組込み用プログラミング言語「Yadorigi」の開発

1. 背景

世の中の多くのプログラマは、主に手続き型言語を利用している。多くの手続き型言語では、大半の問題を簡潔な手段で解決するために十分な機構が用意されているが、特定の問題においては上手くいかない事がある。このプログラムの記述コストに関する問題は、関数型言語を利用する事で解決される事が多い。関数型言語では関数に対する操作の制約が普通のデータと同じであるため、高階関数やラムダ式を利用でき、抽象化の道具が増えるというのがその大きな理由である。しかし、関数型言語はあまり一般に普及しておらず、仕事で使う言語としては敬遠されがちである。

2. 目的

新しい組込み用プログラミング言語「Yadorigi」を開発する。

Yadorigi は関数型プログラミング言語であり、手続き型言語におけるプログラムの記述コストの問題はこれによって解決される。また、組込み用言語であるため手続き型言語と組み合わせて使う事を前提としており、Yadorigi の使用を一部分に留める事で敬遠されがちな関数型言語を簡単に使えるようにしている。

また、Yadorigi は静的型付けで、型推論器を持つ。これによって動的型付けの言語に劣らない記述力と、動的型付けの言語以上の速度を実現する。

3. 開発の内容

Yadorigi 実装は、コンパイラとインタプリタであり、Haskell で実装されている。また、C 言語向けに Yadorigi で書いたプログラムを呼び出すためのライブラリを開発した。実際に動くデモとして、N-Queens Problem プログラムを C と Yadorigi によって記述した。

3. 1 Yadorigi コンパイラ

Yadorigi コンパイラは、Yadorigi で書いたプログラムを解釈し、実行可能なデータを生成する。

- パーサ

  パーサは、Yadorigi のソースコードを読み、その字句と構文を解析し、構文木データを生成する。パーサの記述には、Haskell のパーサコンビネータライブラリ「Parsec」を使用した。Yadorigi のパーサはトップダウンパーサであり、全体に対するパーサがより下位の構文要素を再帰的にパースしていく事でソースコード全体をパースしている。Yadorigi の構文は、特定の構文要素において縦に揃えるといったレイアウト情報を構文の定義に含んでいる。これを実現するために、全てのパーサは対象となる構文要素に対するレイアウト制約を受け取る形で記述している。

- 名前解決

  名前解決では、ソースコード上の全ての名前に対して、その実体の解決を行う。名前解決は以下のような順序で行われる。

  1. モジュール参照の解決

2. スコープ名割り当て
3. 名前解決

モジュール参照の解決では、外部モジュールの参照を解決し、どのモジュールにおいてどの名前が見えているのかを確定させる。

スコープ名割り当てでは、全てのスコープに名前を割り当てる。全てのスコープは、モジュール名とスコープ名の組み合わせによって一意に特定可能な名前を持つ事になる。これによって、違うスコープで同じ名前の束縛名は、スコープの名前によって一意に特定可能となる。

名前解決では、構文木中の全ての名前に対して、その名前の出現する位置で参照される名前のリストから、名前に対応する実体を探し出す。

このような手順によって、ソースコード中の全ての名前に対して、その実体が解決される。

- 型推論

型推論では、式やパターンの持つ型を、型が自明である要素や、型注釈などの情報から推論する。これによって、型注釈を部分的にしか書かなかったとしても、それらの型情報を補完する事ができる。

例えば、C 言語などのプログラミング言語であれば、変数名全てに型情報を書かなければいけないが、型推論器がある事によって、Yadorigi ではその必要が無くなっている。

これによって、ソースコード中には最低限の型情報しか記述する必要が無くなり、プログラムの抽象度が上がるといった利点がある。

3.2 実行環境

実行環境は、Yadorigi コンパイラの出力した実行可能なデータを実際に評価し、計算を行う。

Yadorigi の実装では、型付けの済んだ状態の構文木をそのまま読み、実行している。実行環境はまず、グローバルなスコープにある全ての名前と、それらに束縛された値のテーブルを作成する。このテーブルを環境と呼ぶ。

評価機は、環境と式を受け取り、式を評価した結果を返す。ローカルなスコープで出現する名前は、評価時に環境に追加して扱われる。評価機は自分自身を再帰的に呼び出し、環境を受け渡す事で階層構造のスコープを解決する。

3.3 C 言語向けライブラリ

C 言語から Yadorigi で書いたプログラムを呼び出し、データのやりとりを行うためのライブラリを開発した。このライブラリの行う事は、以下の 3 つである。

1. runyadorigi（Yadorigi インタプリタ）の呼び出し、入出力用パイプの作成
2. C 言語のデータ表現から Yadorigi のデータ表現への変換
3. Yadorigi のデータ表現から C 言語へのデータ表現への変換

runyadorigi の呼び出し、入出力用パイプの作成では、コマンド名を受け取り、入出力用のファイルオブジェクトを 2 つ返すような関数を fork, pipe, execlp, fdopen 関数を利用して記述した。

