Reusable Components

for Formal and Executable Language Specification

L. Thomas van Binsbergen

Royal Holloway, University of London

August 12, 2016

The PLanCompS Approach

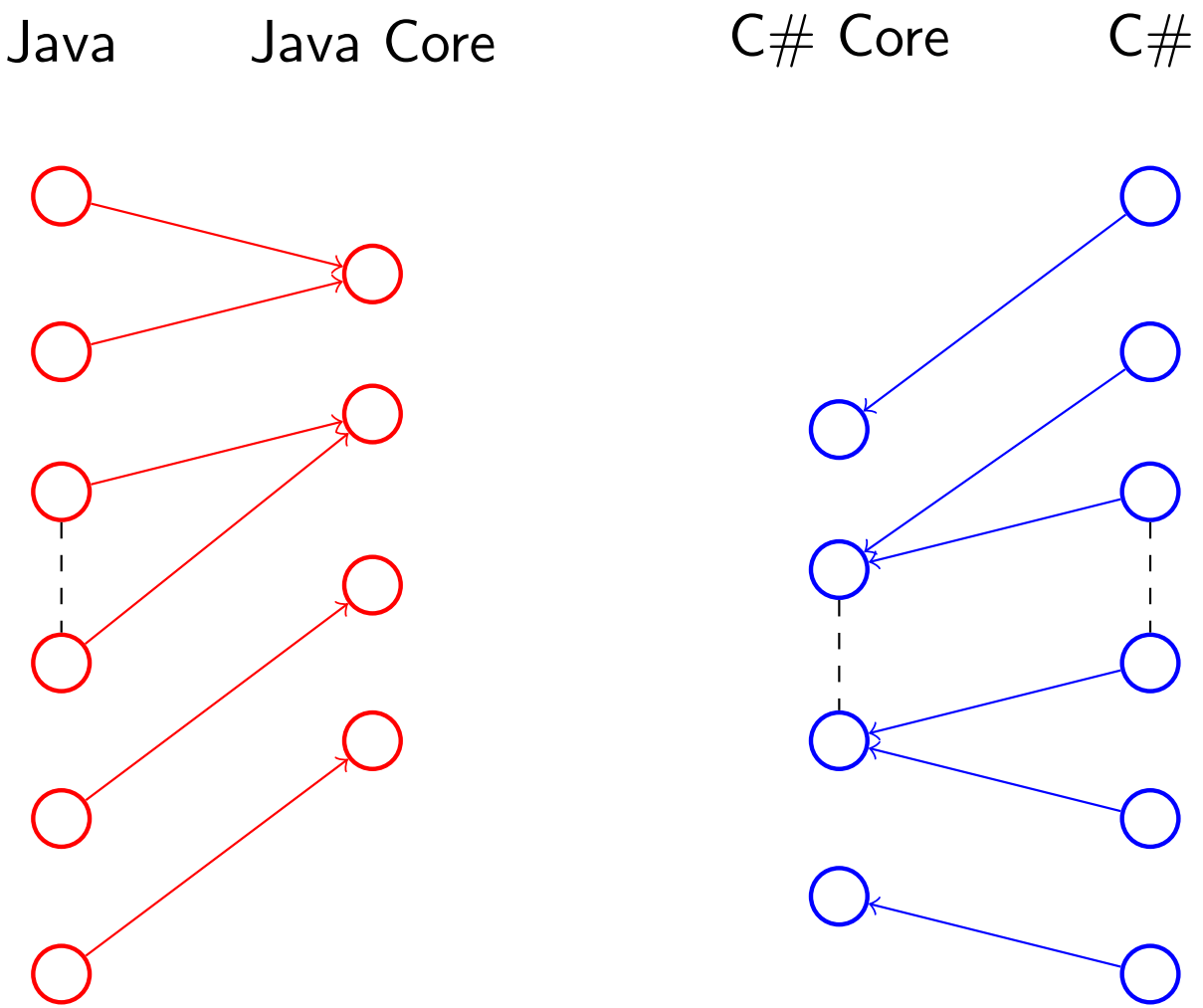
PLanCompS project (2011-2015)

- Swansea University
- Royal Holloway, University of London
- http://plancomps.org

