プログラムの部分的自動生成

7F-8

---　インターフェイス部への表示的意味論の利用　---

三末　和男[1]，　榎本　進[2]
[1]富士通㈱国際情報社会科学研究所，[2]東京理科大学

1．はじめに

ソフトウェアの生産性向上を目的として，我々は表示的意味論[1] を利用したコマンド・プロセッサ・ジェネレータ[2] を設計・開発してきた．そのコマンド・プロセッサ・ジェネレータは対話型システムの一部分であるコマンド・プロセッサを専門に自動生成する．

このようにジェネレータでプログラムの一部分だけを自動生成する場合，プログラムの自動生成される部分と他の部分とのインターフェイスが重要になる．そして，その重要なインターフェイスの部分も含めて自動生成する事，およびその仕様を容易かつ明確に記述する事が望まれる．本稿はコマンド・プロセッサ・ジェネレータを例にとり，部分的な自動生成を行うジェネレータにおけるインターフェイスの問題について考察をする．

2．プログラムの部分的自動生成

プログラムの仕様を与えることにより，その目的とするプログラムの自動生成を行うソフトウェア・ツールを一般にジェネレータと呼ぶ．目的となるプログラムのうち独立可能な一部分を専門に生成するジェネレータを考える．プログラムの一部分を専門に自動生成することにより，専用ジェネレータに期待できるきめ細かな処理と，その他の部分を手書きで作ることによる広範囲の用途に対応可能な汎用性が得られる．

ところが，ジェネレータによりプログラムの独立した一部分を生成する場合，自動生成される部分と手書きの部分は別々のプロセスによって作られるので，それらのインターフェイスが重要になる．ここでいうインターフェイスの機能は，制御構造に関しては一方が他方を呼び出す動作であり，データ構造に関しては，同じデータを表わす異なる構造に対して統一的な表現をとる事である．制御構造との関係から，呼び出す側が自分のデータを，呼び出される側のデータ構造に変換して与え，返されたデータをまた自分のデータ構造に変換するのが自然である．

以上の処理に加えて，プログラムの二つの部分のインターフェイスをどちら側が含むかという問題がある．もしインターフェイス部も制作者が記述するなら，制作者は自動生成される部分の構造について把握する必要がある．しかし，インターフェイス部が自動生成される側に含まれるならば，プログラム制作者には自分自身のプログラム部分についての知識以外は要求されない．よって，以後，自動生成される側にインターフェイス部を含む方向で議論を進める．

3．対話型システムとそのコマンド・プロセッサ

部分的自動生成の対象となるプログラムの例として，TSS などで多く用いられる対話型システムを考える．

対話型システムで仕事をする場合，人間はコマンドによって意志を計算機に伝え，処理を進める．コマンドの入力はシステムにより，様々な形態をとるが，ここではキーボードから入力された文字列をコマンドとする対話型システムを考える．対話型システムはまず入力された文字列の解析を行う．次にその解析結果に基づき，コマンドに対応する対話型システム本体の処理ルーチンを呼び出す．呼び出された処理ルーチンによって人間の期待する処理が進められる．

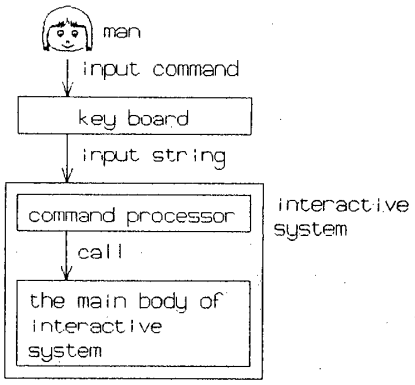

図1　対話型システムにおける
コマンド・プロセッサの位置

On Generating Part of Programs -- an application of denotational semantics to the interface --
Kazuo MISUE[1], Susumu ENOMOTO[2]
[1]International Institute for Advanced Study of Social Information Science, FUJITSU, LIMITED.　[2]Science University of Tokyo

