1J-03

LLDBを用いたソース-to-C型トランスレータ用ソース言語レベルデバッガの実装手法*

柳震†
電気通信大学大学院情報システム学研究科†

小宮常康‡
電気通信大学大学院情報理工学研究科‡

1 はじめに

ソース-to-ソース型トランスレータは高級言語を別の高級言語へ変換する言語処理系であり，C言語へ変換する場合が多い．トランスレータの開発者はトランスレータのソース言語用デバッガを用意することが望まれる．ターゲット言語のデバッガでデバッグすることもできるが，ソース言語レベルでデバッグできることが望ましい．

そこで，ソース言語レベルのデバッグコマンド引数を，対応するC言語のデバッグコマンド引数へ変換することで，C言語デバッガを用いたソース言語レベルデバッガを実現する方法を提案する．本研究ではC言語のデバッガとしてLLDBを用いた．本稿ではScheme言語をC言語へ変換するHobbitコンパイラ[1]で生成されるコードを例として，研究手法を説明する．

2 基本アイデア

C言語のデバッガを用いてトランスレータのソース言語レベルデバッガを実現するために，ソース言語レベルのデバッグコマンドと引数が入力されると，それをC言語レベルのデバッグコマンドと引数へ自動的に変換するようにする．そのために，以下の手法が必要となる：

- ソース言語のコード片とC言語のコード片の対応関係を表す手法
- ソース言語レベルの引数を受け取り，上記の対応関係を用いて，C言語レベルの引数に変換する手法

本研究では，ソース言語のコード片として関数名と変数名のみを扱う．この他にコードの行番号の対応関係などが考えられるが本研究では扱わない．

トランスレータのソース言語とターゲット言語であるC言語の間で識別子を表す規則に違いがある場合，識別子の変換が行われる．さらにソース言語や処理系の実装によっては，ソース言語の局所変数をターゲット言語の局所変数に変換するのではなく，実行時環境をアクセスするC言語コードへ変換することもある．

ソース言語がScheme言語の場合，大域変数はC言語の大域変数として実現され，識別子の規則の違いだけを考えれば十分であることが多い，Hobbitコンパイラの場合はこれに当てはまる．

一方，Scheme言語の局所変数については，関数閉包に閉じ込められる場合，set!操作(代入)される場合などで，C言語への変換方法は異なり，C言語の変数ではなく，より複雑なデータ構造や制御構造を含むC言語コードに変換される．

具体例を示す．ソースコード1のSchemeプログラムでは，変数xが関数閉包に閉じ込められて，set!操作される．これをHobbitコンパイラでC言語へ変換すると，ソースコード2のようになる．

ソースコード 1: Scheme プログラム

```
1 (define (a x)
2   (lambda (y)
3     (set! x (+ x y))
4     x))
```

局所変数がset!操作され得る場合，変数の値をヒープに保存する．4〜5行目のように，Hobbitコンパイラはset!操作され得る変数xの値をベクタで保存して，clargsv_1という変数で保持する．

局所変数が関数閉包に閉じ込められる場合，Hobbitコンパイラは関数閉包オブジェクト内に変数の値を保存する．7行目のように，先ほどのxを保存するベクタは，同じくベクタで作られる関数閉包オブジェクト(newclosure)に格納される．

ソースコード 2: Hobbit による生成コード

```
1  SCM a(x)
2  SCM x;{
3    SCM clargsv_1,newclosure;
4    clargsv_1=make_vector(MAKINUM(1),EOL);
5    VECTOR_SET(clargsv_1,MAKINUM(0),x);
6    newclosure=makcclo(a_cl1_clproc0,2);
7    VECTOR_SET(newclosure,MAKINUM(1),clargsv_1);
8    return newclosure;
9  }
10 SCM a_cl1(closurearg_0)
11 SCM closurearg_0;{
12   SCM closurearg_car_0,clargsv_1,y;
13   . . .
14   return VECTOR_REF(clargsv_1,MAKINUM(0));
15 }
```

そこで，例えば変数xに対するデバッグコマンドの適用はC言語のコードVECTOR_REF(clargsv_1,MAKINUM(0))に対して行うようにする．

*An Implementation Method of Source-Level Debuggers for Source-to-C Translators by Using LLDB

†Liu Zhen
Graduate School of Information Systems
The University of Electro-Communications
‡Tsuneyasu Komiya
Graduate School of Informatics and Engineering
The University of Electro-Communications

3 実装

本研究の実装は二つの部分からなる．一つは Hobbit コンパイラに対する改造であり，もう一つは LLDB に埋め込まれた Python 上のスクリプトによって LLDB を拡張することである．

