Explicit Fusions

Philippa Gardner, Lucian Wischik

MFCS 2000, Bratislava

what is an “explicit fusion”?

```
void P (int &x, int &y)
{ x = x^y;
  y = y^x;
  x = x^y;
}
```

```
fun P x y =
( x := !x xor !y;
  y := !y xor !x;
  x := !x xor !y
);
```

How does P behave?

How does P behave in a context where x=y ?

We write:

$$< x = y > \mid P$$

$$\equiv \quad < x = y > \mid P\{{}^{y}/_{x}\}$$

$$\equiv \quad < x = y > \mid P\{{}^{x}/_{y}\}$$

why explicit fusions?

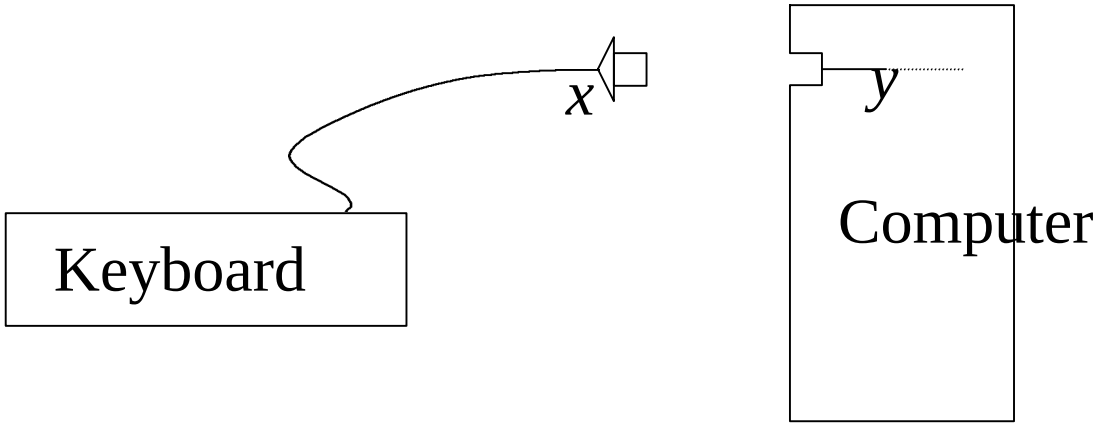
- in real world
- momentum of related work
- simplify syntax
- help implementation

- can add explicit fusions to different calculi. For example:

pi-F calculus

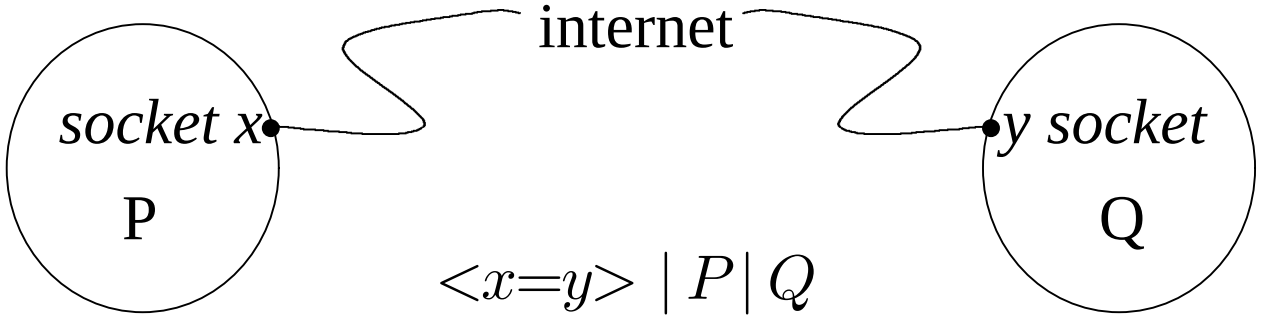
- like pi but with explicit fusions
- LTS, bisimulation
- embeds pi, fusion calculi
- full abstraction for fusion calc.

