TM-1207

CBR における事例ルールを用いた並列部分照合

前田　茂

August, 1992

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

CBRにおける事例ルールを用いた並列部分照合

Parallel Partial Matching using Case-Rule for Case-Based Reasoning

前田 茂
Shigeru MAEDA
maeda@icot.or.jp
(財) 新世代コンピュータ技術開発機構
Institute for New Genration Computer Technology

1 はじめに

事例ベース推論(Case-Based Reasoning: 以下CBR) [Kolodner 87]は，過去の類似の事例を新しい問題に適用することによって問題解決をはかるものである。主要な問題の一つに，類似性の判断がある。これはつまり何をもって類似と判断するかという問題である。

我々は，CBRとルールベース推論を統合した法的推論システムHELIC-II [前田91]を開発した。これはまず過去の判例に現れる判断を事例ルールと呼ばれるある種のルール形式で記述し，また，条文を論理的な条文ルールとして記述しておく。具体的な問題の記述が与えられた時には，CBRが過去の類似の判例(事例ルール)を用いて，より抽象的な法的概念(判断)を作りだし，ルールの左辺部を満たす法的概念が生成された時点でルールベース推論が動き始めるというものである。

事例ルールの左辺部では類似照合を行うことによって類似の状況を検索する。この類似照合は，左辺部に現れる個体間の概念階層による照合と，左辺部の部分照合により行われる。ところで，一般に，判例を用いて判断を行うには大量の判例が必要とされる。なぜなら，判例が多いほど類似の判例が事例ベースに存在する確率が上がるし，また，各々の判例に存在する部分的な判断の組み合わせによって推論を行うことも容易になるだろう。

この判例検索を高速に行おうとすると，当然，並列処理などの手法が要求される。我々は事例ルールの左辺部をRete [Forgy 82]風のプロセスネットワークに展開し，各々を並列に照合する手法により高速化を実現した。以下では，この並列部分照合方式を，開発した並列論理型言語KL1で記述し，並列推論マシンMulti-PSI, PIM上で実現したので，その実現方式について問題点とその解決方式と共に述べる。

2 法的推論における事例ベース推論

条文の記述にはしばしば抽象的な法的概念が存在するので，現実の問題が与えられた時に，直接条文のみを用いて問題解決をはかるのは一般的に困難である。例えば，刑法210条は，“過失により人を死に致したものは過失致死罪により罰せられる”

という内容であるが，この中で“過失”や“人”の定義などは条文には存在しない。したがって，例えば，人を驚かしたら持病の心臓病のため心臓麻痺を起こして死んでしまった場合は過失に当たるかとか，胎児はこの場合の人に当たるか否かなどの判断は，他の何らかの知識を用いなければならない。

この知識として法律の分野では判例が存在する。法律の専門家もしばしばこの問題の解決に過去の判例を用いていると考えられる。つまり，過去の類似の判例を引用して新しい問題の判断を生成する。例えば，脳疾患の人をそれとは気付かずに脅迫して脳出血のために死に致らしめた事件に対し，過失致死の判断がなされた過去の判例があった場合には，これを引用して上の心臓麻痺の事件でも過失と判断するというようなものである。

この推論を計算機上で行う場合，CBRの技術

がまさに適切である。HELIC-IIでは，過去の判例を，その事件の状況と，そこでなされた法的な判断の集まりの組として記述する。基本的に状況の表現は，ある行為や動作(event)と行動主体や物体(object)をノード，それらの関係をリンクとした意味ネットワークである。実際にはこの意味ネットワークをフレーム形式に分解したもので記述し，さらにその間の時間関係を，Allenの時間論理の表現[Allen 83]にしたがって記述する。新しい事件の状況の記述も同様な形式で行う。一方，判例で行われた法的な判断は，事例ルールと呼ばれる形式で記述する。これは，記述形式としては一般的なIf-Thenルールに近い。以下ではこの知識表現について簡単にふれておく。

3 判例および問題の表現

判例および新しく与えられる事件(問題)として，今，次のようなものを考える。

判例 浦田が殺意を持って麻紐で市之助の首を絞めると，被害者が身動きをしなくなったので，死亡したものと思った被告人が，被害者を海辺の砂の上に放置したところ，被害者が砂を吸い込んで死亡した。

