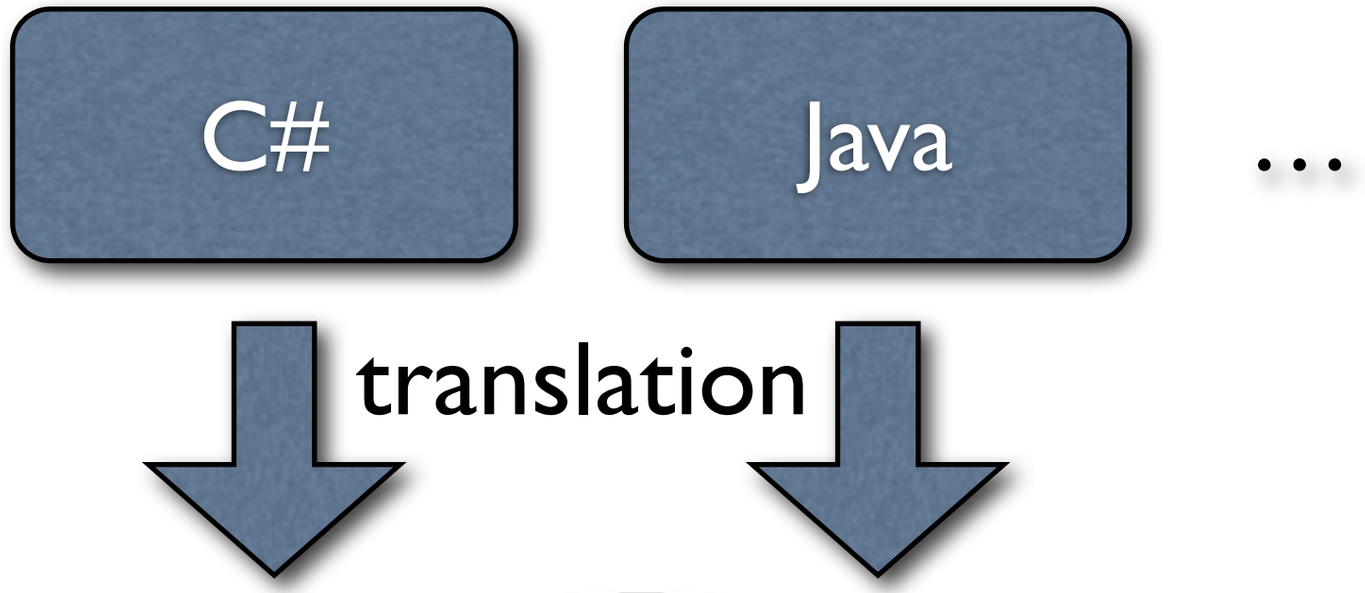
Component-based bisimilarity

Peter D Mosses, Swansea University

IFIP WG 2.11 Meeting, 25-27 June 2012, Halmstad, Sweden

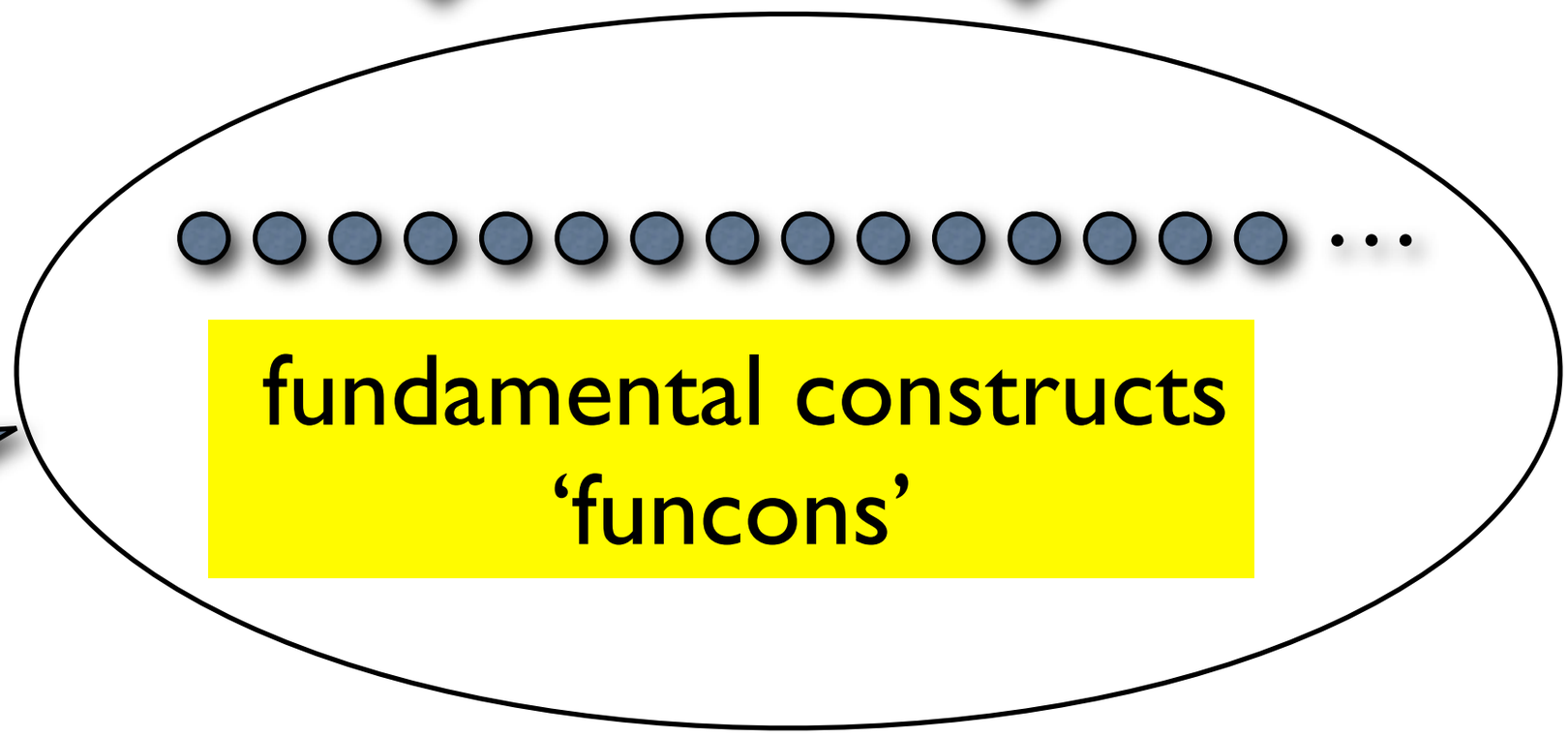
PLANCOMPS

Programming Languages (incl. DSLs)



Components and their Specifications

reusable



Funcon specifications

Structural Operational Semantics?
Reduction Semantics?

- non-modular ☹

Modular SOS?

- modular ☺, but requires *explicit* labels ☹

Implicitly-Modular SOS?

- modular ☺, and labels are left *implicit* ☺

Example funcon

cond (*Expr*, *Expr*, *Expr*) : *Expr*

- *Dynamic semantics:*

$$\frac{E_1 \rightarrow E_1'}{\text{cond}(E_1, E_2, E_3) \rightarrow \text{cond}(E_1', E_2, E_3)}$$