fusions in real world



$$(<x> \mid K) \,@\, (<y> \mid C) \quad \equiv \quad <x{=}y> \mid K \mid C$$

fusions and substitutions

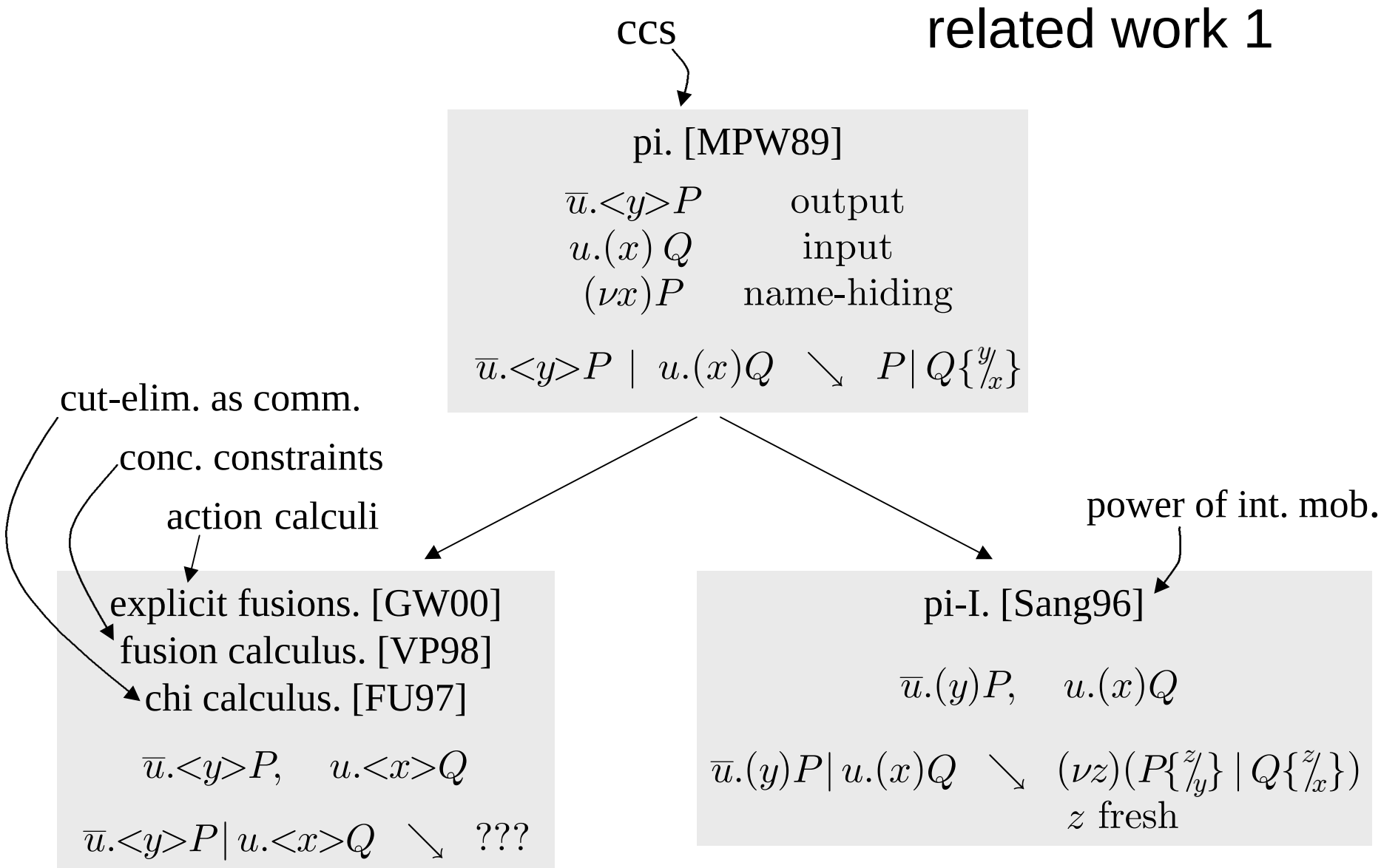


alternative to substitution: $P\{{}^{y}/_{x}\} \mid Q$

“explicitly record and delay the effect of the fusion”
cf. *Explicit Substitutions* [ACCL 91]

explicit fusions - distributed virtual machine for pi-calculus.

