

Compilers and Formal Languages

Email: christian.urban at kcl.ac.uk

Slides & Progs: KEATS (also homework is there)

1 Introduction, Languages	6 While-Language
2 Regular Expressions, Derivatives	7 Compilation, JVM
3 Automata, Regular Languages	8 Compiling Functional Languages
4 Lexing, Tokenising	9 Optimisations
5 Grammars, Parsing	10 LLVM

```
start := 1000;
x := start;
y := start;
z := start;
while 0 < x do {
  while 0 < y do {
    while 0 < z do { z := z - 1 };
    z := start;
    y := y - 1
  };
  y := start;
  x := x - 1
}
```

While-Language

$Stmt ::= skip$
| $Id := AExp$
| if $BExp$ then $Block$ else $Block$
| while $BExp$ do $Block$

$Stmts ::= Stmt ; Stmts$
| $Stmt$

$Block ::= \{ Stmts \}$
| $Stmt$

$AExp ::= \dots$

$BExp ::= \dots$

Aexprs

$$\text{eval}(n) \stackrel{\text{def}}{=} n$$

$$\text{eval}(a_1 + a_2) \stackrel{\text{def}}{=} \text{eval}(a_1) + \text{eval}(a_2)$$

$$\text{eval}(a_1 - a_2) \stackrel{\text{def}}{=} \text{eval}(a_1) - \text{eval}(a_2)$$

$$\text{eval}(a_1 * a_2) \stackrel{\text{def}}{=} \text{eval}(a_1) * \text{eval}(a_2)$$

$$\text{eval}(x) \stackrel{\text{def}}{=} ???$$

Interpreter

$$\text{eval}(n, E) \stackrel{\text{def}}{=} n$$

$$\text{eval}(x, E) \stackrel{\text{def}}{=} E(x) \quad \text{lookup } x \text{ in } E$$

$$\text{eval}(a_1 + a_2, E) \stackrel{\text{def}}{=} \text{eval}(a_1, E) + \text{eval}(a_2, E)$$

$$\text{eval}(a_1 - a_2, E) \stackrel{\text{def}}{=} \text{eval}(a_1, E) - \text{eval}(a_2, E)$$

$$\text{eval}(a_1 * a_2, E) \stackrel{\text{def}}{=} \text{eval}(a_1, E) * \text{eval}(a_2, E)$$

$$\text{eval}(a_1 = a_2, E) \stackrel{\text{def}}{=} \text{eval}(a_1, E) = \text{eval}(a_2, E)$$

$$\text{eval}(a_1 \neq a_2, E) \stackrel{\text{def}}{=} \neg(\text{eval}(a_1, E) = \text{eval}(a_2, E))$$

$$\text{eval}(a_1 < a_2, E) \stackrel{\text{def}}{=} \text{eval}(a_1, E) < \text{eval}(a_2, E)$$

An Interpreter (1)

```
{  
  x := 5;  
  y := x * 3;  
  y := x * 4;  
  x := u * 3  
}
```

the interpreter has to record the value of x before assigning a value to y

An Interpreter (1)

```
{  
  x := 5;  
  y := x * 3;  
  y := x * 4;  
  x := u * 3  
}
```

the interpreter has to record the value of x before
assigning a value to y

`eval(stmt, env)`

Interpreter (2)

$$\text{eval}(\text{skip}, E) \stackrel{\text{def}}{=} E$$

$$\text{eval}(x := a, E) \stackrel{\text{def}}{=} E(x \mapsto \text{eval}(a, E))$$

$$\text{eval}(\text{if } b \text{ then } cs_1 \text{ else } cs_2, E) \stackrel{\text{def}}{=} \\ \text{if } \text{eval}(b, E) \text{ then } \text{eval}(cs_1, E) \\ \text{else } \text{eval}(cs_2, E)$$

$$\text{eval}(\text{while } b \text{ do } cs, E) \stackrel{\text{def}}{=} \\ \text{if } \text{eval}(b, E) \\ \text{then } \text{eval}(\text{while } b \text{ do } cs, \text{eval}(cs, E)) \\ \text{else } E$$

$$\text{eval}(\text{write } x, E) \stackrel{\text{def}}{=} \{ \text{println}(E(x)) ; E \}$$

Interpreted Code

