Ｐｒｏｌｏｇによるコンパイラテストデータの自動生成方式

1Ｒ－6

橋本　辰範　　堀田　博文

ＮＴＴソフトウェア研究所

1　はじめに

テストデータ(TD)の作成作業は，ソフトウェアのテスト作業の中で工数上も品質確保上も重要な役割を担っており，効率的にTD集合を作成することは非常に重要である．ここで「効率的」とは，次の意味で用いる．

・効率的作成　:TD集合を少ない工数で作成すること
・効率的テスト:テスト項目に抜けがなく冗長性が少ないこと

本稿では，これらの条件を満たす方式として，機能テスト用のTD集合を自動的に生成する方式を提案する．基本的な方式自体は種々のソフトのテストに適用可能であるが，ここではコンパイラの言語仕様関連のTD生成方式及びシステム構成を紹介する．

2　方式の特徴

現在，機能テスト用のTD集合を求める方法として，人間が仕様から直接テスト項目を作り出す方法の他に，原因結果グラフ法[1]やAGENT[2]が提案されている．しかし，これらはいずれもテスト項目の抽出にとどまりTD生成の自動化には至っていない．

これに対し，本稿で提案するTD自動生成システムは次の特徴を持つ．

①TD自動生成機能
　仕様の内未テストの部分をテストするために必要なTDの集合を自動的に生成

②未テスト項目抽出機能
　仕様の内TD集合によりまだテストされていない項目を抽出する（①のベース情報とする）

3　基本的な方式

３．１　Prologの活用

次の特徴を持つPrologをシステムの実現に用いた．

①パターンマッチング機能により，入出力の可逆性が実現できる．たとえば，Cプログラムの構文チェッカを作れば，同時に正しい構文を持つCプログラムの生成器が実現されたことになる．

②バックトラック機能により，ある条件を満たすすべての解が次々に生成できる．たとえば，上記のCプログラム生成器がPrologで記述されていれば，正しいプログラムが次々と生成できる．

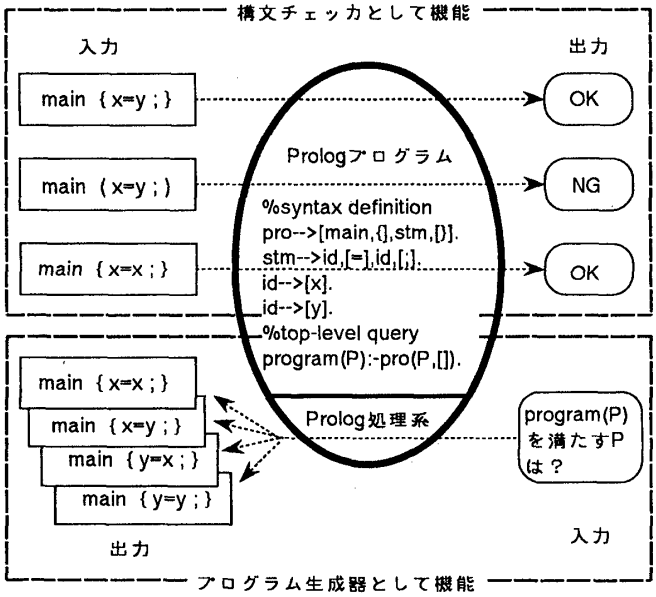

図１　Prologの可逆性とバックトラックの応用

本特徴を簡単な仕様を持つ言語を例にとり図１に示す．

３．２　TD自動生成システムの方式

本節中の記号，用語は次の意味を持つ．

+,-:それぞれ入力パラメタ,出力パラメタ
SP :ソースプログラム
TL :字句列
ST :構文木
CDB:特徴データベース
UC :仕様の内の未テスト部分の特徴

(1)TD自動生成の構成要素

(G1)字句列生成器:lex_gen(-TL,-ST)
　構文上正しい字句列を次々に生成する．このとき，その字句列に対応する構文木の生成も行う．次々に生成するという処理は，Prologの活用により実現されている(３．１②参照)．

(G2)静的意味則チェッカ:semantics(+ST)
　構文木を解析し，静的意味則が満足されていることをチェックする．

(G3)特徴抽出器:characteristics(+ST,-CDB)
　構文木を解析し，対応するプログラムが持つ特徴を抽出する．抽出された特徴がCDBに未登録ならば登録する．

Functional Test Data Generation for Compilers with Prolog
Tatsunori Hashimoto, Hirofumi Hotta
NTT Software Laboratories

(2)未テスト項目抽出の構成要素

(C1)字句解析器:lex(+SP,-TL)
Cプログラムを字句列に分解する.

(C2)構文解析器:syntax(+TL,-ST)
字句列を解析し構文木を出力する.

(C3)特徴抽出器:characteristics(+ST,-CDB)
構文木を解析し，元のプログラムが持つ特徴を抽出する．抽出された特徴がCDBに未登録ならば登録する.

(C4)未テスト項目抽出器:untested(+CDB,-UC)
(C3)で蓄積された特徴の集合と，人手や自動生成で得たテストすべき特徴の全集合との差分を求め，今後テストすべき仕様上の特徴を出力する.

