ImpCore

CS 456 - Programming Languages

The tools of a Semanticist:
Syntax

Concrete	• The actual syntax in which programs are written • Could be ambiguous • eg. 3 + 1 + 4 = (3 + 1) + 4 or 3 + (1 + 4) ? • Makes writing programs practical
Abstract	• Produced after syntactic analysis from concrete syntax • Intermediate Representation (IR) • Unambiguous (ambiguity is resolved in syntax analysis) • Makes program manipulation practical • Captures the **structure** of the program

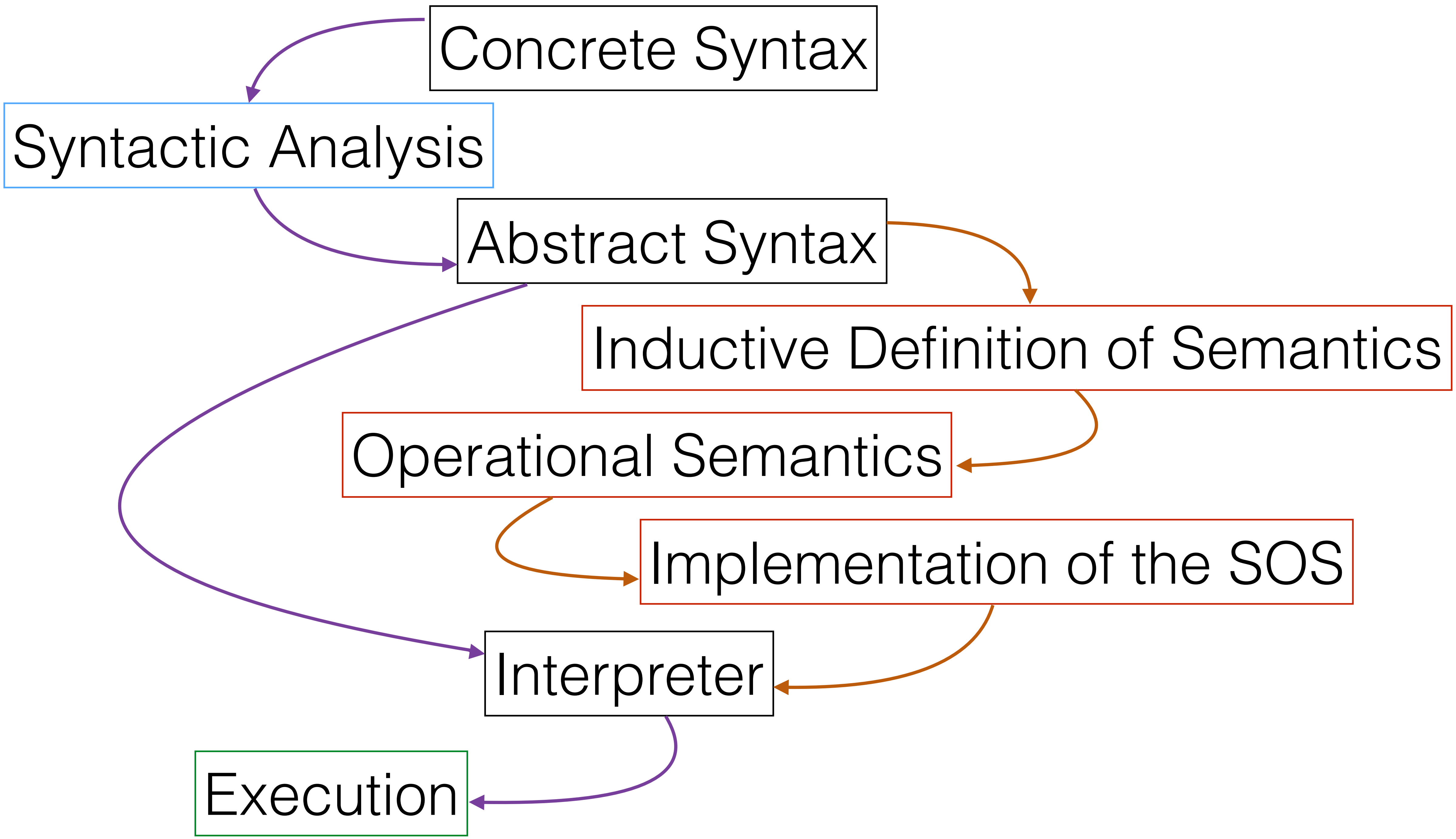
The tools of a Semanticist:
Semantics

Structural Operational (SOS)	• *Syntax Directed* • Gives meaning to entire programs as a relation between the input and the output of the program • No intermediate steps
SOS Small-Step	• Gives meaning of each step of the program • Whole program is the concatenation of all the steps
Denotational	• Gives meaning to the program as a mathematical object (generally a function)
Axiomatic	• Gives meaning to a program as the set of facts that are provable about it

