる事実があれば，その事実をactive状態とする。

(4) RETEﾈｯﾄﾜｰｸと事実DBの関係

従来のPSは，RETEﾈｯﾄﾜｰｸ中に全ての事実が存在することを仮定していた。しかし，事実に状態を導入したことによってこの仮定はくずれることになる。RETEﾈｯﾄﾜｰｸは，activeな事実だけを含み，探索に不要なsleep，killed 状態の事実は含まない。事実DB中の事実をRETEﾈｯﾄﾜｰｸに加える操作をRETE-add,除去する操作をRETE-deleteと呼ぶと，それらの機能は以下の様に整理できる。

RETE-add:

active以外の状態(存在しない状態を含む)から，active状態に変化した事実をRETEﾈｯﾄﾜｰｸに加える。

RETE-delete:

active状態からactive以外の状態(removeされた状態を含む)に変化した事実をRETEﾈｯﾄﾜｰｸから除去する。

(5) 事実の非活性化の例

多重経路探索の防止

図5(1)は冗長な事実をsleep させるﾙｰﾙの例である。これによって，図2に示した多重経路の探索を防ぐことができる。具体的には図2の場合の探索は以下のように行われる。

① TaroとA，AとHanakoの関係を示す事実の内，冗長なものをsleep させる。従って，active状態にあるのは，(sister Taro A)と(sister A Hanako)だけとなる。

② (sister Taro Hanako)が(sister Taro A)と(sister A Hanako)から導びかれる。

③ (relative Taro Hanako)等が，(sister Taro Hanako)から導びかれる。

この結果ﾌﾟﾛﾀﾞｸｼｮﾝの発火回数は，is-aの階層の深さをd，関係を辿る回数をnとすると，従来O(d^n)であったものが，O(d+n)に減少する。

前向き推論の制御

図5(2)はgoalを満たす事実が生成された時に，そのgoalをkillするﾙｰﾙを示している。goalがkillされると，そのgoalから生成されていたgateが除去され，TaroとHanakoの関係の探索が停止する。この様子を図6に示す。3章と同じ事実DBを用いた結果，生成された事実は11個，ﾙｰﾙの発火総数は13回である。(図5(2)のルールは効率が悪いため，実際にはmakeの中で同様の処理を行っている。)

さらに，何らかの理由でgoalを満たす事実が除去された場合には，killed状態のgoalが再びactive状態となり，gateが再度作成され，以前探索を中止した状態から探索が続行される。この結果，異なる経路により，goalを満たす事実が作成されると探索は再び中止される。

```
(defrule relation-compress
  (bind ?1 (?r1 ?n1 ?n2))
  (bind ?2 (is-a ?r1 ?r2))
  (bind ?3 (?r2 ?n1 ?n2))
  -->
  (sleep ?3 :out (?1 ?2)))
```

(1) Example of Sleep

```
(defrule kill-Goals
  (bind ?1 (goal . ?goal-body))
  (test (instanciated-p ?goal-body))
  (bind ?2 ?goal-body)
  -->
  (kill ?1 :out (?2)))
```

(2) Example of Kill

Fig.5 Sleep and Kill

```
Command: (make (goal older-than Taro Hanako))
     =>a  26  (GOAL OLDER-THAN TARO HANAKO)
#<ID 17050261>
Command: (run)
1    GATE-2  (26)
     =>a  27  (GATE OLDER-THAN ? HANAKO)
2    GATE-1  (26)
     =>a  28  (GATE OLDER-THAN TARO ?)
3    RELATION-CHAIN  (5 6)
     =>a  29  (OLDER-THAN TARO 3B)
4    RELATION-CHAIN  (9 10)
     =>a  30  (OLDER-THAN TARO 1B)
5    RELATION-CHAIN  (11 12)
     =>a  31  (OLDER-THAN TARO 2B)
6    RELATION-CHAIN  (14 15)
     =>a  32  (OLDER-THAN 1B HANAKO)
7    RELATION-CHAIN  (16 17)
     =>a  33  (OLDER-THAN 2C HANAKO)
8    RELATION-CHAIN  (20 21)
     =>a  34  (OLDER-THAN 4C HANAKO)
9    RELATION-CHAIN  (7 29)
     =>a  35  (OLDER-THAN TARO 3C)
10   RELATION-CHAIN  (14 30)
     =>a  36  (OLDER-THAN TARO 1C)
11   RELATION-CHAIN  (13 31)
     =>a  37  (OLDER-THAN TARO 2C)
12   RELATION-CHAIN  (10 32)
     =>a  38  (OLDER-THAN 1A HANAKO)
13   RELATION-CHAIN  (30 32)
     =>a  39  (OLDER-THAN TARO HANAKO)
    k<=a  26  (GOAL OLDER-THAN TARO HANAKO)
     <=a  27  (GATE OLDER-THAN ? HANAKO)
     <=a  28  (GATE OLDER-THAN TARO ?)
>> 13 productions are fired.
```

Fig.6 Example of SN Exploration

5　試作・評価

4章で述べた言語機能を備えたﾃﾞｰﾀ駆動型のﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑ PLANET(Production sysytem LAnguage for NETwork exploration)を試作した。

