P. Thiemann, G. Radanne Summersemester 2017/18

Essentials of Programming Languages

https://proglang.informatik.uni-freiburg.de/teaching/konzepte/2018ss/

Project λjs

2018-06-26

λjs is a language which has both functions, in the style of the lambda calculus, and simple objects, similar to Javascript. Your goal will be to implement a dynamic semantics and a type system for λjs using PLT-redex.

1 Project

This project should be worked on in groups of at most two students. The implementation should contain a set of `rkt` files. For the final version, each group should send a file "project-PLT-<NAMES>.tar" to both Prof. Peter Thiemann and Gabriel Radanne by email. The email should be titled "project PLT <NAMES>". The deadline for the project is the **July 31, 2018**.

To help you get started, a file `base.rkt` is provided which contains the grammar definition of the language and some testing infrastructure. All the groups are expected to make a meaningful effort at the first section. Implementations should contain appropriate tests.

2 Language Description and Dynamic Semantics

λjs is composed of constructions we have seen in various other languages:

- Lambdas
- Objects
- Mutability

We thus consider a call-by-value language with first class functions and objects. New objects are declared as a list of variables and methods. The `set` and `get` operations allow to obtain the value and modifies the fields of objects. Methods are simply fields that contains a function. Objects can be recursive by using the `this` variable. Method names can be arbitrary symbols. Numbers are objects with methods `+`, `*`, ...

Your first task is to implement small step reduction rules for this language in three steps:

1. Implement lambdas and objects
2. Add recursion with `this`
3. Add mutability

e	$::=$	x	Variables
	$\mid$	$(e\ e)$	Application
	$\mid$	$\lambda(x : \tau).e$	Abstraction
	$\mid$	(`object` (field e) ...)	Object
	$\mid$	(`send` e field e ...)	Method call
	$\mid$	(`get` e field)	Get field
	$\mid$	(`set` e field e)	Set field
	$\mid$	*number*	Numbers

3 Type System

We consider the type system for λjs as an extension of the simply-typed lambda calculus where the type of an object is a record type, that is, $\langle f_0 : \tau_0; f_1 : \tau_1; \ldots \rangle$ is the type of an object containing the fields f_0 of type τ_0, f_1 of type τ_1, and so on. When an object is created with the `object` construct all the fields are typed and used in the object type. Set and get do not modify the type of the object. For simplicity, it is not possible to change the type of a field or add new fields using `set`. You can see below the typing rules for object creation.

Implement type checking for such a type system. Test that your implementation respects progress and preservation.

$$\textsc{Object}\quad \frac{\text{for all } i, \quad \Gamma,\ (\mathit{this} : \langle f_0 : \tau_0;\ f_1 : \tau_1;\ \ldots\rangle) \vdash e_i : \tau_i}{\Gamma \vdash (\texttt{object}\ (f_0\ e_0)\ \ldots) : \langle f_0 : \tau_0;\ f_1 : \tau_1;\ \ldots\rangle}$$

$$\textsc{Get}\quad \frac{\Gamma \vdash e : \langle \ldots; f : \tau;\ \ldots\rangle}{\Gamma \vdash (\texttt{get}\ e\ f) : \tau}$$

$$\textsc{Set}\quad \frac{\Gamma \vdash e : \langle \ldots; f : \tau;\ \ldots\rangle \qquad \Gamma \vdash e' : \tau}{\Gamma \vdash (\texttt{set}\ e\ f\ e') : \tau}$$