The tools of a Semanticist: Implementation

Interpreter	• The program is analyzed and executed at runtime statement by statement • The interpreter is part of the execution of the input program • No code is “generated” • Usually less efficient than compiled code since we don’t have the whole code of the program
Compiler	• The whole program is analyzed before executing • Code for a target architecture is generated • Usually consists of many (perhaps optimizing) pases • Generally more efficient than interpreted code

The tools of a Semanticist:
Big Picture



Imperative Languages

Pure Imperative

- C
- FORTRAN
- Pascal
- Ada, BASIC, etc …

With Imperative Features

- C++
- Java
- Python
- OCaml, etc…

Characteristics of an Imperative

- Close to the machine architecture (Von Newman)

- *Instructions* that operate one at a time on a small piece of *data* (eg. a register, memory location)

- Arrays and pointers are the most common high-level data structures

- control flow: loops and goto

Syntactic Categories

- Declaration — introduces a *name*
- Definition — *binds* a name
- Statement — produces a *side-effect*
- Expression — produces a *value*

ImpCore: Concrete Syntax

$$\begin{array}{lcl}
\mathit{value} & ::= & \boxed{\mathit{integer}} \\[1em]
\mathit{function} & ::= & \mathit{function\text{-}name} \\
 & | & \mathit{primitive} \\[1em]
\mathit{primitive} & ::= & + \mid - \mid * \mid / \mid = \mid < \mid > \mid \texttt{print} \\[1em]
\mathit{integer} & ::= & \mathit{seq.\ of\ digits} \\[1em]
*\text{-}\mathit{name} & ::= & \mathit{seq.\ of\ chars} - \{\text{“(”},\ \text{“)”},\ \text{“;”},\ \text{“ ”}\}
\end{array}$$

ImpCore: Concrete Syntax

$$
\begin{array}{lcl}
\mathit{def} & ::= & (\texttt{val}\ \mathit{variable\text{-}name}\ \mathit{exp}) \\
 & | & \mathit{exp} \\
 & | & (\texttt{define}\ \mathit{function\text{-}name}\ (\mathit{formals})\ \mathit{exp}) \\
 & | & (\texttt{use}\ \mathit{file\text{-}name}) \\
\\
\mathit{exp} & ::= & \mathit{value} \\
 & | & \mathit{variable\text{-}name} \\
 & | & (\texttt{set}\ \mathit{variable\text{-}name}\ \mathit{exp}) \\
 & | & (\texttt{if}\ \mathit{exp}\ \mathit{exp}\ \mathit{exp}) \\
 & | & (\texttt{while}\ \mathit{exp}\ \mathit{exp}) \\
 & | & (\texttt{begin}\ \{\mathit{exp}\}) \\
 & | & (\mathit{function}\ \{\mathit{exp}\})
\end{array}
$$

$(\texttt{val}\ x\ 3)$ $(\texttt{if}\ 1\ 5\ 8)$ $(\texttt{begin}\ (\texttt{set}\ x\ 8)\ (\texttt{if}\ (=\ x\ 8)\ 3\ 7))$

ImpCore (Examples)

ImpCore: Concrete Syntax

What would change?

$$
\begin{array}{lcl}
exp & ::= & value \\
& | & variable\text{-}name \\
& | & [(\texttt{assign}\ variable\text{-}name\ exp)] \\
& | & [(\texttt{when}\ exp\ exp\ exp)] \\
& | & [(\texttt{whl}\ exp\ exp)] \\
& | & [(\texttt{do}\ \{exp\})] \\
& | & [(function\ \{exp\})]
\end{array}
$$

