解説

京大におけるLispを使ったプログラミング教育

湯淺太一　奥乃　博　尾形哲也
京都大学大学院情報学研究科

はじめに

Lispが世に現れたのは1950年代後半であり，現在も使われているプログラミング言語の中では，Fortranに次いで古い言語である．ハードウェアや実装技術の進歩，プログラミングパラダイムや応用分野からの要望など，さまざまな要因によって，数多くの方言が設計・実装されてきた．1980年代の人工知能ブームを契機に，いくつかの標準規格が作られている[8),9)]．当時と比べると，利用者の数こそ大幅に減少しているが，他に例を見ない強力な言語機能を備えているために，特定の応用分野では現在も実用に供されている．

京都大学工学部情報学科では，2004年度からLispを使ったプログラミング教育を行っている．講義科目としては，1年生後期の「アルゴリズムとデータ構造入門」（以下「データ構造」と略す）と2年生前期の「プログラミング言語」（以下「言語」と略す）があり，実験演習科目としては，3年生後期の「計算機科学実験及演習4」（以下「演習4」と略す）がある[4)]．本稿では，Lispの教育への利用を，これらの科目を実例として紹介する．

SICP

「データ構造」と「言語」は，教科書としてStructure and Interpretation of Computer Programs（略称SICP）[1)]を使ってプログラミングの講義を行っている．SICPは，基礎的なものから高水準なものまで，プログラミングのさまざまな概念をカバーしており，国内外の多くの大学でプログラミングの教科書として採用された実績を持っている．原著は英文であるが，邦訳もあり，また英文のフルテキストを出版社のWebページから無料でダウンロードすることもできる．例題や演習に使うプログラミング言語はLisp（正確には，Scheme[8)]）である．

SICPは次の5つの章から構成されている．

1. 手続きによる抽象化
2. データによる抽象化
3. モジュール化，オブジェクト，状態
4. 超言語による抽象化
5. レジスタ・マシンによる計算

このうち，「データ構造」が第1章と第2章を，「言語」が第3章と第4章をそれぞれ担当している．時間の都合もあるが，第5章の内容は2年生後期の「コンパイラ」の講義がある程度カバーしているので，「言語」の講義では扱っていない．

「データ構造」も「言語」も，毎週あるいは隔週で演習課題を出しており，受講生は実際にLisp処理系を使って課題を解く．講義には演習時間を設けていないので，受講生は各自の都合に合わせて計算機環境や処理系を選択する．演習結果はレポートとして提出し，TAが採点して返却する．一部の演習課題について講義で解説を行ったり，採点過程で気がついた事項を講評することもある．

レポートの提出は，原則としてメールで行う．メールに添付あるいは貼り付けられたプログラムは直ちに実行できるので，提出されたプログラムが

正しく動作するかどうかを確認できるからである．しかし，レポート課題の中には，データ構造や関数閉包内の変数環境などを図示するものもある．描画ツールできれいに描いたり，手書きの図をスキャナーで撮ってメイルに添付する学生が多いが，図示の課題がある場合は手渡しでもよいことにしている．スキャナーで撮るとサイズが大き過ぎてメイルに添付できないことがあるからである．