$$\text{cond}(\text{true}, E_2, E_3) \rightarrow E_2$$

$$\text{cond}(\text{false}, E_2, E_3) \rightarrow E_3$$

Example funcon

seq (*Comm*, *Comm*) : *Comm*

- *Dynamic semantics:*

$$\frac{C_1 \rightarrow C_1'}{\text{seq}(C_1, C_2) \rightarrow \text{seq}(C_1', C_2)}$$

$$\text{seq}(\text{skip}, C_2) \rightarrow C_2 \qquad \text{OR} \qquad \frac{C_1 \not\rightarrow}{\text{seq}(C_1, C_2) \rightarrow C_2}$$

Example funcon

alt (*Proc*, *Proc*) : *Proc*

- *Dynamic semantics:*

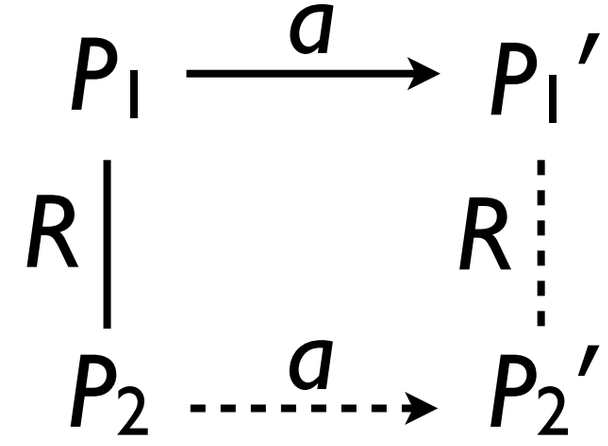
$$\frac{P_1 \xrightarrow{a} P_1'}{\mathrm{alt}(P_1, P_2) \xrightarrow{a} P_1'}$$

$$\frac{P_2 \xrightarrow{a} P_2'}{\mathrm{alt}(P_1, P_2) \xrightarrow{a} P_2'}$$

Standard bisimilarity

[Park 1981; Milner]