問題 甲はまだ幼児の実子太郎の養育に疲れ，厳寒期のある夜，歩道上に太郎を捨てた。そこを乙が発見し，警察に送り届けようとして，自動車に乗せて運転中，誤って事故を起こし，太郎に瀕死の重症を負わせた。乙は，太郎が死んだものと思い，その場に太郎を置き去りにして逃走した。太郎はその夜凍死した。甲と乙の罪責を述べよ。

これらの記述の一例は以下の通りである。

判例

```
case(case128,"錯誤は因果関係を遮断しない",[
...
  during(首絞め1,故意1),
  meets(首絞め1,気絶1),
  during(誤認する1,気絶1),
  before(放置する1,死亡2),
  meets(気絶1,死亡2),
...
]).
...
自然人(浦田,[性別=女]).
自然人(市之助,[性別=男]).
絞首(首絞め1,[主体=浦田,客体=市之助]).
故意(故意1,
  [主体=浦田,行為=首絞め1,目的=死亡1]).
気絶(気絶1,[主体=市之助]).
死亡(死亡1,[主体=市之助,原因=窒息]).
...
```

問題

```
problem(甲女の事件,"昭和60年度",[
...
  before(保護する1,運転する1),
  during(事故を起こす1,運転する1),
  meets(事故を起こす1,傷害1),
  before(運転する1,置き去り2),
  before(置き去り2,死亡2),
...
]).
...
運転する(運転する1,[主体=乙]).
事故を起こす(事故を起こす1,[主体=乙]).
傷害(傷害1,[主体=乙,客体=太郎]).
死亡(死亡2,[主体=太郎,原因=寒さ]).
...
```

この例で，case(...) および problem(...) の部分は，判例および問題の主に時間関係の記述であり，順に，判例および問題のID，コメント，eventおよびobject間の時間関係の集合である。時間関係の記述は，[Allen 83]の記述に従い，relation(A,B)の述語形式で行う。ここで，relationは時間関係，A,Bはそれぞれ，event,objectのいずれかである。

次の記述は，event,objectの定義であり，syntaxは以下の通りである。

```
UpperConcept(Instance, [Attribute=Value, ...]).
```

Instanceは，event,object名であり，UpperConceptはInstanceの直接の上位概念である。HELIC-IIでは，概念の階層の辞書を持ち，このUpperConceptと，概念階層を用いてInstance間の照合を行う。例えば，判例の死亡1と問題の死亡2は，直接の上位概念として，死亡を持つので照合を取ることができる。一方，気絶1と傷害1は直接の上位概念

は照合しないが，概念辞書にそれぞれの直接上位概念気絶と傷害の共通上位概念が存在すれば，照合は可能である。最後に，[と]の中のAttribute, Valueはそれぞれ属性-属性値の組の並びである。

次に，判例に対してはそこでなされた判断を，事例ルールという形式で記述する。今，上の判例で，以下のような判断がなされたとする。

判断 浦田が殺意を持って市之助の首を絞めた行為と市之助の気絶との間に法的な因果関係が認められるのは当然であるが，第一の行為が直接市之助の死の結果を生じた訳ではないので，第一の行為と市之助の死亡との間の因果関係は認められない。

これを，記述したものが以下である。

```
rule001(''因果関係の中断'',
  [条文=第199条,主張者=弁護,勝敗=敗訴],
  [絞首(首絞め1,[
     主体=浦田/trivial,
     客体=市之助/trivial]),
   故意(故意1,[
     主体=浦田/trivial,
     行為=首絞め1/important,
     目的=死亡1/important]),
   死亡(死亡1,[
     主体=市之助/trivial,
     原因=窒息/trivial]),
   before(首絞め1,死亡1),
   因果(因果1,[
     原因=首絞め1/important,
     結果=気絶1/important])]
-->
  [-因果(因果3,[原因=首絞め1,結果=死亡1])]).
```

--> の上が左辺部 (Left Hand Side: LHS)，下が右辺部 (Right Hand Side: RHS) である。LHSでは，ルールID，コメント，主張者などの関連情報と共に，RHSの判断の条件となる状況を，object, eventおよびそれらの間の時間関係で記述する。これらは，状況の記述と同じ形式であるが，フレームの先頭に"-"をつけることで否定を表現できる。そして，LHS要素のインスタンス名，属性-属性値の三つ組に対して，trivial, importantなどの重みを付加する。例えば，故意1の主体が浦田であるという状況は重要ではないが，その目的が死亡1であるという状況は重要であるという場合は，重みとしてそれぞれ，trivial, importantを与える。

