UCM を用いたプログラミング言語の意味記述に関する研究

97B534 金枝 順子　97B603 朝長 武士

指導教員　野呂 昌満

1　はじめに

現在, プログラミング言語は構文規則と意味規則によって定義される. 構文規則の定義には, おもに文脈自由文法とよばれる記法や BNF(Backus Naur Form) やその拡張形である拡張 BNF(Extended BNF) とよばれる記法がもちいられている. 構文規則は, yacc, bison などの構文解析プログラム自動生成系で機械処理できる. 意味規則は, プログラムや構文要素の働きおよび制約を定義することである. 意味規則は現在使用できるどんな記法をもちいても, 構文規則よりはるかに記述が困難である. プログラミング言語の意味を定義するには, 非形式的な記述をするか, 例を示して構文規則が何を意味するのかを自然言語で記述するしかない.

この問題を解決するために, これまでに UCM (Universal Computational Model) が提案された. UCM は, 拡張したメッセージ送信 GMP(Generalized Messeage Passing) にもとづくオブジェクト指向計算モデルである. オブジェクト同士が GMP だけをもちいて計算し, プログラミング言語の意味を記述する. UCM の目的は, プログラミング言語の意味を機械処理することであり, そのためには意味を厳密に定義する必要がある. 意味を厳密に定義するためには, 記述の方法が必要となる.

UCM の研究は, 以下の手順で進められている.

1. モデル (UCM) の構築
2. 手作業での意味記述
3. 言語 (UCM Language) の設計・実現
4. 意味記述に矛盾がないか検証

はじめにモデルを構築する. 構築したモデルをもとに手作業での意味記述をし, 矛盾が生じた場合モデルの再構築をして, 矛盾が生じなければ言語の設計・実現をする. 最後に設計・実現した言語で意味の機械処理ができるかを検証する. モデルの検証は, 随時おこなう.

これまで, UCM の研究では 2 回のモデルの構築がおこなわれた. これまでのモデルをもとに意味記述をした結果, 次のような問題点があることがわかった.

- オブジェクト同士の関係の考察がされていない
- 原始言語要素の定義があいまいである

本研究の目的は,

- UCM の再構築
- UCML の設計

である.

本研究では上記の問題点を考察し, これまで提案されたモデル [1][2] を参考に, 3 回目のモデル構築をおこなった. 本研究は, 研究の手順の, モデルの構築, 手作業での意味記述, 言語の設計にあたる.

2　計算モデル UCM

プログラミング言語の意味規則を厳密に定義し, 機械処理するために UCM が提案された.

UCM には, 次のような利点がある.

- 全てのプログラミング言語の意味を記述することができる
- GMP(拡張したメッセージ送信) だけを用いた単純なしくみ

UCM はオブジェクト指向の考えにもとづいた計算モデルである. UCM のオブジェクトは GMP をもちいて通信する.

GMP とオブジェクトについて説明する.

2.1　GMP

オブジェクトは GMP をもちいて通信する. GMP は UCM のただ一つのしくみである. GMP は以下の性質を持つメッセージ送信である.

- メッセージ送信の結果, オブジェクトを生成することができる.
- メッセージ送信の結果, オブジェクト間に関係ができる.

GMP は, 防護条件部とメッセージ送信部からなる. 防護条件部とは, 論理式からなり, メッセージ送信部を制御する部分である. 防護条件部を評価することによって, True オブジェクトもしくは False オブジェクトが生成される. 防護条件部の評価で True オブジェクトが生成された場合だけ, メッセージ送信部が実行される. True オブジェクト, False オブジェクトについては後述する.

2.2　UCM オブジェクト

オブジェクト間の関係を説明する. 考察した結果, オブジェクト同士の関係は次の二つの関係だけで説明できることがわかった.

- 委託 関係
  メッセージを委託するオブジェクトと委託されるオブジェクトとの関係である.
  オブジェクトは, 処理することができないメッセージを受け取ったとき, メッセージの処理をあらかじめ指定したオブジェクトに委託することができる. これを委託機構という.
- 全体部分 関係
  オブジェクトと, オブジェクトを構成するオブジェクトとの関係である.

UCM のオブジェクトの説明をする.

