計算機言語設計システム

1H-2

大平 剛　武市 正人
(電気通信大学 計算機科学科)

1. はじめに

形式的意味論(formal semantics)をもとにした言語処理系の生成(compiler generation)が注目されている。SIS[1]はこの代表的なシステムであり、形式的意味論として表示的意味論(denotational semantics)を用いて計算機言語の構文仕様と意味仕様から言語処理系を生成する。今回、このSISの方法を継承した言語設計システムLDSを設計した。

言語処理系はプログラムの構文表現からその意味表現(目的コード)への関数であり、この意味表現は入力から出力への関数と考えられる。また、言語処理系は一般に次のように構成される。

(1)構文解析部(parser)
　語句解析、構文解析をおこない、構文木を生成する。
(2)翻訳部(encoder)
　構文木を目的コードへ変換する。

以下では、構文解析されたプログラムは抽象構文木ASTで表現されるものとする。ある計算機言語PLにたいする翻訳部は次の関数としてあらわされる。

Enc : AST[PL] -> Rep(I -> O)

ここでAST[PL]は構文的に正当なPLプログラムの抽象構文木の集合、IとOは入力列と出力列の集合、またRep(I -> O)は入力から出力への関数を表現したものの集合とする。AST[PL]に属するあるPLプログラムP.ASTに対しEnc(P.AST)はそのプログラムの翻訳コードを生成する。

次にDD.FNをある形式的記述法FNによる計算機言語の表示的定義の集合とすると、PLにたいする表示的定義DD.FN[PL]は次のような関数となる。

DD.FN[PL] : AST[PL] -> Rep.FN(I -> O)

つまり、意味方程式は、「PLプログラムの抽象構文木」から「FNで記述された入力から出力への関数」への関数となる。

以上より、表示的意味記述DD.FN[PL]を直接実現することで、翻訳部を生成できることがわかる。つまり、関数DD.FN[PL]を計算するシステムを実現すればよい。

2. SISの概要

P.Mossesの SIS(Semantics Implementation System)は表示的意味記述の直接実現の原理をもとにした言語処理系生成システムである。SISの構文解析部生成系は標準的な方法で構文仕様から構文解析表を生成する。利用者はBNFを拡張した記述法GRAMで計算機言語の語句仕様を含んだ構文仕様を記述することで構文解析部を生成できる。SISはGRAMプログラムを2パスのSLR(1)構文解析部PARSEへ変換する。これは第1パスで語句解析し、第2パスで構文解析をおこない、プログラムPに対してP.ASTを生成する。

また、SISは意味方程式の記述法DSLで記述した意味定義DD.DSLをLAMB(LAMBはScottによるラムダ式記述法LAMBDA[3]を拡張した記述法)で記述された等価な定義に変換する。PLにたいして変換された結果の意味定義は次の関数であらわされる。

DD.LAMB[PL] : AST[PL] -> Rep.LAMB(I -> O)

DD.LAMB[PL]はLAMBを目的コードと考えるとPLの翻訳部として働く。これをEnc.LAMB[PL]とあらわすことにする。P.ASTをEnc.LAMB[PL]に適用したとき、Pの翻訳コードP.CODEをあらわすLAMB式を生成する。このコードに入力列を与えて実行するためには、入力列に対応するLAMB式をP.CODEに適用する。出力列も当然LAMB式で表現される。SISにはLAMB式を実行するLAMB評価系が用意されている。この評価系は還元規則(reduction rules)を適用することで正規形(normal form)を得る。還元規則は基本的にはラムダ算法のベータ規則であり、評価順序は正規順序(normal order)である。

3. SISの問題点

LDSを設計するに当たり、SISを三菱電機のCOSMO 800-3上に移植し、試用した。SISの移植とその評価については[2]にも述べられているが、SISには次のような問題点がある。

(1)不効率
　1) GRAM、DSLプログラムの2段階処理。プログラムを抽象構文木に変換して中間ファイルに出力してから、構文解析表やLAMB式に変換する。
　2) 命令実行の結果がファイルに出力されるため、結果を次の命令の入力として使う場合に効率がわるい。
　3) LAMB評価系による記憶領域回収。
(2)エラー処理
　1)構文エラーの回復をしない。
　2)生成された構文解析部も上と同様。
　3)意味方程式の領域検査(domain check)をおこなわない。
　4) GRAMとDSLの抽象文法の一致を検査しない。
(3)制約
　1) SLR(1)構文解析による制約。
　2)基本領域(basic domains)が非負整数、論理値、文字列しか用意されていない。

