# 意味関数によるC言語の意味形式化*

3U－3

篠崎政久†　宮寺庸造‡　米田信夫§

東京電機大学¶

## 1　はじめに

プログラムの表示的意味論は、1969年D.Scottoらにより提案された。表示的意味論では、意味関数によってプログラムを意味領域上に写像し、そのプログラムの意味を意味領域上で関数として表現する。この意味領域上の関数を調べることにより計算の可能性、停止性を調べることが、さらには、プログラムの検証、意味同値性、デバッグ、自動合成等に応用可能となった[1,2]。表示的意味論は、言語処理系の設計、実装にも応用でき、最近は、意味の設定から言語の設計がなされている。表示的意味論は、既存のプログラム言語の意味解釈にも使用されており、その解釈は、プログラムの高速化、コンパイラ製作の簡略化に役立っている[3]。

しかし、既存の言語に意味が定義された研究結果は多数あるが[1,2]、ポインタに関する議論は十分されていない。ポインタは、マシン、OSへの依存度が高いため、抽象化が難しいと考えられているからである。ポインタをサポートするC言語に対して意味を設定することは、Cプログラムの計算可能性の判断や、自動生成、他言語との変換に必要不可欠であると思われる。そこで本稿では、従来の研究ではなされていなかったポインタや構造体に関する意味記述に特に焦点をしぼり、C言語に対する意味関数の設定を行った。

## 2　準備

C言語に対する意味記述を行うために、下記のような構文領域、意味領域の導入を行い、メタ変数の設定を行う。

本稿で扱うCの構文領域を以下のように設定する。
$\xi \in Id$｛識別子全体｝,$\zeta \in Int$｛整数全体｝,$\tau \in Bool$｛論理値全体｝,$\phi \in Type$｛型全体｝,$\eta \in Mexp$｛単項表現全体｝,$\mu \in Mope$｛単項演算子全体｝,$\omega \in Pfix$｛後置子全体｝,$\beta \in Bope$｛二項演算子全体｝,$\delta \in Vdef$｛変数宣言全体｝,$\varepsilon \in Exp$｛式全体｝,$\gamma \in Cmd$｛文全体｝

*Formal Semantics of Language C by semantic functions
†MasahisaShinozaki
‡YouzouMiyadera
§NobuoYoneda
¶TokyoDenkiUniversity

C言語の意味を表現するために以下のように意味領域を設定する。
$T$｛型｝, $B$｛真偽値｝, $N$｛整数値｝, $err$｛エラー｝, $L$｛アドレス｝, $E = B + N + L$｛記憶可能な値｝,$\nu \in V = E + K + err$｛値全体｝,$\sigma \in S = L \to E$｛状態｝,$\kappa \in K = V \to C$｛式接続｝,$\rho \in Env = Id \to L \times T$｛環境｝$\theta \in C = S \to S$｛命令接続｝,$\chi \in D = Env \to C$｛環境接続｝

ここでは、扱う型を論理型、整数型、ポインタ型に制限しているが、他の型（文字型、浮動小数型）も同様に導入することが可能である。環境$\rho$は、識別子からアドレスと型が得られる関数であり、この得られた型を使用して、配列、構造体の扱いが可能となる。型の領域はポインタ型、構造体の型を表現可能なように次の構成になっている。

$$T = \mathrm{B} + \mathrm{N} + \langle \mathrm{L}, T \rangle + \langle T_1, T_2, \ldots, T_n \rangle$$

ここで、B,N,Lは、それぞれ論理型、整数型、アドレスを表している。$\langle \mathrm{L}, T \rangle$はポインタ型を表しており、Lはアドレスを、Tは Lが指し示す型先を表している。
$T = \langle T_1, T_2, \ldots, T_n \rangle$は、構造体の型を表している。