オブジェクト

オブジェクトは委託変数，全体部分変数，変数オブジェクト，メソッド，インタフェースを持つ．(図１参照)

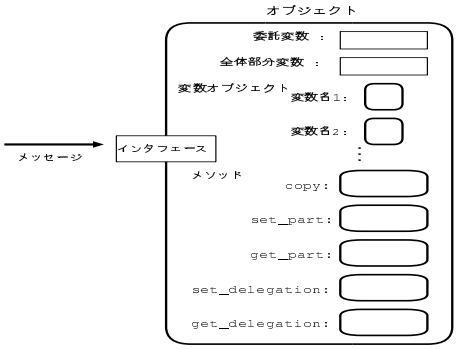

図 1: オブジェクト

インタフェースは，オブジェクトが持っているメソッドの情報を保持している． インタフェースは，オブジェクトが受け取ったメッセージとオブジェクトが持っているメソッドをバインドするものである．
委託変数は，オブジェクトの委託先であるオブジェクトを保持するものである．
全体部分変数は，オブジェクトの構成要素であるオブジェクトを保持するものである．

オブジェクトは以下のメソッドを持つ．

`copy()` オブジェクトのコピーを生成する．
`set_delegation(`引数`)`
委託変数に引数のオブジェクトを保持する．
`get_delegaion()`
委託変数が保持しているオブジェクトを取得する．
`set_part(`引数`)`
全体部分変数に引数のオブジェクトを保持する．
`get_part(`引数`)`
全体部分変数に保持されているオブジェクトを取得する．

UCM はプログラミング言語の意味を記述するプログラミング言語なので，一般的なプログラミング言語ではあらかじめ組み込まれているものもオブジェクトとして定義する．これを原始オブジェクトと呼ぶ．
原始オブジェクトとして次のものを用意する．

変数オブジェクト
変数もオブジェクトとして扱う．
メソッドオブジェクト
オブジェクトが持つメソッドもオブジェクトとして扱う．
メッセージオブジェクト
メッセージ式もオブジェクトとして扱う．
True オブジェクト，False オブジェクト
論理式を評価した結果生成される true，false もオブジェクトとして扱う．

原始オブジェクトの説明をする．

変数オブジェクト

変数オブジェクトは名前を持つ．変数オブジェクトは，オブジェクトの参照を保持するものである． メソッド，インタフェースを持つ．(図２参照)

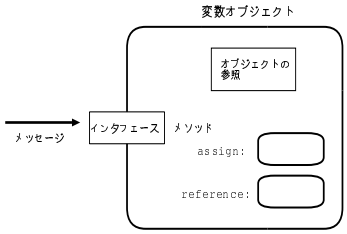

図 2: 変数オブジェクト

変数オブジェクトは以下のメソッドを持つ．

`assign(`引数`)`
引数のオブジェクトを参照する．
`reference()`
変数オブジェクトが参照しているオブジェクトを取得する．

メソッドオブジェクト

メソッドオブジェクトは名前を持つ．メソッドオブジェクトは，全体部分変数，ブロック，インタフェースを持つ．(図３参照)

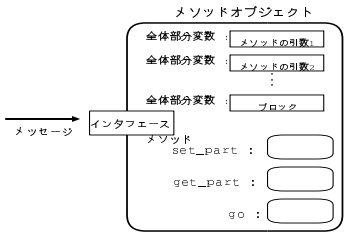

図 3: メソッドオブジェクト

ブロックとは，メソッドの中身であるスクリプトである．
メソッドオブジェクトは以下のメソッドを持つ．

`set_part(`引数`)`

全体部分変数に引数のオブジェクトを保持する．

get_part(引数)
全体部分変数に保持されているオブジェクトを取得する．

go()
メソッドの中身であるブロックを実行する．

メッセージオブジェクト

メッセージオブジェクトは，受信オブジェクト，メッセージ，メッセージの引数であるオブジェクトを保持する．(図 4 参照)

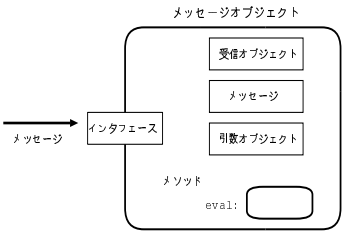

図 4: メッセージオブジェクト

メッセージオブジェクトは以下のメソッドを持つ．

eval()
メッセージ送信をおこない評価する

True オブジェクト，False オブジェクト

True オブジェクトと False オブジェクトは，GMP の防護条件部の論理式を評価することで生成されるオブジェクトである．

3 プログラミング言語 UCML の設計

前章で述べた計算モデル UCM に，構文を与えたプログラミング言語が UCML(UCM Language) である．UCML は，プログラミング言語の意味規則を記述するためのプログラミング言語である．
UCML を設計するにあたって，次のような方針でおこなう．

モデルを素直に実現する．
プログラミング言語の意味を記述するのに必要な言語機能だけを用意する．
見やすく，わかりやすくするためにシンタックスシュガーをまぶす．

3.1 UCML 言語仕様

設計方針にしたがい，UCML にはクラスの概念を持たせない． クラス継承，クラスの実体化は次のように実現する．

クラスの実体化
メッセージ送信によってオブジェクトのコピーを生成することで実体化を実現する．
クラス継承
委託機構によって実現する．
オブジェクトは，処理できないメッセージを受け取ったとき委託先のオブジェクトにメッセージの処理を委託することができる． 委託することによって委託先のオブジェクトの機能を継承しているようにふるまうことができる．