3.1 トランスレータの改造

2 節で述べた対応関係情報は，トランスレータに対する改造を通じて，生成される C 言語プログラムに二つのリストとして埋め込むようにした．一つはソース言語の変数名または関数名を記録して，もう一つは対応する C 言語による変数の実装コードまたは関数名を記録する．

ソースコード 1 の Scheme プログラムを改造された Hobbit コンパイラで変換した結果をソースコード 3 に示す．1〜2 行目，5〜6 行目と 11〜12 行目が改造されたトランスレータによって埋め込まれた対応関係情報である．

ソースコード 3: 改造された Hobbit による生成コード

```
1  char* listCfun[]={"a","a_cl1"};
2  char* listSchemefun[]={"a","a-cl1"};
3  SCM a(x)
4  SCM x;{
5    static char* listScheme[]={"x"};
6    static char* listC[]={"x"};
7    . . .
8  }
9  SCM a_cl1(closurearg_0)
10 SCM closurearg_0;{
11   static char* listScheme[]={"x","y"};
12   static char* listC[]={"-0","y"};
13   . . .
14 }
```

大域スコープのリストは関数と大域変数の対応関係を表す．listSchemefun は Scheme 言語の関数名を記録する．ソースコード 1 の関数閉包は Hobbit コンパイラによって，a_cl1 という名前がつけられた．本研究では関数 a の中の一番目の関数閉包を意味する a-cl1 という名前をソース言語レベルの名前として記録して，Scheme レベルでは名前のついていない関数閉包に対してもアクセスできるようにした．

C 言語の関数内で宣言されたリストは局所変数の対応関係を表す．x に対する“-0”の“-”は clargsv_1 に保存されることを表し，“0”はそのインデックス 0 の位置に保存されることを表す．

3.2 LLDB の拡張

本研究では LLDB に埋め込まれた Python 上のスクリプトを通じて，二つのリストにアクセスして，ソース言語レベルの引数に対応する C 言語レベルの引数を得る．そして，デバッグコマンドとその引数を Python に用意された LLDB の API を用いて LLDB で実行する [2].

ソース言語の変数の値を表示するデバッグコマンド myprint を例として，ソースコード 1 の Scheme プログラムに対するデバッグのステップを図 1 に示す：

1. ユーザが入力するソース言語レベルのコマンド myprint と引数 x を受け取る．
2. Python 上のスクリプトは myprint を C 言語レベルのデバッグコマンド print に変換する．listC から x と対応する引数-0 を取り出して，x の値は clargsv_1 のインデックス 0 に置かれていることを知る．そして，VECTOR_REF(clargsv_1,MAKINUM(0)) という引数を作る．
3. “print VECTOR_REF(...)”を LLDB で実行する．
4. x の値がプリントされる．

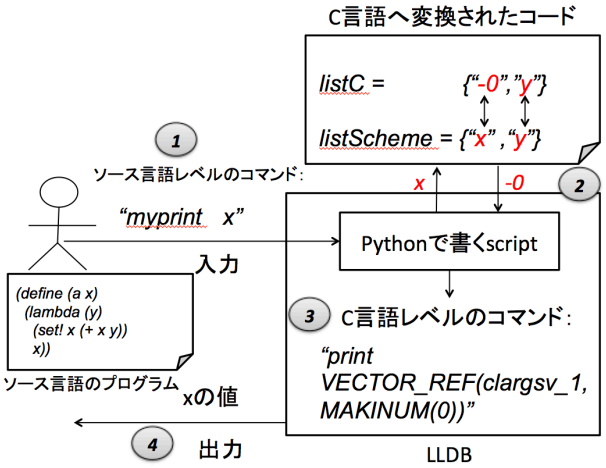

図 1: デバッグコマンドの引数の変換の流れ

4 まとめ

本稿では，C 言語デバッガを用いてトランスレータのソース言語レベルデバッガを実現する方法を提案した．この提案手法では，トランスレータの改造を通じてソース言語と C 言語の対応関係を表すリストを埋め込み，LLDB に埋め込まれた Python 上のスクリプトでそのリストをアクセスして，ソース言語レベルのデバッグコマンドと引数を C 言語レベルへ変換する．

本研究では Hobbit コンパイラを対象としたが，Hobbit 以外にも適用できるはずであり，将来は共通 API を用意することで，様々なトランスレータへ少ない手間で対応することを考えている．

参考文献

[1] The SCM Implementation of Scheme, http://people. csail. mit. edu/jaffer/SCM (2017 年 1 月 11 日参照)

[2] LLDB Python Reference, http://lldb. llvm. org/python-reference. html (2017 年 1 月 10 日参照)