言語変換系を用いたソフトウェア開発におけるソースレベルデバッグ支援

情報科学科　西村 将広　　指導教員：山本 晋一郎

1 はじめに

より簡潔に記述できるプログラミング言語は，コードの見通しを良くしソフトウェア開発のコストを軽減する．しかし，実行環境の制約によりプログラミング言語の選択肢が狭まることがある．このような場合には，あるプログラミング言語で記述されたコードを別のプログラミング言語のコードへ変換する言語変換系の適用が考えられる．以降，変換前のプログラムを S プログラム，変換後のプログラムを T プログラムと呼ぶ．

言語変換系を用いる場合，S プログラムで開発を行い，T プログラムに変換してそれを実行する．そのため，プログラム中の変数の値などの実行時に分かる情報（以降，実行時情報）は T プログラム上の情報であり，S プログラムのどの部分に対応しているかを特定できない．さらに，T プログラムは S プログラムと比較して低レベルなため可読性が低い場合が多い．故に，言語変換系を用いたソフトウェア開発ではデバッグが困難である．

本研究は，言語変換系を用いたソフトウェア開発において，T プログラムの実行時情報を擬似的に S プログラムの実行時情報として表現することにより，S プログラムレベルのデバッグ（以降，ソースレベルデバッグ）が可能な環境を構築し，デバッグに要する負担の軽減を目指す．

2 言語変換系

言語変換系とは，あるプログラミング言語を別のプログラミング言語へ変換するシステムである．例えば，JavaScript へ変換される CoffeeScript では，第 2 引数の配列の要素それぞれに第 1 引数の関数を適用する関数定義は次のように記述できる．

CoffeeScript

```
map = (f, xs) -> (f x for x in xs)
```

次に上に示したプログラムと同じ計算を行う関数定義を JavaScript で直接記述したプログラムを示す．

直接記述した JavaScript

```
map = function(f, xs) {
  var res = [];
  xs.forEach(function(x) { res.push(f(x)); });
  return res;
};
```

言語変換系を用いる場合，変数宣言 (var) などのプログラムにおいて本質でない部分の記述が不要となり，簡潔で見通しの良いソースコードを記述できる．

3 提案手法

ソースレベルデバッグを可能にするため，S プログラムと T プログラムの対応関係を示すテーブルを作成する．そのテーブルを用いて T プログラムの実行時情報から S プログラムの擬似的な実行時情報を生成する．

まず言語変換系の処理中に S プログラム中に現れる関数名や変数名などのシンボル情報を取得する．得られたシンボル情報を元に T プログラムを静的解析し，S プログラムと T プログラム間の対応テーブルを作成する．

T プログラムの実行時情報を取得するために T プログラムを動的解析行し，この解析結果と対応テーブルを用いて擬似的に S プログラムの実行時情報として表現する．

4 実装と評価

本研究では Haskell を JavaScript へ変換する言語変換系 Fay[1] を対象とし，Web 開発ツールである Firebug を拡張することで提案環境の実装を行う．Fay において Haskell プログラム中に現れるシンボルはある一定の規則で変換されるため，シンボル情報を元に両プログラムの行番号の対応を表すテーブルを作成した．このテーブルを用いて，実行中の JavaScript プログラムの行番号から対応する Haskell プログラムの行番号を求め，その時点で JavaScript プログラムが持つシンボルの値を Haskell プログラムのシンボルの値として表現することで，擬似的に Haskell プログラムの実行時情報を生成した．この提案環境により以下のことが可能となる．

- ソースレベルのブレークポイント設置可能
  Haskell プログラムの行にブレークポイントを設置でき，ブレーク時に実行中の関数が持つ引数などの変数の値や戻り値の監視が可能となる．
- Exeption 発生時の Haskell プログラムの行を検出可能

これらの機能により提案環境はソースレベルデバッグを可能にすると考える．

また，本研究の関連技術として Source Maps[2] が挙げられる．本研究は，S プログラムと T プログラムを行単位で対応付ける点で類似しているが，T プログラムの実行時情報を用いて擬似的に S プログラムの実行時情報を表現している点で Souce Maps と異なる．本研究の評価は Source Maps との比較により行う．Souce Maps は，変数の値を常に評価する先行評価の言語から JavaScript へ変換する言語変換系を対象としている．そのためプログラム間の行の対応関係を示すだけで，S プログラム中のシンボルの値が比較的容易に判断できる．しかし，値が必要になるまで評価しない遅延評価を持つ言語からの変換では，プログラム実行時に未評価の値が生じる場合があることから，行の対応を示すだけでは S プログラム中のシンボルの値が判断できない．提案環境では，動的解析を用いることにより未評価の値を含む場合においてもソースレベルデバッグの支援が可能である．本研究の提案手法は，より多くの言語変換系におけるソースレベルデバッグを支援すると考えられる．

5 おわりに

本研究はソースレベルデバッグ支援のための手法を提案し，具体例として Haskell を JavaScript へ変換する言語変換系 Fay を対象に提案環境を構築した．この提案環境では，シンボル情報を用いて Haskell プログラムと JavaScript プログラムを行番号で対応付け，JavaScript プログラムの実行時情報から擬似的に Haskell プログラムの実行時情報を生成することで，ソースレベルデバッグを可能にした．

また本支援環境において，Haskell プログラムの擬似的な実行時情報に JavaScript の表現がそのまま埋め込まれることがある．今後の課題として，そのような JavaScript の表現を Haskell の表現へ変換できるよう改良する．

参考文献

[1] Fay, https://github.com/faylang/fay/wiki
[2] Source Maps,
http://www.html5rocks.com/en/tutorials/
developertools/sourcemaps/