2K−7

知識コンパイルの部分的適用による高速仮説推論法の検討

遠藤裕明　鶴田三郎　石塚満
東京大学

1．はじめに

われわれは、知識の幅を広げる次世代知識ベースへ向けての実用性も備えた枠組みとして、仮説推論に関する研究を進めている[1]。そのための第一段階として、不完全な知識（常に成り立つとは限らない知識）を記述し、それに基づいて推論を行うのが仮説推論である。

仮説推論は推論速度が遅いという欠点があり、その高速化のために様々な研究がなされている。中でもReiter他の提案したＣＭＳ（Clause Management System）[3]は、論理基盤上でＡＴＭＳ[2]を一般化、拡張したものとして位置づけられ、知識ベースのコンパイルにより、推論の高速化を図ると共に、無矛盾性の管理及び欠落した知識の発想を可能とする枠組みでもある。

ここではＣＭＳを仮説推論に応用するための第一段階として、ＣＭＳの演繹推論への応用[4][6]と部分コンパイルについて研究したので、報告する。

2．ＣＭＳ（Clause Management System）

2．1　ＣＭＳの概要

ＣＭＳは、一般の命題論理節を対象とし、公理節集合（Σ）と、問題解決器から送り込まれた質問節（Ｃ）とから、Minimal Support 節（Ｓ）（以下、ＭＳと書く）を生成することにより、質問節の論理的帰結性や、無矛盾での充足可能性を検討する枠組みである。

論理的帰結性を問う質問節の否定に対して、空節ではないＭＳ節が生成された場合（Σ＋～Ｃ⊢Ｓ、Ｓ≠φ）が、質問節が公理節集合から論理的に帰結されない場合であり，ＭＳ 節の否定は証明するために欠けている知識に相当する（Σ＋～Ｓ⊢Ｃ）。即ち、欠けている知識は連言から成り立つ論理式で表現される。

また、ＭＳ節が空節である場合（Ｓ＝φ）は、ＣＭＳ中の公理節集合によって質問節を論理的に帰結することが可能で、欠けている知識がない場合に相当する。

ここで、ＭＳ節は次のように定義される。

① Σ⊢Ｓ∨Ｃ
② Σ⊬Ｓ
③ Ｓの部分集合Ｓ’について、
Σ⊬Ｓ’，Σ⊬Ｓ’∨Ｃ

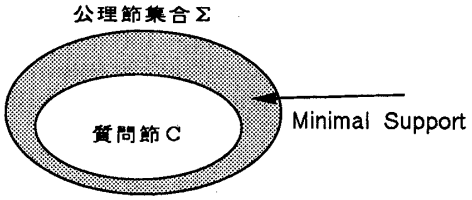

図1　ＣＭＳの概要

更に、効率を向上させるために、公理節集合をコンパイルすることを考える。これは、公理節集合からPrime Implicants（ＰＩ）（以下ＰＩと書く）を導出し、これを利用することにより、質問節に対してＭＳを生成する方法である。ＰＩは以下のように定義される。

① Σ⊢ＰＩ
② ＰＩの部分集合ＰＩ’について、Σ⊬ＰＩ’

なお、コンパイルは基本的に質問がなされる前に行っておけるので、多少時間がかかってもよい。このＰＩを求めてしまえば、もはや導出をする必要はなく、ＭＳ節をＰＩ節と質問節との包含関係から直接高速生成することが可能になる。

