Foundations of Data and Knowledge Bases

A Evaluator for Mini-Python in Python

Joachim Niehren

Links: Linking Dynamic Data
Inria Lille

September 7, 2022

Outline

1 Mini-Python

2 Evaluator

Syntax

Terms

Consider the Mini-Python language whose programs are terms with the following abstract syntax:

$$e \in P ::= \mathit{let}(x, e, e) \mid \lambda(x, e) \mid @(e, e) \mid x \mid n \mid +(e, e)$$

Here, $n \in \mathbb{N}$ ranges over natural numbers and $x \in \mathit{Vars}$ over variables in some fixed set Vars.

Concrete Syntax

For better readability, we will write the terms of Mini-Python with the following concrete syntax:

$$\begin{aligned}
\mathit{let}(x, e, e') &\Rightarrow \mathit{let}\ x = e\ \mathit{in}\ e' \\
\lambda(x, e) &\Rightarrow \lambda x.e \\
@(e, e') &\Rightarrow e(e') \\
+(e, e') &\Rightarrow e + e'
\end{aligned}$$

Example Program

As an example, consider the following Python program:

```
x=100
def y(z):
    return x+z
y(2)
```

In Mini-Python, this Python program can be written as the following term:

$$\mathit{let}\ x = 100\ \mathit{in}\ \mathit{let}\ y = \lambda z.\ x + z\ \mathit{in}\ y(2)$$

This program in P can be represented by a Python value e_0 as follows:

```
e0 = ('let','x',100,
        ('let','y',('lambda','z',('+','x','z')),
         ('@','y',2)))
```

Outline

1 Mini-Python

2 Evaluator

Objective

The next objective is to write an evaluator for Mini-Python in Python. Let *Vals* be the set of all Python values, i.e., the set of values that may be bound to a Python variable, or returned by a Python function.

Environments

The evaluation of a Mini-Python program has to be done relatively to an assignment of variables to Python values, which will take care of the free variables of the program. Such assignments are called environments:

$$Env = Vars \rightarrow Vals$$

Extending Environments

Extensions

We will need a function that extends a given environment for a given variable by a given value:

$$ext : Env \times Vars \times Vals \to Env$$

It should satify for any environment $\alpha \in Env$, variable $x, y \in Vars$, and Python value $v \in Vals$:

$$ext(\alpha, x, v)(y) = \begin{cases} v & if\ x = y \\ \alpha(y) & else \end{cases}$$

Exercise 1

Implement the function *ext* for extending environments in Python.

Evaluator

The evaluator will be a function of the following type:

$$eval : P \times Env \to Vals$$

For all Mini-Python programs $e, e' \in P$, variables $x \in Vars$, natural numbers $n \in \mathbb{N}$, and environments $\alpha \in Env$ we define:

$$\begin{array}{rcl}
eval(@(e, e'), \alpha) & = & eval(e, \alpha)(eval(e', \alpha)) \\
eval(let(x, e, e'), \alpha) & = & eval(e', ext(\alpha, x, eval(e, \alpha))) \\
eval(\lambda x.e, \alpha) & = & f \text{ where } f : Vals \to Vals \text{ with} \\
 & & f(v) = eval(e, ext(\alpha, x, v)) \text{ for all } v \in Vals \\
eval(x, \alpha) & = & \alpha(x) \\
eval(n, \alpha) & = & n \\
eval(e + e', \alpha) & = & eval(e, \alpha) + eval(e', \alpha)
\end{array}$$

Exercise 2

Implement the Mini-Python evaluator function *eval* in Python. Test it by running the Python program:

$$eval(e_0, lambda\ y :' error')$$

were e_0 is the above program and $lambda\ y :' error'$ the environment that maps any variable in $Vars$ to the string $'error' \in Vals$. The value that Python should return is 102.
Note that e_0 does not have any free variable. Therefore, the value of e_0 is independent of which precise environment is chosen for evaluation.
Still, environments are very useful for evaluation, since all subterms are to be evaluated too, and these may have free variables. In the example, the subterm $\lambda z.x + z$ has free variable x while z is not free there.