Interpretation and compilation of programming languages
Project Assignment

João Costa Seco Luís Caires

Document History

Initial release: 23 November 2015
Added conditional expression: 26 November 2015
Added type annotations and reference to unit tests: 27 November 2015
Changed submission date and evaluation rules: 1 December 2015

Important dates

Intermediate submission (mandatory): 30 November 2015
Final submission (mandatory): **6 December 2015**
Project discussions: 14-15 December 2015

1 Introduction

The goal of this project assignment is to implement the interpreter and compiler of a typed imperative programming language whose main elements are functions, records, and high-order functions. The project will be submitted in two phases and evaluated in two grade ranges (**0 to 18** for the base imperative language with lists and records, and 0 to 20 for the complete language).

The source language for the project assignment, whose syntax is given in Figure 1, includes arithmetic operations, comparison operations, boolean operations, the binding expression (`decl`), identifiers, records, lists, and their corresponding operations. The semantics for the interpreter, typechecker, and compiler follow the course lectures and lab classes. The syntax follows the expected operator priority, which can be checked using an ocaml interpreter or Java compiler.

$\langle exp \rangle$	::=	$\langle num \rangle$	Integer Literals
	|	$\langle id \rangle$	Identifiers
	|	$\langle blit \rangle$	Boolean literals
	|	$\langle exp \rangle$`+`$\langle exp \rangle$ | $\langle exp \rangle$`-`$\langle exp \rangle$	Arithmetic operations
	|	$\langle exp \rangle$`*`$\langle exp \rangle$ | $\langle exp \rangle$`/`$\langle exp \rangle$ | `-` $\langle exp \rangle$	
	|	$\langle exp \rangle$`<`$\langle exp \rangle$ | $\langle exp \rangle$`>`$\langle exp \rangle$	Comparator operations
	|	$\langle exp \rangle$`==`$\langle exp \rangle$ | $\langle exp \rangle$ `!=` $\langle exp \rangle$ | ...	
	|	$\langle exp \rangle$ `&&` $\langle exp \rangle$ | $\langle exp \rangle$`||`$\langle exp \rangle$ | `!`$\langle exp \rangle$	Boolean operations
	|	`decl` $(\langle id \rangle$ `=` $\langle exp \rangle)+$ `in` $\langle exp \rangle$ `end`	Declaration
	|	`var(`$\langle exp \rangle$`)`	Imperative constructs
	|	$\langle exp \rangle$`:=`$\langle exp \rangle$	
	|	`*`$\langle exp \rangle$	
	|	`while` $\langle exp \rangle$ `do` $\langle exp \rangle$ `end`	
	|	`if` $\langle exp \rangle$ `then` $\langle exp \rangle$ `else` $\langle exp \rangle$`end`	
	|	$\langle exp \rangle$`;`$\langle exp \rangle$	
	|	`fun` $(\langle id \rangle$`:`$\langle type \rangle)*$ `=>` $\langle exp \rangle$ `end`	Function
	|	$\langle exp \rangle$ `(` $\langle exp \rangle+$ `)`	Function Call
	|	{ $(\langle id \rangle = \langle exp \rangle)+$ }	Record operations
	|	$\langle exp \rangle.\langle id \rangle$	
	|	`[` $\langle exp \rangle+$ `]`	List operations
	|	$\langle exp \rangle$`::`$\langle exp \rangle$	
	|	`hd`$\langle exp \rangle$	
	|	`tl`$\langle exp \rangle$	
	|	`(` $\langle exp \rangle$ `)`	
$\langle num \rangle$	::=	$['1' - '9']['0' - '9']*$	
$\langle blit \rangle$	::=	`true` | `false`	
$\langle id \rangle$	::=	$['a' - 'z','A' - 'Z']['a' - 'z','A' - 'Z','0' - '9']*$	
$\langle type \rangle$	::=	`int` | `bool` | `ref(` $\langle type \rangle$`)`	

Figure 1: Syntax

2 First submission phase

The first submission phase is due on **30 November 2015**, and comprises the parser, interpreter, and compiler for the basic imperative language (no lists, records, or functions). The first submission is graded in the 0-20 range, with a 30% weight on the final grade. Partial deliveries will still be considered, but not without any penalty.

3 Second submission phase

The second submission phase is due on **6 December 2015** and comprises the full project.

4 Evaluation

The project evaluation is divided in two different grade ranges. The full range (0-20) can be accessed by projects that present the parsing, interpretation, typing, and compilation for high-order functions. A project that does not complete this requirement is evaluated in the grade range **0-18**.

Discussions about the project will be held on **14-15 December 2015**.

5 Deliverables

The project should be delivered using the SRTA system in the form of a zip archive containing the source code, a file with language examples, and a 1 page report clearly stating the status of the project with relation to the requirements (the grade range, and the constructs that are totally/partially implemented).