2．2　ＣＭＳの動作

次に、ＣＭＳの実際の動作を示す。

（1）コンパイル

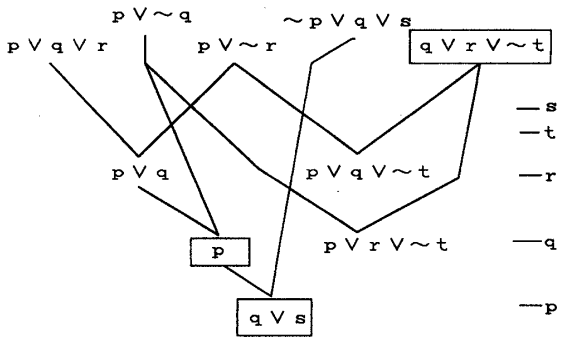

図2　ＣＭＳによる知識のコンパイル

図2は、ＣＭＳにより公理系Σ＝｛p∨q∨r、p∨～q、p∨～r、～p∨q∨s、q∨r∨～t｝をコンパイルして、ＰＩ＝｛p、q∨s、q∨r∨～t｝を導出する様子である。各シンボルについて、正負のものを組み合わせて消すという作業を、全ての文字の組み合わせに対して行なうというものである。その過程で、他のものを含んでいる知識は、冗長なので消される。

（2）証明

上のＰＩに対して、Ｃ＝｛q∨r｝という質問をすると、Ｓ＝ＰＩ－Ｃなので、Ｓ＝｛～q、～r、s、～t｝となる。Ｃを証明するのに必要な知識は～Ｓ＝q∨r∨～s∨tである。

3．部分コンパイル

ＣＭＳを実現するに当たって、コンパイルしてＰＩを導出しておくことの利点はさきに述べたが、次のような問題点もある。

①コンパイルに時間がかかりすぎる(指数オーダー)。
②コンパイルの結果、知識の数が多くなりすぎる。
③矛盾の可能性を持つ仮説を含む場合、空節が導出されてしまい、コンパイルできない。

これらの問題を解決するために、部分コンパイルすることを考える。部分コンパイルには、知識を幾つかの領域に分けてコンパイルする分割コンパイルと、全体をコンパイルするが最後までやらずに途中までで止める中間段階的コンパイルという２つの方法が考えら

A Study on Partial Knowledge Compilation for Efficient Hypothetical Reasoning
Hiroaki ENDO[1], Saburo TSURUTA[2], Mitsuru ISHIZUKA[3]
1,3:Univ. of Tokyo, 2:Tokyo Univ. of Mercantile Marine

れる。

3．1　分割コンパイル

①については、部分に分けてコンパイルするので、並列プロセッサを利用すれば高速化を図ることができる。
②については、各部分毎では比較的コンパクトになる。
③については、矛盾する領域とそうでない領域に分けて別々にコンパイルできるので、解決できる。これにより、初めて仮説推論への応用が可能になる。

(1)一般の場合

次に、矛盾しない場合についてＣＭＳで分割コンパイルした例を示す。

Σ={p∨～q,p∨q∨r,p∨～r,～p∨q∨s}...公理節集合
Σ1={p∨～q,p∨q∨r}、Σ2={p∨～r,～p∨q∨s}.分割
ＰＩ1={p∨～q,p∨r}、ＰＩ2={p∨～r,～p∨q∨s,q∨～r∨s}　.....コンパイル
Ｃ={q∨s}.....質問節
ＭＳ1={～q,～s}、ＭＳ2={～q,～s,～p,～r}.....ＭＳ
～ＭＳ1=q∨s、　～ＭＳ2=q∨s∨p∨r....足りない知識
∴～ＭＳ2 ⊇ＰＩ1. よって、Ｃは Σより証明された。

上の例はうまくいく場合であるが、～ＭＳ⊇ＰＩでないときは、さらに～ＭＳを質問節としてＰＩと導出しなければならないので処理が複雑かつ面倒である。このようなことを考えると、分割コンパイルは一概に良い方法とは言えない。つまり、知識が入り混じって存在しているときに、それをランダムに分割してしまうと、コンパイルしても不充分なＰＩがでるだけで、証明するにはＰＩ同士を何度も調べなければならなくなり効率が悪い。この場合はＰＩ同士で再コンパイル（導出）するか、初めから全部一括コンパイルした方が良いと言うことになる。

(2)特殊な場合・・・・・・・・知識が独立な場合

この場合は、知識を独立な部分に分けても相互に関連がないので、生じたＰＩの扱いも容易である。つまり、ある部分のことについて質問されたら、その部分だけで答えられるからである。そのために、他の領域へ解を探しにいく必要がないので効率的である。ただし、実際どのように知識の独立性、分割可能性を判断するかが問題であり、現在のところ有効な手段はない。ただ、初めにデータを入力する際に、どういう種類のデータかというラベルをつけて区別する方法があるが、できればデータ構造から自律的に判断する機構が望ましい。

