

不確実な知識 における ATMS

董 方清 中川 裕志

横浜国立大学 工学部 電子情報工学科

ATMSは知識ベースの真偽性を維持するシステムであり、自動的に矛盾を検出したり、解消したりする機構を持っている。しかし、ATMSは確定的な知識を対象とし、エキスパートシステムにおいて広く利用された不確実な知識を取り扱えない。一方、不確実な知識自身も非単調性をもつ知識であり、従来の手法で取り扱えば限界がある。それらの問題点を改善するために、本論文ではまずATMSの不確実な知識への取り扱いにおける拡張版EATMSについて述べる。そして、ユーザからもらった情報が少ない場合、その情報を活用するためにpossible nogoodとpossible goodという二つの概念を導入し、EATMSによる推論方法を与える。

Extended ATMS for knowledge with uncertainty

Fangqing DONG Hiroshi NAKAGAWA

Department of Electrical and Computer Engineering,
Yokohama National University, Yokohama 240 Japan.

The ATMS is a truth maintenance system which finds contradictions contained in knowledge base automatically and retracts them efficiently. Although it provides a good way to manipulate definite knowledge, it is hard to deal with knowledge with uncertainty used in expert system.

In this paper, firstly a system called EATMS is presented, which is extended from ATMS and used to knowledge with uncertainty. Secondly two concepts, called possible nogood and possible good, are introduced to make hypothesis about current situation in which the information given for reasoning is not enough, and to retract inconsisitence automatically

1 はじめに

人工知能における研究の初期段階では、古典論理をAIの基礎理論とする。古典論理が公理の完全性を要求するので、定理証明にとっては適当である。しかし、日常生活では、人間は手持ちの情報が殆ど不完全である。にもかかわらず、不完全の知識を用いて正しく推論し、決断を下すことができる。ところが、古典論理で不完全の知識を記述し、推論すれば、推論システムは必ずしも思った通り働かない。そこで、不完全な知識に関する論理と推論システムが必要になる。前者に関して、default reasoning、nonmonotonic logicなどが相次ぎ提案されている。それらの論理は非単調論理とも呼ばれている。一方、後者に関してTMS、ATMSなどが研究されている。それらのシステムは、自動的に矛盾を検出したり、効率的に矛盾を解消する機構を持っている。特に、ATMSはTMSの問題点をうまく解決しているので、そのシステムの応用に関する研究が盛んになされている。

非単調論理の研究と独立に70年代の中間、エキスパートシステムが提案されてから、医学を始め、数多くの分野ではそのシステムが利用されている。エキスパートシステムでは推論に関する知識をルールの形で表現するが、特定の状況に対して適当なルールがない、或はユーザからもらった情報が少ない場合、従来の推論エンジンにより推論された結論は必ずしも正しくない。また、知識を獲得するとき、効率的に知識ベースの一貫性・完全性を維持するために従来、新たなルールを因果ネットワークとマッチし、帰結と前提条件の因果関係を検査する。しかし、一つ一つのルールは正しく見えても組み合わせられると矛盾が出るケースがしばしばある。

それらの課題を解決するために、エ

キスパートシステムにおけるルールベースを非単調性を持つ知識ベースと見てATMSによって知識ベースの真偽性を維持するアプローチを考えることができる。但し、ATMSは確定的な知識を対象とするので、エキスパートシステムの処理対象となる不確実な知識を処理するためにATMS自身を拡張しなければならない。本論文においては、まず、ATMSについて簡単に説明する。次に拡張されたATMS(EATMSと呼ぶ)について述べる。最後にエキスパートシステムにおける推論の問題点に対してEATMSによる改善方法を与える。

2 ATMS

ATMS(Assumption-based TMS)は知識ベースの真偽性を維持するシステムである。非単調論理の観点から考えれば、ATMSは常に幾つかの仮定を信じて、先に推論を進める。後で、矛盾が解いたら、矛盾の原因とみる仮定を削除し、別の仮定リストに移して推論をしていく。問題になるのは、新しい事実が推論された結論を否定すれば矛盾を解消するためにその結論に参照された仮定と前提条件を調べなければならない。

