例外のある性質継承に関する並列アルゴリズム

毛受 哲　森田 幸伯　伊藤 英則
(財)新世代コンピュータ技術開発機構

1. はじめに

「鳥>ハト」のように、概念間に階層関係が成り立つとき、上位の概念のもつ属性(飛ぶ)は、そのまま下位にも継承されていることが多い。そこで、概念間の推論を行う機構があれば、記憶すべき知識の量を大幅に減らせ、また、知識を管理するにも便利である。しかし、ペンギンは鳥だが飛ばない、のように例外がある場合もあり、例外と多重継承を許すと、処理や解釈が複雑になる。これに対し、今まで色々な研究がされてきたが[1,2,3]、ここでは、信心的な解の1つを、ネットワークの最長パスの線型時間で求められる並列アルゴリズムを示し、最後に並列論理型言語GHCとの親和性を述べる。これは、知識管理言語への基礎となる。

2. 継承階層ネットワーク

継承階層ネットワークは、個体と述語の集合と、その間のリンクで<p,+q>、<p,−q>の形をした順序対の集合Γから成る。ここで、pは個体か述語、qは述語を表わし、<p,+q>はis-a、<p,−q>はis-not-aに対応する。また、pからqへのリンクがあれば、pはqの子、qはpの親であると言う。

グラフで表わすときは、個体を白丸、述語を黒丸、is-aリンクを矢印、is-not-aリンクを横線付矢印で示す。

継承階層ネットワークには、同じノード間にis-aリンクとis-not-aリンクが共に存在することはない、リンクによるサイクルがない、という制限がつくものとする。

さらに、$<x_i,+x_{i+1}>$ $(1\leq i\leq n-1)$ があるとき、$<x_n,+y>$があるなら、$<x_1,..,x_n,y>$を正のパス、$<x_n,-y>$があるなら、負のパスと呼ぶ。

3. 属性継承の推論

継承階層ネットワークにおいて、pからqへのリンクがあるとき、それがis-aならpはq、is-not-aならpはqではない、と推論する。さらに、pはrであると推論できるとき、<r,+q>があるならpはq、<r,−q>があるならpはqではない、と推論する。ただし、正負の異なる2つのパス$<x_1,..,x_n,p>$、$<x_1,y_1,..,y_m,p>$ $(n,m\geq 1)$から矛盾する推論ができるとき、x_nからy_mへの正のパスがあるなら、基本的には後者のパスからは推論しないとする。(厳密な定義は[1]参照)

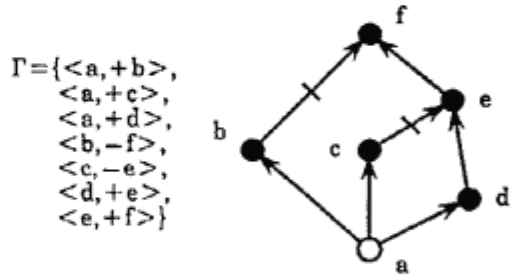

図 1. 継承階層ネットワークの例

ところが、これだけでは解は一意に定まらない。そこで、すべての可能な解の集合を求めることを信心的推論(credulous reasoning)と呼ぶ[1,2]。

他に、上の条件を適用しても相矛盾する推論ができるときは推論しない懐疑的推論(skeptical reasoning)もあり、これは解が一意に定まる[2]。

図1では、aにとって信心的な解は3つあり、

{T={b,c,d,e,f}、F={}、Q={}}、

{T={b,c,d,e}、F={f}、Q={}}、

{T={b,c,d}、F={e,f}、Q={}}、

懐疑的な解は、

{T={b,c,d}、F={f}、Q={e}}、

である。ここで、Tはis-aを推論できる述語、Fはis-not-aを推論できる述語、Qはどちらとも推論できない述語の集合を表わす。

ところが、並列に処理しても信心的推論をする効率のいいアルゴリズムはなく、解の任意の1つを保存し効率良く推論するにも、元のネットワークの構造を変えねばならず、ネットワークの更新をしたいとき処理が複雑になる。また、[2]でも述べられているように、懐疑的推論はネットワークの最長パスの線型オーダーでは時間計算量は抑えられない。しかし、信心的推論で求まる解の一部は、ネットワークの最長パスの線型オーダーで求めることができ、これらを求める並列アルゴリズムを提案する。

A Parallel Algorithm for Inheritance Hierarchies with Exceptions
Satoshi MENJU, Yukihiro MORITA, Hidenori ITOH
Institute for New Generation Computer Technology

4. Parallel Marker Propagation Machine (PMPM)

まずPMPM[1]を示す。

PMPMのプログラムは、すべてのノードまたはリンクに対して並列に実行するactionと、すべてのコンディションが成り立つすべてのノードまたはリンクに対して並列に実行するconditional action、繰り返しを行うloop <body部> loopendから成る。ループの<body部>はconditional actionの列から成り、conditional actionのどれかが実行可能な間はループの内部を繰り返し行う。

