一般部分計算 (GPC) の実験システムの実装*

3K－5

小西 善二郎† 早稲田大学大学院理工学研究科
二村 良彦‡ 早稲田大学理工学部

1 始めに

一般部分計算法 (GPC) は、分かりやすいが能率の悪いプログラムを能率の良いプログラムに自動変換する手法である [2]。この変換過程において定理証明が重要な役割を果たす。

例えば、プログラム中の変数 u が 600 より大きいことが分かっているとき、GPC システムはプログラム中の (`if (> u 500) (f u) (g u)`) という部分を (`f u`) に変換する。このとき定理証明系は、論理式 $u > 600 \rightarrow u > 500$ を証明している。また、u が 400 より大きな整数であることが分かっていて、その中で u が 401 以下の場合を処理する場合、GPC システムはプログラム中の `u` を `401` に置き換える。この整数 401 を導き出すのも定理証明系の役割である。

GPC には、関数の畳み込みと展開とよばれる過程がある。ここでも定理証明系が働く。例えば、変換のある段階 N では、u は整数でプログラムは (`f u`) であるとする。後の段階で、u は偶数で、プログラム中に (`f (- u 1)`) という部分が現れたとする。このときこの f に N を畳み込むためには、

$$\{u-1 | u \in Even\} \subset \boldsymbol{Z} \qquad (1)$$

という条件を確かめなくてはいけない。また、畳み込んだあと、(`N (- u 1)`) の N を展開するためには、条件 (1) を真部分集合に変えて確認する必要がある。

従来の定理証明系は、このような処理には向いていなかったので、GPC の自動化は実現できなかった。

そこで、著者は GPC に特化した定理証明系を作成した。これは、直接証明、反例発見、環境簡約という 3 種類の機能から成り立ち、これらの機能を組み合わせることで、上記のような処理を実現している。

なお、GPC システムを実用的にするため、定理証明系の処理は、ある一定時間内に終わらなければ打ち切られるものとしている。このため、本来なら証明できる論理式が証明できないようなことが起こり得る。このような場合でも誤った処理がなされないようにした。

2 定理証明系

定理証明系の 3 つの機能それぞれについて説明する。

直接証明は、閉論理式

$$\forall x (A_1 \wedge \cdots \wedge A_n \rightarrow B)[u := x] \qquad (2)$$

が成り立つことを確かめるルーチンである。ここで $A_1, \ldots, A_n, B$ は節 (多くの場合単位節) である。これは、節集合 $\{A_1, \ldots, A_n, \neg B\}$ と公理系から矛盾を導くことによって実現している。

反例発見は、論理式 (2) が成り立たないことを確かめるルーチンである。これは、あらかじめ用意したサンプル・データを順次あてはめて反例を探している。

環境簡約は、$A_1 \wedge \cdots \wedge A_n \wedge B$ と $A_1 \wedge \cdots \wedge A_n \wedge \neg B$ を同値でなるべく単純なものにするルーチンである。これは、書き換え規則に基づいて行っている。

3 SDFU 計算

GPC のメインである SDFU 計算について、定理証明系がどのように使われるかを説明する。なお、以下で言う時間切れとは、形式的体系が弱くて証明できないことや、反例のデータが足りないことも含んでいる。

3.1 簡略化 (S)

簡略化では、環境 (プログラム中の変数が満たす条件) に基づいた分岐除去に定理証明系を用いる。これは、GPC 木の節 N の環境 $I(N)$ から分岐条件 p が導ければ else 式を取り除き、$\neg p$ が導ければ then 式を取り除くことである。

---

*An implementation of an experimental system for generalized partial computation (GPC)

†Zenjiro Konishi. Graduate School of Science and Engineering, Waseda University. 4-1, Okubo 3-chome, Shinjuku-ku, Tokyo 169-8555, Japan.

‡Yoshihiko Futamura. School of Science and Engineering, Waseda University. 4-1, Okubo 3-chome, Shinjuku-ku, Tokyo 169-8555, Japan.

