3L-7

携帯電話向け Scheme 言語処理系の開発

松崎 泰裕† 並木 美太郎‡
東京農工大学工学部情報コミュニケーション工学科 †/東京農工大学大学院共生科学技術研究部 ‡

1 はじめに

近年の携帯電話は Java VM を搭載しており、プログラマーは携帯電話上で動作するアプリケーションを自由に開発できる。しかし、使用できる言語は Java に限られている。このことは Java 以外のプログラマーにとって携帯電話向けアプリケーション開発のハードルをあげていると言える。また、既存プログラムを携帯電話へ移植する際には言語を越えた移植が必要となり、多大な手間が生じている。

携帯電話で動作する言語処理系には既存のものがいくつか存在し、例えば Scheme の JAKLD[1] が挙げられる。しかし、携帯電話向けではないことや、関数型言語である Scheme に必須である末尾呼び出しの最適化や継続に非対応であるなどの問題がある。

そこで本研究では携帯電話の Java VM 上で動作する Scheme 言語処理系の設計と実装を行った。

2 本研究の目標

本研究では携帯電話の Java VM 上で動作する Scheme 言語処理系を開発し、Scheme プログラマーの携帯電話向けプログラミングを助ける共に、既存の Scheme プログラムの有効活用を目指す。Scheme は R5RS[2] で定義されているものとする。本処理系の核部分は PC でも動作するものとする。

本処理系では、携帯電話の機能を操作するための手続きを提供する。これにより GUI やグラフィック、ネットワークを用いたプログラムを Scheme によってプログラミング可能となる。また、Scheme の特徴である継続をファーストクラスオブジェクトとして扱うことに対応し、携帯電話のモバイル性を活かしたマイグレーションや分散処理などの応用を期待する。

3 本処理系の概要

本処理系の全体構成は図 1 のとおりである。本処理系のパッケージ内にプログラムを組み込んで実行する方式と、携帯電話上で作成したプログラムを実行する方式をサポートする。

プログラムを処理する方式は、携帯電話の Java VM の制限を考慮して、S 式をそのまま辿りながら実行するツリートラバース型インタプリタ方式を採用する。

4 設計

4.1 オブジェクト表現

本処理系では、すべての Java クラスの基底である Object クラスで S 式を表わす。各データを表わすクラスは、図 2 のとおりである。点線で囲まれたクラスは Java の標準クラスであり、これらを多く用いることにより、処理系サイズの低減を期待する。

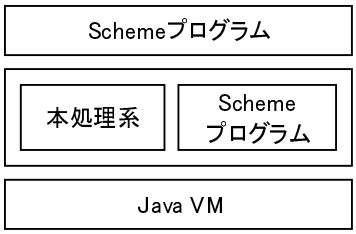

図 1: 本処理系の全体構成

真偽値、文字や数値は対応する Java の各プリミティブ型の配列で保持する。Java には各プリミティブ型を保持するための標準クラスが存在するが、より速度の速い要素数が 1 の配列を採用する。

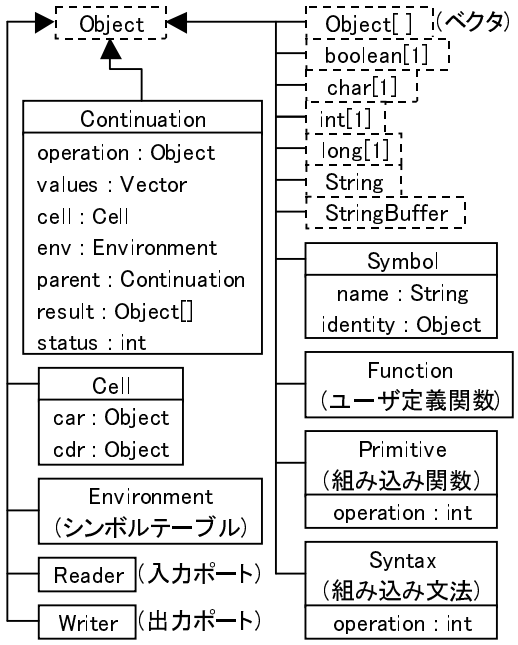

図 2: 各データを表わすクラス

4.2 評価器

Scheme ではコンテキスト情報である継続をファーストクラスオブジェクトとして扱う。これに対応するために、式の部分評価において再帰呼び出しは用いず、コンテキスト情報を Continuation クラスで保持し、このオブジェクトをループで切り替えながら処理を行う。

シンボルテーブルは Environment クラスで保持する。スコープを表わすための親環境への参照と、実際に値を保持するハッシュテーブルを持つ。Scheme ではマクロにより同じ名前で別束縛を持つ場合があるため、ハッシュテーブルの走査は変数名ではなく識別用のオブジェクトで行う。