## 3　意味関数

2節で設定した構文領域の各要素に意味関数の定義を行う。

### 3.1　変数宣言の意味関数

変数宣言に対しては、次のような型を持つ意味関数$\mathcal{D}$を定義する。

$$\mathcal{D} : Vdef \to Env \to D \to C$$

**変数宣言の意味関数**

$$\mathcal{D}[\phi\xi]\rho\chi\sigma = \chi(\rho[< L_\xi, \phi > /\rho]\sigma[\perp /L_\xi])$$

変数宣言$\delta$は環境$\rho$の更新ととることができ、環境接続$\chi$によって更新された環境を後のプログラムに引き継ぐことができる。

## 3.2 式の意味関数

式に対しては、次のような型を持つ意味関数$\mathcal{E}$を定義する。

$$\mathcal{E} : Exp \to Env \to K \to C$$

**&演算子の意味関数**

$$\mathcal{E}[\&\xi]\rho\kappa\sigma = \kappa(\pi^1(\rho[\xi]))\sigma$$

&演算子の意味関数では、識別子の意味関数のみをここでは示したが、単項表現に適用した場合もこの定義と同様に定義可能である。

**変数の意味関数**

$$\mathcal{E}[\xi]\rho\kappa\sigma = \mathcal{E}[\&\xi]\rho\{\lambda\nu.cond(\kappa(\sigma(\nu\mid_L)), fail)(\lceil\nu \in L\rceil)\}\sigma$$

***演算子の意味関数**

$$\mathcal{E}[*\eta]\rho\kappa\sigma = \mathcal{E}[\eta]\rho\{\lambda\nu.cond(\kappa(\sigma(\nu\mid_L)), fail)(\lceil\nu \in L\rceil)\}\sigma$$

**配列の意味関数**

$$\begin{aligned}\mathcal{E}[\omega[\varepsilon]]\rho\kappa\sigma =& \mathcal{E}[\omega]\rho\{\lambda\nu_0.cond(A_0, fail)(\lceil\nu_0 \in L\rceil)\}\sigma\\ A_0 =& \mathcal{E}[\varepsilon]\rho\{\lambda\nu_1.cond(A_1, fail)(\lceil\nu_1 \in N\rceil)\}\sigma\\ A_1 =& cond(\kappa(\sigma(A_2)), fail)(ge(\nu_1\mid_N)(\bar{0}))\\ A_2 =& add(\nu_0\mid_L)(mult(\nu_1\mid_N)(A_3))\\ A_3 =& GetSize(\pi^2(\rho[\omega]))\end{aligned}$$

配列の意味関数では、指定するデータのアドレスを求めることが要求される。そのため、配列の1ブロックのサイズを調べなければならない。1ブロックのサイズは、その配列の型に依存するので、次のような補助関数$GetSize$を用意する。

$$GetSize : T \to N$$

ここで、N はT のサイズ

補助関数$GetSize$は、C言語のSize of演算子と同等の機能を持ち、これによって1ブロックの大きさが求められることになる。

**構造体の意味関数**

$$\begin{aligned}\mathcal{E}[\omega.\xi]\rho\kappa\sigma =& \mathcal{E}[\&\omega]\rho\{\lambda\nu.cond(\kappa(\sigma(B_0)), fail)\\ & (\lceil\nu \in L\rceil)\}\sigma\\ B_0 =& add(\nu\mid_L)(GetAdr[\langle\omega,\xi\rangle])\\ \mathcal{E}[\omega{\to}\xi]\rho\kappa\sigma =& \mathcal{E}[\omega]\rho\{\lambda\nu.cond(\kappa(\sigma(B_0)), fail)(\lceil\nu \in L\rceil)\}\sigma\end{aligned}$$

