プログラミング言語処理系
OCamlの環境設定

末永 幸平

この資料について

- 京都大学工学部専門科目「プログラミング言語処理系」の講義資料
- 講義 Web ページ:
  - https://kuis-isle3sw.github.io/IoPLMaterials/
- 講義をする人: 末永幸平
  - https://researchmap.jp/ksuenaga/
  - https://twitter.com/ksuenaga

今日の内容

- 環境設定とOCaml の復習
  - 環境設定について
    - OCaml 言語処理系のインストール
    - OCaml プログラムの実行の仕方
    - OCaml プログラミングのためのツール
  - OCaml 復習

Disclaimer

- このスライドは 2020 年時点での内容で書いてあります

OCaml (https://ocaml.org/) とは

- フランスの INRIA (https://www.inria.fr/fr) で作られている関数型プログラミング言語
  - 関数型言語でありながら命令形言語の良さも備えている
  - 実行前の型検査によって，実行時型エラーを起こさないことが保証されている
  - 型推論によって型アノテーションを書かなくても書ける
  - オブジェクト指向もある

- 関数型言語と命令形言語
  - 関数型言語: OCaml, Haskell, LISP, Scheme, …
    - （基本的に）計算は式を評価することで行われる
    - 関数を値として使える（関数を受け取ったり渡したりできる）
  - 命令形言語: C, Java, …
    - 計算は命令を実行することで行われる

OCaml インストールのすすめ

- 計算機科学コースの人はどっちみち実験3SWで動かす必要
- そうでない人もこんな効果が
  - 教科書を読むだけで一応理解はできる
  - 動かしてみたほうがもっと理解できる
  - 改造してみるとさらによく理解できる
  - 関数型言語を使う貴重な機会

OCaml を自分のマシンに入れるには

- 公式のドキュメント https://ocaml.org/docs/install.html を読み
- 自分の環境に合ったインストール方法を理解し
- インストールする

英語じゃない情報

- 講義リポジトリに無保証の翻訳あり
  - https://kuis-isle3sw.github.io/IoPLMaterials/textbook/setting-up-ocaml.html

もう少しくわしく

- OCaml のパッケージマネージャである OPAM をインストール
  - 京大工学部専門科目「プログラミング言語」の「プログラミング言語で用いる開発環境の整備方法」という資料の「開発環境のセットアップ」のパートと「OPAMの初期設定」のパートを読んで実行
    - http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/setup.html
- その後 https://kuis-isle3sw.github.io/IoPLMaterials/textbook/setting-up-ocaml.html の指示に従って必要なライブラリをインストール

- **わからない場合はこの講義の Slack か PandA で質問！**

OPAM のインストールについて もう少しくわしく: Linux の場合

- https://ocaml.org/docs/install.html から自分のディストリビューションを探してそこに書いてあるコマンドを実行
  - Debian なら: `apt-get install opam`
  - Ubuntu なら:
    - ```
      add-apt-repository ppa:avsm/ppa
      apt update
      apt install opam
      ```
  - 和訳も少ししておきました https://kuis-isle3sw.github.io/IoPLMaterials/textbook/install_opam.jp.html
      ```

OPAM のインストールについて もう少しくわしく: MacOS の場合

- まず xcode コマンドラインツールをインストール
  - ターミナルで `xcode-select --install` を実行
- Homebrew をインストール
  - Homebrew: https://brew.sh/index_ja
- OCaml と OPAM をインストール
  - `brew install gpatch ocaml opam`

OPAM のインストールについて
もう少しくわしく: Windows 10 以上の場合

- WSL (Windows Subsystem for Linux) を入れる
  - https://docs.microsoft.com/ja-jp/windows/wsl/install-win10 を参考に
- Ubuntu 18.04 を入れる
  - https://www.microsoft.com/store/apps/9N9TNGVNDL3Q
- Linux の場合と同様に OPAM を入れる

OPAM のインストールについて
もう少しくわしく: Windows 9 以前の場合

- VirtualBox (https://www.virtualbox.org/) を使って Ubuntu 18.04 の仮想環境を準備
- その上で OPAM をインストール

もう少しくわしく

- OCaml のパッケージマネージャである OPAM をインストール
  - 京大工学部専門科目「プログラミング言語」の「プログラミング言語で用いる開発環境の整備方法」という資料の「開発環境のセットアップ」のパートと「OPAMの初期設定」のパートを読んで実行
    - http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/setup.html
- その後 https://kuis-isle3sw.github.io/IoPLMaterials/textbook/setting-up-ocaml.html の指示に従って必要なライブラリをインストール

- **わからない場合はこの講義の Slack か PandA で質問！**

この演習で必要な OCaml ライブラリのインストール

- https://kuis-isle3sw.github.io/IoPLMaterials/textbook/setting-up-ocaml.html を見てやる

- 以下のコマンドを順に実行．ログの最後に `eval $(opam env)` を実行せよみたいなメッセージが出たら，`eval $(opam env)` を実行してから次のコマンドを実行．

```
opam install depext
opam install user-setup
opam depext menhir dune ounit
opam install menhir dune ounit tuareg
opam user-setup install
```