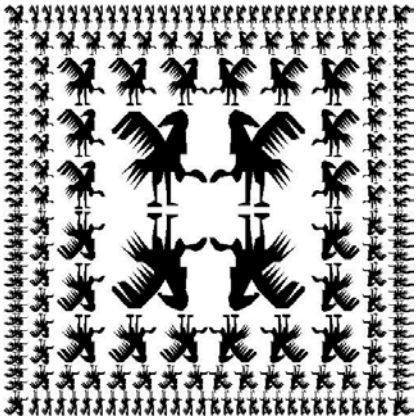
図 -1　図形言語による再帰的な描画例

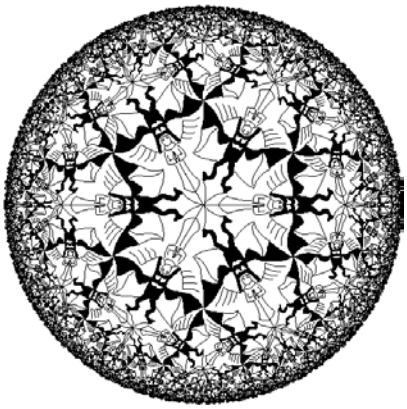
図 -2　キャンバスを円形にする

SICP は Lisp 言語を教えるための教科書ではない．新しい概念が出てきたときに，必要に応じて Lisp の機能を紹介する形式をとっている（たとえば，代入を行う `set!` は，第 3 章で代入の概念といっしょに紹介される）．だから，先頭から順に読み進めば自然と Lisp の言語も理解できるように工夫されている．しかし，講義を始めるにあたって，Lisp の全体像を受講生がある程度把握しておけるように，Lisp の簡単な入門講義を行っており，後述する演習用処理系の使用法をついでに説明している．

「データ構造」では，第 2 章までの知識を基礎にして，講義名にあるデータ構造とアルゴリズムに関する講義を行っている．また，学生のレベルに応じて，advanced な課題を出すこともある．SICP の第 2 章では，再帰的計算を画像生成に応用するための図形言語（**図 -1** 参照）が登場するが，描画キャンバスは正方形に限定されている．これを円形にする（**図 -2** 参照）とどうなるか，といった難題もある [3]．

ロボットプログラミング

「演習 4」では 6 件のテーマが用意されており，受講生はその中から 2 つを選択し，学期の前半と後半に 1 つずつ取り組む．テーマの 1 つである「ロボットプログラミング」[2] では，実ロボットのプログラミングを通じて，実世界と繋がったプログラミングを体験する．実ロボットとして，Lego Mindstorms[6] を使っている．ブロック玩具で知られる Lego 社が販売しているロボットキットで，8bit CPU を搭載した RCX というコントローラ・ブロックに，モータや各種センサのブロックを組み合わせることによって，ロボットを自作できる（**図 -3** 参照）．

受講生には，次のような課題がいくつか与えられる．

【演習○】

光センサを 1 つだけ使用して黒い線に沿って動くロボットを作成せよ．さらに，黒い線のコース上に障害物を置いたときに，それをタッチセンサで検出し，回避してコースに復帰する機能を付加せよ．

課題の中には，やや難易度の高い次のような「自由課題」も含まれている．

【自由課題○】

迷路課題（省略）において，ロボットがゴールに到着した後，スタート地点まで“効率的に”帰還するプログラムを，リスト操作を使用して作成せよ．

演習に使用する言語は，NQC[7] と XS Lisp[11] である．NQC（Not Quite C）は，C 言語の機能を大胆に削り落とし，ロボット部品との API を加え，マルチスレッド機能を追加したものである．これによって，RCX の 32Kbyte という極小のメモリ空間でも，C 言語風のロボット制御プログラムを動作させることができる．NQC で書かれたプログラムは，フロントエンドの PC でクロス・コンパイルされ，バイナ

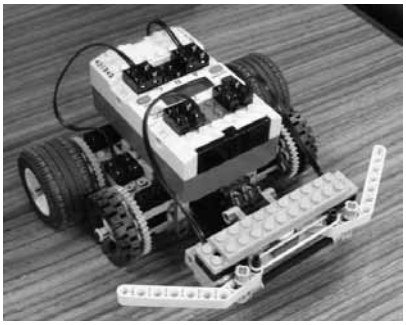

図 -3　移動ロボットの例

リを RCX に送り込んで実行される．

もう一方の XS Lisp は，筆者の一人が Mindstorms 用に開発した超小型の Lisp 処理系である（“XS” は，eXtra Small の意）．RCX の中で Lisp インタプリタが動作し，フロントエンド PC から Lisp の式を 1 つ受け取っては解釈実行する．結果はフロントエンドのディスプレイに表示され，あたかも通常の Lisp 処理系を使っているような感覚でプログラムを開発することができる．言語仕様は，文献 8)の仕様をベースに，大幅に機能削減し，ロボット部品を制御する関数を追加している．NQC よりもはるかにメモリの制約が厳しいために，マルチスレッドではなく，非同期イベントのための割り込み処理構文 `with-handler` を備えている．

