

2D-2

仮説推論高速化のための知識ベースのリフォーメーション

堂前宣夫, 石塚満
東京大学

1. はじめに

高次AI機能の枠組みの一つである仮説推論は, 診断問題, 設計問題などに直接的に応用できるという実用性を備えており, 将来の知識ベースシステムとして有用な枠組みである。しかし, 仮説推論が本質的に計算量が大きく難しい問題であることから, これを実用化するためには様々な高速化技術を開発しなくてはならない[1]。本稿では, 知識ベースをあらかじめ変形しておくことによる推論実行時の実行時間短縮のアプローチについて報告する。

2. 仮説推論

本研究では, ホーン節述語論理知識ベースでの仮説推論を対象とする。完全知識の集合を Σ (無矛盾), 仮説の集合を H (矛盾の可能性を有する), ゴールを G とすると, 仮説推論は,

$h \subseteq H$ (h は H の部分集合)
 $\Sigma \cup h \vdash G$ ($\Sigma \cup h$ から G が証明される)
 $\Sigma \cup h \not\vdash \perp$ ($\Sigma \cup h$ は無矛盾)
 上記の3条件を満たす h' ($h' \subset h$) は存在しない
 (解仮説の極小性)

なる h を見いだすこととなる。本研究では, 論理否定を陽に表現できないホーン節を用いるため, 知識ベース(Σ)中に矛盾を陽に書き表すことによって矛盾を取り扱う。

$\text{inconsistent} \leftarrow a \ \& \ b$

a と b が同時に成り立つ場合は矛盾である。

3. 知識ベースのリフォーメーション

知識ベースのリフォーメーションは2つのフェーズからなる。すなわち, 第1のフェーズがPrime Implicatesへのコンパイルであり, 第2のフェーズが延べノード数の最小化である[2]。

知識ベースをあらかじめ部分的に Prime Implicates にコンパイルしておくことで仮説推論での推論の一部をメモリ探索に置き換え高速化を実現する手法については既にその有効性を確かめた[2]。命題論理知識ベースでは, このコンパイルで得られる知識は十分効率的であるが, 述語論理知識ベースでは, 同じユニフィケーションの失敗を何度も繰り返すことで冗長推論となり, 効率向上が妨げられることがある[2]。そこで, 本稿で提案するリフォーメーションの第2フェーズでは, 知識の前提部を多段化することで延べノード数を減らすアプローチにより, 同じユニフィケーションの失敗をできるだけ減らし効率向上の妨害を取り除き, さらに効率的な知識ベースを得る。例えば図1の例で, $F(X)$ がユニフィケーションに失敗して $T(X)$ が証明できない, という場合, 第1フェーズ後の知識では $C(X), D(X), E(X), F(X)$ に関して全く同じ操作をそれぞれ3回ずつ行う可能性があるが, 第2フェーズ後の知識では1回ずつしか行わない。

$T(X) \leftarrow S(X) \& C(X) \& D(X) \& E(X) \& F(X)$
 $S(X) \leftarrow A(X)$
 $S(X) \leftarrow B(X)$
 $\text{inc} \leftarrow B(X) \& C(X)$
 オリジナル知識

$T(X) \leftarrow St(X) \& C(X) \& D(X) \& E(X) \& F(X)$
 $T(X) \leftarrow A(X) \& C(X) \& D(X) \& E(X) \& F(X)$
 ターゲット $T(X)$ に関する
 第1フェーズ後の知識

$T(X) \leftarrow \text{New}(X) \& C(X) \& D(X) \& E(X) \& F(X)$
 $\text{New}(X) \leftarrow St(X)$
 $\text{New}(X) \leftarrow A(X)$
 第2フェーズ後の知識

図1 知識ベースリフォーメーション

4. 前提部の多段化

リフォーメーションの第2フェーズでは知識の前提部を多段化することで延べノード数の最小化を目指す。

命題論理知識ではターゲットノード T をヘッド部とする全ての節をまとめると、

$$T \leftarrow (A11 \& A12 \& \& A1i) \text{ or } (A21 \& A22 \& \& A2j) \text{ or} \\ \text{or } (Ak1 \& Ak2 \& \& Ak1)$$

と書き表せる。このボディ部は積和形論理式で表されており、代数表現すると、

$$\text{Body} = A11 \& A12 \& \& A1i + A21 \& A22 \& \& A2j + \\ + Ak1 \& Ak2 \& \& Ak1$$

となる。これに一種の因数分解（あるいは論理式の割り算）を行うことで、多段の論理式を生成する。多段化処理に関して、LSI-CADでの多段論理回路合成技術[3][4]が直接的に利用できる。