(3)TD自動生成と未テスト項目抽出の融合

(G1)～(G4)，(C1)～(C4)には次の共通点がある.

①extract_patternは双方にある

②syntaxとlex_genは引数の入出力関係が違うだけ

③lexとprog_genは引数の入出力関係が違うだけ

①の処理はTD自動生成と未テスト項目抽出の間で共用可能である．さらに，②③のように引数の入出力関係だけが違う要素は，Prologの「入出力の可逆性」の活用により一つに統一可能である.

以上のように，Prologの使用により，TD自動生成処理と未テスト項目抽出処理を融合し，少ない要素でシステムが構成できることが分かる(図2).

4．主な機能の実現方法

(1)自動生成の方法

まず，言語仕様を形式的に定義することが必須である．本システムでは，仕様を記述する道具としてProlog上のプリプロセサによりPrologに変換されるDCG[3]を採用し，次々と生成するためにはPrologのバックトラック機能そのものを用いる.

なお，バックトラックによる生成では，無限に字句列が生成される．そこで，無限の生成を行わせないために，つまりプログラムの大きさを制限するために各構文則の適用回数を限定する.

(2)冗長なTDを生成しない方法

lex_genにおいては，同一の特徴をもつTDは２個以上生成しないようにしている.

さらに，それだけでは冗長度抑制は不十分であるため，字句要素，式，文といった構文上の規則毎に詳細は無視して別々に構文木を生成し，あとで構文木上で詳細部分を置き換える方式で解決する.

また，宣言部と実行部など，木を垂直に分割しそれぞれ独立して生成しつなぎ合わせる方式も用いる.

5 おわりに

効率的に機能テスト用TDを作成する方式を提案した.

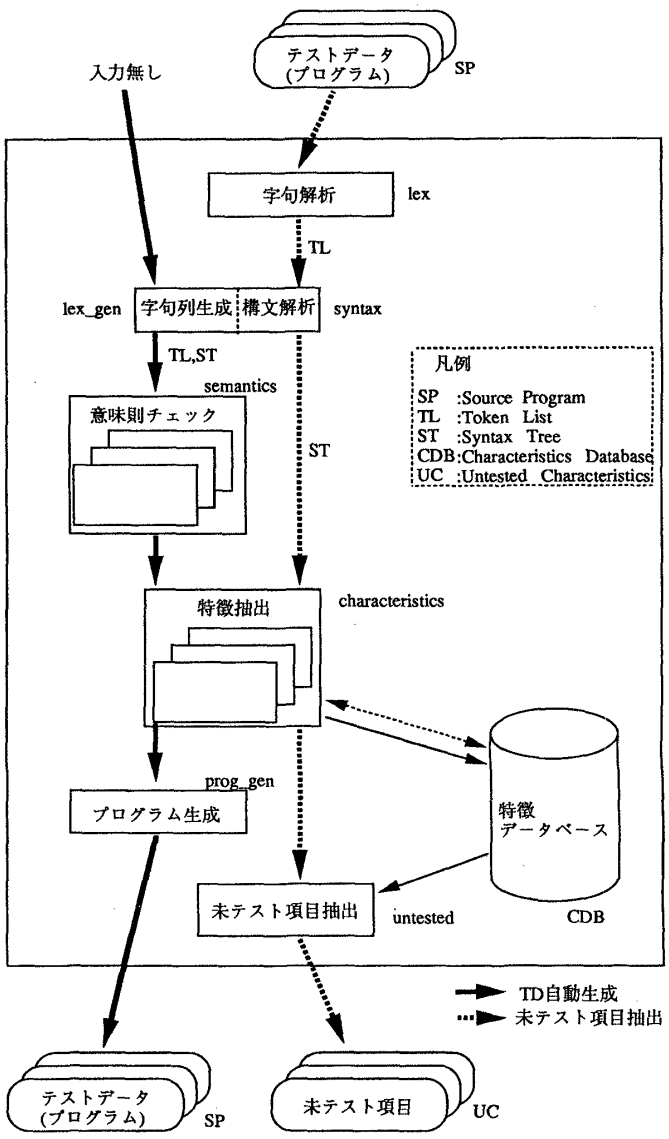

図２　システム構成と処理の流れ

現在，本方式をＣ言語に対しシステム化中であり，一部の仕様を対象としてではあるが有効性を確認している．今後はまず，対象とする仕様の領域を拡張していく．さらに，現在は生成されたプログラムに対し実行確認にも使えるよう人手で修正(printfを挿入するなど)を加えているが，そのまま使えるTDの作成，異常系TDの生成方法に関しても検討を加える.

((参考文献))

[1]藤野,花田:ソフトウェア生産技術,電子情報通信学会,1985,pp.127-140

[2]古川,野木,徳永:AGENT:機能テストのためのテスト項目作成の一手法，情処論，Vol.25，No.5，1984

[3]L.Sterling,E.Shapiro:The Art of Prolog, The MIT Press, 1986, pp.256-265