Lecture 5 – Identifiers (2)

COSE212: Programming Languages

Jihyeok Park

PLRG

2023 Fall

Recall

- **Identifiers**
  - Bound identifiers
  - Free identifiers
  - Shadowing
- **VAE – AE with variables**
  - Concrete syntax
  - Abstract syntax
- In this lecture, let's learn **natural semantics** for VAE, and implement an **interpreter** for VAE.

Contents

Contents

Evaluation with Environments

Let's evaluate the following VAE expressions:

```
/* VAE */
val x = 1; {        // [ x -> 1 ]
  val y = 2; {      // [ x -> 1, y -> 2 ]
    x + y           // x + y = 1 + 2 = 3
  }
}
```

How to evaluate the expression `x + y` into the value 3?

$$\vdash x + y \Rightarrow 3$$

We need to keep track of the **environment** that maps identifiers to values:

$$[x \mapsto 1, y \mapsto 2] \vdash x + y \Rightarrow 3$$

Evaluation with Environments

```scala
type Value = BigInt                    // values
def interp(expr: Expr): Value = ... // interpreter
```

$$\boxed{\vdash e \Rightarrow n}$$

Originally, the interpreter takes an expression and returns a value.

Evaluation with Environments

```scala
type Value = BigInt                                  // values
type Env = Map[String, Int]                          // environments
def interp(expr: Expr, env: Env): Value = ...        // interpreter
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

Now, we extend the interpreter to take an **environment** as well.

For example, we want to evaluate the expression `x + y` into the value 3 with the environment $[x \mapsto 1, y \mapsto 2]$:

```scala
val env : Env   = Map("x" -> 1, "y" -> 2)   // [ x -> 1, y -> 2 ]
val expr: Expr  = Expr("x + y")             // Add(Id("x"), Id("y"))
val v   : Value = interp(expr, env)         // 3
```

$$[x \mapsto 1, y \mapsto 2] \vdash x + y \Rightarrow 3$$

Evaluation with Environments

For VAE, we need to 1) implement the **interpreter** with **environments**

```scala
def interp(expr: Expr, env: Env): Value = ???
```

and 2) define the **natural semantics** with **environments**.

$$\boxed{\sigma \vdash e \Rightarrow n}$$

We read it as "*the* **expression** *e evaluates to the* **number** *n with the* **environment** σ."

We use the following notations:

Expressions	e	(`Expr`)
Environments	$\sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{Z}$	(`Env`)
Integers	$n \in \mathbb{Z}$	(`BigInt`)
Identifiers	$x \in \mathbb{X}$	(`String`)

Contents

Summary

PLRG

The **interpreter** for VAE:

```
def interp(expr: Expr, env: Env): Value = expr match
  case Num(n)         => ???
  case Add(l, r)      => ???
  case Mul(l, r)      => ???
  case Val(x, e, b) => ???
  case Id(x)          => ???
```

The inference rule of each case for the **natural semantics** of VAE:

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$
\begin{array}{llll}
\text{Expressions} & e ::= & n & (\texttt{Num}) \\
 & \mid & e + e & (\texttt{Add}) \\
 & \mid & e \times e & (\texttt{Mul}) \\
 & \mid & \texttt{val}\ x = e;\ e & (\texttt{Val}) \\
 & \mid & x & (\texttt{Id})
\end{array}
$$

Numbers

```scala
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Num(n)       => ???
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Num}\ \frac{???}{\sigma \vdash n \Rightarrow ???}$$

Numbers

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Num(n)       => n
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Num}\ \frac{}{\sigma \vdash n \Rightarrow n}$$

The **expression** n evaluates to the **number** n with the **environment** σ.

Addition

```scala
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Add(l, r)     => ???
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Add}\ \frac{???}{\sigma \vdash e_1 + e_2 \Rightarrow ???}$$

Addition

```scala
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Add(l, r)    => interp(l, env) + interp(r, env)
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Add}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2}$$

The **expression** $e_1 + e_2$ evaluates to the **number** $n_1 + n_2$ with the **environment** σ when

1. The **expression** e_1 evaluates to the **number** n_1 with the **environment** σ.
2. The **expression** e_2 evaluates to the **number** n_2 with the **environment** σ.