Development of a Scheme Interpreter for Cellular Phones

† Yasuhiro Matsuzaki
Department of Computer, Information and Communication Sciences, Tokyo University of Agriculture and Technology
‡ Mitaro Namiki
Graduate school of Engineering, Tokyo University of Agriculture and Technology

4.3 携帯電話用 API

携帯電話の機能を操作するための API は、Java で提供されるものと同様のものを提供する。ただし Java と違い Scheme はオブジェクト指向ではないため、作成されたオブジェクトを引数に渡す形で操作を行う。

GUI やネットワーク通信処理における非同期イベントの処理記述に関しては、Scheme における procedure を登録する形とする。Scheme における procedure は組み込み手続きやユーザ定義手続き、継続を含むため、プログラマーはコールバック関数や継続などで処理を記述できる。

プログラム例を図 3 に示す。このプログラムでは、コールバック関数によりキャンバスにアニメーション描画を行いながら、バックグランドでネットワーク通信を行う。通信の終了は継続切り替えにより通知され、切り替え先の継続の処理により終了する。

```
(define cvs (make-Canvas))
(show cvs)
(define g (Canvas-getGraphics cvs))
(define i 0)
(Canvas-setPaint cvs
  (lambda (con cvs g)
    (Graphics-setColor g #xFFFFFF)
    (Graphics-fillRect g 0 0 240 320)
    (Graphics-setColor g #xFF00FF)
    (Graphics-drawString g
      (number->string i) 10 50)))
(define http (Connector-open "http://hoge/"))
(call-with-values
  (lambda ()
    (call/cc (lambda (con)
      (http-connect http con))))
  (lambda (con http in)
    (display "done")))
(let loop ()
  (set! i (+ i 1))
  (Canvas-repaint cvs)
  (sleep 100)
  (loop))
```

図 3: プログラム例

5 実装と評価

本研究では、実数を扱う標準手続きや携帯電話用 API の一部を除いて、継続やマクロへの対応を含めた実装が完了した。図 3 のプログラムも含めて、動作確認できている。

実装を行った結果、処理系の核部分のバイナリサイズは約 40KB となった。API の追加によりサイズは増加するが、一般的な携帯電話のサイズ制限である 100KB[3] の半分以下に抑えることができた。また JAKLD の約 40KB と比較しても、継続などの機能を追加した上で同程度のサイズを実現できた。

携帯電話の実機として DoCoMo SH902iS を用い、実行時のメモリ消費量を測定した。起動時におけるメモリ使用量は、約 200KB となった。また、car 部に空リストを格納した長さ 1000 のリストを作成した時の消費メモリは、約 20KB となった。これは、数 MB〜10MB[4] のヒープメモリを搭載している最近の携帯電話において問題ない消費量である。

次に携帯電話と PC 上で実行速度を測定した。比較のために、Java で記述された他の 2 つの Scheme 処理系においても測定を行った。結果は表 1 と表 2 のとおりとなった。ループは 1 ループあたりの時間である。携帯電話向けとして問題ない速度を確保できている。また PC において一部のコードでは、他の処理系よりも速い結果が得られた。

表 1: 携帯電話 (SH902iS) 上での実行速度

コード	本処理系	JAKLD
(tak 18 12 6)	247[s]	25.2[s]
単純ループ	1.8[ms]	-
関数付ループ	2.3[ms]	-
継続ループ	2.2[ms]	-

表 2: PC(Pentium4-3GHz) 上での実行速度

コード	本処理系	JAKLD	SISC[5]
(fib 25)	890[ms]	1.24[s]	1.64[s]
(tak 18 12 6)	352[s]	367[ms]	79[ms]
単純ループ	3.04[μ s]	-	0.97[μ s]
関数付ループ	3.97[μ s]	-	1.52[μ s]
継続ループ	3.72[μ s]	-	4.50[μ s]

6 おわりに

本論文では、携帯電話向け Scheme 言語処理系の設計と実装について述べた。本処理系により携帯電話上での Scheme プログラムの実行が可能となった。

今後の課題は、携帯電話上で Scheme プログラムを入力するための実用的なエディタの開発や、Scheme 言語の特徴である継続を活用したマイグレーションや分散処理などの応用システムの開発である。

参考文献

[1] 湯浅太一: Java アプリケーション組み込み用の Lisp ドライバ, 情報処理学会 PRO 研究会, 2002. http://www.yuasa.kuis.kyoto-u.ac.jp/~yuasa/jakld/

[2] Richard Kelsey, William Clinger and Jonathan Rees, editors: The revised5 report on the algorithmic language Scheme. In Higher-Order and Symbolic Computation 11(1), pages 7-105, 1998.

[3] NTT DoCoMo: i アプリコンテンツ開発ガイド for DoJa-4.x (2006)

[4] NTT DoCoMo: i アプリ対応端末の情報 http://www.nttdocomo.co.jp/service/imode/make/content/spec/iappli/

[5] SISC: http://sisc.sourceforge.net/