# 言語Cとのインターフェイスを持つScheme処理系

2F−9

寺川次郎，早川栄一，下村秀樹，並木美太郎，高橋延匡
東京農工大学

## 1．はじめに

我々の研究室では，日本語情報処理のためのフル2バイトコードを採用した独自開発の言語Cコンパイラ CAT[1]を利用してOS/omicron[2]などのシステムプログラムの開発を行ってきた．OS/omicron 上で日本語情報処理やヒューマンインターフェイスの研究を進めて行く過程でプロトタイピングを行う頻度が増加し，より開発効率の高いプログラミング言語が必要になった．このため，新たにプロトタイピング用のプログラミング言語としてScheme[3]の処理系を実現した．このScheme処理系の実現においては，（1）プログラマの言語処理系の移行にともなう負担軽減（2）研究室内に蓄積された言語Cによるソフトウェア資産の有効利用を考慮し，Schemeの仕様を拡張して言語Cとのインターフェイスを提供した．

本稿では，我々が，言語Cとのインターフェイスのために Schemeに行った仕様拡張の一方式について述べる．

## 2．言語Cによるプロトタイピングの問題点

プロトタイピングを行う場合には，プログラム作成の目的は，アルゴリズム，処理方式，処理内容の検証なので，記憶領域の確保や解放は本質的ではない．しかし，言語Cの仕様では記憶領域の管理は事実上プログラマに委ねられている．

また，言語Cに限らずコンパイルを行う言語処理系は，プログラムの編集，実行，修正のサイクルが長い．何度も実行と修正を繰り返す傾向のあるプロトタイピングでは，このサイクルが短いインタープリタ型の言語処理系が適している．

## 3．プロトタイピング用言語処理系の開発目標

プロトタイピング用言語処理系の開発目標は，プロトタイピングの場面においてプログラムの開発効率を高めることである．これにより，効果的なアルゴリズムやインターフェイスの調査を効率よく行うことが可能になる．

特に我々の研究室では，言語Cでは記述性が低く開発に時間がかかる種類のプログラムがあり，それらを効率的に開発できるようにすることが効果的である．

さらに，一般に言語処理系は，処理系の信頼性，実用的な実行効率を確保することが要求される．

## 4．本処理系の設計方針

上に述べた言語Cの特徴と，本処理系の開発目標から検討した結果，（1）静的スコープの採用によりプログラムの可読性が高い（2）仕様が小規模であるため実現が容易である（3）形式評価の方法が限られているため言語仕様が把握しやすいなどの特徴を持つ，Schemeを本処理系の仕様とした．

これらのSchemeの特徴はプロトタイピング用の言語として十分なものだが，我々の研究室特有の要求として，言語Cのソフトウェア資産の活用がある．このため，我々はSchemeの仕様に拡張を行い，本処理系上に言語Cとのインターフェイスのための関数群を実現した．

## 5．言語Cとのインターフェイス関数の設計方針

言語Cとのインターフェイス関数の設計で問題となるのは，言語CとSchemeの数値の表現範囲，引数や戻り値に言語Cのポインタや構造体を含む場合の扱い，言語C関数の呼出しの記述方法などである．

本処理系の設計では，これらの問題に対し次のような方針を立てた．

・CATのライブラリの利用を考慮し，CATにより生成されたオブジェクトモジュール中の言語C関数を呼び出す．

・言語CとSchemeの間で数値が渡されるときは表1の変換規則に従い変換を行い，変換結果がデータを渡された側の表現範囲にない場合には，エラーとする．

・言語CとSchemeの間のポインタの受渡しをサポートするため，Schemeにメモリ参照の関数をサポートする．

・言語C関数の呼出しがSchemeの関数の呼出しと同じ形式で記述できるようにする．

## 6．言語Cインターフェイス関数の設計

表2に我々がSchemeに拡張した言語Cインターフェイス関数を示す．ここでは，これらの拡張された関数の設計の詳細について述べる．

### 6．1　オブジェクトモジュールを指定するための関数

方針で述べたように，本処理系の言語Cインターフェイスはオブジェクトモジュール中の関数単位で行う．このため，言語Cの関数を呼び出すための準備として，使用する関数が含まれるオブジェクトモジュールを指定する必要がある．我々はオブジェクトモジュールの指定のために `use-c-module` 関数と `unuse-c-module` 関数を拡張した．`use-c-module` 関数は引数で指定されたCATのオブジェクトモジュールをメモリ上にロードし，実行のための準備を行う．`unuse-c-module` 関数は引数で指定されたCATのオブジェクトモジュールを解放する．

---

A Scheme interpreter providing a C function interface

Jiro TERAKAWA, Eiichi HAYAKAWA, Hideki SHIMOMURA, Mitarou NAMIKI and Nobumasa TAKAHASHI
Tokyo University of Agriculture and Technology

### 6．2　言語 C 関数の宣言のための関数

CAT　のオブジェクトモジュールには関数の型と引数の型の情報が含まれていないので，プログラマは言語 C の関数を呼び出すために，Scheme 処理系に対して言語 C 関数の宣言を行う必要がある．この宣言を行うための関数が，`c-func`　である．`c-func`　は，引数に宣言された言語 C 関数の引数と戻り値の型の情報から，「言語 C 呼出しオブジェクト」を生成し，評価結果として返す．

