ソフトウェア工学演習Ⅲ

システムの実装とレポート

グループ 1

2018SE016 平工真基

2018SE034 児玉春司

2018SE036 黒川誠史

問 13.5

(1) Minimal^{0}の構文木を表すデータ型を定義する。コンピュータサイエンス入門アルゴリズムとプログラミング言語の p265, p266 に Minimal の型と型定義、パターンと式が書かれているのでこれを用いて Minimal^{0}の文法を考える。まず、Minimal^{0}では代入可能な変数の宣言が禁止されているので Minimal に存在する var variable = expr { and variable = expr}は Minimal^{0}では存在しないものとする。また、Minimal^{0}では複数の宣言文の後にひとつの式がある構文に制限されるので commands ::= command {; command} は commands ::= command とし、BNF 記法の右辺に現れる commands は全て expr とする。なので、Minimal^{0}の構文は BNFC を用いて記述すると AA.cf に書いたようになる。BNFC で構文を記述する時、複数回繰り返す可能性のある記号列はその記号列を使用する時と使用しないときの 2 つのパターンを書いた。また、繰り返される記号列を Expsec ::= "," Exp とした時、Expsec ::= Expsec Expsec と記述することで何回でも繰り返せるようにした。

(2) (1)で記述した Minimal^{0}を問 11.7 で定義したΛの構文木を表すデータ型に変換する関数を定義する。まず、問 11.7 で定義したΛの構文は BNFC を用いて記述すると AB.cf に書いたようになる。次に、Ocaml を用いて Minimal^{0}をΛに変換する関数を作成する。Trans.ml というファイルを作成し、その中に関数を記述していく。Ocaml のパターンマッチングを用いて Coms x -> Prog x という風に書き、AA.cf に書いた構文に当てはまるものがあればそれを対応する AB.cf に書いた構文に変換しようとしたが Syntax error や型エラーが起きてしまい実装できなかった。なので型ごとに trans_型名という関数を作成し、与えられた構文の内終端記号はΛの構文に変換し非終端記号はその非終端記号の型に対応する関数に渡すことでΛの構文に変換しようとした。例えば、翻訳したい構文の中に型が exp の部分と型が type の部分が現れたら前者は trans_exp に渡し、後者は trans_type に渡す。また、終端記号である関数宣言の fun が現れたら fn に変換する。しかし、minimal^{0}の構文をΛの構文にどのように翻訳すれば良いかが分からず、また(a, b, c, d)は(pair a (pair b (pair c (pair d ()))))と変換すれば良いと考えてもそれをどのように書けばそのように変換できるのかが分からず実装できなかった。

(3) read.ml というファイルを作成し、問 11.7 で作成した read_print 関数を変更して Minimal^{0}の構文とその変換結果を出力する関数を作成する。まず、標準入力で構文を受け取り受け取った構文を字句解析し、構文解析したものを cha に代入する。その後、minimal^{0}と出力した後に `let out = PrintAA.printTree PrintAA.prtCommands cha` と書き out を出力することで入力が minimal^{0}の構文であることを表す。さらに、out を Trans.ml の trans_coms に渡した結果を result に代入し、Lamda と出力した後に result を出力することで変換結果がどこに出力されているのかを分かりやすくした。最後に、

read_print();;と書き再帰関数にすることで何度も関数が実行できるようにした。すると minimal^{0} の構文とその構文をΛの構文へと変換した結果を出力する read_print()関数は read.ml に書いたようになる。