(1)　PLANETの構成

PLANETの構成を図7に示す。PLANETは大きく以下の3ﾓｼﾞｭｰﾙに分かれる。

①　事実DBﾓｼﾞｭｰﾙ

事実DBに対する操作をｻﾎﾟｰﾄしている。match，join等の操作が行える。

②　PSﾓｼﾞｭｰﾙ

RETEﾈｯﾄﾜｰｸを用いたﾃﾞｰﾀ駆動型のPSである。事実DBからRETEﾈｯﾄﾜｰｸへの事実のﾛｰﾃﾞｨﾝｸﾞ，ｱﾝﾛｰﾃﾞｨﾝｸﾞをきめこまかく管理することが可能である。

③　ﾏﾝﾏｼﾝｲﾝﾀﾌｪｰｽﾓｼﾞｭｰﾙ

ﾋﾞｯﾄﾏｯﾌﾟﾃﾞｨｽﾌﾟﾚｰを用いて，ﾈｯﾄﾜｰｸを図形表示するﾓｼﾞｭｰﾙである。

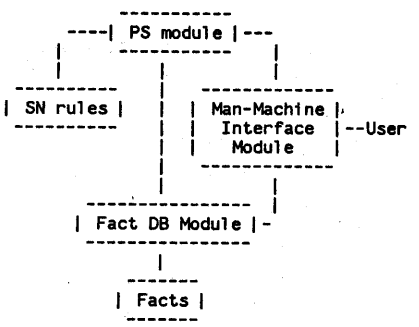


Fig.7 System Configuration of PLANET

(2) PLANETの使用例

図8にPLANETを用いたSN探索例を示している。match は事実DBの内容検索を行うもので，関係を辿る探索は行わない。一方，ask は，SNの規則を用いて探索を行い，match では得られなかった事実を導き出す。ask では，必要な個数を与え探索を途中で停止させることも可能である。

```
Command: (match (older-than Taro ?))
   5 : (OLDER-THAN TARO 1A)
   9 : (OLDER-THAN TARO 2A)
   11 : (OLDER-THAN TARO 3A)
>> 3 facts are found.
NIL
Command: (ask (older-than Taro Hanako))
    =>  39  (OLDER-THAN TARO HANAKO)
>> 1 fact is found.
>> 10 facts are created.
>> 14 productions are fired.
NIL
Command: (ask (older-than Taro ?) :no 2)
    =>  42  (OLDER-THAN TARO 5A)
    =>  43  (OLDER-THAN TARO 5B)
>> 2 facts are found.
>> 3 facts are created.
>> 10 productions are fired.
NIL
Command: (more 2)
    =>  44  (OLDER-THAN TARO 5C)
    =>  45  (OLDER-THAN TARO 5D)
>> 2 facts are found.
>> 2 facts are created.
>> 2 productions are fired.
```

Fig.8 SN Exploration using PLANET

6 むすび

ﾃﾞｰﾀ駆動型のﾌﾟﾛﾀﾞｸｼｮﾝｼｽﾃﾑに前向き推論を制御する機能，および事実の非活性化機能を導入することにより，効率的に意味ﾈｯﾄﾜｰｸを探索できることを示した。現在，PLANETを用いてSNの構築実験を行っている。SNのﾓﾃﾞﾙには，文献[7]に示したものを用いている。今後の課題としては，RETEﾈｯﾄﾜｰｸへのｵﾝﾃﾞﾏﾝﾄﾞな事実のﾛｰﾃﾞｨﾝｸﾞ機能，及びﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑを用いた大規模な事実DBの管理等が残されている。

謝辞：本研究の機会を与えて戴いた堀内敬之知識ﾍﾞｰｽ研究室長，服部文夫主幹研究員に感謝します。

参考文献

[1] Fox,M.S.: On Inhereitance in Knowledge Representation, 6th IJCAI, 1979.

[2] Forgy,C.L.: OPS5 User's Manual, Tech. Report CS-79-132, Dept.of Computer Science, Cranegie Mellon University, 1979.

[3] ART Reference Manual, Inference Corp., 1986.

[4] Forgy, C.L.: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence 19, pp.17-37,1982.

[5] 広瀬紳一： 強力なﾊﾟﾀｰﾝ・ﾏｯﾁﾝｸﾞ機能を持ったﾌﾟﾛﾀﾞｸｼｮﾝ・ｼｽﾃﾑ記述言語POPS2，日本ｿﾌﾄｳｪｱ科学会，知識ﾌﾟﾛｸﾞﾗﾐﾝｸﾞｼﾝﾎﾟｼﾞｳﾑ資料，KP-85-8，1985.

[6] Doyle,J.: A truth maintenance system,Artificial Intelligence 12, pp.231-272, 1979.

[7] 石田亨，森原一郎，古屋博行，服部文夫：知識ﾍﾞｰｽの段階的詳細化に適した知識表現ﾓﾃﾞﾙ，知識工学と人工知能研究会，46-7，1986.