「言語 C 呼出しオブジェクト」には，宣言された言語 C の関数の名前，引数の型，戻り値の型が記録されている．「言語 C 呼出しオブジェクト」は，手続きとして評価されると，各引数を評価した後，表 1 に従い　`c-func`　で宣言された型に変換する．さらに，変換された引数を呼び出す言語 C 関数に渡して実行を移す．言語 C 関数の実行から戻ると戻り値を `c-func`　で宣言された型とみなし，表 1 に従い変換し，評価結果として返す．`c-func` で指定できる言語 C の型は，表 1 に含まれている型である．

「言語 C 呼出しオブジェクト」の導入により，`c-func`　関数で言語 C の関数の宣言をした後は，Scheme のプログラム中で言語 C 関数が Scheme の手続きと同様に使用できる．

### 6．3　ポインタを扱うための関数

言語 C 関数が引数や戻り値にポインタを宣言している場合には実際に渡したいデータはアドレスの内容であるため，表 1 のような単なる表現の変換だけでは不十分である．ポインタの扱いに関しては，戻り値として言語 C から Scheme 処理系が受け取る場合，引数として Scheme 処理系から言語 C に渡す場合に分けて考察する．

戻り値がポインタの場合，実際に意味があるのはポインタの指している番地の内容なので，　ポインタの内容を得る手続きが必要になる．　CAT の関数がポインタとして返す値のサイズは，32bit の値である．本 Scheme 処理系では，この値を 32bit 整数値として受け取る．このため Scheme のプログラム中で，言語 C のポインタに対し演算を行うことが可能である．

ポインタの指している内容を得るために，Scheme の整数値をアドレスとみなしてアドレスの内容を得る `peek`　関数を拡張した．言語 C 関数が戻り値として構造体へのポインタや配列を返す場合には，`peek`　関数と，Scheme の標準の算術演算を組み合せて使用すれば，ポインタの示す構造体のメンバや配列の各要素を参照することができる．

ポインタが引数の場合，ポインタの指しているアドレスの内容を設定できることが必要になる．ポインタを使用して言語 C に渡すオブジェクトは，実際にはポインタの指し示すメモリのイメージである．つまり，ポインタの引数を言語 C に渡すためにはメモリイメージを作成できる必要がある．本処理系では，「整数値ベクタ」と呼ばれる要素に 32bit 整数値だけを許すベクタ型が拡張されている．「整数値ベクタ」は，タグなどが収められているヘッダの部分以外はすべてデータ領域になっているため．数値ベクタの要素に数値を代入してやることによって，構造体や配列のメモリイメージをこのベクタのデータ部分に作成してやることができる．「整数値ベクタ」の利用によってポインタに指されるべき言語 C のオブジェクトのメモリイメージを設定することができる．

さらに，Scheme オブジェクトの内容が置かれているアドレスを数値として取り出す手続きを Scheme に拡張する必要がある．このための拡張関数が `get-pointer`　である．

`get-pointer`　は引数のオブジェクトをガーベジコレクションで移動されない領域にコピーし，そのオブジェクトのデータ部分の先頭番地を指すポインタの値を整数として返す．`get-pointer`　を使用することによって，言語 C 関数にポインタ引数を渡すことができる．

### 7．おわりに

Scheme の仕様を拡張し「言語 C 呼出しオブジェクト」を導入することにより，言語 C の関数呼出しの記述を Scheme の関数呼出しと似たインターフェイスで可能にした．

今回拡張した仕様では，次のような問題点がある．

・構造体と配列の受渡しの記述に手間がかかる
・型の宣言に使用できる型が言語 C の基本的な型に限られている

今後は，今回拡張した Scheme の関数をプリミティブとしてこれらの問題の解決を行うための関数を実現して行くつもりである．

表 1　言語 C と Scheme のデータ変換表

| Scheme | 言語 C |
|---|---|
| 整数（`LONG`） | `SHORT(8bit)`<br>`INT(16bit)`<br>`LONG(32bit)` |
| 浮動小数点数（`FLOAT`） | `FLOAT(32bit)`<br>`DFLOAT(64bit)` |
| 文字（`CHAR`） | `CHAR(16bit)` |
| 数値ベクタ（`LONG[n]`） | 構造体 |
| 整数（`32bit`） | ポインタ（`32bit`） |

表 2　言語 C 追加リンクのための関数

| | |
|---|---|
| 言語 C モジュールの指定に関する関数 | `use-c-module`<br>`unuse-c-module` |
| 言語 C 関数の宣言のための関数 | `c-func` |
| オブジェクトのポインタを得る関数 | `get-pointer` |

### 参考文献

[1]高橋：“研究プロジェクト総説 OS/omicron の開発”，情報処理学会オペレーティングシステム研究会報告 39-5，1988

[2]並木，他：“OS/omicron 用システム記述言語 C 処理系 Cat のソフトウェア工学的見地からの方式設計”，電子情報通信学会論文誌 D，Vol. J71-D，No.4，pp.652-660，1988-4

[3]Jonasan Rees，William Clinger 他：“Revised[3] Report on the Algorithmic Language Scheme”，ACM SIGPLAN Notices 21 (12)，ACM，1986