related work 1



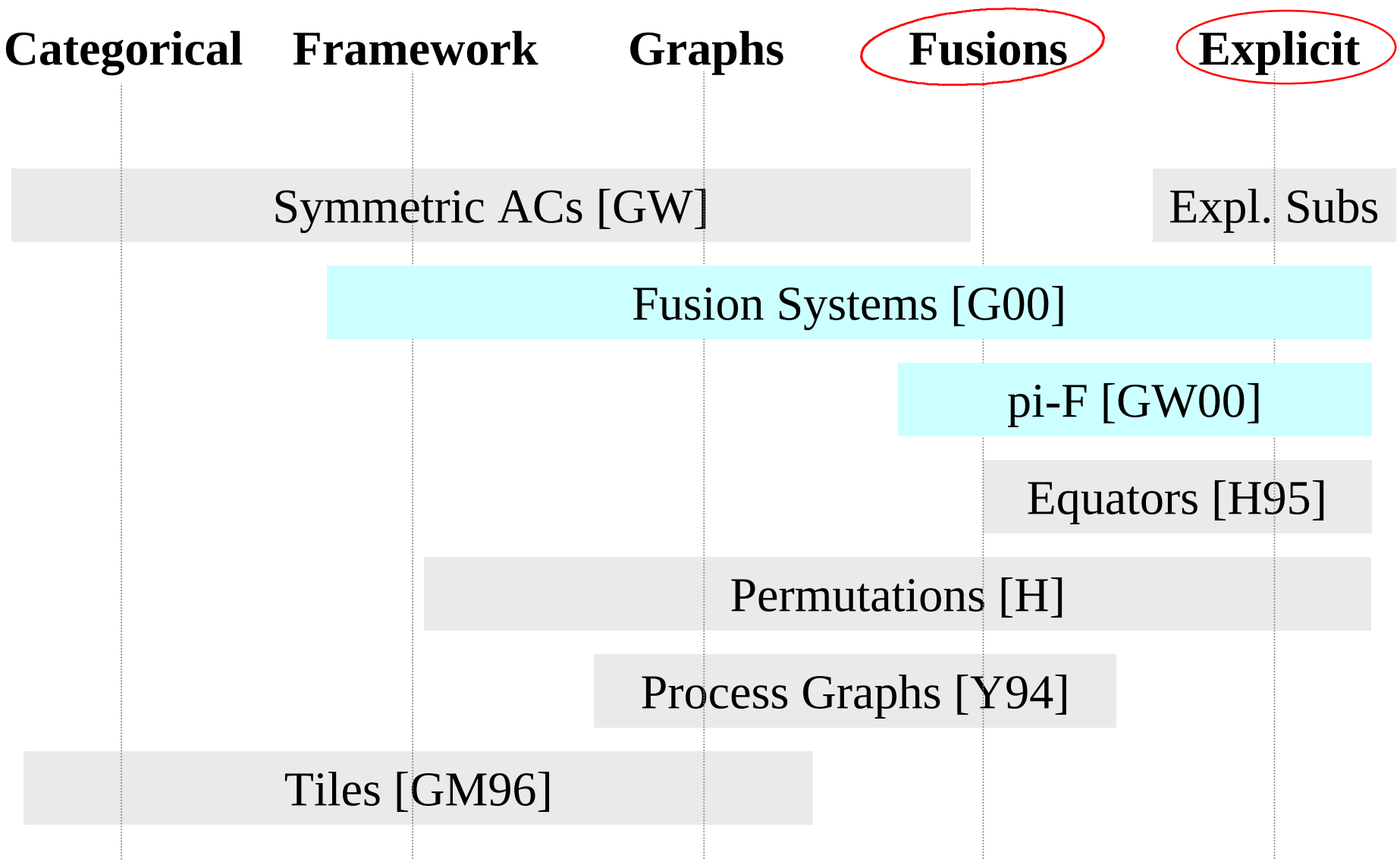
pi-calculus

- comm. between output, input
  $\overline{u}\!<\!y\!>\!.P \,|\, u(x).Q \;\searrow\; P \,|\, Q\{{}^{y}\!/_{x}\}$
- mobile scope of names/channels
  $(\nu y)\big(\overline{u}\!<\!y\!>\big) \,|\, u(x).P$
  $\equiv\; (\nu y)\big(\overline{u}\!<\!y\!> \,|\, u(x).P\big)$
- dynamic creation of names/channels

concretions+abstractions [Mil99]

