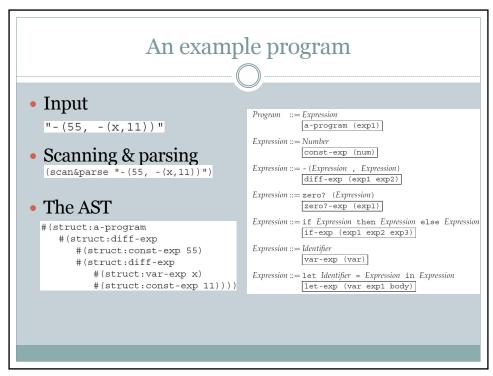
# Lecture 12 Let

T. METIN SEZGIN

1

# LET: our pet language Program ::= Expression [a-program (exp1)] Expression ::= Number [const-exp (num)] Expression ::= - (Expression , Expression) [diff-exp (exp1 exp2)] Expression ::= zero? (Expression) [zero?-exp (exp1)] Expression ::= if Expression then Expression [if-exp (exp1 exp2 exp3)] Expression ::= Identifier [var-exp (var)] Expression ::= let Identifier = Expression in Expression [let-exp (var exp1 body)]



# Nugget Steps of inventing a language

### Components of the language

- Syntax and datatypes
- Values
- Environment
- Behavior specification
- Behavior implementation
  - Scanning
  - Parsing
  - Evaluation

5

### Syntax data types (define-datatype program program? (a-program (exp1 expression?))) Program ::= Expression (define-datatype expression expression? a-program (exp1) (const-exp (num number?)) Expression ::= Numberconst-exp (num) (diff-exp (exp1 expression?) (exp2 expression?)) (zero?-exp $\textit{Expression} ::= \underline{- (Expression \ , \ Expression)}$ diff-exp (exp1 exp2) (exp1 expression?)) $\textit{Expression} ::= \underline{\texttt{zero?}} \ \ (\textit{Expression})$ (if-exp zero?-exp (exp1) (exp1 expression?) (exp2 expression?) (exp3 expression?)) $\label{eq:expression} \textit{Expression} \ \text{then} \ \textit{Expression} \ \text{else} \ \textit{Expression}$ if-exp (exp1 exp2 exp3) (var-exp (var identifier?)) (let-exp Expression ::= Identifiervar-exp (var) $Expression ::= \texttt{let} \ \textit{Identifier} \ = \ \textit{Expression} \ \texttt{in} \ \textit{Expression}$ (var identifier?) let-exp (var exp1 body) (exp1 expression?) (body expression?)))

Nugget

### Values

7

### Values

- Set of values manipulated by the program
  - Expressed values
    - Possible values of expressions
  - Denoted values
    - Possible values of variables
- ExpVal = Int + BoolDenVal = Int + Bool

- Interface for values
  - Constructors
  - Observers

 $\begin{array}{lll} \textbf{num-val} & : Int \rightarrow ExpVal \\ \textbf{bool-val} & : Bool \rightarrow ExpVal \\ \textbf{expval->num} & : ExpVal \rightarrow Int \\ \textbf{expval->bool} & : ExpVal \rightarrow Bool \\ \end{array}$ 

Q

### **Environments**

- Same model of environment from before
  - $\rho$  ranges over environments.
  - [] denotes the empty environment.
  - $[var = val]\rho$  denotes (extend-env  $var \ val \ \rho$ ).
  - $[var_1 = val_1, var_2 = val_2]\rho$  abbreviates  $[var_1 = val_1]([var_2 = val_2]\rho)$ , etc.
  - [ $var_1 = val_1, var_2 = val_2, \ldots$ ] denotes the environment in which the value of  $var_1$  is  $val_1$ , etc.
- Use [y=7]

[x=3] [y=7] [u=5]ρ

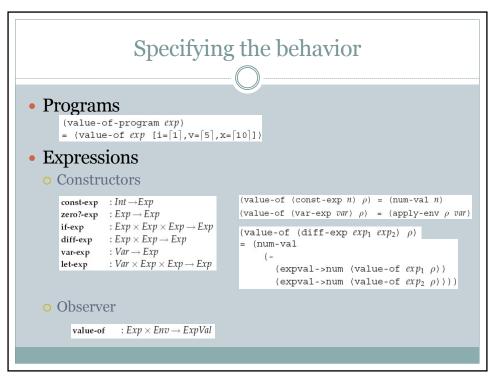
to abbreviate

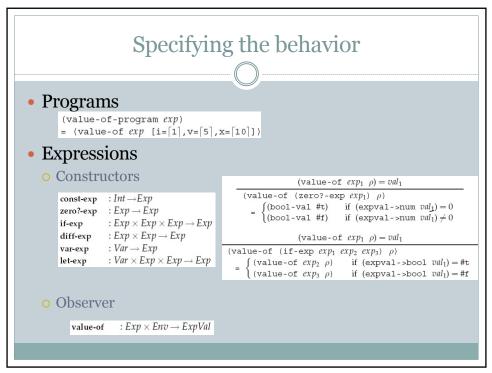
(extend-env 'x 3 (extend-env 'y 7 (extend-env 'u 5 ρ)))

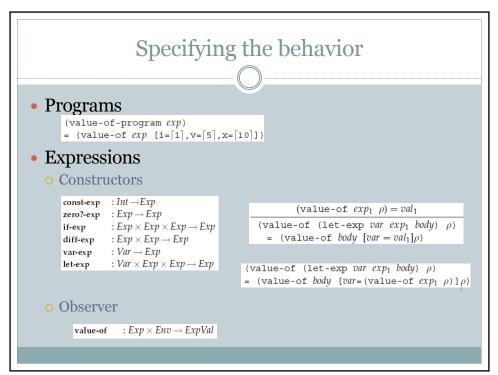
C

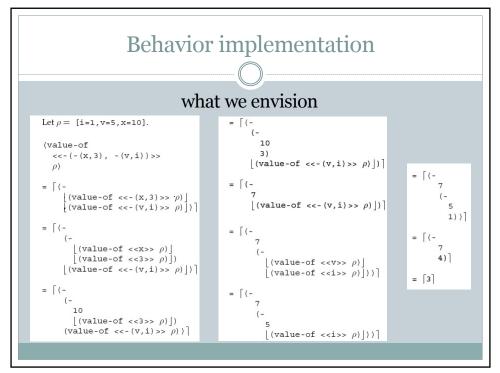
### Nugget

We specify the meaning of expressions first









## Lecture 13 Let – Implementation

T. METIN SEZGIN

15

## 

### Nugget

# Intro to implementation It all revolves around **value-of**

17

### The Interpreter

