Lecture 5 – Identifiers (2)

COSE212: Programming Languages

Jihyeok Park

2026 Fall

Recall

- **Identifiers**
  - Bound identifiers
  - Free identifiers
  - Shadowing
- VAE – AE **with variables**
  - Concrete syntax
  - Abstract syntax

- In this lecture, we will
  - implement the **interpreter** for VAE
  - define the **natural semantics** for VAE

Contents

1. Evaluation with Environments

2. Interpreter and Natural Semantics for VAE
   - Numbers
   - Addition and Multiplication
   - Variable Definition
   - Variable Lookup

3. Examples

Contents

1. Evaluation with Environments

2. Interpreter and Natural Semantics for VAE
    - Numbers
    - Addition and Multiplication
    - Variable Definition
    - Variable Lookup

3. Examples

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?

$$\vdash \mathrm{x} + \mathrm{y} \Rightarrow 3$$

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

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

Evaluation with Environments

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

For AE, the interpreter takes an expression and returns a number.

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

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

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

$$\sigma \vdash e \Rightarrow n$$

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

For example, the interpreter should be able to evaluate like this:

```
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
```

$$[\mathtt{x} \mapsto 1$$

Interpreter and Natural Semantics 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)        => ???
```

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

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

$$\textsc{Add}\ \frac{???}{\sigma \vdash e_1\ \texttt{+}\ e_2 \Rightarrow ???} \qquad \textsc{Mul}\ \frac{???}{\sigma \vdash e_1\ \texttt{*}\ e_2 \Rightarrow ???}$$

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

Addition

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

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

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

Numbers

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

$$\boxed{\texttt{`}\, e \,)\, n}$$

$$\text{Num}\ \frac{}{\texttt{`}\, n \,)\, n}$$

With the environment , the expression n evaluates to the number n.

Exercise #2

https://github.com/ku-plrg-classroom/docs/tree/main/cose212/vae

- Please see above document 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, so we **strongly recommend** solving it by yourself.

Next Lecture

- First-Order Functions

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