(with-handler ((*event*$_1$. *handler*$_1$))
　　...
　　((*event*$_n$. *handler*$_n$))
　. *body*)

この式を実行すると，基本的に本体 *body* が実行されるが，その間に各条件式 *event*$_i$ が定期的に評価され，その値が真になると本体の実行に割り込んで，対応する処理 *handler*$_i$ を実行する．処理が終わると，再度本体の実行に戻る．

演習では，それぞれの課題について，NQC と XS の両方でプログラムを書く．静的な手続き型言語と動的な関数型言語を同時に使用することによって，これらの相違を体得することが重要な狙いである．また，これらの言語で書いたプログラムを実際のロボットで動作させることによって，アルゴリズム以外に，時間やインタラクションの概念を扱うことができるマルチスレッドや割込みといった動的制御テクニックを学ぶことが期待されている．

JAKLD

講義科目の「データ構造」も「言語」も，JAKLD[10] という Lisp 処理系を使って演習問題を解くことを奨励している．

SICP による講義を始めた当初は，自家製の処理系を改造して，描画機能として Tcl/Tk の Tk 機能を追加した TUTScheme/Tk[5] を用意し，これを計算機センターの PC にインストールした．しかしセンターの利用時間には制限があり，多くの学生は自分の PC に TUTScheme/Tk をインストールしようとした．TUTScheme/Tk は C 言語で記述されており，もともと Unix 系 OS の上で動作していた．これを受講生たちが自分の Windows PC で動かすのは容易ではなく，本来の演習に専念できない状態が続いていた．

このような状況で数年間講義を行ってきて，その間に明らかになったさまざまな不都合を解消し，受講生が演習問題を解くことに専念できるような環境を提供するために，Java で記述した JAKLD に手を加え，演習に必要な諸機能を追加した．演習用の PC にどんな OS が動いていようと，Java VM さえあれば，jar ファイルをダウンロードするだけで直ちに起動することができる．携帯電話で使えるバージョンもある．

JAKLD に追加した機能としては，図形言語はもちろんのこと，SICP の第 3 章で使われる並列処理機能（実際はマルチスレッド機能）や，第 4 章で必要となる遅延評価機構も含まれる．これらはすべて Java の標準的なライブラリを利用して実装されている．また，処理系依存の振る舞いは，できるだけ SICP 本文の例題と一致するよう調整し，学生が違和感なく演習できる環境を提供している．

SICP の第 4 章「超言語による抽象化」には，いくつかのプログラミングパラダイムを実現する metacircular なインタプリタが登場する．これら

のインタプリタはLispで記述され，パラダイムを採用したLisp風言語のプログラムを解釈実行する．実行エンジンはインタプリタごとに異なるが，基本的なLisp関数は，インタプリタが動作する処理系のものを流用する．流用するためのコードは，処理系が提供する関数群に依存する．演習用の処理系をJAKLDに統一することによって，各インタプリタのコードがすべての演習用PCで共通に使えることになった．

ちなみに，昨年度の「データ構造」の受講生の1人が，「自由課題」としてJAKLDをAndroidスマートフォンでも使えるようにした．AndroSchemeと名付けており，Androidマーケットから無償で入手可能である．

おわりに

「Lispによる」教育についての原稿を依頼されたということは，Lispでプログラミング教育を行うのは珍しいのであろう．C言語やJavaといったポピュラーなプログラミング言語を使うのが一般的かもしれない．限られた誌面で，京都大学で行っているLispを使った教育を簡単に紹介してきたが，その範囲で「なぜLispなの？」に答えたいと思う．