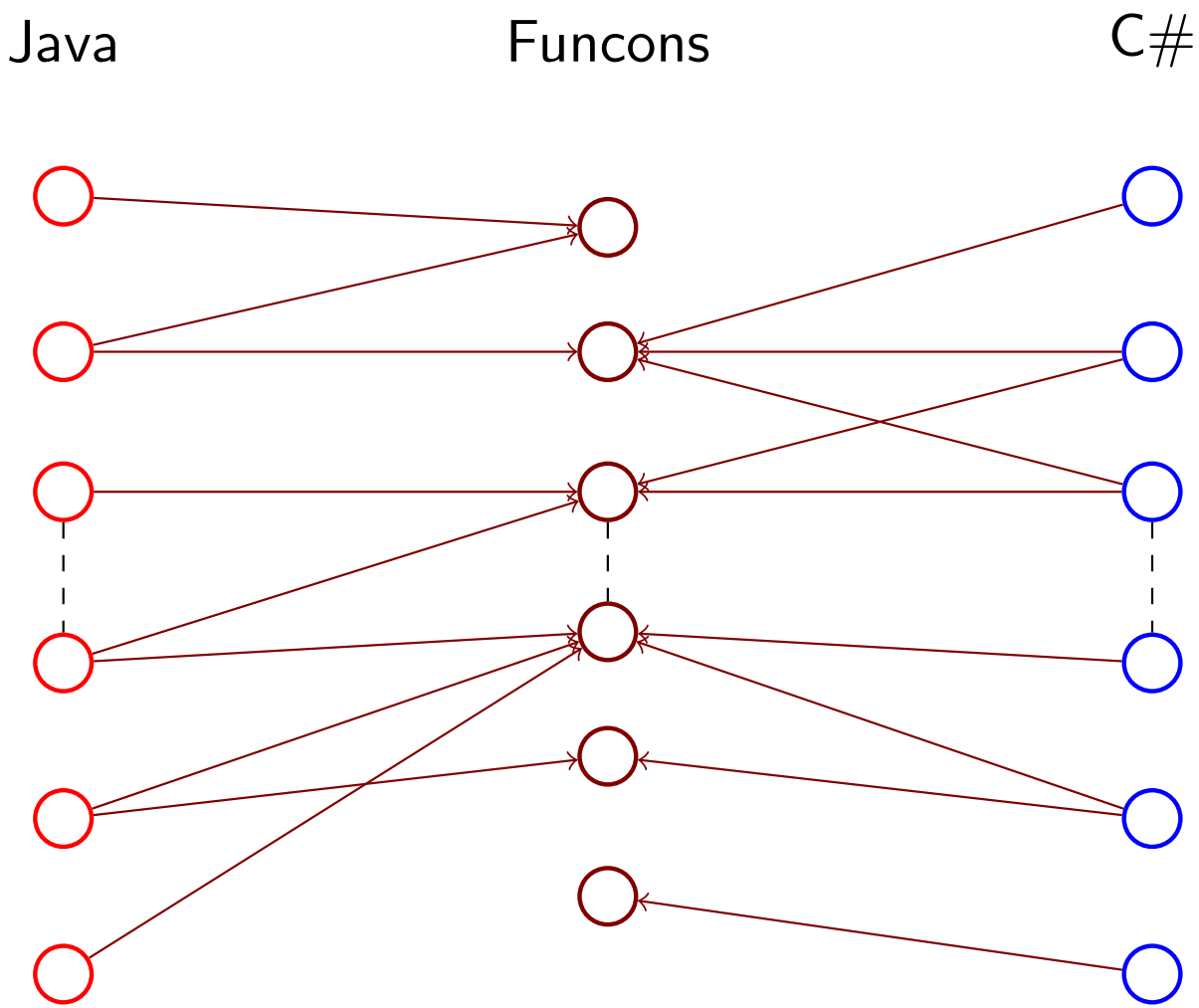
Approach

- Component based approach towards formal semantics.
- Highly reusable, fundamental constructs: *funcons*.
- A language is defined formally via a translation to funcons.
- Each funcon has a formal definition in I-MSOS.