ATMSにとっては基本的な課題は三つある。

- ① 非単調性をもつ知識における表現
- ② 矛盾の定義
- ③ 矛盾の検出・解消

古典論理による知識表現と違い、非単調性をもつ知識では、仮定の概念が導入されている。ATMSでは

- ① premise: 何の前提条件にも依存せずに成り立つ事実
- ② choice: 確実な根拠がないまま下した仮定
- ③ assumption: choiceのシステムに

おける内部表現

④ node: 事実と仮定によって推論された結論

という四つの基礎概念が導入される。
nodeは一般的に

< datum, label, justification >

(2-1)

のように与える。datumはnodeに対応する知識の中身であり、justificationはnodeの推論された際に参照した前提条件である。labelは一般的にいくつかの環境と呼ばれる仮定リストからなる。環境をそのnodeの推論される迄に参照した総ての仮定と考えれば良い。特にpremiseとassumptionをそれぞれ

premise P: < P, {{}}, {()} >
assumption A: < A, {{A}}, {{A}} >

とする。また、ATMSでは環境に含まれば矛盾が導かれる仮定リストをnogoodというデータベースに記録し矛盾を判断する条件とする。

ATMSの基本操作は前提条件により環境を生成することである。例えば、直角三角形に関して、

- ① 直角三角形の性質により $X^2 + Y^2 = Z^2$ を満たす。
- ② X を 3 とする。
- ③ $\{X \neq V-4\}$ を満たす。

ということが解る。そこで

premise: < $X^2 + Y^2 = Z^2$, {{}}, {()} >
assumption: < X=3, {{X=3}}, {{X=3}} >
node: < Y=4, {{W=6}}, {{V=7}}, {...} >
 < Z=5, {{X=3, W=6, U=8}}, {...} >
nogood {X=V-4}

により①、②と③の知識を記述する。

また、

$$X^2 + Y^2 = Z^2, X=3, Y=4 \Rightarrow Z=5 \quad (2-2)$$

が解る。次に(2-2)に与える前提条件により四つのステップでZ=5の新たな環境が構成される。

I 前提条件の環境により帰結の新たな環境を生成する。

$$\begin{array}{ccc} X=3 \text{ の環境} & Y=4 \text{ の環境} & X^2+Y^2=Z^2 \text{ の環境} \\ \Downarrow & \Downarrow & \Downarrow \\ \{\{X=3\}\} \times \{\{W=6\}, \{V=7\}\} \times \{\{\}\} & & \\ = \{\{X=3, W=6\}, \{X=3, V=7\}\} & & \\ \text{ア} & \text{イ} & \end{array}$$

II 冗長な環境を検出する。この例の場合、アとZ=5の元の環境を比べれば、

$$\{X=3, W=6, U=8\} \supset \{X=3, W=6\}$$

ということが解る。 $\{X=3, W=6, U=8\}$ は冗長な環境と考え、捨ててしまう。

III 矛盾する環境を取り除く。イがちょうどnogoodの条件を満たすので、この環境を削除しなければ矛盾が引き起こされと考え、イを捨ててしまう。

IV 最後に新たに生成された環境をさらにZ=5の帰結に伝播していく。

結果として

$$\begin{array}{l} < Z=5, \\ \{\{X=3, W=6\}\}, \\ \{(X^2+Y^2=Z^2, X=3, Y=4), \dots\} > \end{array} \quad (2-3)$$

が生成された。

ATMSは環境で上例のようにnogoodのsupersetを無くすように知識の無矛盾性を維持する。その他、動的に矛盾を検出する場合、矛盾と宣言されたノードの環境を総てnogoodとし、自動的に