RHSでは，以上の条件の大半が満たされた場合に生成される判断を記述する。この例では，首絞め1と死亡1との間に因果関係が認められないという否定の判断が生成される。HELIC-IIでは，考えられる全ての判断を生成する仕組みになっているので，Working Memory(WM)要素の修正や削除は行われない。

4 並列部分照合の概要

以上述べた知識を用いてHELIC-IIのCBR部では，部分照合を並列に行う。並列部分照合の処理の概略を図1に示す。図の四角で囲まれた部分がLHSの照合処理である。事例ルールのLHSは，プロセスネットワークに展開され，初期事実の入力や新しい判断の生成の度に照合処理が行われる。

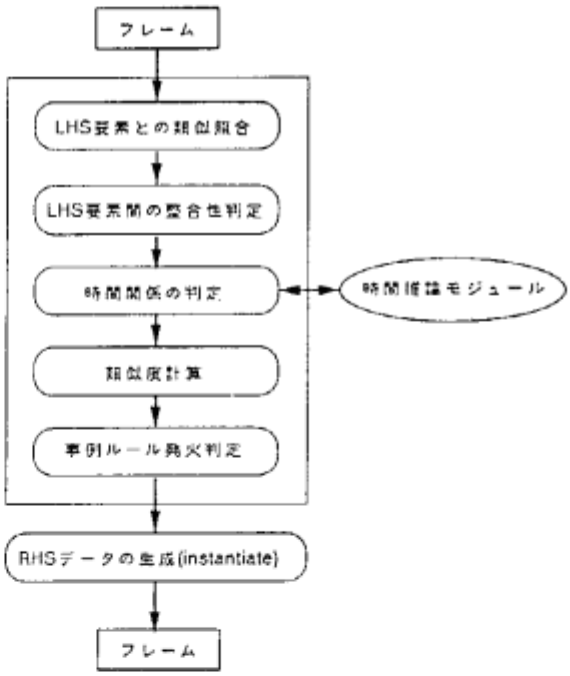

図1: 並列部分照合の流れ

ネットワークを流れるトークンの形式は，状況記述などのフレーム形式そのものである。以下に例をあげる。

死亡(死亡2,[主体=太郎,原因=寒さ])

フレーム形式のトークンが入力されると，まず先に述べた通り，各々のLHS要素とトークンの照合が行われる。この時，インスタンス名，属性値の照合は直接上位概念および概念階層辞書を用いて行い，共通の上位概念が存在する場合，照合がなされたと判断し照合した部分のトークンを流す。属性名に対しては現在の所，同一性の判断しかしていない。

次に，各々のLHS要素でなされた照合結果をもとに，LHS要素間の整合性を判定する。これは，Reteでいうところのinter-elementのチェックである。例えば，以下のLHSにおいて，首絞め1の客体の属性値に照合した値と死亡1の主体の属性値に照合した値が同じかどうかを判定するものである。これが例えば同じ太郎である場合は，そのトークンの組み合わせは許されるが，太郎と花子である場合は，その組み合わせは許されないので，ここを通過しない。これら二つの部分の処理の詳細と問題点，その解決方法は次章で詳しく説明する。

[絞首(首絞め1,
  [主体=浦田/trivial, 客体=市之助/trivial]),
 死亡(死亡1,
  [主体=市之助/trivial, 原因=窒息/trivial]),
...

