3H-01

命題レベル高速推論法を利用した 述語論理版仮説推論 HYPER システム*

山本 哲† Helmut Prendinger† 石塚 満†

‡東京大学工学系研究科情報工学専攻/電子情報工学専攻

1 はじめに

述語論理式による仮説推論は、命題論理版にくらべ記述能力は高いが、その推論速度は実用上有効な高速化が未だ達成されていない。一方、コストに基づく命題論理版についてはいくつかの高速推論方法が提案されている。そこで、本研究における述語版コストに基づく仮説推論システム HYPER では、一階述語論理式の知識ベースを命題論理に展開し、命題版高速推論システム利用して解を求める。この際、知識ベースを事前にリフォメーションをする事で効率的な命題論理式への展開をできるようにする。

2 述語版コストに基づく仮説推論システム HYPER の構成

システムの概要は次のようになる。関数なしの一階述語論理式の知識を入力とし、与えられたクエリーの形から、知識ベース中の用いられる事のない部分を削除する。さらに知識の変形を行い、各変数を展開して命題論理表現へと変換する。この結果を命題論理版高速仮説推論システムである SL 法 [3] によって解き、(準)最適解仮説を求める。

3 述語知識のリフォメーション

知識ベース中の不要変数 (節中のボディに現れてヘッドに現れない変数) を取り除くことによって、効率的な命題論理表現への変換ができるようする。この際、節の形によって次のような分類を行い、それぞれの場合について不要変数除去手続きを行なう点が、先行研究 [1] と HYPER との差異である。

3.1 ブロック

節のボディ中アトムが変数を共有する事によって結ばれる集合をブロックと呼ぶ。ブロック中に 2 つ以上の不要変数がある場合、新しい述語を生成することによって不要変数を除去する。

例えば

$$q(X,Y) \leftarrow p1(X,Z1,Z2) \wedge p2(Z1) \wedge p3(Z2,Y)$$

のボディは一つのブロックから構成される。この節に対しては次のような新しい述語が定義される。

$$new1(X,Z2) \leftarrow p1(X,Z1,Z2) \wedge p2(Z1)$$

そして元の節は

$$q(X,Y) \leftarrow new1(X,Z2) \wedge p3(Z2,Y)$$

となる。

3.2 chain

ブロック中の全てのとなりあうアトムが変数を共有し、ブロック中の最初と最後のアトムが少なくとも一つのヘッド中の変数を持つ時、このブロックを chain と呼ぶ。次の節

$$g(X,Y) \leftarrow p1(X,Z1) \wedge p2(Z1,Z2) \wedge p3(Z2,Y)$$

は chain であり、新しい述語が定義され、次のようになる。

$$\begin{aligned} g(X,Y) &\leftarrow new2(X,Z2) \wedge p3(Z2,Y) \\ new2(X,Z2) &\leftarrow p1(X,Z1) \wedge p2(Z1,Z2) \end{aligned}$$

3.3 孤立ブロック部

ブロック中において他のアトムと共有されない変数を孤立変数と呼び、孤立変数を含むアトムを孤立ブロック部と呼ぶ。

$$q(X,Y) \leftarrow p1(X,Z) \wedge p2(Z,Y) \wedge p3(Z,Z1)$$

において、$Z1$ は孤立変数であり、$p3(Z,Z1)$ は孤立ブロック部である。

この場合には新しい述語 $new3(Z) \leftarrow p3(Z,Z1)$ が生成される。

4 述語論理式から命題論理式への展開

述語論理式から命題論理式への展開に先行研究 [1] では QSQR 法を用いていたが、HYPER では一階述語論理用の探索木である Query-tree [4] を用いる。Query-tree ではクエリーの形から探索木のうち探索に必要な

---

* First-Order Hypothetical Reasoning System HYPER using Propositional Hypothetical Reasoning

† Tetsu Yamamoto, Helmut Prendinger, Mitsuru Ishizuka

‡ University of Tokyo,7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

e-mail: {tetsu,helmut,ishizuka}@miv.t.u-tokyo.ac.jp

部分だけが生成される。その過程は大きく分けてボトムアップフェーズとトップダウンフェーズの2つからなる。

ここでは次の例の場合について実行過程を示す。

(r_1) $p(X,Y) \leftarrow q1(X,Y) \wedge q2(X,Y).$
(r_2) $q1(X,Y) \leftarrow r1(X,Y) \wedge h(X,Y).$
(r_3) $q2(X,Y) \leftarrow r2(X,Y).$
(r_4) $q2(X,Y) \leftarrow r3(X,Y).$
(f_1) $r1(a,b).$ (f_2) $r1(a,c).$
(f_3) $r2(a,b).$ (f_4) $r2(c,d).$ (f_5) $r3(b,d).$