矛盾が解消できる。

3 EATMS

ATMSは確定的な知識を対象とするが、同様な手法で不確実な知識の真偽性を維持するためにはATMS自身を拡張しなければならない。

確定的な知識と比べ、不確実な知識は

- ① 確信度付き
- ② 異なるルールにより推論された帰結項はCOMB関係により帰結項の結合確信度を求める。

という二つの特徴を持っている。

不確実な知識の特徴を反映するために、幾つかのところでATMSを拡張すべきである。拡張されたシステムをEATMSと呼ぶ。ここでは、MYCINの不確実な知識における記述法を採用している。但し、EATMSの原理と不確実な知識における表現法と独立であるから、どの不確実な知識における記述法を利用してもかまわない。

EATMSにおいては次のようにノードを定義する。

$$\begin{aligned} <[\text{datum}, CF], \{\text{環境}\}, \text{justification}> \\ \text{justification} = \{\text{just}_1, \dots, \text{just}_n\} \\ \text{just}_i = (\text{rule}_i, [\text{antecedent}, CF_i]) \end{aligned} \quad (3-1)$$

但し、 CF_i は rule_i により計算された帰結の確信度である。特にassumptionは

$$<[A, CF_A], \{\{A\}\}, \{(A, [], CF_A)\}\rangle \quad (3-2)$$

のように記述される。

ATMSと同様にEATMSの基本操作は環境の生成であるが、ATMSに比べ幾つかの所で違う。

I 環境の合成。ATMSでは、前提条件

により生成された環境を一つのリストとして、labelに記録する。EATMSでは異なる前提条件から生成された環境は皆COMB関係をもつと考え、それらの環境の和集合をEATMSにおける環境という所に記録する。即ち

$$\begin{aligned} \text{ATMS } \text{label} &= \{\text{環境}_1, \dots, \text{環境}_n\} \\ &\quad \downarrow \qquad \qquad \downarrow \\ \text{EATMS } \text{環境} &= \text{環境}_1 \cup \dots \cup \text{環境}_n \end{aligned} \quad (3-3)$$

のようになる。

II 冗長な環境。ATMSでは、ノードの環境にはsubset関係をもつ環境があれば、supersetの方を冗長な環境と考えるが、EATMSでは、冗長な環境を検査しない。なぜなら、EATMSにおける環境は皆異なる前提条件によって生成され、お互いにCOMB関係をもつと考えるからである。COMB関係における環境の求め方はIに与えられている。

例えば、rule1とrule2によりFが推論されてしまう。そこで

$$\begin{aligned} <[a, 100], \{\{\}\}, \{(\{\})\}\rangle \\ <[A, 50], \{\{A\}\}, \{(A, [], 50)\}\rangle \\ <[b, 100], \{\{\}\}, \{(\{\})\}\rangle \\ <[B, 70], \{\{B\}\}, \{(B, [], 70)\}\rangle \\ <[F,], \{\}, \{\}\rangle \\ \text{rule1: } a \wedge A \rightarrow F, 80 \\ \text{rule2: } b \wedge B \rightarrow F, 70 \end{aligned} \quad (3-4)$$

rule1により Fの環境₁ = $\{\}\cup\{A\} = \{A\}$
rule2により Fの環境₂ = $\{\}\cup\{B\} = \{B\}$

(3-4)によりFの環境は

$$F \text{ の環境} = \text{環境}_1 \cup \text{環境}_2 = \{A, B\}$$

のようになる。生成されたノードFは

$\langle [F, 69.4],$
 $\{\{A, B\}\},$
 $\{(rule1, [[a, 100], [A, 50]], 40)\},$
 $\{(rule2, [[b, 100], [B, 70]], 49)\}\rangle$
 $69.4 = 40 + 49 - 40 * 49 / 100 \quad (3-5)$

のように与える。

EATMSでは矛盾に関する情報をATMSのnogoodと同様にして表す。但しEATMSにおけるnogoodは確信度によって異なる主張を持っている。例えば、“鳥”に関して、確信度が-20以下の場合、“鳥でない”と、確信度が20以上の場合、“鳥である”と、確信度が(-20, 20)にある場合、“鳥かunknown”と仮定する。