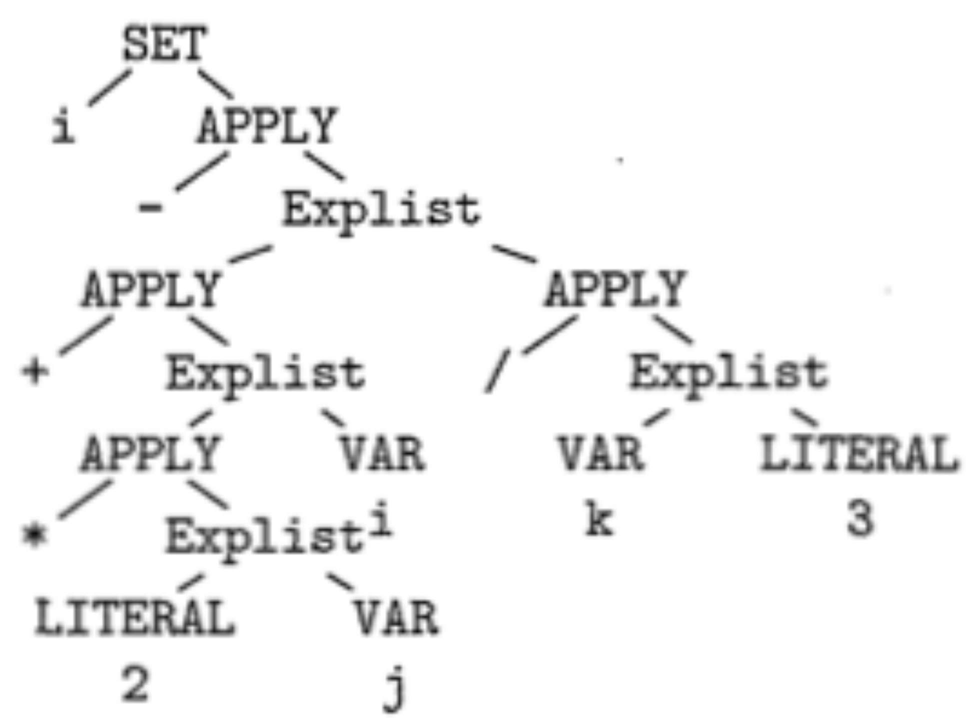
NOTHING!

ImpCore: *Abstract* Syntax (AST)

$$
\begin{array}{lcll}
\texttt{Def} & = & \texttt{VAL} & (\texttt{Name}, \texttt{Exp}) \\
 & | & \texttt{EXP} & (\texttt{Exp}) \\
 & | & \texttt{DEFINE} & (\texttt{Name}, \texttt{Exp}) \\
 & | & \texttt{USE} & (\texttt{name}) \\
\\
\texttt{Exp} & = & \texttt{LITERAL} & (\texttt{Value}) \\
 & | & \texttt{VAR} & (\texttt{Name}) \\
 & | & \texttt{SET} & (\texttt{Name}, \texttt{Exp}) \\
 & | & \texttt{IF} & (\texttt{Exp}, \texttt{Exp}, \texttt{Exp}) \\
 & | & \texttt{WHILE} & (\texttt{Exp}, \texttt{Exp}) \\
 & | & \texttt{BEGIN} & (\texttt{Explist}) \\
 & | & \texttt{APPLY} & (\texttt{Name}, \texttt{Explist})
\end{array}
$$

ImpCore: *Abstract* Syntax (AST)

(set i (- (+ (* 2 j) i) (/ k 3)))



Emphasis is on the *structure*

Environments

- We need a way to formalize the *current state*

$(\texttt{val}\ x\ 0)$

$(\texttt{val}\ y\ 0)$

$(\texttt{set}\ x\ 8)$

$(\texttt{val}\ z\ 3)$

x	8
y	0
z	3

ρ

Query

$$\rho(x) = 8$$
$$\rho(y) = 0$$
$$\rho(z) = 3$$
$$\rho(h) = \perp$$

Update

$$\rho\{x \mapsto v\}(y) = \begin{cases} v & \text{if } y = x \\ \rho(y) & \text{otherwise} \end{cases}$$

$$\rho\{x \mapsto 3\}(x) = 3$$
$$\rho\{x \mapsto 3\}(y) = 0$$

ImpCore: Operational Semantics

$$\textsc{Rule Name}\quad \frac{Antecedent1 \quad Antecedent2}{Consequent}$$

$$Antecedent1 \ \wedge \ Antecedent2 \ \Rightarrow Consequent$$

$$Consequent \ \text{if} \ Antecedent1 \ \wedge \ Antecedent2$$

$$\textsc{Syllogism}\quad \frac{\text{All men are mortal} \qquad \text{Socrates is a man}}{\text{Socrates is mortal}}$$

Generally known as Natural Deduction or Sequent Calculus

ImpCore SOS: State

Expression being evaluated
(eg. (+ 2 (* 4 5)))

Value Env:
Values of Global Vars

$$\langle e, \xi, \phi, \rho \rangle$$

Function Env:
Binding of Function names

Formals Env:
Values of Formal parameters

For definitions $\langle d, \xi, \phi \rangle$

ImpCore SOS: Judgment

$$\langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle$$

Reads: evaluating e in environments ξ ,ϕ and ρ produces value v and environments ξ', ϕ and ρ'

$$\langle d, \xi, \phi \rangle \rightarrow \langle \xi', \phi' \rangle$$

Reads: defining d in environments ξ and ϕ produces new environments ξ' and ϕ'