他のプログラミング言語における型 (integer や real など) も，UCML ではオブジェクトとして実現する．これを Builtin オブジェクトと呼ぶことにする． 本研究で用意した Builtin オブジェクトは，以下のものである．

Integer オブジェクト
Real オブジェクト
Char オブジェクト
Array オブジェクト
String オブジェクト

GMP 式

UCML の命令文は，すべて GMP 式である． GMP 式は次のように記述する．

```
| [ 防護条件部 ] 受信オブジェクト message( 引数 ) |
```

記述のさい，防護条件部は省略可能である． 防護条件部が省略された場合は常に True オブジェクトが生成されるものとする．

実際の記述法

UCM はすべて GMP で通信するが，そのまま GMP 式として記述すると括弧を多用することになる． 見にくく，実行する文の順序もわかりづらいので， 実際に記述するときにはシンタックスシュガーをまぶして記述することにする．

Object 定義部
ユーザがオブジェクトを定義するとき，オブジェクトを参照している変数名，オブジェクトが持つ変数，オブジェクトが持つメソッド，オブジェクトの委託先を一つ一つ GMP 式で記述する．これを オブジェクトの定義部として，実際の記述では次のように記述することにする．

```
OBJECT  名前  IS
        変数１, 変数２, … ;
  MESSAGE メソッド名１( 引数 ) ブロック
              …
  DELEGATION 委託先名 ;
END 名前 ;
```

代入演算子
変数オブジェクトへの値の代入は,

```
変数オブジェクト名 assign(値)
```

とするが, これを := をもちいて, 次のように記述することにする.

```
変数オブジェクト名 := 値
```

メソッド呼び出し
メソッド呼び出しは

```
( ( ( (変数名 reference() ) get_part( メソッド名 ) )get_part(仮引数)) assign ( 実引数 ));
( (変数名 reference() ) get_part( メソッド名 ) )go()
```

とするのを, 次のように記述することにする.

```
変数名 メソッド名 ( 実引数 )
```

4 意味記述

プログラミング言語の意味規則は, 非形式的な記述か, 例を示して構文規則が何を意味するのかを自然言語で記述されているのが現状である.UCM はそれを解決する計算モデルである. 本章では構築したモデルで, 実在する手続き指向プログラミング言語 Pascal の意味記述をする. 意味記述をすることによって UCM の検証も同時にすることができる. 副プログラムである手続きの宣言を例にあげる.
Pascal の

```
Procedure  aSub_name  parameter_list;
       declaration_list
   begin
       subpragram_block
   end
```

という記述は, 以下のように記述できる.

```
(((aSub_name reference()) set_part(
    parameter_list))
    set_part(declaration_list))
    set_part(subprogram_block)
```

副プログラムもひとつのオブジェクトとして考えることができる. 副プログラムである手続きは,

- 仮引数部
- 局所変数宣言部
- ブロック部

で構成されている. 副プログラムオブジェクトと以上の 3 つは UCM で定義された全体部分関係にあたる. 仮引数部, 局所変数部, ブロック部のそれぞれの構成要素を副プログラムオブジェクトの構成要素に追加する. Pascal のもうひとつの副プログラムである関数も返り値の情報を副プログラムに構成要素に追加するだけなので, 手続きと同様に定義できる. 本研究で構築した UCM では, 上記のように副プログラムを記述することができる.

5 おわりに

プログラミング言語の意味規則を厳密に定義するために, 計算モデル UCM が提案された. 本研究では, これまで構築されてきたモデルを参考に, 新しくモデルを構築し提案した. そして, 構築したモデルを素直に実現したプログラミング言語 UCML を設計した. 今後の課題は, 以下のものである.

- UCML の実現
- 関係の原始言語要素の考察
  - 全体部分関係に対する原始言語要素
    われわれが構築したモデルは, 構成するオブジェクトを取得するとき使用者がオブジェクトを指定するようになっている. 取得するオブジェクトの特定方法, 必要性などを考慮する必要がある.
  - 委託関係に対する原始言語要素
    われわれが構築したモデルは単一継承を実現している. 多重継承の実現を考慮する必要がある.

参 考 文 献

[1] Masami Noro and Tsuneo Ajisaka: Manipulating Software Semantics with Uni ed Computational Model and Software Quark Model, Asia-Paci c Software Engineering Conference, pp.476-483,1999.

[2] 熊崎 敦司, 中田 義也 : UCML 処理系の設計と実現, 南山大学経営学研究科修士論文, 2000.

[3] 原田 賢一 : コンパイラ構成法, 共立出版, 1999