ISLISPコンパイラの実装

4L−6

西村祥治，湯淺太一，八杉昌宏，小宮常康
京都大学大学院情報学研究科
通信情報システム専攻

1. はじめに

ISLISPは1997年5月にISO/IEC13816[1]、1998年7月にJIS X 3012[2]として制定されたLisp言語の国際規格である．その言語仕様はCommon Lisp[3],[4]のサブセット＋オブジェクト指向機能＋例外処理機能＋αとなっている．ISLISPの処理系としてKCL(Kyoto Common Lisp)[5]上で実装されたISLISPインタプリタISL/CL[6],[7],[8]があり，本研究ではこのISL/CL上で動作するISLISPコンパイラを実装した．このコンパイラの実装はKCL上で動作するCommon Lispコンパイラである KCLコンパイラを改造することで行った．

本稿ではまずISLISPの仕様で特徴的なところをCommon Lispの言語仕様と対比させながら取り上げる．次にISLISPコンパイラの構成について述べ，Common LispとISLISPの言語仕様上の類似点，相違点をうまく利用し，実装する方法について述べる．

2. ISLISPの仕様

Common Lispとの主な相違は次の通り．

変数と動的変数　Common Lispではglobal変数とlexical変数は静的束縛，special変数は動的束縛され，これらはシンタックスとしてはどちらも同じ形であり，同じ名前空間に属する．しかし，ISLISPでは，静的束縛される変数と動的束縛される変数がシンタックスとしても分離される．前者を変数，後者を動的変数と呼び，その識別子は別々の名前空間を持つ．例えば，`var`とすると`var`という識別子を持つ変数の値が参照され，`(dynamic var)`とすると`var`という識別子を持つ動的変数の値が参照される．また，変数と動的変数を束縛する式もシンタックスとして分離される．定義形式はそれぞれ`defglobal`式，`defdynamic`式を用い，一時的に束縛するにはそれぞれ`let`式，`dynamic-let`式を用いる．

ラムダクロージャ　Common Lispでは`function`式の引数にラムダ式を与えることでラムダクロージャをつくる．ISLISPではラムダ式のみでつくる．ISLISPでは，`function`式は関数の名前空間の識別子を引数に取り，その識別子に束縛されている関数への参照を表す．ただし，本コンパイラではCommon Lispと同じように`function`式を用いてラムダクロージャがつくれるように拡張している．これによって，本コンパイラ自身の実装が容易になった．

ラムダリスト　ラムダリストでは必須パラメタと残余パラメタのみ指定できる．Common Lispにあったオプショナルパラメタ，キーワードパラメタ，補助変数などは指定できない．

その他　変数の定義や`let`変数リストで初期値を明示的に指定しなければならない．Common Lispの場合は省略すると`nil`が初期値となる．

3. ISLISPコンパイラの実装

ISLISPコンパイラの元となるKCLコンパイラは2パスコンパイラである．第1パスではファイルなどからプログラムを読み込み，プログラム解析を行い，内部コードを生成する．第2パスではこの内部コードからCのソースを生成し，外部のCコンパイラを起動してオブジェクトファイルを生成する．ISLISPはCommon Lispの言語仕様の部分集合に近いということと元になるKCLコンパイラが2パスコンパイラであることを利用して，改造箇所ができるだけ少なくするため，できる限り第1パスでその差を吸収する．

Implementation of an ISLISP compiler
Shoji Nishimura, Taiichi Yuasa, Masahiro Yasugi, Tsuneyasu Komiya
Department of Communications and Computer Engineering,
Graduate School of Informatics, Kyoto University
shoji-n@kuis.kyoto-u.ac.jp

コンパイラ記述言語の置換　KCL コンパイラは Common Lisp で記述されているので ISLISP インタプリタ上で動くようにするために，記述言語を ISLISP に置換する．ラムダリストの書き直し，変数の初期値を明示的に指定，special 変数を動的変数に置換，動的束縛を行っている `let` 式を `dynamic-let` 式へ置換，組み込み関数名や引数の順序の置換などを行い，処理内容は変更しないようにする．

構文規則の相違　ISLISP の構文規則に合うようにプログラム解析部を改造する．第 1 パスで生成する内部コードは KCL コンパイラのものを流用するので第 2 パスのコードはそのまま利用できる．

変数と動的変数の解析　KCL コンパイラのプログラム解析では変数を global 変数，lexical 変数，special 変数に区別し，それらに対する内部コードを生成する．ISLISP コンパイラでは最上位有効範囲を持つ変数を global 変数に，局所的に束縛される変数を lexical 変数に，動的変数を special 変数に対応させ，内部コードもこれに合わせる．ISLISP では変数と動的変数の構文規則は異なるので別々にプログラム解析を行う．変数のプログラム解析は KCL コンパイラのものを流用し，動的変数の解析は `dynamic` 式のプログラム解析を新たにつくり，そこで行うようにする．また，KCL インタプリタ内部では global 変数，special 変数の値は記号の special 変数スロットの値として，局所変数はスタック上の値として表現する．しかし，このままだと最上位有効範囲を持つ変数と動的変数の名前空間が同一になってしまい都合が悪いのでインタプリタに次のような改造を施す．変数スロットを追加し，最上位有効範囲を持つ変数の値をそこへ格納し，動的変数を special 変数スロットへ格納する．変数，動的変数の値が適切なスロットへ格納されるように第 2 パスの変数，動的変数の代入，参照のコード生成部を改造する．

新しい構文規則　変数，動的変数を定義する `defglobal` 式，`defdynamic` 式が追加されたので Common Lisp の `defvar` 式を改造する．また，動的変数を局所的に束縛する `dynamic-let` 式が追加されたのでそのプログラム解析を追加する．`dynamic-let` 式のセマンティクスは Common Lisp で special 変数を `let` 式で束縛するものと同じなので，生成する内部コードはこの時と同じとする．このため，第 2 パスのコード生成部を新たにつくる必要がない．

4. まとめ

本稿は ISLISP と Common Lisp の言語仕様の相違点に着目し，Common Lisp コンパイラである KCL コンパイラを改造することによって ISLISP コンパイラを実装した．KCL コンパイラの構造をうまく利用し，できる限り改造箇所を少なくするための工夫も示した．

参考文献

[1] ISO: *ISO/IEC 13816, Information technology – Programming languages, their environments and system software interfaces – Progamming Language ISLISP* (1997).

[2] 日本工業標準調査会: プログラム言語 ISLISP(JIS X 3012) (1998).

[3] *Common Lisp the Language, 2nd Edition*,
http://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html.

[4] 湯淺太一, 萩谷昌己: Common Lisp 入門, 岩波書店 (1986).

[5] Taiichi Yuasa: *Design and Implementation of Kyoto Common Lisp*,
Journal of Information Processing, vol.13, no.3, pp.284-295 (1990)

[6] 須藤竜也: 特別研究報告書: ISLISP の Common Lisp における実装 (1998).

[7] 早川祐志: 特別研究報告書: ISLISP オブジェクトシステムの Common Lisp における実装 (1998).

[8] 大西貴子: 特別研究報告書: KCL における ISLISP オブジェクトシステムの高速化 (1999).