LHS要素間の整合性が満たされたトークンの組み合わせのbindingsが，次に，時間関係の判定に用いられる。前述の事例ルールの例では，before(首絞め1,死亡1)という記述があるが，まずこの首絞め1と死亡1を新しい事件の値に置き換える。HELIC-IIでは，[Allen 83]をもとにした時間推論モジュールを持つ。ここにはあらかじめ問題の時間関係の記述が与えられてあり，queryが入力されると，解としてその時間関係が1)一意に特定できる，2)存在する可能性がある，3)存在しないのいずれかを返す。したがって，たとえば，首絞め1に対して傷害1が，死亡1に対して死亡2が対応したとすると，queryとしてquery(before(傷害1,死亡2))(傷害1は死亡2の前に起こったか?)が時間推論モジュールに与えられる。その解が1)または2)の場合，照合処理は全て成功し次の類似度計算に移る。

類似度計算では，直接上位概念と概念階層辞書を用いて照合したインスタンス名間と属性値間のリンクの数(距離)と，LHS要素の重みを用いて計算される。そして，それが与えられた閾値を越えた場合，RHSデータの生成の処理に移る。

RHSデータの生成では，流れてきたトークンのbindingsを用いてRHSをインスタンシエートする。ここで問題となるのは，事例ルールのLHSの照合は部分照合であるので，必要な値のbindingsが存在しないことがある場合である。したがって，たとえば，以下のRHSで結果に対するbindingsが存在しない場合は，RHSデータは生成されないことになる。

-因果(因果3,[原因=首絞め1,結果=死亡1])

5 並列部分照合プロセスネットワーク

前述した通り，事例ルールのLHSはRete風の部分照合プロセスネットワークに展開され，その各々の照合ノードの処理は並列に実行される。処理の一例を図3に示す。ここで，丸い箱はLHS要素類似照合ノードを示し，ここでは一つのLHS要素との類似照合が行われる。mergeノードは，二つのノードの出力の整合性を判定するとともに前回までに照合された結果を保存するmemoryを左右に二つ持つ。

ただし，Reteとは異なり，ルール間にまたがるパターンの共有は行わない。その理由は，以下の通りである。1)Reteの完全照合と異なり，概念階層辞書を用いた照合なので，共有するともとのインスタンス名などの情報を保持するのが困難になる。つまり，binding情報を作るのが困難になる。2)ルールがプロセッサ間にまたがる場合，共有ノードがあるプロセッサと他のLHS要素があるプロセッサの間の通信コストが問題になる。

まず最初に，LHS要素類似照合ノードの照合処理を説明しよう。例を図2に，アルゴリズムの説明を以下に示す。ここでも，各属性-属性値の組の照合も並列なプロセスに展開され実行される。ただし，memoryはトークンが流れている間だけ有効である。

1. トークンの入力
2. トークンとLHS要素の符号(否定か否か)は同一か?
   2.1 同一なら3へ。
   2.2 異なるならEndへ。

3. トークンとLHS要素のインスタンス名の直接上位概念は同一か？

3.1 同一ならインスタンス名間の距離を2にして4へ。

3.2 異なるなら距離の初期値を2にして概念階層辞書をたどって共通上位概念を探索する。共通上位概念が見つかった場合は，そこまでの距離を初期値に足してbindingsを作成し4へ。見つからなかった場合はEndへ。

4. 概念階層辞書を用いて同じ属性名を持つ属性値の照合をインスタンス名の照合と同様に行う。

4.1 共通上位概念が見つかった場合は，bindingsを作成する。

End. bindingsがあれば出力する。

図2では，以下のLHS要素の類似照合処理に，トークンとして以下が入力された場合を例にとって示している。LHS要素は，大きな四角内のようなプロセスに展開される。

LHS要素 死亡(死亡1,[主体=市之助,原因=窒息])

トークン 死亡(死亡2,[主体=太郎,原因=寒さ])

まず，トークンの符号が調べられ，LHS要素の符号と同一なので，次に直接上位概念の同一判定に移る。直接上位概念はいずれも死亡で一致するので，bindingsとして{死亡2/死亡1}を生成し，次の属性-属性値組リストの類似照合を行う。この時トークンは，属性-属性値組リストに変更される。このトークンは各々の属性-属性値組照合ノードに分配され，各ノードでは照合が並列に行われる。属性主体の属性値の照合では，市之助と太郎が共通上位概念自然人を持つため照合するので，bindingsとして{太郎/市之助}を加える。ところが，属性原因の属性値は，窒息と寒さなので照合しない。以上の照合結果のbindingsをmergeすると，{死亡2/死亡1,太郎/市之助}というbindngsが得られる。

単一LHS要素での照合処理については，以上の通りであるので，次に，LHS全体の処理の説明を行う。例としては，上と同じトークンが入力された場合を考える。

まず，首絞め1との照合では，共通上位概念が存在しなかったと考える。同様に，故意1との照合でも存在しなかったと考えるとする。したがって，新しいデータは下に流れない。

