法的推論システム HELIC-II (3) *
- 判例を用いた類似検索と判断生成 -

前田 茂　　小野 昌之　　新田 克己†

(財) 新世代コンピュータ技術開発機構‡

1 はじめに

われわれは，並列法的推論システム HELIC-II [前田 91] を開発してきた。法律における推論は，条文を形式的に記述して推論を行うだけでは，その中に現れる法的概念の解釈の問題があるので，うまく行かないことが多い。たとえば，刑法 210 条の内容は，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であるが，この中の法的概念過失などの定義は条文には存在しない。

そこで，判例を用いてこの問題を解くことが自然に考えられる。類似の事件での判例があれば，それを適用することで新しい問題を解決することができる。したがって，この点で，事例ベース推論 (Case-Based Reasoning: 以下 CBR) の必要性が認められることになる。HELIC-II の判例を用いた推論は CBR の枠組を応用したものである。したがって，事例となる知識を変更することで，汎用の問題解決器になることも期待できる。

ここでは，HELIC-II の類似論理構築部として，事例ルールと呼ばれる事例知識を用いて推論を行う方式を並列推論マシン Multi-PSI/PIM 上で実現したので，今回は特にその照合方式について詳しく述べる。一般に事例を用いて推論する場合，推論能力を考えると大量の事例が必要とされる。そこで，高速化が要求されるが，ここでは，事例ルールの左辺部を Rete [Forgy 82] 風の照合ネットワークに展開し，並列部分照合を行うことでこの問題を解決する。その際起こる問題点とその解決方法についても報告する。

2 判例および問題の知識表現

HELIC-II では，過去の判例を，その事件の状況の記述とそこでなされた法的な判断の組として表現する。事件の状況は，そこに現れる行為や動作 (event) と行動主体や物体，状態 (object) をノードとし，それらの間の関係をリンクとした意味ネットワークとして表される。実際の記述は，意味ネットワークをあるノードから発生するリンク一段階のノードまでを含む部分に分解し，フレーム形式にしたものである。

一方，判例でなされた法的判断は，その判断のもとになる状況を左辺部に，判断自身を右辺部に持つ If-Then 形式で表現される。これを事例ルールと呼ぶこととする。事例ルールと一般のプロダクションルールとの違いは，事例ルールでは左辺部の照合が，概念階層を用いた類似照合と左辺部の部分照合によってなされる点である。

例として扱う問題と判例の状況を以下に示す。

問題 捨て子であった幼児の太郎を乙が発見し，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。太郎はその夜衰死した。乙の罪責を述べよ。

判例 浦田が殺意を持って麻縄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。

判例および問題の状況の記述は，object, event の記述とそれらの間の時間関係とで表される。object, event の記述形式は以下の通りである。

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

ここで，Instance は，object/event 名を表すが，これだけではただの記号でしかないので，その直接上位概念 UpperConcept を定義し最小の抽象化を行うことで，他の Instance との照合の手掛りとする。もし，共通の直接上位概念がない場合は，HELIC-II では，あらかじめ用意されている概念階層を用いて照合を試みる。リスト内の Attribute, Value は，属性とその属性値を記述したものである。なお，次章で述べる部分照合プロセスネットワークを流れるトークンは，基本的に上の形式であると考えて良い。

次に，判例の判断の記述例を示す。今，上の判例の状況でなされた以下の判断を，事例ルールで記述した場合を示す。

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。

```
rule001("因果関係の中断",
  [条文 = 第 199 条, 主張者 = 弁護, 勝敗 = 敗訴,
   対立ルール =rule002, 学説 = 相当因果関係説],
  [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])]).
```

--> の左側が左辺部 (LHS) でありルール ID や説明生成のための付加情報と共に，右辺部 (RHS) の判断の条件となった状況を記述する。それらの条件には，その判断に対する重要度を，important, trivial によって付加する。照合時の類似度の計算は，照合したアトムの概念階層内における距離とこの重要度から計算される。RHS では，LHS の条件のもとになされた判断を記述する。ここでは，否定

