Lecture 4. DefineLang – Global Variables

February 1, 2021

Local and Global Variables

- Local variable: available based on the scope of the let expression
- Global variable: available during the entire iteration with the interpreter
  - (define Sun 0)
  - (define a 97)

- DefineLang: new feature added to VarLang, called *define declaration*
- Syntax: keyword define, name, initial value
- Example:
  (define i 1)
  (define ii 2)
  (* i ii)

Examples

```
$ (define R 8.3145) // The gas constant R
$ (define n 2) // 2 moles of gas
$ (define V 0.0224) // Volume of gas 0.0224 m^2
$ (define T 273) // Temperature of gas 273 K
$ (define P (/ (* n R T) V)) // Using Boyles law to compute pressure
$ P      //What is the pressure?
202665.93750000003
```

```
$ (define F 96454.56) (define R 10973731.6)
```

The Definelang language also permits defining one or more constants and then computing the value of an expression.

```
$ (define R 8.3145) (/ (* 2 R 273) 0.0224)
202665.93750000003
```

DefineLang Demo

```
Type a program to evaluate and press the enter key, e.g. (let ((a 3) (b 100) (c 84) (d 279) (e 277)) (+ (* a b) (/ c (
Press Ctrl + C to exit.
(define P 3.1415926)
$
P
$ 3.1415926
(let ((P 3)) (* 2 2 P))
$ 12
P
$ 3.1415926
(define P (* 3.14 2 2))
$
P
$ 12.56
(define P (let ((r 2)) (* 3.14 r r)))
$
P
$ 12.56
(define P 3.14) (define r 2)
$
P
$ 3.14
r
$ 2
```

Grammar

`Program`	::=	`DefineDecl*` `Exp?`	*Program*
`DefineDecl`	::=	`(define Identifier Exp)`	*Define*
`Exp`	::=		*Expressions*
		`Number`	*NumExp*
	|	`(+ Exp Exp+)`	*AddExp*
	|	`(- Exp Exp+)`	*SubExp*
	|	`(* Exp Exp+)`	*MultExp*
	|	`(/ Exp Exp+)`	*DivExp*
	|	`Identifier`	*VarExp*
	|	`(let ((Identifier Exp)+) Exp)`	*LetExp*
`Number`	::=	`Digit`	*Number*
	|	`DigitNotZero Digit+`	
`Digit`	::=	`[0-9]`	*Digits*
`DigitNotZero`	::=	`[1-9]`	*Non-zero Digits*
`Identifier`	::=	`Letter LetterOrDigit*`	*Identifier*
`Letter`	::=	`[a-zA-Z$_]`	*Letter*
`LetterOrDigit`	::=	`[a-zA-Z0-9$_]`	*LetterOrDigit*

Extending AST (syntax): Read Phase

1. New AST node: DefineDecl
2. Modify program to store DefineDecl
3. Modify visitor interface to support DefineDecl
4. Modify formatter regrading print DefineDecl

Extending Semantics: Eval

- Varlang: evaluate a program starting in an empty environment; DefineLang: when a program starts running, the declared global variables are defined; the program can have free variables
- Unitval: A UnitVal is like a void type in Java. It allows programming language definitions and implementations to uniformly treat programs and expressions as evaluating to 'a value' – e.g., a define declaration

`Value`	::=		*Values*
		`NumVal`	*Numeric Values*
	|	`UnitVal`	*Unit Values*
`NumVal`	::=	`(NumVal n)`, where `n` $\in$ the set of doubles	*NumVal*
`UnitVal`	::=	`(UnitVal)`	*NumVal*

- each definition changes the global initEnv to add a new binding from name to value.

Review and Further Reading

Definelang: support globals

- Syntax: definition declaration (AST node, visitor interface)
- Semantics: modify environment for each run; unitval;

Further reading:

- Rajan: CH 4, Sebesta Ch 7, 8