次に，死亡1との照合では，共通上位概念死亡が存在し，また，属性主体の属性値市之助に対しても太郎との間に共通上位概念自然人が存在する。しかし，窒息と寒さの間には存在しないので，以下の照合データがmergeプロセスに送られる。本来は，binding情報も同時に送られるのだが，ここでは省略する。

死亡(死亡2,[主体=太郎])

ここで，完全照合と一番違う点はmergeプロセスである。完全照合では，故意1のノードに照合したトークンもなく，memoryの記憶もない場合，必ずしもそれ以下のノードでの照合は行う必要はないが，並列部分照合では，以下の照合も並列に行わる。それ以上に，たとえあるノードで照合するトークンがなくても，他の多くのノードが照合すればよいという部分照合であるので，あるトークンに対してLHS要素全体の照合が全て完了するまで，発火条件が分からない。

また，mergeノードでは，たとえあるLHS要素に照合したトークンがあっても，それが照合していない可能性も考えたデータを作る必要がある。なぜなら，それは，最後にデータをマージする時に，整合性が取れないデータかも知れないし，そのデータさえ照合しないことにすれば他の十分多くのLHS要素は整合性が取れて照合可能かも知れないからである。これは，組み合わせ的爆発を生む原因にもなるが，各mergeノードでローカルに整合性判定をしていくことである程度防ぐことができる。つまり，同一の属性値にbindingされる値が同一であるかどうかの判定を各mergeノードであらかじめ行っておく訳である。

実際のmergeノードの処理は以下の通りである。ここで，mergeノードに対する入力として新しいトークンに対する照合結果のうち，図3の左側からの入力を左入力，右側からの入力を右入力とする。図では，左入力は常に上からmergeされた出力が流れてくることになる。また，mergeノードのmemoryのうち，左側のを左memory,右側のを右memoryと呼ぶことにする。