3) 直和領域の処理が不便。
4) 入力/出力は等価なLAMB式でなければならない。

4. LDSの設計

SISを改良して、表示的意味定義から効率的で応用範囲の広い翻訳部を生成する言語設計システムLDS (Language Developement System)の設計と実験的なシステムの作成をおこなった。LDSは現在、構文解析部の生成系にたいしてはまったくサポートしておらず、言語の意味定義の実現しかできないが、次のような特徴がある。

翻訳部生成系は静的な領域検査をおこない、不当な意味方程式を検出する。これにより翻訳部生成時に意味方程式の正当性の検査をある程度おこなうことができるようになる。また、翻訳部生成系は意味定義をUEC ML[3]のプログラムに変換することで意味を実現する。UEC MLは多様型をもつ関数型言語MLの処理系である。

意味方程式の評価系としてMLを利用した理由は

(1) MLの型(type)が表示的意味記述の領域(domain)に類似している.またMLの型検査が領域検査と考えられる。

(2) ラムダ式を扱うための一般的構文を持ち、表示的意味記述も記述できるものと考えられる。また高階関数(higher order function)を扱うことができる。

(3) コンパイルすることで機械語を生成する。また関数閉包(function closure)に対する環境処理のための効率的なコードを生成するので、表示的意味記述の高階関数に対して効率のよいコードが期待できる。

以上のようにSISが意味方程式の評価系としてLAMB評価系を使うのに対して、LDSはMLの評価系を利用したシステムとして実現されている。

5. 領域検査とMLへの変換系

MLの型が表示的意味論の領域によく対応していることからLDSにMLの評価系を利用したが、LDSではMLの型検査を使わずに表示的意味定義をMLプログラムへ変換する以前に領域検査をおこなっている。この理由は、意味記述上では領域方程式は単に領域の同値性の定義として扱われ、意味方程式では領域の変換が自動的におこなわれているかのように記述される。ところがMLでは意味方程式中の型の変換は自動的におこなわれず、型変換のための変換関数を利用者が埋め込まなければならない。この型変換関数を自動的に埋め込むために事前に領域検査が必要になる。MLの型検査が型変換関数を必要とするのに対してLDSの領域検査は、領域の同値性を判定するために、領域変換の可能性を自動的に判定する。したがって、意味方程式はMLで記述されたものよりも簡潔で見やすいものとなっている。

MLへの変換系は領域検査で許された意味定義に対しておこなわれる。この変換系は領域方程式から自動的に型変換関数の定義を生成し、意味方程式の中へ型変換関数を埋め込みながらMLプログラムへ変換する。

6. 評価と問題点

意味定義の実現にUEC MLを利用したことにより実行速度が改善された。しかし、UEC MLの評価系は評価順序が作用的順序(applicative order)であるため、停止性に問題を残している。この点は意味定義の記述の際に注意を払うことで、ある程度解決されるが、問題点として残る。また、MLを正規順序(normal order)で評価するように改良することも考えられる。

構文解析部の生成系については適応範囲の広いものが要求されるが、これは標準的な方法や生成系が確立されているので、評価系とのインターフェースを設計すれば容易に解決できる。

評価系にたいしてはML以外のものを使うことも考えられる。領域検査部はMLへの変換系とは独立に設計されているので、変換系を変えることでほかの評価系へのソースプログラムを生成するシステムを容易に作成できる。この評価系として結合子(combinator)による評価系などが考えられる。

参考文献

[1] Mosses,P.D., SIS - Semantics Implementation System Reference Manual and User Guide, Computer Science Departement, Arhus University, (August 1979)

[2] Bodwin,J., Bradley,L., Kanda,K., Litle,D., and Pleban,U., Experience with an Experimental Cmpiler Generation Based on Denotational Semantics, ACM SIGPLAN NOTICES vol.17 no.6, (June 1982)

[3] 中条 秀和、武市 正人、多様型をもつ関数型言語MLの処理系の移植について、情報処理学会第28回全国大会、1984