ImpCore SOS: Judgment

$$\langle (+\ 3\ 5), \xi, \phi, \rho \rangle \Downarrow \langle 8, \xi, \phi, \rho \rangle$$

$$\langle (\texttt{set}\ x\ 5), \xi, \phi, \rho \rangle \Downarrow \langle 5, \xi\{x \mapsto 5\}, \phi, \rho \rangle \quad \text{if } x \notin \mathsf{dom}(\rho)$$

$$\langle (\texttt{set}\ x\ 5), \xi, \phi, \rho \rangle \Downarrow \langle 5, \xi, \phi, \rho\{x \mapsto 5\} \rangle \quad \text{if } x \in \mathsf{dom}(\rho)$$

$$\langle (f\ 2\ 3), \xi, \phi, \rho \rangle \Downarrow \langle 5, \xi, \phi, \rho \rangle \qquad \text{if } \phi(f)(m\ n) = (+\ m\ n)$$

$$\langle (\texttt{val}\ x\ 5), \xi, \phi \rangle \rightarrow \langle \xi\{x \mapsto 5\}, \phi \rangle$$

$$\langle (\texttt{define}\ sum\ (m\ n)\ (+\ m\ n)), \xi, \phi \rangle \rightarrow \langle \xi, \phi\{sum \mapsto [(m\ n)(+\ m\ n)]\} \rangle$$

ImpCore SOS

$$\textsc{Value}\quad \frac{}{\langle \mathrm{LITERAL}(v), \xi, \phi, \rho\rangle \Downarrow \langle v, \xi, \phi, \rho\rangle}$$

$$\textsc{FormalVar}\quad \frac{x \in \mathsf{dom}(\rho)}{\langle \mathrm{VAR}(x), \xi, \phi, \rho\rangle \Downarrow \langle \rho(x), \xi, \phi, \rho\rangle}$$

$$\textsc{GlobalVar}\quad \frac{x \notin \mathsf{dom}(\rho) \qquad x \in \mathsf{dom}(\xi)}{\langle \mathrm{VAR}(x), \xi, \phi, \rho\rangle \Downarrow \langle \xi(x), \xi, \phi, \rho\rangle}$$

ImpCore SOS

$$\textsc{FormalAssignment}\;\frac{x \in \mathsf{dom}(\rho) \qquad \langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle}{\langle \mathrm{SET}(x, e), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho'\{x \mapsto v\} \rangle}$$

$$\textsc{GlobalAssignment}\;\frac{x \notin \mathsf{dom}(\rho) \quad x \in \mathsf{dom}(\xi) \quad \langle e, \xi, \phi, \rho \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle}{\langle \mathrm{SET}(x, e), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi'\{x \mapsto v\}, \phi, \rho' \rangle}$$

ImpCore SOS

$$\textsc{IfTrue}\quad \frac{\langle e_1, \xi, \phi, \rho \rangle \Downarrow \langle v_1, \xi', \phi, \rho' \rangle \qquad v_1 \neq 0 \qquad \langle e_2, \xi', \phi, \rho' \rangle \Downarrow \langle v_2, \xi'', \phi, \rho'' \rangle}{\langle \mathrm{IF}(e_1, e_2, e_3), \xi, \phi, \rho \rangle \Downarrow \langle v_2, \xi'', \phi, \rho'' \rangle}$$

$$\textsc{IfFalse}\quad \frac{\langle e_1, \xi, \phi, \rho \rangle \Downarrow \langle 0, \xi', \phi, \rho' \rangle \qquad \langle e_3, \xi', \phi, \rho' \rangle \Downarrow \langle v, \xi'', \phi, \rho'' \rangle}{\langle \mathrm{IF}(e_1, e_2, e_3), \xi, \phi, \rho \rangle \Downarrow \langle v, \xi'', \phi, \rho'' \rangle}$$