1. 左入力と右入力を整合性のチェックを行いmergeし，これをoutput1とする。

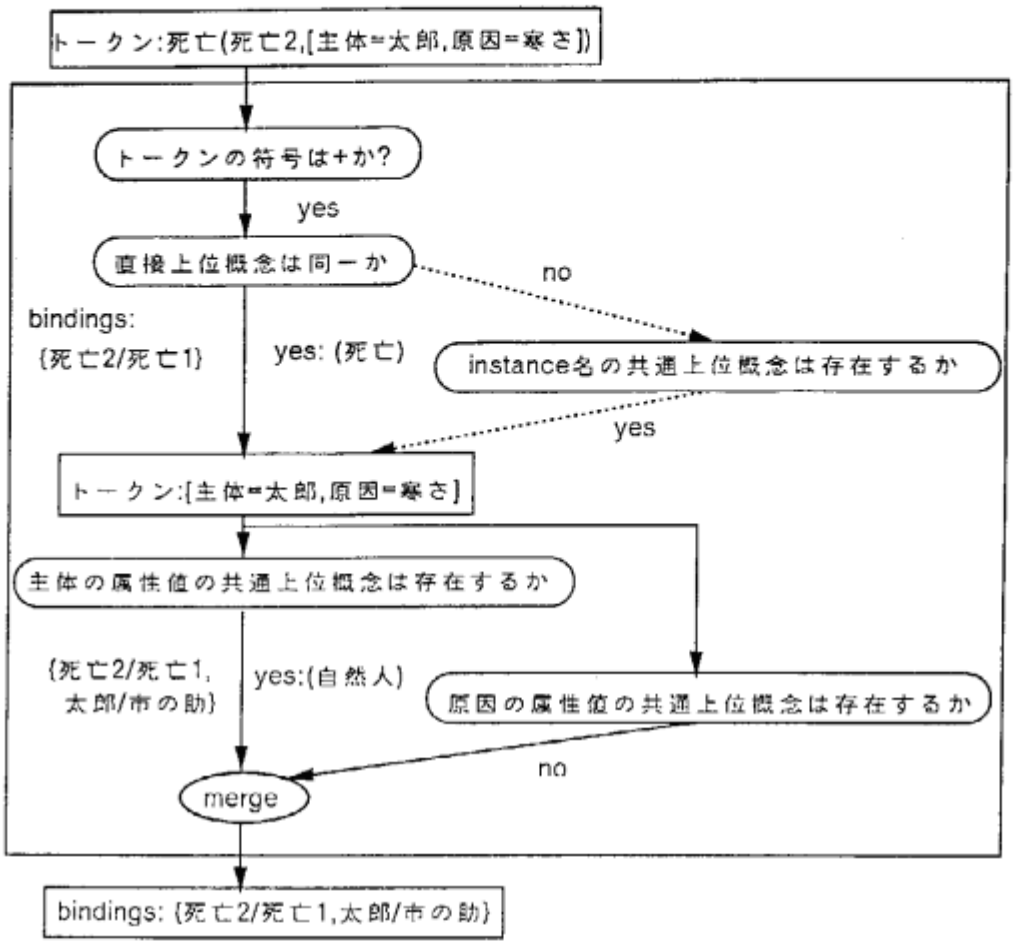

図 2: LIIS 要素類似照合ノードの処理例

2. 左入力と右 memory を整合性のチェックを行い merge し，これを output2 とする。

3. 左 memory と右入力を整合性のチェックを行い merge し，これを output3 とする。

4. 左入力の最後に，照合しなかった印の none を付けたデータを output4 とする。

5. 右入力の先頭に，上にある LIIS 要素類似照合ノードの数だけ none を付けたデータを output5 とする。このため，各ノードは，そのノードのプロセスネットワーク内の順番数を記憶している。

6. output1 から output5 までのデータを append し最終出力とする。

上から 2 番目の merge プロセスでの出力を上の処理に沿ってそれぞれ下に示す。

```
1. output1 = []

2. output2 = []

3. output3 =

    [傷害(傷害1,[主体=乙,客体=太郎])
     & 死亡(死亡2,[主体=太郎])]

4. output4 = []

5. output5 =

    [none
     & none
```

& 死亡(死亡2,[主体=太郎])]

実際にはこれらがappendされたものが出力されるが，図では，説明の簡単のためにoutput3のみを示している。同様な処理を次のmergeプロセスに対しても行う。最後に生成されたデータ全てに対し，類似度が閾値を越えるかどうかの判断を行い，越えたデータに対し事例ルールを発火しRHSデータを作る。上のように，部分照合では，たとえLHS要素に照合するトークンが存在しても，それが存在しないものと仮定したデータをも生成するので，閾値による発火条件の制御は必要である。

RHSデータの作成は，binding情報を用いて事例ルールのRHSのアトムをインスタンシエートするだけである。ただし，インスタンシエートされるべきbinding情報が存在しない場合，データは作成されない。

6 並列処理による効果

疎結合型Multi-PSIのプロセッサ数64台を使用して並列処理の効果を評価する。判例数は62である。判例一つに対し数個から十数の事例ルールが存在する。なお，メモリー資源などの関係から，プロセッサ数16以上でのデータとなっている。負荷分散は単純に各判例をプロセッサに割り当てているだけである。

PE数	16	32	64
時間(sec)	1012.6	650.2	355.7
台数効果	1.00	1.56	2.85

表1: 並列処理による速度向上

この結果は，計算上，プロセッサ1台に比べて64台用いると45倍程度の速度向上を望むことができ，単純な負荷分散でもかなりの効率向上があがっている。今後は，クラスター内密結合型のマシンで，より並列度をあげたインプリメントを行い，計測，評価を行いたい。

7 おわりに

事例ベース推論を行うために，過去の事例を事例ルールと呼ばれる形式で表し，並列類似照合を行う手法を提案した。事例ルールのLHSは，並列部分照合プロセスネットワークに展開され，LHS要素内でのインスタンス名および属性値の照合では概念階層辞書による類似照合を，LHS内では部分照合を用いることでこれを実現した。また，この部分照合により，各並列処理結果をマージする時の問題点を指摘し，この解決方法も示した。

さらに，並列処理では，Reteの同一ノードの共有は問題があることを指摘し，また，それを行わなくても十分な速度向上がはかれるという結果を得た。

参考文献

[Kolodner 87] "Extending Problem Solving Capabilities Through Case-Based Inference", Proceedings of thr 4th Annual International Machine Learning Workshop, 1987.