一方、人間はいろいろな状況を想定し、各状況において、推論を進めることができる。ATMSでは、このような状況をcontextと呼ぶ。厳密に表現すればcontextを

- ① 仮定集合Aと前提条件集合Jから導かれた総てのノードN
- ② Nはお互いに矛盾しない

のように定義する。便宜上のためにcontextを $\{A, J\}$ で表す。特に、Jを特別に定義しなければ、contextを仮定集合Aから導かれたノード全体とする。

ATMSにおいてはノードの定義でcontextを陽に表現していないが、EATMSでははっきりと記述しなければならない。というのは、ある帰結に対してcontextによって成り立つルールが違い、確信度を求める場合、各contextにおいて成り立つルールを区別しなければ、確信度が一つ求められてしまい、間違っただけになるからである。例えば(3-4)の例に対しては、

$C_1 = \{\{A\}\}$
 $C_2 = \{\{B\}\}$

という二つのcontextを設定し、各con

textに対応するFが

$\langle F, \{C_1(40), C_2(49)\},$
 $\{C_1\{A\}, C_2\{B\}\},$
 $\{C_1\{(rule1,$
 $[[a, 100], [A, 50]], 40)\},$
 $C_2\{(rule2,$
 $[[b, 100], [B, 70]], 70)\}\}\rangle$
 $(3-6)$

のようになる。(3-6)と(3-5)を比べれば、その区別がはっきりと解るだろう。

contextに関しては集合のsubsetにより半順序関係 $\leq$ が定義できる。二つのcontextに対して

$C_1 = \{A_1, J_1\}$
 $C_2 = \{A_2, J_2\}$
 かつ
 $A_1 \subseteq A_2, J_1 \subseteq J_2 \quad (3-7)$

を満たせば、 $C_1 \leq C_2$ とする。 C_1 を C_2 のsubcontextと言い、 C_2 を C_1 のsupercontextと言う。

contextを考慮する上、定義したEATMSにおいて、指定されたcontextに対してノードを生成したり、前提条件を挿入したりする。但し、ノードを調べる時、まず指定されたcontextで探す。見つけられない場合、更にそのcontextのsubcontextで解を求める。

より効率的にcontextを操作するために、EATMSでは総てのcontextを一つの列で記録する。各contextにcontext列における番号を与え、その番号に基づいて効率的にcontextを操作できる。しかし、contextを削除する場合、context列における順番を保つために、contextを削除しても対応するところを残さなければならない。このような“ゴミ”を処理しなければ、context列がどんどん長くなる一方、メモリが消費されてしまう。この点を解決するために、

EATMSでは、更に時刻の概念を導入している。各contextのcontext列における位置を

<時刻, context列における番号>

(3-8)

のように記述する。新たなcontextを挿入する場合、常に、“ゴミ”のところを先に利用する。但し、新たなcontextの対応する時刻をもとより1増やす。それにより、context列における番号が同じでも、取り出されたcontextが常に最も新しい時刻におけるcontextになる。

4 EATMSによる推論法

エキスパートシステムを用いて推論する場合、適当なルールがない、或はユーザからもらった情報が少ない場合、推論エンジンは必ずしも正しく働かない。例えば、次の知識ベースにより条件を満たす動物を求める。

足が二本ある	two_feet
体が白い	white_body
口が細長い	long_mouth
首が細長い	long_neck
魚を食べる	eat_fish

```
rule1: have_feather ^ lay_egg
      → bird, 95
rule2: fly → bird, 100
rule3: bird ^ (swim V have_weddes)
      → bird, 100
rule4: bird ^ eat_fish
      → water_fowl, 70
rule5: two_feet ^ white_body
      → bird, 40
rule6: (have_feather, unknown) ^
      (lay_egg, unknown) ^
      (fly, unknown) ^ two_feet
      → bird, -30
rule7: (bird, false)
      → water_fowl, -100
```

```
rule28: bird ^ white_body ^
        long_mouth ^ eat_fish
        → egent, 70
```

```
rule29: bird ^ white_body ^
        long_neck ^ eat_fish
        → egent, 70
```

```
rule5 ⇨ CFbird += 40
rule6 ⇨ CFbird -= 30
```