したがって、まず反例発見によって論理式

$$\forall x(I(N) \to p)[u := x] \tag{3}$$

の反例を探してみる。見つからなかったら、次に直接証明によって論理式 (3) を証明してみる。証明できたら else 式を取り除く。反例が見つかったり証明ができなかったら、反例発見によって論理式

$$\forall x(I(N) \to \neg p)[u := x] \tag{4}$$

の反例を探してみる。見つからなかったら、最後に直接証明によって論理式 (4) を証明してみる。証明できたら then 式を取り除く。

時間切れによって分岐除去ができなくても、then 式と else 式の両方が残るので問題はない。

3.2 分配 (D)

簡略化によって、分岐のない式、または除去できなかった分岐の残る式が得られる。後者について分配が行われ、分岐条件 (およびその否定) が環境に追加される。この時、新しい環境をなるべく単純にする必要があるので、定理証明系を用いて環境簡約を行ってみる。

時間切れによって環境簡約ができなくても、情報不足で SDFU 計算が進まないだけなので問題はない。

3.3 畳み込み (F)

GPC 木の節 N の式が $E(N)$ のとき、節 H の式 $C(E(N)[u := k(u)])$ の $E(N)$ に N を畳み込むための条件は、$k(Ext(I(H))) \subset Ext(I(N))$ である。(ここで、$Ext(I) = \{x|I[u := x]\}$.) これは、

$$\forall x(I(H) \to I(N)[u := k(u)])[u := x] \tag{5}$$

ということだから、次のようにすればよい。

まず反例発見によって論理式 (5) の反例を探してみる。見つからなかったら、次に直接証明によって論理式 (5) を証明してみる。証明できたら畳み込む。

時間切れによって畳み込めない場合は、この節に関する SDFU 計算が止まるだけなので問題はない。

3.4 展開 (U)

GPC 木の節 N の式 $C(H(k(u)))$ の H を展開するための条件は、$k(Ext(I(N))) \subset Ext(I(H))$ である。ただし、これは H が定義関数名の場合であり、H が節名の場合は真部分集合でなくてはいけない。これは、

$$\forall x(I(N) \to I(H)[u := k(u)])[u := x] \tag{6}$$

ということであり、H が節名ならば、さらに

$$\exists x(\neg I(N) \land I(H)[u := k(u)])[u := x] \tag{7}$$

ということである。

したがって、まず反例発見によって論理式 (6) の反例を探してみる。見つからなかったら、H が節名の場合にのみ、次に反例発見によって論理式 (7) を示してみる。示せたら、最後に直接証明によって論理式 (6) を証明してみる。証明できたら展開する。

時間切れによって展開ができない場合は、この節に関する SDFU 計算が止まるだけなので問題はない。

4 終わりに

この定理証明系を用いることで、いくつかの例 (F71 関数 [2] など) について自動的に GPC (SDFU 計算) が行えた。

直接証明のルーチンは、定理証明系 TPU [1] を改造したものである。オリジナルの単位二元導出に加え、等号調整と単純な書き換えを行っている。ただし、組み合わせ論的爆発を避けるため、証明図には公理が高々 1 回だけ現れるように制限した。そのかわり、工夫された公理を数多く用意して、ある程度の証明を可能としている。

反例発見は、直接証明の前処理の役割を持ち、原理的には必要ない。唯一必要なのは、展開における真部分集合の判定のときである。より実用的な展開の条件として W-redex が提案されている [3] が、これについてはある程度集合の濃度計算をしなければならず、反例発見を拡張する必要がある。

また、環境簡約については、数式処理系 REDUCE を利用して、より確実に処理することも検討している。

参考文献

[1] Chang, C. and R. C. Lee: *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, 1973.

[2] Futamura, Y., K. Nogi and A. Takano: Essence of generalized partial computation, *Theoretical Computer Science*, vol. 90 (1991), pp. 61–79.

[3] 二村良彦, Song Litong, 小西善二郎: 一般部分計算 (GPC) における制御構造と停止条件, 日本ソフトウェア科学会第 15 回大会論文集, 1998, pp. 313–316.