[Forgy 82] "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence 19 (1982) pp.17-37.

[前田 91] 法的推論における並列推論, 情処研報 Vol.91, No.62, 1991.

[Allen 83] "Maintaining Knowledge about Temporal Intervals", CACM, Vol. 26, No.11, Nov. 1983.

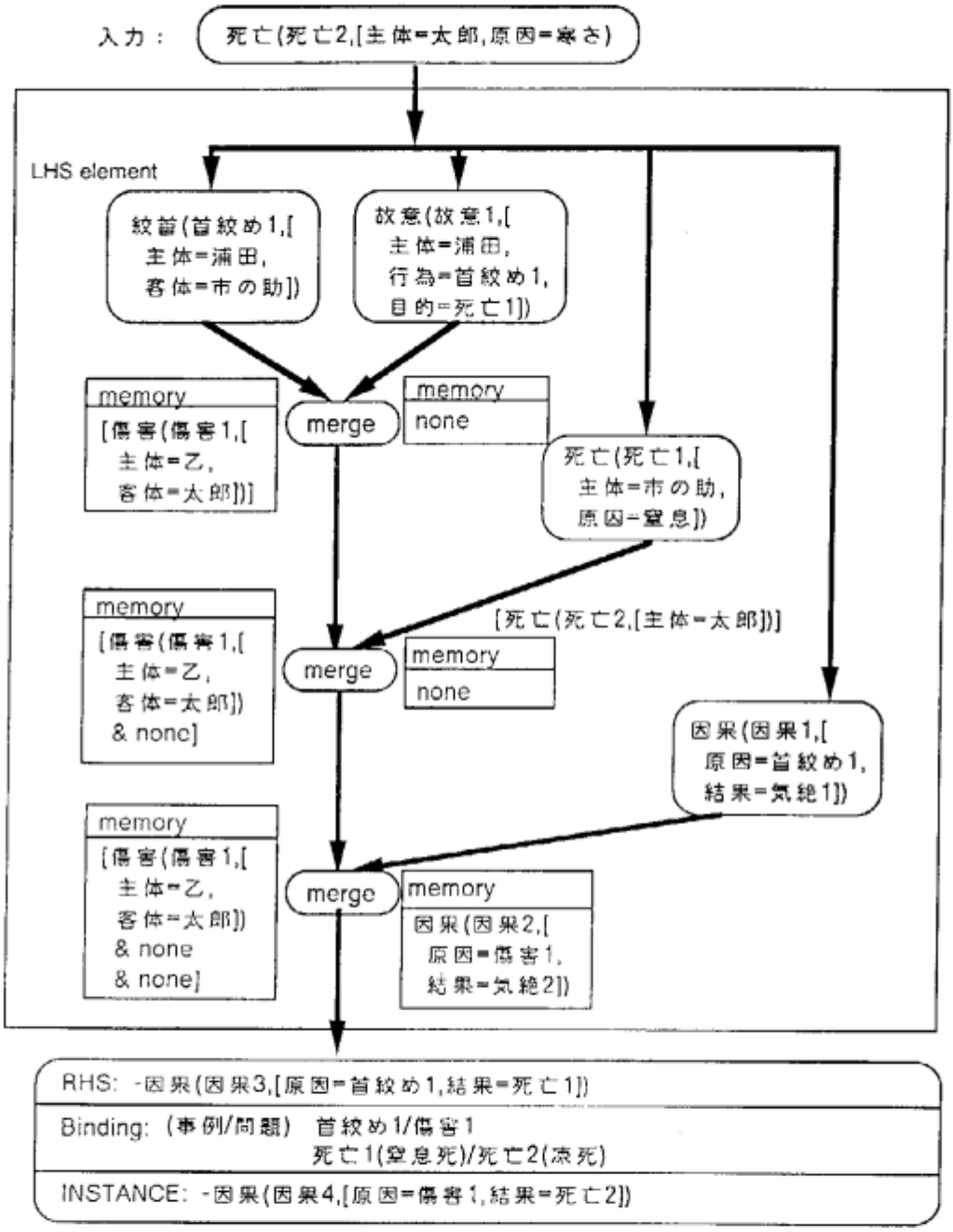

図 3: 並列部分照合プロセスネットワーク