---

*Case retrieval and making legal argument using precedents

†Shigeru MAEDA, Masayuki ONO, Katsumi NITTA

‡Institute for New Generation Computer Technology

の表現である"-"を用いて因果関係が認められないという判断を示している。

3 並列部分照合プロセスネットワーク

事例ルールのLHSは，図1に示されるような並列部分照合プロセスネットワークに展開され，高速に照合処理が行われる。

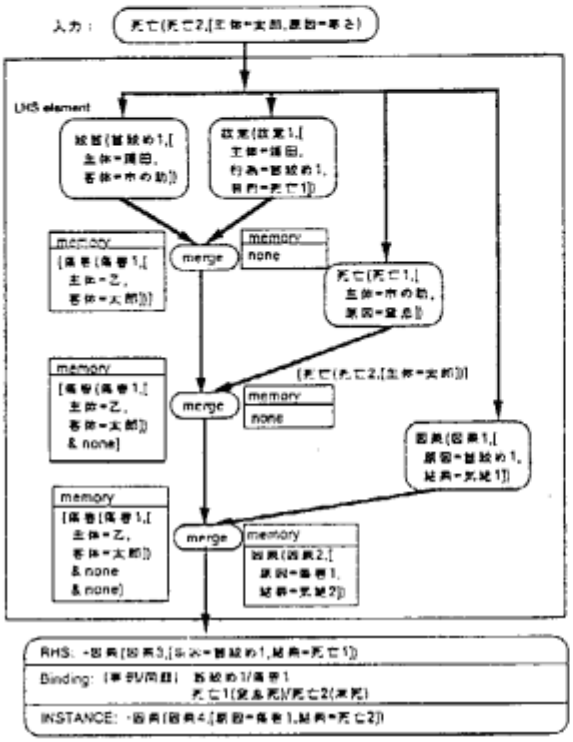

図1: 並列部分照合プロセスネットワーク

図で，丸い四角は，各LHS要素の照合プロセスである。ここでは，Instance，Value共に概念階層を用いて各々並列に照合が試みられる。(もちろん，Instanceは，UpperConceptも共に用いられる)。mergeと書かれたノードは，上位の二つの照合ノードの過去の照合結果をメモリーとして持ち，新しい照合情報とこれらを用いて整合性のある組み合わせを下流に流す。

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) 概念階層を用いた照合なので，共有すると bindingを作るのに必要な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。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。

今，トークンとして死亡(死亡2,[主体＝太郎, 原因＝寒さ])が流されたとしよう。これは，LHSの死亡(死亡1,[主体＝市之助, 原因＝窒息])のノードと照合され，死亡(死亡2,[主体＝太郎])が結果として出力される。他のノードとは照合されないとする。

Reteなどの完全照合と異なる点はmergeノードの処理にも現れる。部分照合では，必ずしも全てのノードが照合しなくても良いので，そういう組み合わせも考慮しなければならない。たとえば，上から2番目のmergeノードでは，トークン死亡2が流れてきた時は，[傷害1 & none & 死亡2]の組み合わせの他に，実際には照合された要素が照合されなかったと仮定した時の組み合わせ，[none & none & none]，[none & none & 死亡2]，[傷害1 & none & none]をも作る。なぜなら，これまでの照合結果とこれより下のノードで照合された結果が矛盾する可能性も考慮して組み合わせを作る必要があるからである。たとえば，死亡1と死亡2が照合しなかったことにした方が，最良照合が得られる場合も考えられる。

このような処理を行う必要上，並列部分照合では，逐次の完全照合と比べて組み合わせの数が多くなる。この問題に対しては，組み合わせの整合性のチェックを各mergeノードである程度ローカルに行うことで部分的に解決する。それにもかかわらず，最終的な照合結果も整合の取れた全ての組み合わせになるので，やはり大量のデータが生成される可能性がある。そこで，類似度がある指定された閾値を越えた物だけを発火させることにすることで，データの生成を抑えることとした。

4 並列処理の評価

並列処理による速度向上の評価を行うために，疎結合型並列推論マシン PIM/m の上で計測を行った(表1参照)。計測値はプロセッサ(Processor Element: 以下 PE) 16台から64台までで，それぞれ3回の平均値を用いた。判例数は，63で，各判例には数個から十数の事例ルールが含まれている。なお，各PEには，判例数がなるべく均等に成るように事例ルールが割り当てられるので，かならずしも事例ルール数は各PEで均等になるわけではない。

PE数	16	32	64
時間(sec)	692.5	273.9	134.7
台数効果	1.00	2.53	5.14

表1: 並列処理による速度向上

表より，スーパーリニアーな速度向上が見てとれる。原因を推測するために各PEの稼働状況を調べてみると，PE数32の時は，PE数64の時に比べて，より処理時間のかかるデータがある特定のPEに固まるというデータの非均一性が原因であることがわかった。PE数16の時は，ガベージコレクション(以下GC)が頻繁に起こっていることが分かった。したがって，PE数16に対してPE数32は約2.5倍の速度向上を示していて，それがPE数64の台数効果にも大きく影響していることが分かる。ちなみに，PE数8以下ではメモリ容量の関係で計測不可能だったことからも，PE数16の場合のGCにかかる処理時間が多いことが容易に推測される。

5 おわりに

以上，事例ルールという表現を用いたCBRの一手法について，その高速な実現方法について述べた。その際起こるプロセスネットワークでのマージの処理での問題点とその解決方法についても述べた。高速手法としては，Reteのノードの共有なしに十分な速度向上をはかれることを示した。

参考文献

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

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