(3)矛盾する場合

この場合、これは分割せざるを得ないので分割することについては問題はない。しかし、互いに矛盾するような知識の可能性のうち、どれを取るかを決定するために、知識間のランク付けが必要である。そのためには、例えば例外は一般常識よりレベルが高いと言うようなレベルを各知識毎につけるか、知識毎にそれ自身と矛盾するような知識のリストをもたせるなど、する必要がある。

3．2　中間段階的コンパイル

①については、途中までしかコンパイルしないので、コンパイルのための計算時間は短縮される。（しかし、後で導出をしなければならないので、その分の時間はかかる。）
②についても、途中までしかコンパイルしないので、比較的コンパクトになる。
③については、途中までとはいえ矛盾を含んだままコンパイルしてしまうので空節を導出してしまう可能性があるが、空節が出たらその組み合わせについては導出をしないとか、矛盾する知識は外に出して別の知識グループに分けるなどすれば一応解決できる。なお、矛盾を含んだ知識ベースのコンパイルについては、幾つか研究がなされているが（Freuderのk-consistency）[5]、具体的な成果は未だなく、今後の課題である。

ただし、段階的コンパイルとは言っても、どこでコンパイルを止めるかが問題で、Ｎ個のリテラルまでで止めるとか、ある決まった回数までしか導出しないというような方法がありうるが決定的なものはない。ここでは図１の例について３つのリテラルまでで導出を止める例を試みたが、その結果、調べなければならない組み合わせの数が減ったため、その後の導出に要する計算量が減少した。
（図１の例で、ｓ、ｔ、ｒまで導出。質問節Ｃ＝｛q｝。前処理時の導出回数６回、なにもしないときの導出回数１０回であった。）

4．結論

ＣＭＳは知識をコンパイルしてしまえば導出が不要になり、推論の高速化が図れるが、その反面、コンパイルに時間がかかる、知識の数が増え過ぎる、矛盾する場合に困るという問題があるため部分コンパイル（分割コンパイルと中間段階的コンパイル）を検討した。

・分割コンパイルについて

知識が入り混じって存在しているときに、それをランダムに分割することは無用な手間を増やすだけで効率の悪化を招くので避けた方がよい。一方、図２のように知識が独立な場合は分割に適しており、この場合は分割した部分だけで証明できるので効率的であるが、知識の独立性、分割可能性の判断が困難である。結局、効率的な分割方法が必要である。また、矛盾する知識については、分割することについては問題はない。しかし、矛盾する知識同士のうち、どれを選択するかの決定手続きを定めなければならない。

・中間段階的コンパイルについて

途中までしか計算しないので、コンパイルによる負荷（計算量、知識の数）は軽くなるが、そのために後の処理が当初ほど高速でなくなる。また、知識が矛盾する場合のコンパイル法、コンパイルをどこで中断するかというアルゴリズムが今後の課題である。

5．参考文献

[1] 石塚満：不完全な知識の操作による次世代知識ベースシステムへのアプローチ、人工知能学会誌, Vol.3, No.5, pp.22-32(1988)
[2]J.de Kleer:"An Assumption-based TMS",Artificial Intelligence 28, pp127-162(1986)
[3]R.Reiter,J.de Kleer:"Foundations of Assumption-based Truth Maintenance Systems:Preliminary Report",Proc.AAAI-87, pp183-188(1987)
[4]鶴田三郎、石塚満：不完全な知識が関わる知識システムへのＣＭＳの応用に関する基礎研究、人工知能学会全国大会（第３回）、pp.179-182(1989)
[5]J.de Kleer:"A Comparison of ATMS and CSP Techniques". Proc.of Eleventh International Joint Conference on Artificial Intelligence.pp.290-296(1989)
[6]鶴田三郎、石塚満：命題論理知識ベースのコンパイル法、情報処理学会、人工知能研資70-6(1990.5)