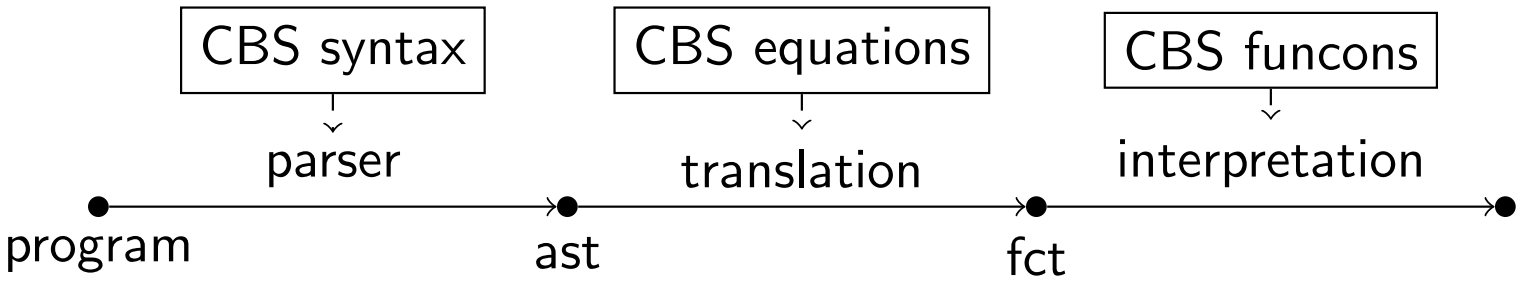
Reusable Components: Funcons



Reusable Components: Funcons



The CBS Language



SOS rules for a let-binding construct

$$\frac{\rho \vdash E_1 \rightarrow E_1'}{\rho \vdash \textbf{let } I = E_1 \textbf{ in } E_2 \rightarrow \textbf{let } I = E_1' \textbf{ in } E_2}$$

$$\frac{\rho[I \mapsto V] \vdash E \rightarrow E'}{\rho \vdash \textbf{let } I = V \textbf{ in } E \rightarrow \textbf{let } I = V \textbf{ in } E'}$$

$$\rho \vdash \textbf{let } I = V_1 \textbf{ in } V_2 \rightarrow V_2$$

SOS rules for a let-binding construct

$$\frac{E_1 \quad \rightarrow \quad E_1'}{\textbf{let } I = E_1 \textbf{ in } E_2 \quad \rightarrow \quad \textbf{let } I = E_1' \textbf{ in } E_2}$$

$$\frac{\rho[I \mapsto V] \vdash \; E \quad \rightarrow \quad E'}{\rho \vdash \; \textbf{let } I = V \textbf{ in } E \quad \rightarrow \quad \textbf{let } I = V \textbf{ in } E'}$$

$$\textbf{let } I = V_1 \textbf{ in } V_2 \quad \rightarrow \quad V_2$$

SOS rules for a let-binding construct

$$\frac{\rho \vdash E_1 \rightarrow E_1'}{\rho \vdash \textbf{let } I = E_1 \textbf{ in } E_2 \rightarrow \textbf{let } I = E_1' \textbf{ in } E_2}$$

$$\frac{\rho[I \mapsto V] \vdash E \rightarrow E'}{\rho \vdash \textbf{let } I = V \textbf{ in } E \rightarrow \textbf{let } I = V \textbf{ in } E'}$$

$$\rho \vdash \textbf{let } I = V_1 \textbf{ in } V_2 \rightarrow V_2$$

SOS rules for a let-binding construct

$$\frac{\rho \vdash \langle E_1, \sigma \rangle \rightarrow \langle E_1', \sigma' \rangle}{\rho \vdash \langle \textbf{let } I = E_1 \textbf{ in } E_2, \sigma \rangle \rightarrow \langle \textbf{let } I = E_1' \textbf{ in } E_2, \sigma' \rangle}$$

$$\frac{\rho[I \mapsto V] \vdash \langle E, \sigma \rangle \rightarrow \langle E', \sigma' \rangle}{\rho \vdash \langle \textbf{let } I = V \textbf{ in } E, \sigma \rangle \rightarrow \langle \textbf{let } I = V \textbf{ in } E', \sigma' \rangle}$$

$$\rho \vdash \langle \textbf{let } I = V_1 \textbf{ in } V_2, \sigma \rangle \rightarrow \langle V_2, \sigma \rangle$$

SOS rules for a let-binding construct

$$\frac{\rho \vdash \langle E_1, \sigma \rangle \xrightarrow{\alpha} \langle E_1', \sigma' \rangle}{\rho \vdash \langle \textbf{let } I = E_1 \textbf{ in } E_2, \sigma \rangle \xrightarrow{\alpha} \langle \textbf{let } I = E_1' \textbf{ in } E_2, \sigma' \rangle}$$

$$\frac{\rho[I \mapsto V] \vdash \langle E, \sigma \rangle \xrightarrow{\alpha} \langle E', \sigma' \rangle}{\rho \vdash \langle \textbf{let } I = V \textbf{ in } E, \sigma \rangle \xrightarrow{\alpha} \langle \textbf{let } I = V \textbf{ in } E', \sigma' \rangle}$$

$$\rho \vdash \langle \textbf{let } I = V_1 \textbf{ in } V_2, \sigma \rangle \xrightarrow{[]} \langle V_2, \sigma \rangle$$

An Intermediate Language for (I-M)SOS Rules

- Design an intermediate language (IL) for interpreting inference rules
- The IL programs are sequential (not declarative)
- Translate SOS/I-MSOS rules into IL programs

  from a variety of sources
- Perform program manipulations on IL programs:
  - Simplifications
  - Scheduling
  - Left-factoring
- Generate interpreters in a variety of host languages