C 言語のデータ表現は、言語そのものが持つ配列機能などを利用した物であ

り、Yadorigi のデータ表現は、表面上ではテキストでデータ構造を素直に表現した物である。変換機能を提供する事で、この 2 つのデータ表現の差を埋める。

C 言語のデータ表現から Yadorigi のデータ表現への変換では、配列やタプルなどのデータを上で説明した入力用ファイルオブジェクトに対して書き込むための関数群を記述した。

Yadorigi のデータ表現から C 言語のデータ表現への変換では、出力用ファイルオブジェクトから読み出したデータを、簡単なトップダウンパーサにより C 言語のデータ表現に変換した。

3．4　デモ

Yadorigi の実用例、利点を示すため、C と Yadorigi による簡単なデモを作成した。

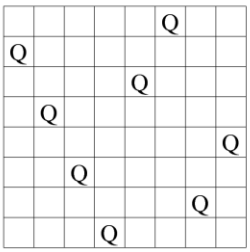

図 1. N-Queens Problem

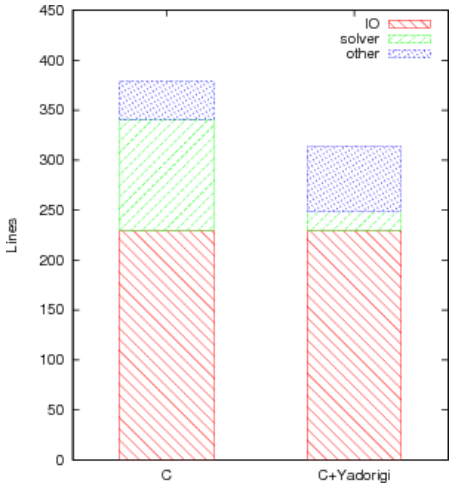

図 2. N-Queens ソルバの比較

このデモでは、ユーザの入力に従って N-Queens Problem を解き、その結果を表示する。N-Queens Problem とは、図 1 のように、N×N の盤面上に縦横斜めに重ならないように N 個の駒を配置するパズルである。入力画面で実際にボードを見ながら入力でき、複数画面に渡る結果を読みやすいように出力できるようにするため、入出力には ncurses を利用した。出力はページャのような挙動をするように作っている。

図 2 が C 単体と C+Yadorigi で実装した場合の行数の比較である。どちらの実装も、プログラム全体は入出力部分、ソルバ、その他雑多な部分に分けられる。入出力部分はどちらも共通なので長さも同じだが、ソルバの行数に大きな違いがある。C+Yadorigi での実装では、ソルバのみを Yadorigi で記述した。また、上述の C 言語向けのライブラリを使用している。

図 3 に Yadorigi による N-Queens ソルバを示す。Yadorigi は Haskell に似た構文を持つ。抽象度の高い幾つかのコンビネータを組み合わせる事で、プログラム全体を簡潔に記述できる。図 3 にある通り、「<束縛名> :: <型名>」の形式で型シグネチャ、「<束縛名> <引数リスト> = <式>」の形式で関数値の束縛となる。

```
module Nqueens where          -- 本 module の宣言
import Data.List              -- import で他のモジュールを読み込む
import Control.Applicative
import Control.Monad
{- ボードのサイズ(Int)から、N-Queens の解答リスト([[Int]])を返す関数 -}
nqueens :: Int -> [[Int]]
nqueens n = filter (all f.tails) $ permutations [0..n-1] where
    f [] = True
    f (x:xs) = and $ zipWith ((/=).abs.subtract x) xs [1..]
{- ボードのサイズ(Int)と、どの位置に駒があるべきかという制約([(Int,Int)])
   から、解答リスト([[Int]])を返す関数 -}
nqueensInterface :: Int -> [(Int,Int)] -> [[Int]]
nqueensInterface n cond = foldr (¥(x,y) -> filter $ (y==).(!!x))
(nqueens n) cond

main :: IO ()                 --入出力の main 関数
main = liftM2 nqueensInterface readLn readLn >>= print
```

図 3. Yadorigi による N-Queens ソルバ

4. 従来の技術(または機能)との相違

従来の関数型言語の多くは、その言語単体で全ての問題を解決する事を前提とした物が多く、その結果として敷居が高くなってしまっていた。それに対し Yadorigi では、一般的に利用される手続き型言語に対して組込む事を前提としており、部分的に Yadorigi を取り入れる事ができるため、関数型言語を初めて触る人の負担を減らす事ができる。

組込み用言語として代表的な言語には Lua があるが、Lua は動的型付けの手続き型言語であるのに対し、Yadorigi は静的型付けの関数型言語となっている。

5. 期待される効果

Yadorigi を使う事で、今までソースコードがどうしても長くなりがちだったプログラムをより簡潔に記述でき、開発にかかる時間の短縮などが期待できる。また、Yadorigi がきっかけで関数型言語に触れる人が増え、関数型言語の普及に貢献できる事を期待している。

現時点では、github で開発リポジトリを公開している。

7. クリエータ名(所属)

坂口和彦(木更津工業高等専門学校情報工学科)

(参考)関連URL

開発リポジトリ:
http://github.com/pi8027/yadorigi