⇨ CF_{bird} = 10

このままでは鳥の確信度が小さいので、鳥か否かを判明できない。そこで、それ以上のルールは発火されなくて、何の結論も得られない。しかし、人間の場合を考えれば、この例の場合、鳥に関する確実な情報、例えば、飛ぶ、羽を持っている、卵を生むなど、を提供していないが、人間は類似な白鳥、鶴などの鳥を思い付くだろう。人間は情報が少ない場合、手持ちの情報を十分に活用し、適当な仮定を下し、先に推論を進めることができる。エキスパートシステムは人間と違って、只ルールに従ってのみ推論を進める。特定な状況に対して、適当なルールがなければ推論できないのである。この点を改善するために、エキスパートシステムにおける知識を不完全な知識と考え、ATMSの考え方に基づいて、一つの解決方法を与える。

推論の場合、例の鳥のような肯定の確信度と否定の確信度とも推論され、結合すればunknownになる時、鳥をpossible nogoodと呼ぶ。鳥に関して、“鳥か不明”、“確信度40で鳥を信じる”、“確信度30で鳥を信じない”という三つの可能性がある。possible nogoodの結論が導かれれば、鳥かunknownを先に選んで推論してみる。結論を得られない場合、肯定と否定の内、確信度の絶対値の大きい方を信じて、推論を進める。後で、良くないことが解れ

ば、他方を信じて推論する。このような信じられた方を possible goodと呼ぶ。

この例の場合、鳥に関しては、

鳥である確信度 $CF_{bird}=40$

鳥でない確信度 $CF_{not_bird}=30$

ということが解るので、先に確信度40で”鳥”を信じて推論を進め、鷺が推論されてしまう。

rule28 $\Rightarrow CF_{agent1}=28$

rule29 $\Rightarrow CF_{agent2}=28$

$CF_{agent}=28+28-28*28/100=48$

そこで、手持ちの情報を活用し、いい結果が導かれることができる。

一つの possible nogood に対して、三つの可能性がある。しかし、possible nogood の数が増えると、可能な組合せは膨大になる。それらの組合せを総て調べれば、推論の時間が長くなるし、その内、無駄な組合せも含まれる。ここで、次の例を通じて possible good の絞り込みを説明する。例えば、鳥と水鳥に関して

鳥		水鳥	
鳥か	unknown	水鳥か	unknown
鳥	50	水鳥	30
鳥でない	40	水鳥でない	45

ということが解る。鳥と水鳥の可能性を組合せれば九つの可能性が得られる。

①	鳥か	unknown	水鳥か	unknown
②	鳥	50	水鳥か	unknown
×③	鳥でない	40	水鳥か	unknown
？④	鳥か	unknown	水鳥	30
⑤	鳥	50	水鳥	30
×⑥	鳥でない	40	水鳥	30
？⑦	鳥か	unknown	水鳥でない	45
⑧	鳥	50	水鳥でない	45

*⑨ 鳥でない 40 水鳥でない 45

まず、探索の必要でない組合せを捜し出す。”？”が付いた組合せに関しては、現実には、鳥、鳥でないかの何れかが必ず成立するから鳥か unknown の情報量は決して両者より多くない。そこで、④の情報量は⑤或は⑥より少ない。すると、④を考慮しなくてもよい。同様に⑦も考えなくてもよい。

”×”が付いた組合せに対して、システムの持っている知識により、水鳥が鳥の一種類なので、鳥でないことが解れば、水鳥でないことも解るはずである。そこで、③、⑥と⑨のいずれかを調べる時、自然に”水鳥でない”が推論され、残る二つの組合せを調べる必要性がなくなる。つまり、③、⑥と⑨の内、只一つを調べればよい。そこで、⑨だけを調べればよい。これは、常識により推論の例であり、システム自身の知識により実現できる。

