前向き自動帰結演算システムの効率化

5AH－3

西 和則　　程 京徳　　牛島 和夫
九州大学大学院 システム情報科学研究科 情報工学専攻

1. はじめに

現在の知識処理システムでは、設計者がシステムを構築する際にあらかじめ考えておいた問題にしか対応できず、既に分かっている知識を基にして新しい知識を獲得するようなことはできない。新しい知識の獲得を実現するためには、既に分かっている帰結関係に基づいて自動的に新しい帰結関係を生成することを実現しなければならない。

2. 汎用前向き自動帰結演算システム EnCal

EnCal(Entailment Calculus System) は帰結演算に基づいて推論規則の自動生成と自動検証、知識の自動獲得および自動定理発見を支援するために我々の研究室で開発されてきた汎用前向き帰結演算システムである[1]。前向き自動帰結演算では、その実行に伴う膨大な資源と時間の消費という大きな問題も生じる為、消費する資源の最小化、実用的な時間で停止させる為の高速化といった様々な効率化が必要とされる。

3. 省メモリ化

現在の EnCal には実行時に大量にメモリを消費するという実用上の問題点がある。それは前向き推論における莫大な数の論理定理図式 (以下、論理定理と略す) の導出に起因する。EnCal では、各論理定理は図 1 で定義される Wff クラスの複合体からなり、例えば、相関論理体系 $T_{\Rightarrow}$ の 30 万個の論理定理の集合を格納する為には、202M バイトという多大な資源が必要とされる。

```
class Wff {
  char symbol;
  Wff *left;
  Wff *right;
}
```

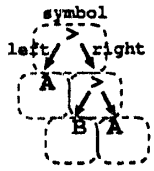

図 1: EnCal において、論理定理を構成するクラス

そこで効率的に論理定理を格納する為に、複数の論理定理式が共通の部分論理式を共有することで、冗長なメモリの削減を目指す。また、部分論理式をハッシュテーブルで管理することで、一元性を保つとともに、高速に引き出すことができる。図 2 にその再構成の様子を載せる。

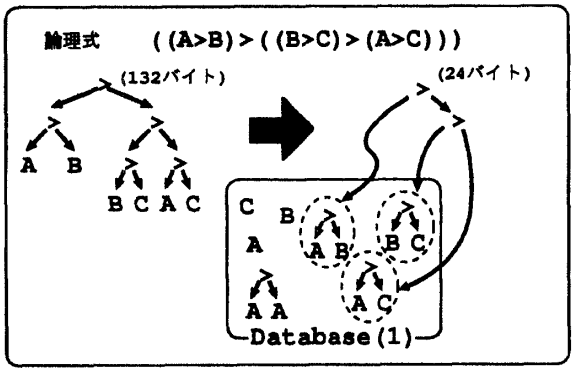

図 2: 論理式を再構成する様子

この改良により、上述の集合を、29M バイト (従来の 14%) で格納することができた。再構成に要する時間は、古典数理論理体系 CMLin の 11888 個の論理定理を例にすると、入力ファイルから論理定理を読み込むのに 2.0 秒かかり、その後再構成を行うのに 1.0 秒しかかからない。これは同論理体系の公理 (5 個) から演繹されるサブセット (11888 個) を導出に要する時間である 7 分から見ると微々たるもので、ファイル IO の誤差とみなせるぐらい小さな値である。実験は Sun の Ultra-SPARC(200MHz) で行った。また一般的に、前向き推論における論理定理図式の導出に要する時間は何百、何千時間になる場合も多いことからも、再構成に関するオーバーヘッドは殆どないと言える。表 1 にこの再構成によるメモリ使用変化を載せる。

表 1: 再構成によるメモリ消費の推移

論理体系	論理定理数	従来	再構成
CMLin	11888	6M	1.8M(30%)
Rc	32448	24M	5.5M(22%)
Te	300000	202M	29M(14%)

Improving the Performance of Automated Forward Deduction System for General-Purpose Entailment Calculus
Kazunor Nishi, Jingde Cheng and Kazuo Ushijima
Department of Computer Science and Comm. Eng.
Kyushu University

