OCaml 学習者の syntax error 調査

北川舞 (指導教員：浅井 健一)

1 はじめに

syntax error は、その名の通りプログラムの構文が誤っているときに発生するエラーである。初学者が殊更陥りやすいエラーであるにも関わらず、OCaml コンパイラが出力するエラーメッセージは簡素であるため、原因の特定が難しい。

本研究では、本大学の関数型言語の授業で OCaml を初めて使う学生 40 名の、2018 年上半期のプログラム実行ログを解析し、さらに syntax error に関してより詳細な分類を行い、その傾向を把握した。

2 関連研究

関数型言語の初学者が犯すエラーの分析について、いくつかの関連研究 [2, 3] がある。これらは、OCaml と同じ関数型言語である Haskell の初学者が犯したエラーについて分析を行なっている。

3 ログ解析

ログ解析は (1) OCaml コンパイラが出すエラーと警告メッセージによるもの (2) menhir [1] を使用した自作パーサが出すエラーメッセージによるもの の 2 段階で行う。

3.1 調査対象

本研究で調査対象となるプログラム実行ログは、学生が本学計算機室で OCaml プログラムを実行するたびに取られている。ログには、実行日時、実行マシン番号、実行ファイル本体、実行ファイルを実行するのに必要なサブファイル群、等が含まれている。また、調査にあたり、課題の変化が学生の犯すエラーに違いをもたらすことが想定される。そこで、課題が出される授業日から週ごとにログを分け、module が導入される前週、第 11 週までを調査対象とした。

3.2 OCaml コンパイラ

OCaml コンパイラは、実行プログラムに誤りがある場合、誤りの種類によって、様々なエラーや警告メッセージを出す。実行結果から出力されるエラーは、メッセージの共通点を取り出すことで、分類することができる。主だったエラーには、`Error:Syntax error..` のように出力される構文エラー、`Error:This expression has type int but..` のようなメッセージで型の不一致を指摘する型エラー、不適切な字句を使っていて出される `Error:Illegal character..` などがある。

3.3 menhir を用いたパーサ

3.3.1 パーサの作成方法

menhir は ocamlyacc と同じ働きをするパーサジェネレータである。ほぼ上位互換であるため、そのまま ocamlyacc の代わりに使うことができる。加えて、parse error が出た時に、指定したエラーメッセージを出すことができる。

menhir に parser.mly を渡すと default.messages というファイルができる。この default.messages に出力したいエラーメッセージを書き込む。このファイルを利用してコンパイルすることで、指定したエラーメッセージ付きのパーサを作成できる。

default.messages では、文法がオートマトンにコンパイルされた際の状態ごとに、異なるエラーメッセージを出力するよう設定できる。メッセージ作成の一助として、現在の状態の具体的な内容、この文法規則に至るまでに読み込んでいたと断定できるトークンなどが記されており、それらを参考にメッセージを作成した。

3.3.2 作成したパーサ

作成したパーサは、関数型言語第 11 回までに用いられる文法規則をほぼ網羅している。menhir を使ってエラーメッセージを作成した結果、状態の違いによって、173 種類のエラーメッセージを出すパーサができた。

例えば、

```
let abs x = if x > 0 then x else then -x
```

のような `else` 節が誤ったプログラムを実行すると、「if 文内で、else 節の式が式になっていません (259)。」というエラーメッセージを出力する（末尾の数字は、オートマトンの状態番号)。また、

```
let sokutei p = match p with
  {namae = n; shincho = s} ->
    if s < 152.0 then "小" else "大"
let test1 = sokutei
    {namae = "kitagawa"; shincho : 170.5 }
```

のようなレコード内の構文の書き方を誤った、前の例とは異なる構文エラーをもつプログラムを実行すると、「レコード内で、フィールド名の次に=(あるいはセミコロン、閉じ中かっこ) 以外のものが来ています (183)。」という、別のエラーメッセージが出力される。

4 分類結果と考察

4.1 OCaml コンパイラ

本大学の関数型言語の授業で OCaml を初めて使う学生 40 名の、調査範囲のログを生成した全 26,952 件の OCaml ファイルに対し、コンパイラによる実行結果を分類した（分類結果：図 1)。

- Haskell 初学者の調査 [2] では、学習の経過とともに、syntax error は減少し、type error が増加する傾向が観察された。一方、本研究では、syntax error, type error 共に一定割合発生している。これは、カリキュラムの難易度が学生の成長に見合っており、また、本学の OCaml 教育に導入されている型デバッガの、一定の効果を表すと考える。

- unbound error には、Unbound record field, Unbound value, Unbound constructor などが含まれる。Unbound value error は週が進むごとに減少するのではないかと考えたが、回が進んでも Un-