Multiplication

PLRG

```scala
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Mul(l, r)     => interp(l, env) * interp(r, env)
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Mul}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 \times e_2 \Rightarrow n_1 \times n_2}$$

The **expression** $e_1 \times e_2$ evaluates to the **number** $n_1 \times n_2$ with the **environment** σ when

1. The **expression** e_1 evaluates to the **number** n_1 with the **environment** σ.
2. The **expression** e_2 evaluates to the **number** n_2 with the **environment** σ.

Variable Definition

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Val(x, e, b) => ???
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Val}\ \frac{???}{\sigma \vdash \texttt{val}\ x = e_1;\ e_2 \Rightarrow ???}$$

Variable Definition

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Val(x, e, b) => ... interp(e, env) ...
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\text{VAL}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \ldots}{\sigma \vdash \texttt{val}\ x = e_1;\ e_2 \Rightarrow ???}$$

The **expression** $\texttt{val}\ x = e_1;\ e_2$ evaluates to the **number** ??? with the **environment** σ when

1. The **expression** e_1 evaluates to the **number** n_1 with the **environment** σ.
2. ...

Variable Definition

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Val(x, e, b) => ... env + (x -> interp(e, env)) ...
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Val}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma[x \mapsto n_1] \qquad \ldots}{\sigma \vdash \texttt{val}\ x = e_1\texttt{;}\ e_2 \Rightarrow \texttt{???}}$$

The **expression** `val` $x = e_1$`;` e_2 evaluates to the **number** ??? with the **environment** σ when

1. The **expression** e_1 evaluates to the **number** n_1 with the **environment** σ.
2. ... the **environment** $\sigma[x \mapsto n_1]$.

Variable Definition

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Val(x, e, b) => interp(b, env + (x -> interp(e, env)))
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Val}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma[x \mapsto n_1] \vdash e_2 \Rightarrow n_2}{\sigma \vdash \texttt{val}\ x = e_1;\ e_2 \Rightarrow n_2}$$

The **expression** $\texttt{val}\ x = e_1;\ e_2$ evaluates to the **number** n_2 with the **environment** σ when

1. The **expression** e_1 evaluates to the **number** n_1 with the **environment** σ.
2. The **expression** e_2 evaluates to the **number** n_2 with the **environment** $\sigma[x \mapsto n_1]$.

Variable Lookup

```
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Id(x)        => ???
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Id}\ \frac{???}{\sigma \vdash x \Rightarrow ???}$$