構造体の意味関数においても指定するデータのアドレスを求めるために、親変数に対するメンバの相対位置を求める補助関数$GetAdr$を用意した。

$$GetAdr : Pfix \times Id \to N$$

ここで、N はId の相対位置

この関数で求められた相対位置と親変数のアドレスを加算することによって、指定したデータのアドレスを得ることができる。

## 3.3 文の意味関数

文に対しては、次のような型を持つ意味関数$C$を定義する。

$$\mathcal{C} : Cmd \to Env \to C \to C$$

**代入文の意味関数**

$$\begin{aligned}\mathcal{C}[\eta = \varepsilon]\rho\theta\sigma =& \mathcal{E}[\&\eta]\rho\{\lambda\nu_0.cond(D_0, fail)(\lceil\nu_0 \in L\rceil)\}\sigma\\ D_0 =& \mathcal{E}[\varepsilon]\rho\{\lambda\nu_1.cond(\theta(D_1), fail)(\lceil\nu_1 \in E\rceil)\}\\ D_1 =& \sigma[\nu_1\mid_E/\nu_0\mid_L]\end{aligned}$$

代入文の意味関数では、アドレスを求める段階と代入する値を求める段階に分けることができ、それぞれには、式の意味関数が適用される。識別子、配列、構造体、代入等の意味関数は、&演算子の意味関数に基づいている。

**連接文の意味関数**

$$\mathcal{C}[\gamma_0;\gamma_1]\rho\theta\sigma = \mathcal{C}[\gamma_0]\rho\{\mathcal{C}[\gamma_1]\rho\theta\}\sigma$$

**条件文の意味関数**

$$\begin{aligned}\mathcal{C}[if(\varepsilon)\gamma_0 else\gamma_1]\rho\theta\sigma =& \mathcal{E}[\varepsilon]\rho\{\lambda\nu.cond(E_0, fail)\\ & (\lceil\nu \in B\rceil)\}\sigma\\ E_0 =& cond(\mathcal{C}[\gamma_0]\rho\theta, \mathcal{C}[\gamma_1]\rho\theta)(\nu\mid_B)\end{aligned}$$

**繰り返し文の意味関数**

$$\begin{aligned}\mathcal{C}[while(\varepsilon)do\gamma]\rho\theta\sigma =& fix(\lambda\theta'.\mathcal{E}[\varepsilon]\rho\{\lambda.cond(F_0, fail)\\ & (\lceil\nu \in B\rceil)\}\sigma\\ F_0 =& cond(\mathcal{C}[\gamma]\rho\theta', \rho\theta)(\nu\mid_B)\end{aligned}$$

**複文の意味関数**

$$\begin{aligned}\mathcal{C}[\{\delta;\gamma\}]\rho\theta\sigma =& \mathcal{D}[\delta]\rho\{\lambda\rho'.\mathcal{C}[\gamma]\rho'\theta\}\sigma\\ \mathcal{C}[\{\gamma\}]\rho\theta\sigma =& \mathcal{C}[\gamma]\rho\theta\sigma\\ \mathcal{C}[\{\}]\rho\theta\sigma =& \theta\sigma\end{aligned}$$

# 4 終わりに

本稿では、手続き型言語の意味を取る際に良く使用されるアドレスを含めた環境$\rho : Id \to L$を、$\rho : Id \to L \times T$に拡張することによって、C言語の構文に意味関数を設定した。また従来の研究では充分議論されていなかった、ポインタ、配列、構造体に対する議論が行えた。ポインタ等を使用している言語は多数あるが、同様な手法で意味を与えることができる。

今後の研究課題として、意味領域上で表された関数の数学的性質を明らかにすること、さらに、C言語と関数型言語との意味同値性、変換に関する研究がある。

## 参考文献

[1] 中島玲二:数理情報学入門, 朝倉書店,1982.
[2] VanLeeumen:Handbook of Theoretical Computer Science Volume B.Formal Models and Semantics, The MIT Press,1990.
[3] 井田哲雄:ラムダ計算とそのモデルによるCOMMON LISPの解釈と新しい処理系, コンピュータソフトウェア, Vol.4,No.4,pp.33-44,(1984).