述語論理知識の場合も、知識を変えない変数名付け換え操作を行うことで命題論理知識の場合と同様の方法を用いることが可能となる。変数名付け換え操作が知識を変えないためには、その操作が、一つの節（オアだけで接続されたもの）の中の複数の変数名間に新たな関係をもたらす操作でなければよい。例えば、

$$G(UX, UY) \leftarrow A(UX, U1)A(U1, U2)A(U2, U3)A(U3, UY) \\ G(VX, VY) \leftarrow A(VX, V1)A(V1, V2)A(V2, VY) \\ G(WX, WY) \leftarrow B(WX, W1)A(W1, W2)A(W2, WY)$$

に関しては代入 $UX=VX=WX=X$, $UY=VY=UY=Y$, $U1=V1=Z1$, $U2=V2=Z2$ を行うことができる。

$$G(X, Y) \leftarrow A(X, Z1)A(Z1, Z2)A(Z2, U3)A(U3, Y) \\ G(X, Y) \leftarrow A(X, Z1)A(Z1, Z2)A(Z2, Y) \\ G(X, Y) \leftarrow B(X, W1)A(W1, W2)A(W2, Y)$$

ボディ部をまとめて多段化すると、

$$G(X, Y) \leftarrow A(X, Z1)A(Z1, Z2)[A(Z2, U3)A(U3, Y) \\ \text{or } A(Z2, Y)] \text{ or } B(X, W1)A(W1, W2)A(W2, Y)$$

となる。そして新述語 N を導入し、

$$G(X, Y) \leftarrow A(X, Z1)A(Z1, Z2)N(Z2, U3, Y) \\ G(X, Y) \leftarrow B(X, W1)A(W1, W2)A(W2, Y) \\ N(Z2, U3, Y) \leftarrow A(Z2, U3)A(U3, Y) \\ N(Z2, U3, Y) \leftarrow A(Z2, Y)$$

ところで、代入の選び方によって多段化の結果が変わってくる場合がある。例えば上記の例でも、代入 $UX=VX=WX=X$, $UY=VY=UY=Y$, $U2=V1=W1=Z1$, $U3=V2=W2=Z2$ を行うで多段化した場合は、

$$G(X, Y) \leftarrow N(X, U1, Z1)A(Z1, Z2)A(Z2, Y) \\ N(X, U1, Z1) \leftarrow A(X, U1)A(U1, Z1) \\ N(X, U1, Z1) \leftarrow A(X, Z1) \\ N(X, U1, Z1) \leftarrow B(X, Z1)$$

となる。

現在は、延べノード数をできるだけ小さくするため、ヒューリスティクスを用いるアプローチを採用している。

5. 評価実験とその結果

全加算器故障診断システムの例に適用し推論時間の評価を行った。多段化処理に関して、MIS[4]での weak deviation のアルゴリズムを用いた。

Inference Time
(CPU time [sec])

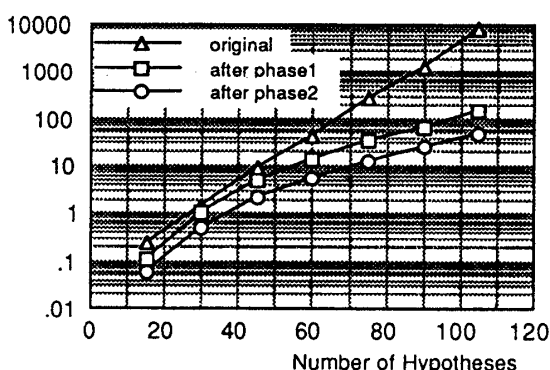


図2 実験結果

6. おわりに

仮説推論の高速化のための知識ベースリフォーメーションについて述べた。このリフォーメーションはそもそもは知識ベースの効率化が目的である。その目的のための副産物として、リフォーメーションの後半部分である前提部多段化処理において、新述語の発見が行われていると見ることができる。ここで発見される新述語にどのような意味があるのか考察することが期待される。

<参考文献>

- 1) 石塚：人工知能の夢への接近，情報処理，Vol. 32, No. 1, pp. 7-9(1991)
- 2) 堂前，石塚：最適解計算仮説推論高速化のための知識ベースの部分コンパイル法，情報学会人工知能研資，AI-85-4, pp. 25-32(1992)
- 3) 藤田：多段論理回路合成技術の動向，信学論，Vol. J74-A, pp. 152-161(1991)
- 4) R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli and A. R. Wang: MIS: A Multiple-Level Logic Optimization System, IEEE TCAD, Vol. CAD-6, 6, pp. 1062-1081(1987)