そして探索の必要になる組合せに対して次の原則に従って、調べ順序を決める。

I まず、エキスパートシステムの推論方式に従って、unknownの方を先に調べる。

II possible nogood の内、確信度の絶対値の大きい方を先に調べる。

そこで、まず、①の可能性があるので原則 I により①を先に調べる。

①の次、鳥の確信度が最も大きいから、鳥か unknown を捨てて、原則 II により、確信度50で鳥を信じて②を調べる。その後の調べ順序が原則 II に基づく。結果としてその順序は

① → ② → ⑧ → ⑤ → ⑨

のようになる。

possible goodの生成と変更は、その可能な組合せを全て展開し、その内最も良さそうなものを選択するわけではなくて、DDBのように自動的にpossible goodを選ぶことにより行われる。その手続きは次に示される。

I. 新たなpossible nogoodが宣言される場合、確信度がunknownの方を先に信じる。例えば、鳥unknownを信じる場合、水鳥をpossible nogood（水鳥の確信度 = 30、水鳥でない確信度 = 45）と宣言すれば、水鳥に対して、“水鳥unknown”を仮定する。

II. 推論の途中、現在信じているpossible goodを捨てて、新たなpossible goodを選びたい場合、

II.1 まず信じられているpossible goodの内、確信度がunknownのもの、例えばx,y、があれば、

$$CF = \max\{CF_x, CF_y, CF_u, CF_v\}$$

を満たすものを新たなpossible goodと仮定する。例えば、鳥と水鳥の確信度をunknownと仮定し、推論が進まない場合、

$$CF_{bird} = \max\{CF_{bird}, CF_{water_fowl}, CF_{bird}, CF_{water_fowl}\}$$

鳥の方が上式を満たすので、鳥unknownを捨てて、鳥50を新たなpossible goodと仮定する。

II.2 信じられているpossible good、例えばx,y,z,w、が皆unknownでない場合、

$$CF = \min\{CF_x, CF_y, CF_z, CF_w\}$$

を満たすもの、例えばx、に対して、仮定 x CF_x を捨てて、 $\neg x$ をpossible goodとして仮定する。特に、 $\neg x$ が既に仮定されたことがある場合、 x とい

うpossible nogoodを無視し、残ったy,z,wに対して同様な手続きを実行する。

以上で、鳥と水鳥の例を通じて原理を示した。一般的に、possible nogoodが三つ以上の場合、手続きは全く同じである。

5 まとめ

本論文では、まずATMSについて説明した。そしてATMSの不確実な知識への拡張版EATMSについて述べた。最後に不確実な知識における推論の場合、情報の不十分さにより推論が進まない問題点に対して、possible nogoodとpossible goodという二つの概念を導入し、改善方法を与えた。

今後の課題として、更に大規模な知識ベースを対象としてEATMSの有効性を実証したい。また、知識獲得のためにEATMSにより知識を組織し、知識ベース全体の一貫性・完全性を保つため、EATMSによるアプローチを研究したい。

【参考文献】

- [1] Johan de Kleer: "AN ASSUMPTION-BASED TMS", XPARC, P1-59
- [2] D. McDermott, and J. Doyle: "Non-Monotonic Logic I" AI 13(1980), P41-71
- [3] R. Reiter: "A Logic for Default Reasoning" AI 13(1980), P81-131
- [4] Elaine Rich: "ARTIFICIAL INTELLIGENCE", P173-198 McGRAW-HILL INTERNATIONAL BOOK COMPANY, 1983
- [5] 中川: "非単調論理", 寺子屋, 1986.12.16 資料
- [6] 中島: "知識表現とProlog/KR", 産業図書(聯60)
- [7] 日経AI, 1985.11.18 付録