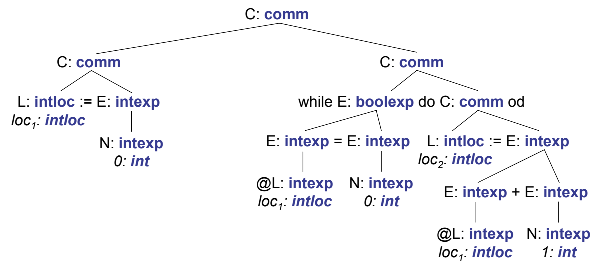
Exercise Session Informatik III

1. Simple Imperative Language L_c

Attributed Syntax Tree



globis

Blackbox Function *m(x)*

- To prove this exercise we define the following **blackbox function** *m(x)*
- The function *m(x)* will return **true** iff $x \in C$ contains the same number of **if** as **fi**, and **false** otherwise.

$$m(x) = \begin{cases} \textit{true} & \#if = \#fi \\ \textit{false} & \textit{otherwise} \end{cases}$$

globis

Proof

Now we have to check if *m(x)* = *true* holds for all $x \in C$

- *L := E* *m(x)* = *true* (#if = #fi)
- **skip** *m(x)* = *true* (#if = #fi)
- **if** E **then** C_1 **else** C_2 **fi** ...

globis

Semantic Valuation Functions

[[C]] = [[loc$_1$:= 1; loc$_2$:= @loc$_1$ + 1; **skip**]]
[[C]] = [[C$_1$; C$_2$; C$_3$]] = [[C$_3$]]([[C$_2$]]([[C$_1$]](s)))
s = $\langle 0,0,0,0 \rangle$
[[**skip**: comm]]([[loc$_2$:= @loc$_1$ + 1: comm]]([[loc$_1$:= 1: comm]](s)))
[[**skip**: comm]]([[loc$_2$:= @loc$_1$ + 1: comm]](*update*([[loc$_1$: intloc]], [[1: intexp]](s), s)))
s = $\langle 1,0,0,0 \rangle$
[[**skip**: comm]]([[loc$_2$:= @loc$_1$ + 1: comm]](s))
[[**skip**: comm]](*update*([[loc$_2$: intloc]], [[@loc$_1$ + 1: intexp]](s), s))
[[**skip**: comm]](*update*([[loc$_2$: intloc]], *plus*([[@loc$_1$: intexp]](s), [[1: intexp]](s)), s))
[[**skip**: comm]](*update*([[loc$_2$: intloc]], *plus*(lookup([[loc$_1$: intloc]], s), [[1: intexp]](s), s))
[[**skip**: comm]](*update*([[loc$_2$: intloc]], *plus*([[1: intexp]](s), [[1: intexp]](s), s))
[[**skip**: comm]](*update*([[loc$_2$: intloc]], [[2: intexp]](s), s))
s = $\langle 1,2,0,0 \rangle$
[[**skip**: comm]](s)
s = $\langle 1,2,0,0 \rangle$

globis

Extending L_c with Booleans

To change L_c to handle **boolean values**, we have to extend

- syntax rules
- typing rules
- semantic valuation functions
- operations

globis

Syntax Rules

B := true | false

E := N | ... | $E_1 = E_2$ | B

Typing Rules

$\{true, false\} : bool$

$$\frac{B : bool}{B : boolexp}$$

$$\frac{L : intloc}{@L : \tau exp} \quad for\ \tau \in \{int, bool\}$$

$$\frac{L : intloc \quad E : \tau exp}{L := E : comm} \quad for\ \tau \in \{int, bool\}$$

Semantic Valuation Functions

$[[B : boolexp]](s) = [[B : bool]]$

$[[B : bool]] = b$

$[[L := E : comm]](s) = update([[L : intloc]], [[E : \tau exp]](s), s)$
for $\tau \in \{int, bool\}$

$[[@L : \tau exp]](s) = lookup([[L : intloc]], s)$
for $\tau \in \{int, bool\}$

Operations

$Store = \{<n_1, n_2, \ldots n_m> \mid n_i \in \{int, bool\}, 1 \le i \le m, m \ge 1\}$

$lookup : Location \times Store \to \{int, bool\}$

$update : Location \times \{int, bool\} \times Store \to Store$

Simple Calculator

- **P** Program $\to$ Nat
  - [[ON S]] = [[S]](0)
- **S** ExprSequence $\to$ Nat $\to$ Nat* (Output)
  - [[E SUM S]](n) = *let* n' = [[E]](n) *in* n' **cons** [[S]](n')
  - [[E SUM OFF]](n) = [[E]](n) **cons** nil
- **E** Expression $\to$ Nat $\to$ Nat
  - $[[E_1 + E_2]](n) = plus([[E_1]](n), [[E_2]](n))$
  - $[[E_1 * E_2]](n) = times([[E_1]](n), [[E_2]](n))$
  - $[[E_1 - E_2]](n) = minus([[E_1]](n), [[E_2]](n))$

Simple Calculator

- **E** Expression $\to$ Nat $\to$ Nat
  - $[[IF\ E_1, E_2, E_3]](n) =$
    if $[[E_1]](n)$ equals zero then $[[E_2]](n)$ else $[[E_3]](n)$
  - [[IT]](n) = n
  - [[(E)]](n) = [[E]](n)
  - [[N]](n) = [[N]]
- **N** Numeral $\to$ Nat
  - Maps numeral N to corresponding $n \in$ Nat

Unicity of Typing

To proof the exercise we have to a case analysis for all $c \in C$

i) L := E
ii) **skip**
iii) C_1; C_2
iv) **while** E **do** C **od**
v) **if** E **then** C_1 **else** C_2 **fi**

That's all folks!