ImpCore SOS

$$\textsc{WhileFalse}\quad \frac{\langle e_1, \xi, \phi, \rho \rangle \Downarrow \langle 0, \xi', \phi, \rho' \rangle}{\langle \mathrm{WHILE}(e_1, e_2), \xi, \phi, \rho \rangle \Downarrow \langle 0, \xi', \phi, \rho' \rangle}$$

$$\textsc{WhileTrue}\quad \frac{\begin{array}{c}\langle e_1, \xi, \phi, \rho \rangle \Downarrow \langle v_1, \xi', \phi, \rho' \rangle \qquad v_1 \neq 0 \\ \langle e_2, \xi', \phi, \rho' \rangle \Downarrow \langle v_2, \xi'', \phi, \rho'' \rangle \\ \langle \mathrm{WHILE}(e_1, e_2), \xi'', \phi, \rho'' \rangle \Downarrow \langle v_3, \xi''', \phi, \rho''' \rangle\end{array}}{\langle \mathrm{WHILE}(e_1, e_2), \xi, \phi, \rho \rangle \Downarrow \langle v_3, \xi''', \phi, \rho''' \rangle}$$

ImpCore SOS

$$\frac{\textsc{EmptyBegin}}{\langle \mathrm{BEGIN}(), \xi, \phi, \rho \rangle \Downarrow \langle 0, \xi, \phi, \rho \rangle}$$

$$\textsc{Begin}\quad \frac{\begin{array}{c} \langle e_0, \xi, \phi, \rho \rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1 \rangle \\ \langle e_1, \xi_1, \phi, \rho_1 \rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2 \rangle \\ \vdots \\ \langle e_{n-1}, \xi_{n-1}, \phi, \rho_{n-1} \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle \end{array}}{\langle \mathrm{BEGIN}(e_0, e_1, \ldots, e_{n-1}), \xi, \phi, \rho \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle}$$

ImpCore SOS

$$\textsc{ApplyUser}\quad \frac{\begin{array}{c}\phi(f) = \mathrm{USER}(\langle x_1, \ldots, x_n \rangle, e) \\ (\forall\ i, x_i \notin \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}) \\ \langle e_0, \xi, \phi, \rho \rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1 \rangle \\ \vdots \\ \langle e_{n-1}, \xi_{n-1}, \phi, \rho_{n-1} \rangle \Downarrow \langle v_n, \xi_n, \phi, \rho_n \rangle \\ \langle e, \xi_n, \phi, \{x_1 \mapsto v_1, \ldots, x_n \mapsto v_n\} \rangle \Downarrow \langle v, \xi', \phi, \rho' \rangle \end{array}}{\langle \mathrm{APPLY}(f, e_0, \ldots, e_{n-1}), \xi, \phi, \rho \rangle \Downarrow \langle v_n, \xi', \phi, \rho_n \rangle}$$

ImpCore SOS

$$\frac{\overset{\textsc{ApplyPrimitive}}{\begin{array}{cc}\phi(f) = \mathrm{PRIMITIVE}(\oplus) & \oplus \in \{+,-,*,/,=,<,>\} \\ \langle e_0, \xi, \phi, \rho\rangle \Downarrow \langle v_1, \xi_1, \phi, \rho_1\rangle & \langle e_1, \xi_1, \phi, \rho_1\rangle \Downarrow \langle v_2, \xi_2, \phi, \rho_2\rangle\end{array}}}{\langle \mathrm{APPLY}(f, e_0, e_1), \xi, \phi, \rho\rangle \Downarrow \langle v_1 \oplus v_2, \xi_2, \phi, \rho_2\rangle}$$

$$\frac{\overset{\textsc{ApplyPrint}}{\phi(f) = \mathrm{PRIMITIVE}(\texttt{print}) \qquad \langle e, \xi, \phi, \rho\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle}}{\langle \mathrm{APPLY}(f, e), \xi, \phi, \rho\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle \quad \text{and ``print'' v}}$$

ImpCore SOS

$$\textsc{DefineGlobal}\quad \frac{\langle e, \xi, \phi, \{\}\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle}{\langle \mathrm{VAL}(x, e), \xi, \phi\rangle \to \langle \xi'\{x \mapsto v\}, \phi\rangle}$$

$$\textsc{DefineFunction}\quad \frac{(\forall\ i, x_i \notin \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\})}{\langle \mathrm{DEFINE}(f, \langle x_1, \ldots, x_n\rangle, e), \xi, \phi\rangle \to \langle \xi, \phi\{f \mapsto \mathrm{USER}(\langle x_1, \ldots, x_n\rangle, e)\}\rangle}$$

$$\textsc{EvalExp}\quad \frac{\langle e, \xi, \phi, \{\}\rangle \Downarrow \langle v, \xi', \phi, \rho'\rangle}{\langle \mathrm{EXP}(e), \xi, \phi\rangle \to \langle \xi'\{\mathtt{it} \mapsto v\}, \phi\rangle}$$

ImpCore SOS

Some examples on the board