Programming Languages Recitation

Standard ML : Installation and Basics

Arpit Jain

Computer Science Department
Courant Institute of Mathematical Sciences
New York University

arpit.jain@cs.nyu.edu

October 23, 2014

Overview

1.
2.
3.
4.

ML Introduction

- Originally developed for use in writing theorem provers.

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed
- Static typing

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed
- Static typing
- Structural equivalence

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed
- Static typing
- Structural equivalence
- **Type inference**

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed
- Static typing
- Structural equivalence
- **Type inference**
- Advanced module system

ML Introduction

- Originally developed for use in writing theorem provers.
- Functional (First class values)
- Garbage collected
- Applicative order evaluation
- Strongly typed
- Static typing
- Structural equivalence
- **Type inference**
- Advanced module system
- Exceptions

SML of New Jersey

- Download from SML/NJ
- For Unix download config.tgz , update it and run install.sh
- For Mac download smlnj-x86-110.77.pkg and add /usr/local/smlnj/bin to bash profile
- For Windows download smlnj-110.77.msi and configure path variables

- Run sml in Terminal/Command Prompt and the sml interactive environment should be launched

Using Sublime Text (Optional)

- Download from Sublime Text

Using Sublime Text (Optional)

- Download from Sublime Text
- Download and Install Package Control

Using Sublime Text (Optional)

- Download from Sublime Text
- Download and Install Package Control
- Install SML REPL
  - From Tools → Command Palette, type "Install Package" and click Package Control: Install Package
  - Search for sublimeREPL and click the filtered result

Using Sublime Text (Optional)

- Download from Sublime Text
- Download and Install Package Control
- Install SML REPL
  - From Tools → Command Palette, type "Install Package" and click Package Control: Install Package
  - Search for sublimeREPL and click the filtered result
- Now go to preferences and browse package

Using Sublime Text (Optional)

- Go to folder User and create a new folder SML with two files from https://github.com/arpitjain2811/Programming-Languages-Recitation-Fall-2014-NYU under Lec 8 → Sublime Configuration → SML

Using Sublime Text (Optional)

- Go to folder User and create a new folder SML with two files from https://github.com/arpitjain2811/Programming-Languages-Recitation-Fall-2014-NYU under Lec 8 → Sublime Configuration → SML
- In Main.sublime-menu file edit line 17 "cmd" with the path to smlnj bin.

Using Sublime Text (Optional)

- Go to folder User and create a new folder SML with two files from https://github.com/arpitjain2811/Programming-Languages-Recitation-Fall-2014-NYU under Lec 8 → Sublime Configuration → SML
- In Main.sublime-menu file edit line 17 "cmd" with the path to smlnj bin.
- Run the REPL by going to tools → sublimeREPL → SML

Basics

- val i=2; ⇒ val i=2 : int
- i * i; ⇒ val it = 4 : int
- [1, 2, 3]; ⇒ val it = [1,2,3] : int list
- ["*hello*","*world*"]; ⇒ val it = ["hello","world"] : string list
- 1 :: [2, 3]; ⇒ val it = [1,2,3] : int list

List Operations

- null [1, 2] $\Rightarrow$ val it = false : bool
- null [ ]; $\Rightarrow$ val it = true : bool
- hd [1, 2, 3]; $\Rightarrow$ val it = 1 : int
- tl [1, 2, 3]; $\Rightarrow$ val it = [2, 3] : int list
- [] $\Rightarrow$ val it = [ ] : 'a list

Functions

- Function declaration : fun name parameter = body
- Function expression : fn parameter = body

Example uploaded