ここで h は仮説を表している。

まずボトムアップフェーズにおいて、各述語がとりうる定数の制約をファクトや仮説といったベース述語からスタートし、求めていく。ルールにおいてもとりうる定数を求めていき、各述語の制約を求める。

この例ではボトムアップフェーズによって次のような制約が得られる。

$r1^{\{\langle a,b\rangle,\langle a,c\rangle\}}$ $r2^{\{\langle a,b\rangle,\langle c,d\rangle\}}$ $r3^{\{\langle b,d\rangle\}}$
$h^{\{\langle a,a\rangle,\langle a,b\rangle,\ldots\}}$
$p^{\{\langle a,b\rangle\}}$ $q1^{\{\langle a,b\rangle,\langle a,c\rangle\}}$ $q2^{\{\langle a,b\rangle,\langle c,d\rangle,\langle b,d\rangle\}}$

次にトップダウンフェーズおいて、証明すべきゴールクエリーからトップダウンに木構造をたどっていき、満たされ得ない制約を削除していく。

このように Query-tree を構築していく過程で、各変数の具体化が同時に行われ、上記例では次のような結果を得る。

(r'_1) $p(a,b) \leftarrow q1(a,b) \wedge q2(a,b).$
(r'_2) $q1(a,b) \leftarrow r1(a,b) \wedge h(a,b).$
(r'_3) $q2(a,b) \leftarrow r(a,b).$
(f'_1) $r1(a,b).$ (f'_3) $r2(a,b).$

各アトムは変数を含まないことから、命題アトムへと書き換えられ、命題論理表現が得られる。

5 実験結果

Chain の例を用いて、知識ベースリフォメーションを行い、定数の数を変化させた時の推論時間を図1に示す。なお、この推論時間は述語知識のリフォメーション、述語知識から命題知識への変換、SL法によるコストに基づく準最適解の計算時間を含んでいる。

これはリフォメーションの効果が最も良く現れるケースであり、大きな効果が得られている。他の形態の述語知識の場合にも、リフォメーションの効果は大きいものがある。

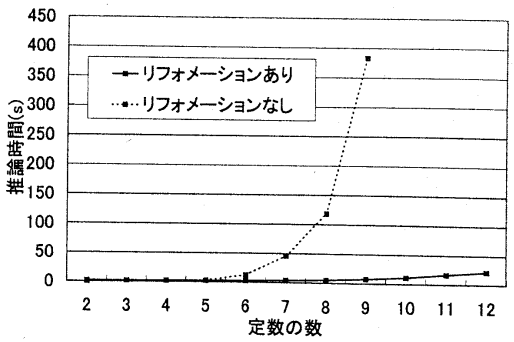

図 1: 知識リフォメーションによる推論時間の短縮

以上より、HYPER は述語論理版コストに基づく仮説推論に対しても、指数オーダの計算時間を回避して準最適解を計算する事ができる。

参考文献

[1] 棚橋, 福田, 石塚: 命題レベルの高速開放の利用を図るコストに基づく述語論理版仮説推論法, 人工知能学会誌, vol. 14, No. 6, pp. 1100–1107, 1999.

[2] A. Y. Levy, R. E. Fikes and Y. Sagiv: Speeding up inferences using relevance reasoning: a formalism and algorithms, *Aritificial Intelligence*, vol. 97, pp. 83–136, 1997.

[3] Matsuo, Y. and Ishizuka, M.: SL method for computing a near-optimal solution using linear and non-liniear programming in cost-based hypothetical reasoning, In *Proceedings 5th Pacific Rim Conference on Artificial Intelligence (PRICAI-98)*, pp. 611–625, 1998.

[4] Proietti, M. and Pettorossi, A.: Unfolding — definition — folding, in this order, for avoiding unnecessary variables in logic programs, *Theoretical Computer Science*, vol. 142, pp. 89–124, 1995.