言うまでもなく，これらの講義・演習は，Lispプログラマを養成するためのものではない．Lispという具体的な言語を通して，プログラミングを学習するためのものである．ここで，プログラミングというのは，単に与えられた問題をコーディングする技術ではなく，計算機科学の成果であるさまざまな概念や機能を駆使できる能力である．この意味で，講義で採用しているSICPは，きわめて適切な教科書である．プログラミング言語の学習にあまり時間をかけずに，抽象化，metacircular，遅延評価，並列実行，制約プログラミングといった高度な概念や言語機能を学習することができる．SICPは，これらを単に概念として紹介するのではなく，Lispでどのように実現されているかを具体的に解説している．これによって，学生たちは，次々と紹介される概念を，より深く学習することが期待される．

レポートの演習課題を解くために対話的なLisp処理系を利用できる点も，学習効率を高めている．SICPの最初の演習課題（Exercise 1.1）は，(+ 5 3 4)などのLisp式の値を実際の処理系を使って求めよ，というものである．学生たちは処理系を起動して，“(+ 5 3 4)”と打ち込むだけでよい．同じことをC言語やJavaで行えば，初心者の学生にとって結果を得るまでにどれほどの時間がかかるだろうか．

演習に使っているXS Lispについても，同様の学習効率をねらっている．Lisp処理系に共通する対話機能を利用して，関数を定義しては直ちに（コンパイラを呼び出すことなく）実行して動作確認する，といったプログラム開発方法が使える．さらに，マニュアルに書いてあるモータやセンサのためのAPIも，式を入力してテストすれば，簡単に動作が確認できる．

参考文献

1) Abelson, H. and Sussman, G. J. : Structure and Interpretation of Computer Programs, 2nd Edition, http://mitpress.mit.edu/sicp/, MIT Press, Cambridge, MA (1996).
2) 尾形哲也：計算機科学実験及演習4—ロボットプログラミング，http://winnie.kuis.kyoto-u.ac.jp/~ogata/le4-robot/wiki/
3) 奥乃　博：「アルゴリズムとデータ構造入門」講義情報，http://winnie.kuis.kyoto-u.ac.jp/~okuno/Lecture/10/IntroAlgDs/
4) 京都大学工学部シラバス，http://www.t.kyoto-u.ac.jp/syllabus-s/
5) 小宮常康：TUTScheme, TUTScheme/Tkの処理系，http://www.spa.is.uec.ac.jp/~komiya/download/
6) LEGO.com Mindstorms: Home, http://mindstorms.lego.com/
7) Not Quite C, http://bricxcc.sourceforge.net/nqc/
8) Revised(4) Report on the Algorithmic Language Scheme, http://people.csail.mit.edu/jaffer/r4rs_toc.html
9) Steele, G. L. Jr. : Common Lisp the Language, Second Edition, http://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html
10) 湯淺太一：Javaアプリケーション組込み用のLispドライバ，http://www.yuasa.kuis.kyoto-u.ac.jp/~yuasa/jakld/index-j.html
11) 湯淺太一：XS: Lego MindStorms用Lisp処理系，http://www.xslisp.com/index-j.html

(2011年5月31日受付)

湯淺太一（正会員） yuasa@i.kyoto-u.ac.jp
1977年京都大学理学部卒業．同大数理解析研究所，豊橋技術科学大学を経て，現職．プログラミング言語処理系の研究・開発に従事．

奥乃　博（正会員） okuno@i.kyoto-u.ac.jp
1972年東京大学教養学部基礎科学科卒業，博士（工学）．NTT，JST，東京理科大学を経て，現職．プログラミング言語，人工知能，音環境理解の研究に従事．

尾形哲也（正会員） ogata@i.kyoto-u.ac.jp
1993年早稲田大学理工学部卒業．学振特別研究員，理化学研究所を経て，2005年より京都大学情報学研究科准教授．さきがけ「情報環境と人」研究員．神経回路モデルと人間ロボットのコミュニケーションの発達研究に従事．