このようなシステムにおいて，入力文字列の解析から処理ルーチンの呼び出しまでの一連の処理を行う部分をコマンド・プロセッサと呼ぶ（図１参照）．

対話型システム全体からコマンド・プロセッサをみた場合，コマンド・プロセッサは入力コマンドに応じて，個々のコマンドに対応する処理ルーチンを呼び出す親ルーチンの役割を演じている．そこで，コマンド・プロセッサ・ジェネレータと呼ばれる専用ジェネレータによって自動生成されるコマンド・プロセッサについて，手書きで作られる個々の処理ルーチンとのインターフェイスについて考えてみる．

コマンド・プロセッサの構成は図２のようになる．基本的にはその動作により，字句解析部，構文解析部，インターフェイス部の三つの部分に分けられる．字句解析部，構文解析部により入力文字列が解析され構文木ができる．インターフェイス部は構文木を参照し，対話型システム本体の各コマンドに対応する処理ルーチンへパラメータを渡し，制御を移す働きをする．対応する処理を行う事がコマンドの意味であることから，インターフェイス部は，言い替えれば，解析の結果をもとに，コマンドに対し意味付けを行うと言える．

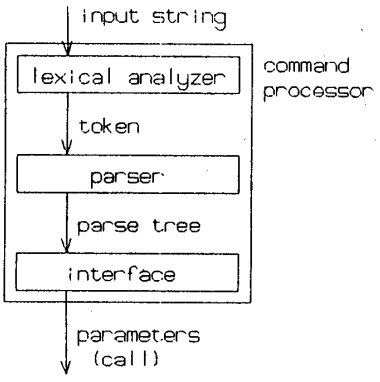

図２　コマンド・プロセッサの構成

４．表示的意味論の利用

コマンドに意味付けを行うコマンド・プロセッサのインターフェイス部の働きを表現するために，表示的意味論を利用する．例として，パラメータとして数値を一つとるコマンド'a'を考える．このコマンドは下記のように定義される意味関数Ｃで表される．定義で使われる補助関数ｆと意味関数Ｅは他で定義されているとする．

生成規則 <cmd> → 'a' <num> に対して

C : Syn → ⟦S → S⟧

C⟦'a' <num>⟧ σ=f(E⟦<num>⟧ σ, σ)

インターフェイス部は入力文字列の解析の結果を参照し，対応する処理ルーチンにパラメータを渡し，制御を移さなければならない．部分的自動生成を行うにあたり重要な位置を占めるこの過程は次のように説明できる：インターフェイス部は得られた解析結果に対して，意味関数Ｃの評価を行う．意味関数Ｃの評価に伴い，意味関数の定義で使用された補助関数ｆが適当なパラメータを伴い評価される．

インターフェイス部を含むコマンド・プロセッサ側の処理はこれだけである．ただし，ｆの評価に当たり，値 E⟦<num>⟧ σ をｆの引数の型に変換しておく必要がある．また，おそらくｆはその値として第２引数のσを単に返すであろう．プログラム上は，補助関数ｆの評価が対話型システム本体の処理ルーチンｆの呼び出しに当たり，意味関数の評価によって適当な処理が行われる事になる．

これにより，パラメータの受け渡しを含む処理ルーチンの呼び出しが，コマンド・プロセッサにとっては疑似的な補助関数によって，自然に表現できる．そして，仕様記述で定義された意味関数と等価な関数を実現する事により，仕様通りのコマンド・プロセッサが実現できる．

５．おわりに

あらゆるインターフェイスへ表示的意味論を応用できるわけではないが，インターフェイスが構文への意味付けと解釈できる処理は多い．その一例としてコマンド・プロセッサ・ジェネレータの作成をもとに，部分的に自動生成されたプログラムと他の部分とのインターフェイスへの表示的意味論の利用について述べた．今後は，表示的意味論の持つ厳密さなどの特徴も有効に生かせるよう検討を加える．

参考文献

[1] Stoy,J.E.:Denotational Semantics - The Scott-Strachey Approach to Programming Language Theory, MIT Press, 1977.

[2] 三末和男他：表示的意味論を用いたコマンド・プロセッサ・ジェネレータの設計，日本ソフトウェア科学会第２回大会7A-4, 1985.