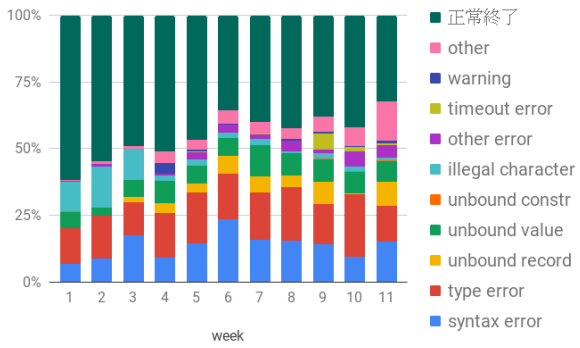

図 1: OCaml コンパイラ による分類

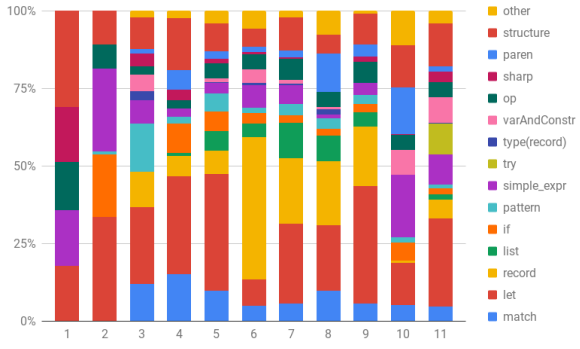

図 2: menhir を用いた自作パーサ による分類

bound value error, Unbound record field error が一定割合起こっていることが確認された。

- illegal character のような字句エラーは、初期（第 1 週〜第 3 週）には多く見受けられるが、その後は収束する。
- 初期に見られなかった、other に分類される実行結果は、`Exception:Not_found` のような、例外が発生して終了したもの、あるいは、`Cannot find file...` のようなロードしようとしたファイルが見つからなかった場合など、実行時エラーのファイルが主である。
- 第 9 週には、一定数の timeout error が確認される。これは、この週の課題が実行に数時間を要するプログラムであるためだと考えられる。

4.2 menhir を用いた自作パーサ

OCaml コンパイラによる分類から、実行結果が syntax error となったファイル全 3,639 件に対し、自作パーサによる詳細な分類を試みた（分類結果：図 2）。自作パーサは、 173 種類のエラーメッセージを出力するが、ここではメッセージの内容の違いとその件数によって、 16 のグループにまとめた。なお、うち 58 件については、自作パーサではエラーが検出されなかった。

- if 文は第 2 週で初めて習う構文である。初週は「else 節がない場合 then 部分は unit 型にする必要がある」という OCaml の特徴を知らないユーザが、誤りを犯すことが一定数確認されるが、その後収束する。対して、 第 3 週に初めて習う match 文や第 1 週から用いる let 文のエラーは、後半週に至るまで一定割合確認されており、対策の必要がある。
- 様々なデータ構造とその記法はプログラミング言語毎に異なり、初学者を混乱させる。本研究でも、レコードやリストといったデータ構造の扱いに関するエラーが常に多く見られた。また、第 10 週では、かっこに関するエラー（paren）が一定割合見られるが、週のテーマである木構造を扱う OCaml の記法が学生を悩ませたためだと考えられる。
- simple_expr や pattern に含まれるエラーは、OCaml の基本的な字句のまとまりがわかっていない場合に起こる。大半は第 3 週までに起こっており、時期に収束する。
- other に含まれるエラーには、例外を定義する exception 文、匿名関数を作る fun 文、ヴァリアントを定義する type 文などのエラーが含まれている。

5 パーサを使ったエラー解消の問題点

menhir を用いたパーサにより、既存の OCaml コンパイラよりも情報量の多いエラーメッセージをユーザに与えることができる。3.3 節で見たように、このエラーメッセージは syntax error を解消するのに有益な情報となる一方、限界もある。例えば、状態番号 282 でのメッセージは、「let 文で束縛された式の次に、in 以外のものが来ています (282)。」とした。しかし実際、このエラーを発生させるプログラムは、

```
let chocolate n =
let test1 = chocolate 216 = 2
```

などである。エラー解消のためにユーザがすべき修正は、`chocolate` 関数の中身を記述することであって、プログラムの末尾に `in` を書き加えることではない。

パーサでの解決が難しいこのようなエラーは、ブロックプログラミング等プログラム全体の構造が把握しやすい方法で、ユーザに理解を促さざるえないのではないかと考える。

6 まとめ

本研究では、 OCaml コンパイラと menhir を用いた自作パーサにより、本学の OCaml 初学者の実行データを分析し、学生の syntax error の傾向を把握した。得られた知見から、今後授業の改善に取り組む。

参考文献

[1] http://gallium.inria.fr/~fpottier/menhir/.

[2] Singer Jeremy and Archibald Blair. Functional baby talk: Analysis of code fragments from novice haskell programmers. Vol. 270 of *Electronic Proceedings in Theoretical Computer Science*, pp. 37–51. Open Publishing Association, 2018.

[3] Boldizsár Németh, Eunjong Choi, Erina Makihara, and Hajimu Iida. Investigating compilation errors of students learning haskell. In *Trends in Functional Programming in Education*, 2018.