プログラム変換による CPS コンパイラの最適化に関する研究

7B－6

楯 武士　西谷 泰昭
群馬大学 工学部 情報工学科

1 概要

ソフトウェアの開発においてコンパイラは不可欠な要素であり、多くの最適化手法が研究されてきた。MLコンパイラはその中間表現としてCPS (continuation passing style) を用いて、CPS 化されたプログラムに対して最適化を行っている [1]。

また、ソフトウェア生産性、信頼性の向上のためにソフトウェアの再利用技術が研究され、鈴木 [2] らによってプログラムの作成過程の再利用が提案されている。本発表ではコンパイラの最適化にプログラム変換の手法を取り入れることを提案する。

2 プログラム変換

プログラムの作成過程として、仕様からプログラムへ、ルールの適用を繰り返しながら変換を行なう過程を記録したものを変換履歴と言い、それを一般化したものを変換履歴スキーマと言う。変換履歴スキーマは、プログラミング手法を表現していると考えられる。鈴木 [2] は変換履歴を自動的に一般化する手法を提案し、二分法、対数時間での計算法を与えている。図1は対数時間で計算するための変換履歴スキーマの仕様を展開して得られたプログラムと変換後のプログラムを示している。

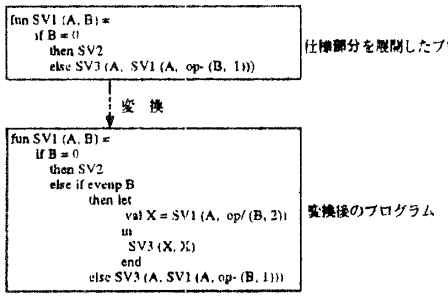

図 1: 変換履歴スキーマ (一部)

本発表では変換履歴スキーマの仕様部分を展開したプログラムを変換前パターン、変換後のプログラム部分を変換後パターンと呼び、2つのパターンの組をルールと呼ぶ。

3 CPS プログラムの変換

CPS プログラムの変換は、CPS プログラムをプログラムの制御に基づく基本ブロック [3] という単位に分割する。そして、CPS プログラムの制御構造と変換前パターンの制御構造によって、それぞれ対応している基本ブロックを定める。その後各基本ブロック間の照合を行い、照合に成功した場合に CPS プログラムからそのプログラムの固有部分を抽出し、ルールの変換後パターンに埋め込むことによって変換を行う。

照合に成功したとき、変換前パターンとプログラムで対応するパターン変数とプログラム中の変数、文あるいは値の対の集合を照合結果とする。埋め込みは照合結果をもとに変換後パターンへパターン変数とそれに対応するプログラム中の変数、文、値を置き換えることによって行われる。

照合の際には以下の問題点がある。

問題点

1. プログラムの意味が同じであるプログラムは数通り書くことができ、意味的には変換が可能であっても文法上の違いから照合に成功しないことがある。原因として次の2つが考えられる。
   (a) **プログラムに評価の余地がある。**
   例えば、ソースプログラム中に2つの連続する文 (x:=1; y:=3) とパターン中の連続する文 (y:=3;x:=1) があったときにこれらは本質的には同じものであるが、構文上は別のものとして扱われる。
   (b) **文の順序が異なる。**
   ある基本ブロック内の2文が互いに変数の参照、被参照の関係がないとき、順序を入れ換えてもプログラムの意味は同じである。
   例えば、ソースプログラム中に2つの連続する文 (x:=1; y:=x+1) とパターン中の連続す

Optimization on CPS compiler by program transformation processes
Takeshi Tate and Yasuaki Nishitani
Department of Computer Science, Gunma University