4. 高速化

また、現在の EnCal のもう1つの問題点である実行時間に関しても、その効率化を目指す技法について順次説明する。まず、論理定理をハッシュテーブルで管理する技法について述べる。ハッシュテーブルで用いるハッシュキーには、論理定理を完全2分木の木構造と見なし、各要素が存在するかどうかを「根からの通し番号」番目のビットとしたときの先頭16ビットで構成された正数を用いる。これにより、目的とする論理定理を高速に検索することができると共に、論理定理が重複しているかどうかの検査(以下、重複検査と略す)の必要条件としても利用可能である。同一のハッシュキーを持つ論理定理に関しては、図3に示すように線形リストとして保存する。

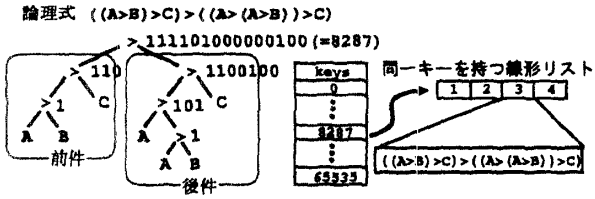

図 3: 木構造によるハッシュテーブルへの登録

次に、推論規則モーダス・ポネンス(以下MP)特有の効率化に関しては、「前件による分類」と「後件のインスタンス検査」という技法がある。前者は論理定理を前件によって分類しておくことで、MPにおけるパターンマッチングの必要条件である、ある論理定理の前件ともう一方の論理定理とが表現能力において包含関係にあるという条件の充足性を、同一の前件を持つ論理定理に関しては一括して検査することを可能とする。

また後者は、MPによって演繹される論理定理は常に後件部分のインスタンスである、という必要条件を早期に適用する、つまりパターンマッチングを行う前に、後件部分が他の論理定理のインスタンスになっているかを検査することで冗長なパターンマッチングを減少させる技法である。ここで、インスタンスとはその論理定理の表現能力がもう一方の論理定理の表現能力に完全に包含されている状態を意味する。

そして、前向き推論における「同じ論理定理が何度も生成される」という特徴から、論理定理の重複情報のキャッシュするという技法も、高速化に大きく役立つ。論理定理の重複検査は、単なる文字列による比較でなく、表現能力の包含関係に依存した比較が必要とされる為、前者に比べて非常に複雑かつ処理が重い。その為、通常の重複検査を行う前に重複履歴を参照することで、負荷が高い重複検査を単純な文字列比較に置き換えることができる。当然、履歴内にある全ての文字列との比較を避ける為に、前述のハッシュテーブルで管理しておき、最低限の比較で済ませることが効率化に繋がる。

これら全ての技法を取り入れた時の実行時間の結果は表2の通りで、従来と比べると、5〜80倍の処理速度の向上が見られる。相関論理体系 Ten に関しては演繹全体に要した時間が短いことから、ファイルIOやその他の処理の影響が強いと推定できる。また、相関論理体系 Te に関しては含意のみの論理体系であることからハッシュの効率が悪いと推定できる。それら以外は、相関論理体系 Ren の16倍、さらには CMLin の81倍というように、導出する論理定理の個数が大きくなるに従って、本改良が効果を発揮しているのがわかる。このことからも、現在実行中である、さらに大きな集合に関しては、より良好な結果を期待することができる。実験は同じく Sun の UltraSPARC(200MHz) で行った。

表 2: 高速化による実行時間の変化

論理体系	論理定理数	従来の時間	改良後の時間
Ten	263	0:00:04.27	0:00:00.77
Ren	1472	0:03:30.32	0:00:14.15
CMLin	11888	10:30:49.39	0:07:43.58
Te	11156	3:44:23.68	0:34:08.21

5. 最後に

本稿では、汎用前向き自動帰結演算システム EnCal における効率化の技法として、ハッシュテーブルの利用、重複履歴の保存、部分論理式を共有するシステムに関しても述べた。これらの技法は、EnCal やその他の前向き自動帰結演算システムに限らず、論理式を扱う多くのアプリケーションでの応用が期待される。

参考文献

[1] J. Cheng, "EnCal: An Automated Forward Deduction System for General-Purpose Entailment Calculus," in N. Terashima and E. Altman (Eds.), "Advanced IT Tools, IFIP World Conference on Advanced IT Tools, IFIP96 - 14th World Computer Congress," pp. 507-514, Chapman & Hall, 1996.