

YuchiKaml

Yuchiki

2018 Dec.

Contents

1	Introduction	1
2	YuchiKaml Language	1
2.1	Syntax	1
2.1.1	Syntax Sugar	2
2.2	Semantics	2
2.2.1	Value	2
2.2.2	Evaluation	3
3	YuchiKaml Interpreter	5
3.1	Usage	5
3.2	Preprocess	5

1 Introduction

YuchiKaml is a toy language. and YuchiKaml interpreter is an implementation of interpreter of YuchiKaml. Both are created in order to get accustomed to Sprache, a C#Parser Combinator Library. In this article, I introduce both the language and the interpreter.

2 YuchiKaml Language

YuchiKaml is a dynamic typed language with-ML like surface grammar.

2.1 Syntax

Expressions of YuchiKaml are defined by the following BNF equations:

$$\begin{aligned} e ::= & () \mid x \mid n \mid \text{true} \mid \text{false} \mid s \mid (e) \\ & \mid e \ e \mid !e \\ & \mid e * e \mid e / e \\ & \mid e + e \mid e - e \\ & \mid e \leq e \mid e < e \mid e \geq e \mid e > e \\ & \mid e = e \mid e \neq e \\ & \mid e \& e \\ & \mid e \parallel e \\ & \mid e \triangleright e \mid e \gg e \\ & \mid \text{if } e \text{ then } e \text{ else } e \mid \text{let } x \ \tilde{a} = e \text{ in } e \mid \text{let rec } f \ a_1 \ \tilde{a} = e \text{ in } e \mid \lambda x. e \\ & \mid e ; e \end{aligned}$$

The operators defined in earlier rows have stronger precedences than the operators defined in later rows. For example, $1 + 2 * 3$ is not parsed as $(1 + 2) * 3$, but $1 + (2 * 3)$.

In real source codes, the symbols above are notated as follows:

\leq	$<=$
\geq	$>=$
\neq	$!=$
$\&$	$\&\&$
\parallel	\parallel
\triangleright	$ >$
\gg	$>>$
$\lambda x. e$	$\backslash x - > e$

Example 2.1 (GCD). This is an example of a YuchiKaml source code of a program which calculates the greatest common divisor of 120 and 45.

Listing 1: Samples/gcd

```
let rec gcd m n =
  if m > n then gcd (m - n) n
  else if m < n then gcd m (n - m)
  else m
in gcd 120 45
```

2.1.1 Syntax Sugar

Some of the expressions shown above are syntactic sugars. We show the list and syntax sugars and how they are desugared.

$$\begin{array}{l}
 e_1 \triangleright e_2 \\
 e_1 \gg e_2 \\
 \text{let } x \tilde{a} = e \text{ in } e \text{ let rec } x \ a_1 \ \tilde{a} = e \text{ in } e \qquad e ; e =
 \end{array}$$

TODO: explain it.

2.2 Semantics

Then we define the semantics of the expressions.

2.2.1 Value

Values of YuchiKaml is listed as below:

$$\begin{aligned}
v(\text{value}) &:: = n \mid b \mid s \mid \text{cl} \mid f_b \\
\rho(\text{valuation}) &\in \text{Var} \not\rightarrow \text{Val} \\
f_b(\text{built-in function}) &\in \text{Val} \not\rightarrow \text{Val} \\
\text{cl}(\text{closure}) &:: = (x, e, \rho)
\end{aligned}$$

Here Var is the set of the variables and Val is the set of the values.

2.2.2 Evaluation

We define the *evaluation* process of expression by a big-step semantics shown below.

An *evaluation relation* is a four-term relation of the form $\rho \vdash e_1 \longrightarrow e_2$.

The evaluation rules of YuchiKaml are shown below:

$$\begin{aligned}
&\frac{\rho(x) = v}{\rho \vdash x \longrightarrow v} && (\text{E-VAR}) \\
&\frac{\rho \vdash e_1 \longrightarrow e'_1}{\rho \vdash e_1 e_2 \longrightarrow e'_1 e_2} && (\text{E-APPLEFT}) \\
&\frac{\rho \vdash e_2 \longrightarrow e'_2}{\rho \vdash v_1 e_2 \longrightarrow v'_1 e'_2} && (\text{E-APPRIGHT}) \\
&\frac{\rho' \cup \{x \mapsto v\} \vdash e_1 \longrightarrow e'_1}{\rho \vdash (x, e_1, \rho') v_2 \longrightarrow (x, e'_1, \rho') v_2} && (\text{E-APPCLS}) \\
&\frac{}{\rho \vdash (x, v_1, \rho') v_2 \longrightarrow v_1} && (\text{E-APPCLS2}) \\
&\frac{f_1 @ v_2 = v_3}{\rho \vdash f_1 v_2 \longrightarrow v_3} && (\text{E-APPBUILTIN}) \\
&\frac{\rho \vdash e_1 \longrightarrow e'_1}{\rho \vdash e_1 \text{ op } e_2 \longrightarrow e'_1 \text{ op } e_2} && (\text{E-BINOP-LEFT}) \\
&\frac{\rho \vdash e_2 \longrightarrow e'_2}{\rho \vdash v_1 \text{ op } e_2 \longrightarrow v_1 \text{ op } e'_2} && (\text{E-BINOP-RIGHT}) \\
&\frac{n_1 \llbracket * \rrbracket n_2 = n_3}{\rho \vdash n_1 * n_2 \longrightarrow n_3} && (\text{E-MUL}) \\
&\frac{n_1 \llbracket / \rrbracket n_2 = n_3}{\rho \vdash n_1 / n_2 \longrightarrow n_3} && (\text{E-DIV})
\end{aligned}$$

$$\begin{array}{c}
\frac{n_1 \llbracket + \rrbracket n_2 = n_3}{\rho \vdash n_1 + n_2 \longrightarrow n_3} \quad (\text{E-PLUS}) \\
\\
\frac{n_1 \llbracket - \rrbracket n_2 = n_3}{\rho \vdash n_1 - n_2 \longrightarrow n_3} \quad (\text{E-MINUS}) \\
\\
\frac{n_1 \llbracket \leq \rrbracket n_2 = b_3}{\rho \vdash n_1 \leq n_2 \longrightarrow b_3} \quad (\text{E-LEQ}) \\
\\
\frac{n_1 \llbracket < \rrbracket n_2 = b_3}{\rho \vdash n_1 < n_2 \longrightarrow b_3} \quad (\text{E-LT}) \\
\\
\frac{n_1 \llbracket \geq \rrbracket n_2 = b_3}{\rho \vdash n_1 \geq n_2 \longrightarrow b_3} \quad (\text{E-GEQ}) \\
\\
\frac{n_1 \llbracket > \rrbracket n_2 = b_3}{\rho \vdash n_1 > n_2 \longrightarrow b_3} \quad (\text{E-GT}) \\
\\
\frac{b_1 \llbracket \& \rrbracket b_2 = b_3}{\rho \vdash b_1 \& b_2 \longrightarrow b_3} \quad (\text{E-AND}) \\
\\
\frac{b_1 \llbracket || \rrbracket b_2 = b_3}{\rho \vdash b_1 || b_2 \longrightarrow b_3} \quad (\text{E-OR}) \\
\\
\frac{(v_1 = v_2) = b_3}{\rho \vdash v_1 = v_2 \longrightarrow b_3} \quad (\text{E-EQ}) \\
\\
\frac{(v_1 \neq v_2) = b_3}{\rho \vdash v_1 \neq v_2 \longrightarrow b_3} \quad (\text{E-NEQ}) \\
\\
\frac{\rho \vdash e_1 \longrightarrow e'_1}{\rho \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \longrightarrow \text{if } e'_1 \text{ then } e_2 \text{ else } e_3} \quad (\text{E-IFCOND}) \\
\\
\frac{}{\rho \vdash \text{if true then } e_2 \text{ else } e_3 \longrightarrow e_2} \quad (\text{E-IFTRUE}) \\
\\
\frac{}{\rho \vdash \text{if true then } e_2 \text{ else } e_3 \longrightarrow e_3} \quad (\text{E-IFFALSE}) \\
\\
\frac{\rho \vdash e_1 \longrightarrow e'_1}{\rho \vdash \text{let } x = e_1 \in e_2 \longrightarrow \text{let } x = e'_1 \text{ in } e_2} \quad (\text{E-LETBODY}) \\
\\
\frac{\rho \cup \{x \mapsto v_1\} \vdash e_2 \longrightarrow e'_2}{\rho \vdash \text{let } x = v_1 \text{ in } e_2 \longrightarrow \text{let } x = v_1 \text{ in } e'_2} \quad (\text{E-LETREM})
\end{array}$$

$$\frac{}{\rho \vdash \text{let } x = v_1 \text{ in } v_2 \longrightarrow v_2} \quad (\text{E-LETVALUE})$$

(E-LETREC)

TODO: define it.

$$\frac{}{\rho \vdash \lambda x. e \longrightarrow (x, e, \rho)} \quad (\text{E-ABS})$$

3 YuchiKaml Interpreter

3.1 Usage

3.2 Preprocess