る文(x:=1;y:=2)があったときにも、本質的には同じものであるが別のものとして扱われる。

2. 変換前パターンに存在しない部分がCPSプログラム中の基本ブロックに存在すると照合に成功しない。

解決法

1. の問題はプログラムを基本ブロック毎に整形を行うことで解決する。プログラムの整形は次の2つの手順によって実現される。

- 1(a). の問題を解決するために照合を行う前にプログラムの評価できる部分は予め評価、つまり部分評価を行う。
- 1(b). の問題を解決するためにプログラムの標準化を行う。参照、被参照による文の依存関係を保存しつつ、予め設けられた構文毎の優先順位に基づいてソートを行う。

[例 1] プログラム整形

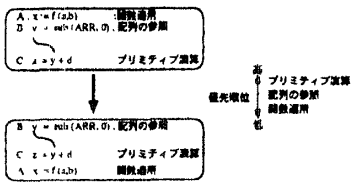

図 2: プログラムの整形

図2に示すブロックで文Cは構文の優先順位によるとソートによって文Bの前になければならないが、B,C間に依存関係があるので文の順序を保存して、図2下のようになる。 □

しかし、上の手順によって整形されたプログラムであっても依存関係がない同種の構文である文の間には順序が定まらずにプログラムが一意に定まらない。そのような場合は照合の機能を拡張し、独立で同じ構文の列について順序に関係なく照合を行う。

2. の問題が生じるとき、変換前パターンに存在しない部分は、そのプログラムのアルゴリズムとは直接関係がない非本質部分であると考えられる。したがって、この非本質部分を取り除いて照合を行い、成功したときにそれらを各基本ブロック毎に集め、照合結果に添付する。そして、埋め込みの際に変換後パターンで予め指定された位置にそれを埋め込むことを行う。

しかし、変換後のプログラムは基本ブロック間の関係が変化する可能性があるので変換後のプログラムを実行したときの非本質部分の実行結果は変換前のプログラムを実行したときのそれと同じであるという保証はされない。

照合と埋め込みの例を示す。

[例 2] 照合と埋め込みの例

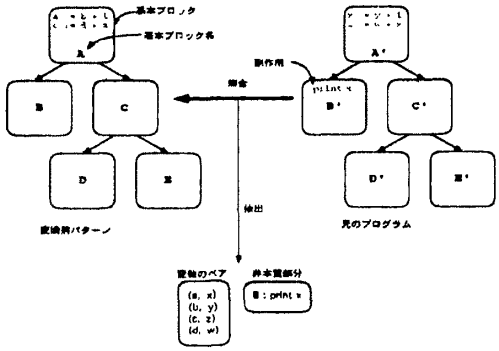

図 3: 照合

図3でCPSプログラムの整形された各ブロックA'～E'について変換前パターンのブロックA～Eがそれぞれ照合に成功し、変数のペアとBブロックの非本質部分print xが抽出される。そしてその結果の埋め込みを示したのが図4である。ここで変換後パターンのB,Dの非本質部分を埋め込むことを予め指定されたブロックに照合で抽出されたBの非本質部分を埋め込む。

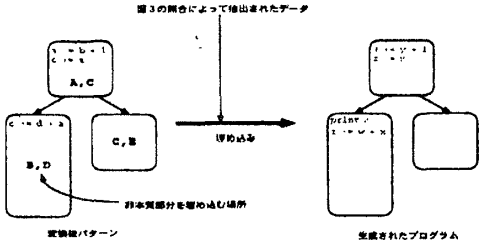

図 4: 埋め込み

4 結論 □

プログラム変換をコンパイラの最適化に取り入れるためにプログラムとパターンの照合、照合結果の埋め込みの手法について述べた。またその際に問題となるプログラムの整形、非本質部分の扱いについて述べた。

参考文献

[1] Andrew W. Appel: "Compiling with Continuations", Cambridge University Press, 1992.

[2] 鈴木秀明: "プログラム変換履歴の一般化", 情処ソフトウエア工学研究会, SE-90, pp. 81–88, 1993.

[3] A.V.Aho, R.Sethi, J.D.Ullman : "Compilers", Addison-Wesley Publishing co., 1986.