Variable Lookup

```scala
def interp(expr: Expr, env: Env): Value = expr match
  ...
  case Id(x)        => env.getOrElse(x, error(s"unknown identifier: $x")
    )
  ...
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Id}\ \frac{x \in \mathsf{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

The **expression** x evaluates to the **number** $\sigma(x)$ with the **environment** σ when

1. The **variable** x is in the domain of the **environment** σ.

Contents

Example 1

$$
\textsc{Val}\ \dfrac{\textsc{Num}\ \dfrac{}{\varnothing \vdash 1 \Rightarrow 1} \quad \textsc{Add}\ \dfrac{\textsc{Id}\ \dfrac{x \in \text{Domain}([x \mapsto 1])}{[x \mapsto 1] \vdash x \Rightarrow 1} \quad \textsc{Num}\ \dfrac{}{[x \mapsto 1] \vdash 2 \Rightarrow 2}}{[x \mapsto 1] \vdash x + 2 \Rightarrow 3}}{\varnothing \vdash \texttt{val}\ x = 1;\ x + 2 \Rightarrow 3}
$$

We can name environments σ_i to make the derivation tree concise.

$$
\textsc{Val}\ \dfrac{\textsc{Num}\ \dfrac{}{\varnothing \vdash 1 \Rightarrow 1} \quad \textsc{Add}\ \dfrac{\textsc{Id}\ \dfrac{x \in \text{Domain}(\sigma_0)}{\sigma_0 \vdash x \Rightarrow 1} \quad \textsc{Num}\ \dfrac{}{\sigma_0 \vdash 2 \Rightarrow 2}}{\sigma_0 \vdash x + 2 \Rightarrow 3}}{\varnothing \vdash \texttt{val}\ x = 1;\ x + 2 \Rightarrow 3}
$$

where

$$
\sigma_0 \quad = \quad [x \mapsto 1]
$$

Example 2

$$
\textsc{Val}\ \dfrac{
  \textsc{Num}\ \dfrac{}{\varnothing \vdash 1 \Rightarrow 1}
  \quad
  \textsc{Val}\ \dfrac{
    \textsc{Num}\ \dfrac{}{\sigma_0 \vdash 2 \Rightarrow 2}
    \quad
    \textsc{Add}\ \dfrac{
      \textsc{Id}\ \dfrac{x \in \mathsf{Domain}(\sigma_1)}{\sigma_1 \vdash x \Rightarrow 1}
      \quad
      \textsc{Id}\ \dfrac{y \in \mathsf{Domain}(\sigma_1)}{\sigma_1 \vdash y \Rightarrow 2}
    }{\sigma_1 \vdash x + y \Rightarrow 3}
  }{\sigma_0 \vdash \texttt{val}\ y = 2;\ x + y \Rightarrow 3}
}{\varnothing \vdash \texttt{val}\ x = 1;\ \{\texttt{val}\ y = 2;\ x + y\} \Rightarrow 3}
$$

where

$$
\begin{array}{rcl}
\sigma_0 & = & [x \mapsto 1] \\
\sigma_1 & = & [x \mapsto 1, y \mapsto 2]
\end{array}
$$

Example 3

$$
\textsc{Val}\ \dfrac{
\textsc{Num}\ \dfrac{}{\varnothing \vdash 1 \Rightarrow 1}
\quad
\textsc{Add}\ \dfrac{
\textsc{Val}\ \dfrac{
\textsc{Num}\ \dfrac{}{\sigma_0 \vdash 2 \Rightarrow 2}
\quad
\textsc{Id}\ \dfrac{x \in \mathrm{Domain}(\sigma_1)}{\sigma_1 \vdash x \Rightarrow 2}
}{\sigma_0 \vdash \texttt{val}\ x = 2;\ x \Rightarrow 2}
\quad
\textsc{Id}\ \dfrac{x \in \mathrm{Domain}(\sigma_0)}{\sigma_0 \vdash x \Rightarrow 1}
}{\sigma_0 \vdash \{\texttt{val}\ x = 2;\ x\} + x \Rightarrow 3}
}{\varnothing \vdash \texttt{val}\ x = 1;\ \{\texttt{val}\ x = 2;\ x\} + x \Rightarrow 3}
$$

where

$$
\begin{array}{rcl}
\sigma_0 & = & [x \mapsto 1] \\
\sigma_1 & = & [x \mapsto 2]
\end{array}
$$

Example 4

$$
\textsc{Val}\ \dfrac{\textsc{Val}\ \dfrac{\textsc{Num}\ \dfrac{}{\varnothing \vdash 1 \Rightarrow 1} \quad \textsc{Id}\ \dfrac{x \in \mathsf{Domain}(\sigma_0)}{\sigma_0 \vdash x \Rightarrow 1}}{\varnothing \vdash \texttt{val}\ x = 1;\ x \Rightarrow 1} \quad \textsc{Id}\ \dfrac{\color{red}{x \notin \mathsf{Domain}(\varnothing)}}{\varnothing \vdash x \Rightarrow \color{red}{\mathsf{FAIL}}}}{\varnothing \vdash \{\texttt{val}\ x = 1;\ x\} + x \Rightarrow \color{red}{\mathsf{FAIL}}}
$$

where

$$
\sigma_0 \quad = \quad [x \mapsto 1]
$$

Summary

```scala
def interp(expr: Expr, env: Env): Value = expr match
  case Num(n)       => n
  case Add(l, r)    => interp(l, env) + interp(r, env)
  case Mul(l, r)    => interp(l, env) * interp(r, env)
  case Val(x, e, b) => interp(b, env + (x -> interp(e, env)))
  case Id(x)        => env.getOrElse(x, error(s"unknown variable: $x"))
```

$$\boxed{\sigma \vdash e \Rightarrow n}$$

$$\textsc{Num}\ \frac{}{\sigma \vdash n \Rightarrow n}$$

$$\textsc{Add}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 \times e_2 \Rightarrow n_1 \times n_2} \qquad \textsc{Mul}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 \times e_2 \Rightarrow n_1 \times n_2}$$

$$\textsc{Val}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma[x \mapsto n_1] \vdash e_2 \Rightarrow n_2}{\sigma \vdash \texttt{val}\ x = e_1;\ e_2 \Rightarrow n_2} \qquad \textsc{Id}\ \frac{x \in \mathrm{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

Exercise #2

- Please see this document[1] on GitHub.
  - Implement `interp` function.
  - Implement `freeIds` function.
  - Implement `bindingIds` function.
  - Implement `boundIds` function.
  - Implement `shadowedIds` function.
- It is just an exercise, and you **don't need to submit** anything.
- However, some exam questions might be related to this exercise.

[1] `https://github.com/ku-plrg-classroom/docs/tree/main/cose212/vae`.

Next Lecture

- First-Order Functions

Jihyeok Park
jihyeok_park@korea.ac.kr
`https://plrg.korea.ac.kr`