A symmetric relation R on **ground** terms is a **bisimulation** when:



P_1 , P_2 are **bisimilar** ($P_1 \underline{\leftrightarrow} P_2$) when there exists a bisimulation relating them

Standard bisimilarity proofs

Example: $\text{alt}(P_1, P_2) \underline{\leftrightarrow} \text{alt}(P_2, P_1)$

- prove for all ***ground terms*** P_1, P_2
- ***re-prove*** whenever the current language is extended
- bisimilarity is ***not guaranteed to be preserved*** by language extension

Non-preservation of standard bisimilarity

$$\frac{P_1 \xrightarrow{a} P_1'}{\mathrm{alt}(P_1, P_2) \xrightarrow{a} P_1'} \qquad \frac{P_2 \xrightarrow{a} P_2'}{\mathrm{alt}(P_1, P_2) \xrightarrow{a} P_2'}$$

- Suppose *Proc* includes only 'alt' and some constant with ***no*** transitions at all. We have $\boldsymbol{P_1 \underline{\leftrightarrow} P_2}$ for ***all* $\boldsymbol{P_1}$, $\boldsymbol{P_2}$.**

  If we now extend *Proc* with a further constant having ***some*** transition (with label *a*), $\boldsymbol{P_1 \underline{\leftrightarrow} P_2}$ **no longer holds** for ***all* $\boldsymbol{P_1}$, $\boldsymbol{P_2}$.**

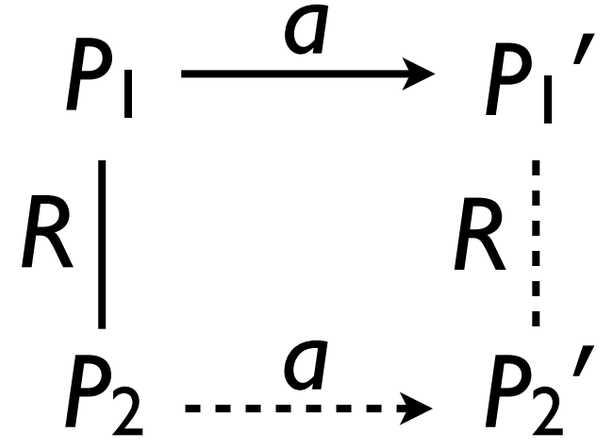
- We have $\mathbf{alt(}\boldsymbol{P}\mathbf{,}\ \boldsymbol{P}\mathbf{)} \underline{\leftrightarrow} \boldsymbol{P}$ for **all *P*.**

  If we add a further constant that can make a transition with a ***new label*** *b*, $\mathbf{alt(}\boldsymbol{P}\mathbf{,}\ \boldsymbol{P}\mathbf{)} \underline{\leftrightarrow} \boldsymbol{P}$ **no longer holds…**

FH-bisimilarity

[De Simone 1985, Rensink 2000]

A symmetric relation R on **open** terms is a **formal-hypotheses bisimulation** when for all sets of hypotheses $x \mathrel{-}a\mathrel{\rightarrow} y$ about ***variables*** x, y :



P_1 , P_2 are **fh-bisimilar** ($P_1 \underline{\leftrightarrow}_{\mathbf{fh}} P_2$) when there exists an fh-bisimulation relating them

FH-bisimilarity proofs

Example: $\mathrm{alt}(x_1, x_2) \underline{\leftrightarrow}_{\mathbf{fh}} \mathrm{alt}(x_2, x_1)$

- prove it for the specified ***open terms***
- ***instantiation*** with ground terms P_1, P_2 gives $\mathrm{alt}(P_1, P_2) \underline{\leftrightarrow} \mathrm{alt}(P_2, P_1)$
- fh-bisimilarity is ***guaranteed*** to be preserved by language extension

Published results

[M, Mousavi, Reniers, in Proc *EXPRESS* 2010]

For rules in positive **GSOS** format:

- Disjoint extension with ***no new labels*** **always** preserves fh-bisimilarity
- Disjoint extension ***with new labels*** **usually** preserves fh-bisimilarity – but not always 'improper' equivalences, e.g.:

$$\mathrm{alt}(x, x) \underline{\leftrightarrow}_{\mathbf{fh}} x \qquad \mathrm{alt}(x, 0) \underline{\leftrightarrow}_{\mathbf{fh}} x$$

Component-based bisimilarity

Components

- *funcons:* simpler than language constructs

Funcon specifications

- *I-MSOS rules:* independent

Funcon equivalences

- *fh-bisimilarity:* preserved

PLANCOMPS

Programming Languages (incl. DSLs)

C#

Java

…

translation

Components and their Specifications

reusable