やったか！？

- ocaml コマンドを入力してインタプリタが立ち上がれば成功
  - こんなかんじ

```
        OCaml version 4.06.1

Findlib has been successfully loaded. Additional directives:
  #require "package";;      to load a package
  #list;;                   to list the available packages
  #camlp4o;;                to load camlp4 (standard syntax)
  #camlp4r;;                to load camlp4 (revised syntax)
  #predicates "p,q,...";;   to set these predicates
  Topfind.reset();;         to force that packages will be reloaded
  #thread;;                 to enable threads

# 
```

もう少しくわしく

- OCaml のパッケージマネージャである OPAM をインストール
  - 京大工学部専門科目「プログラミング言語」の「プログラミング言語で用いる開発環境の整備方法」という資料の「開発環境のセットアップ」のパートと「OPAMの初期設定」のパートを読んで実行
    - http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/setup.html
- その後 https://kuis-isle3sw.github.io/IoPLMaterials/textbook/setting-up-ocaml.html の指示に従って必要なライブラリをインストール

- **わからない場合はこの講義の Slack か PandA で質問！**

OCaml プログラミングのやり方

- OCaml インタプリタを起動して直接プログラムを入力
- ファイルにプログラムを書いてインタプリタから読み込ませる
- ファイルにプログラムを書いて実行可能ファイルをビルド

OCaml インタプリタを起動して直接プログラムを入力

- `ocaml` コマンドを起動
- 出てきたインタプリタにプログラムを入力して実行

- 長所
  - すぐに実行してすぐに結果を確認できる
- 短所
  - ocaml コマンドのインターフェイスが貧弱
    - Emacs なら M-x shell でシェルを起動して実行することでまあまあなんとかなる
    - 高機能なインターフェイスを持つ `utop` を使うのも手かも
  - 入力内容を保存できない
    - 課題をやる上では辛い

ファイルにプログラムを書いて インタプリタから読み込ませる

- 好きなファイル（例えば “main.ml”）にプログラムを書く
- ocaml コマンドを起動
- インタプリタで #use “<ファイル名>”;; と入力して
  - 例えば #use “main.ml”）

- 長所
  - 書いたプログラムがファイルとして保存される
  - ファイルを git 等のバージョン管理システムで管理すれば変更履歴も追える
- 短所
  - 少し手間なくらい？
    - だが，いちいち書き直すほうが手間

ファイルにプログラムを書いて実行可能ファイルをビルド

- プログラムをモジュール単位に分割して，
  各モジュールの実装とインターフェイスを各ファイルに書く
- ビルドツール（dune や make 等）を使って
  実行可能ファイルをビルド
- 生成された実行可能ファイルを実行

- 長所
  - プログラムを修正したときに，影響を受ける部分のみをコンパイルすればよい
  - 自動テストとの組み合わせが比較的容易
- 短所
  - ビルド用ファイルを書くのがちょっと大変

OCaml プログラミングのためのツール

- VSCode や Emacs や Vim の OCaml 用プラグインを使うのが吉
  - Merlin
    - https://github.com/ocaml/merlin
  - OCaml LSP
    - https://github.com/ocaml/ocaml-lsp

- 便利機能がたくさん
  - Syntax highlighting
  - 自動テストの実行やビルド等がボタン一発
  - 指定した変数や式の型を一発で見ることが可能
  - 自動補完もしてくれる

- **ベストプラクティスを見つけたら Slack や PandA 等でシェアしよう**

今日の内容

- 環境設定と OCaml の復習
  - 環境設定について
  - OCaml 復習

今日の残りの内容

- できるだけ早く OCaml を書けるようになってもらうための OCaml 復習講義

とはいうものの

- プログラミング言語を学ぶには,
  自分でプログラムを書いて動かしてもらう以外にない
  - なんぼ OCaml の講義をここでしたとしても,
    自分でやらねばプログラムは書けない

というわけで

- 今日は OCaml をある程度書けるようになるために
  やっておいてほしい資料を書いておきます
- 講義動画では，そのさわりの部分をライブコーディングして見せます
- 残りの部分は各自で進めてください

資料

- 工学部専門科目「プログラミング言語」配布資料のうち以下
  - 最低ここは読んで書いて動かしてほしい
    - OCaml 爆速入門: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/03-ocaml.html
    - 2分探索木 in OCaml: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/04-bst-ocaml.html
  - できればここも読んで書いて動かしてほしい
    - 再帰と繰り返し: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/08-rec-iter.html
    - 「高階関数」のうち「高階関数 in OCaml」の節: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/10-hofuns.html
  - ここもやるとラクになる
    - 「多相性」のうち「多相的2分木 in OCaml」の節: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/09-polymorphism.html
    - 「多相的2分探索木」のうち OCaml に関する部分: http://www.fos.kuis.kyoto-u.ac.jp/~igarashi/class/pl/11-polyBST.html