次に、後でアルゴリズムを記述するのに使用するオペレーションのリストを示すが、Mはマーカーの並び、Tはリンクのタイプの並びを表わす。

```
action:
  set[M]              ノードにMを置く。
  clear[M]            ノードのMを取る。
  set-head[M]         リンクの先にMを置く。
  clear-head[M]       リンクの先のMを取る。
conditional action:
  cond1,..,condn ⇒ action1,..,actionm
condition:
  off[M]              ノードがMもたない。
  name[x]             ノードがxという名前をもつ。
  link-type[T]        リンクがTのどれかである。
  off-head[M]         リンクの先がMをもたない。
  on-tail[M]          リンクの元がMをもつ。
  on-all-child[M]     子のノードすべてがMをもつ。
```

5. 並列アルゴリズム

信心的な解の1つを求める並列アルゴリズムの一例を、PMPMのプログラムとして示す。他の解を求めるアルゴリズムに修正することは、容易にできる。アルゴリズムは、is-aを推論するノードにTMマーカー、is-not-aを推論するノードにFMマーカー、どちらとも推論しないノードにQMマーカー、推論を進めていいノードにGMマーカーを置く。ただし、ネットワークには、rootという特別なノードと、rootから子を持たないすべてのノードへのリンクを加えるものとする。これらを加えても、ネットワークの構造は保存される。

```
procedure (x:node) = begin
 clear[TM,FM,QM,GM];
 name[x]      ⇒ set[TM,GM];
 name[root]   ⇒ set[QM,GM];
  loop
   on-tail[TM,GM], off-head[TM,FM], link-type[is-a]
              ⇒ set-head[TM], clear-head[QM];
   on-tail[TM,GM], off-head[TM,FM], link-type[is-not-a]
              ⇒ set-head[FM], clear-head[QM];
   on-tail[FM,GM], off-head[TM,FM]
              ⇒ set-head[QM];
   on-tail[QM,GM], off-head[TM,FM,QM]
              ⇒ set-head[QM];
   off[GM], on-all-child[GM] ⇒ set[GM]
  endloop
end
```

このように、この並列アルゴリズムは、マーカーを並列に伝播することによって推論を行う。各ノードは、最初に受けたTMかFMを自分のマーカーとし、自分のすべての子からマーカーが来たら、すべての親に自分が持つマーカーとリンクのタイプに合ったマーカーを送り、処理を終わる。

従って、ここで示したアルゴリズムは、継承階層ネットワークの構造を基本的に変えることなく、ネットワークの最長パスの長さに比例した時間で、信心的な解の1つを求めることができる。

図1の例における、このアルゴリズムの解は、

{T={b,c,d,e}、 F={f}、 Q={}}、

であり、信心的な解に含まれる。

6. GHCでの実現

上のアルゴリズムは、並列論理型言語GHC[4]によって容易に記述することができる。

図1の構造は図2のように書ける。genによってノードに対応したnodeというプロセスを立ち上げる。nodeは、第1引数が名前、第2引数が入力のリスト、第3引数がノードから出ているis-aリンクの先のノードへの出力、第4引数がノードから出ているis-not-aリンクの先のノードへの出力である。そして、リンクは同じ名前のストリーム変数によって実現され、質問やマーカーが流れる。node以下のプログラムは省略してあるが、入力が来ると部分計算をし、すべての入力が来たら出力をするように同期を取っている。このように、本アルゴリズムは、GHCの機能を使って容易に実現することができる。

```
gen(Root) :- true |
             node(a,[Root]      ,TM_a,_    );
             node(b,[TM_a]      ,_    ,FM_b);
             node(c,[TM_a]      ,_    ,FM_c);
             node(d,[TM_a]      ,TM_d,_    );
             node(e,[FM_c,TM_d],TM_e,_    );
             node(f,[FM_b,TM_e],_    ,_    ).
```

図 2. GHCによるプログラム例

7. おわりに

知識管理言語への基礎として、継承階層ネットワークに対し、解の候補の1つを効率良く求める並列アルゴリズムを示した。これは、GHCで自然に実現できることも示した。今後の課題としては、他の種類の推論を効率良く実行する方法や、様相の概念を取り入れることなどが考えられる。

[参考文献]

[1] Touretzky, D. S. ,"The Mathematics of Inheritance Systems", Morgan Kaufmann, 1986.

[2] Horty, J.F. et al. "A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks", Proc. of AAAI, 1987.

[3] 坂間他「非単調並列継承ネットワーク」内部資料.

[4]「並列論理型言語GHCとその応用」、共立出版、1987.