2D-4

ＣＣＦＧ（Coupled Context Free Grammar）におけるプログラム変換

中田育男　山下義行
筑波大学　電子・情報工学系

１．はじめに

ＣＣＦＧ（Coupled Context Free Grammar：組文法）［１］はいくつかのＣＦＧ（文脈自由文法）を組み合わせたものであり、各文法はそれぞれあるデータ構造を定義し、それらの組み合わせ方によってそれらのデータ構造の間の関係を定義する。従って、その一つが入力データの構造を定義し、その他が出力データの構造を定義すると考えれば、そのＣＣＦＧは入力データを出力データに変換するプログラムを表現している。ＣＣＦＧプログラムは一般には非決定性の動きをするものが考えられるが、ここでは、決定性の動きの出来るものだけを取り上げ、ＣＣＦＧプログラムを通常の手続き型プログラムに変換する方法、ＣＣＦＧプログラムの合成法、合成されたプログラムをより効率の良いプログラムに変換する方法、等を述べる。

２．ＣＣＦＧとは

ＣＣＦＧの定義は［１］にあるが、ここでは、例を使って簡単に説明する。

ＣＣＦＧは４項組Ｇ＝（Ｎ，Ｔ，Ｐ，Ｓ）とする。Ｎは非終端記号（Nonterminal）の集合、Ｔは終端記号（Terminal）の集合、Ｐは生成規則の集合の集合、Ｓは開始記号の集合である。

例：Ｇ１＝（Ｎ，Ｔ，Ｐ，Ｓ）

Ｎ＝{Ｉ，Ｏ}，Ｔ＝{ａ，ｂ}，Ｓ＝{Ｉ，Ｏ}，Ｐ＝{｛Ｉ→ａＩ，　Ｏ→Ｏａ｝，　①
｛Ｉ→ｂＩ，　Ｏ→Ｏｂ｝，　②
｛Ｉ→ε　，　Ｏ→ε　｝｝　③

文形式集合は例えば次のように書き換えられる。

｛Ｉ，Ｏ｝⇒｛ａＩ，　　Ｏａ｝　①による書き換え
⇒｛ａｂＩ，　Ｏｂａ｝　②による書き換え
⇒｛ａｂｂＩ，Ｏｂｂａ｝　②による書き換え
⇒｛ａｂｂ，　ｂｂａ｝　③による書き換え

ａｂｂが入力、ｂｂａが出力と考えれば、Ｇ１はａ，ｂの文字列を反転するプログラムである。

３．ＣＦＧとＣＣＦＧの拡張

ＣＣＦＧを実用的なプログラム言語とするためには整数や実数や構造データも扱えることが必要である。ＣＦＧにおいて、Ａ→ａＢなる生成規則の右辺ａＢは″ａ″と″Ｂ″からなる文字列であるが、それは、ａと，Ｂの形をしたある文字列（例えばｂｂｃ）との連接をとったもの（例えばａｂｂｃ）がＡの形であることを定義している。すなわち、ａＢはａとＢ（の値）

Program Transformation in CCFG(Coupled Context Free Grammar)
Ikuo Nakata and Yoshiyuki Yamashita
University of Tsukuba

の集合としての積を表す式と見ることも出来る。そこで、一般に右辺は演算子を含む式であるとする。生成規則A→αは、式αの値（それを［α］と書くことにする）がAの値に含まれる、すなわち［A］⊃［α］、を意味する。

4．ＣＣＦＧプログラムから手続き型プログラムへの変換

出発記号集合の中の一つの出発記号が入力データの構造を定義し、その他の出発記号が出力を定義しているとすれば、入力データの構文解析をしながら出力データを求めるプログラムに変換することが出来る。この場合、出力データの文法は属性文法における合成属性の評価規則に当たる。ただし、入力データが列であれば通常の構文解析と同じであるが、入力データが例えば数値であるときは入力データに関する計算を進めるのが構文解析に当たる。

例えば次のような整数の列の和を求めるプログラムのＩを入力の出発記号として再帰的下向きパーサを作ると再帰的なプログラムが得られる。

{Ｉ→ε，　Ｓ→0　}

もとのＩ→ＮＩをＩ→ＩＮに変えてＬＲパーサを作ると繰り返し型のプログラムが得られる。

{Ｉ→ＮＩ，Ｓ→Ｓ＋Ｎ}

{Ｎ→integer}

5．ＣＣＦＧプログラム間の変換

fold/unfoldの例：　ｃの文字列をａｂの列に変換し、それを反転するプログラム

{Ａ→Ｘ，　Ｙ＝Ｗ，　Ｂ→Ｚ}　(1)

{Ｘ→ε，　Ｙ→ε　}　(2)

{Ｘ→ｃＸ，Ｙ→ａｂＹ}　(3)

{Ｗ→ε，　Ｚ→ε　}　(4)

{Ｗ→ａＷ，Ｚ→Ｚａ}　(5)

{Ｗ→ｂＷ，Ｚ→Ｚｂ}　(6)

ここでＹ＝Ｗは、ＹとＷが同一の値を取ることを意味する。これは以下のように変換できる。

{Ａ→ε，Ｂ→ε}　(7)　(1)を(2)、(4)でunfold

{Ａ→ｃＸ，ａｂＹ＝Ｗ，Ｂ→Ｚ}　(8)　(1)を(3)でunfold

{Ａ→ｃＸ，ａｂＹ＝ａｂＷ，Ｂ→Ｚｂａ}　(9)　(8)を(5)、(6)でunfold

{Ａ→ｃＸ，Ｙ＝Ｗ，Ｂ→Ｚｂａ}　(9')

{Ａ→ｃＡ，Ｂ→Ｂｂａ}　(10)　(9')を(1)でfold

(7)、(10)が求めるプログラムである。

6．おわりに

データ構造を陽に突き合わせることによって、プログラムを直感的に分かりやすく表現することが出来る。さらに、プログラム変換も、中間的なデータ構造が陽に表現されているので、関数型言語などにおけるそれより見通し良く簡単に行なうことが出来る。

参考文献　［１］山下、中田：文脈自由文法の拡張とそれに基ずく計算モデル
情報処理学会ソフトウェア基礎論研究会報告１５－５

［２］中田、山下：ＣＣＦＧ(Coupled Context Free Grammar)におけるプログラム変換
電子通信学会技術研究報告，Vol.86，No.85，17-24