- syntactic change: $\overline{u}.\!<\!y\!>\!P$
- makes a *commitment* $\overline{u},\;\; u$
- effects sorted out by $@$
  $\overline{u}.\!<\!y\!>\!P \,|\, u.(x)Q \;\searrow\; <\!y\!>\!P @ (x)Q$

related work 2



pi-F calculus

- grammar
- reaction relation
- structural congruence
  (e.g. alpha conversion)

labels, bisimulation

- to describe interaction with environment
- to compare two processes without detail of internal state

embedding pi, fusion

- how to encode bound inp.
- results: reaction preserved, full abstraction for fusion & not pi

pi-F calculus

P ::=	$\mathbf{nil}$	empty process
	$P \mid P$	parallel composition
	$(\nu x)P$	name-hiding
	$<x>$	datum
	$<x{=}y>$	explicit fusion
	$x.P$	input
	$\overline{x}.P$	output

datums

- are like concretions
- polyadic, through parallel composition
- not commutative

$<x>|<y>| P \cdots <xy>P$

explicit fusion

- denotes an equivalence relation
- finite basis
- has substitutive effect

pi-F reaction

$$\overline{u}.P \,|\, u.Q \;\searrow\; P@Q$$

$$\begin{aligned}
\overline{u}.(<y>|P) \;\mid\; u.(\nu x)(<x>|Q) \;&\searrow\; (<y>|\,P) \;@\; (\nu x)(<x>|\,Q) \\
&\equiv\; (\nu x)\big(<x{=}y>|P|\,Q\big) \\
&\equiv\; P\{{}^{y}\!/_{x}\} \,|\, Q\{{}^{y}\!/_{x}\}
\end{aligned}$$

effect of @ is to fuse datums

- use *standard form:* dats, fus, restrictions ("interface") factored out
- interfaces are unique up to alpha-conversion
- if datums, then explicit fusions. $<x>@<y> \;\equiv\; <x{=}y>$

encode bound input $u.(x)Q$

encode lazy input $(\nu x)(u.<x>|\,Q)$

struct.cong. rules for fusions

$$
\begin{array}{lll}
 & (\nu x)(\overline{x}.\mathbf{nil}) & \\
\equiv & (\nu x)(\nu y)(<x{=}y>|\,\overline{x}.\mathbf{nil}) & \text{create fresh bound name } y \text{ as alias for } x \\
\equiv & (\nu x)(\nu y)(<x{=}y>|\,\overline{y}.\mathbf{nil}) & \text{substitute } y \text{ for } x \\
\equiv & (\nu y)(\overline{y}.\mathbf{nil}) & \text{remove the now-unused bound name } x
\end{array}
$$

$$
\begin{array}{lcll}
<x{=}y> \mid <y{=}z> & \equiv & <x{=}y> \mid <x{=}z> & \text{transitivity} \\
<x{=}y> & \equiv & <y{=}x> & \text{symmetry} \\
\text{⚠}\ <x{=}x> & \equiv & \mathbf{nil} & \text{reflexivity}
\end{array}
$$

$$
\begin{array}{lcll}
<x{=}y>|\,\overline{z}.P & \equiv & <x{=}y>|\,\overline{z}.(<x{=}y>|\,P) & \\
<x{=}y>|\,\overline{x}.P & \equiv & <x{=}y>|\,\overline{y}.P & \text{and other small-step substitutions}
\end{array}
$$

$$
\text{⚠}\quad (\nu x)(<x{=}y>) \equiv \mathbf{nil} \qquad \text{restriction delimits scope of fusion}
$$

standard presentation

- reaction relation $\searrow$
  labelled transitions
  bisimulation

labelled transitions

- "commitment" labels $\xrightarrow{\overline{u}}$ $\xrightarrow{u}$ $\xrightarrow{\tau}$
- $\xrightarrow{\tau}$ corresponds to $\searrow$

bisimulation relation $\sim$

- "P can do all the transitions that Q can, and vice versa"
- bisimulation is a congruence
- It equals fusion-bisimulation, but not pi-open-bisimulation

pi-F labels, bisimulation

$$\overline{u}.P \xrightarrow{\overline{u}} P \qquad u.P \xrightarrow{u} P \qquad \overline{u}.P \mid u.Q \xrightarrow{\tau} P@Q$$

Bisimulation is standard, augmented for fusion contexts:

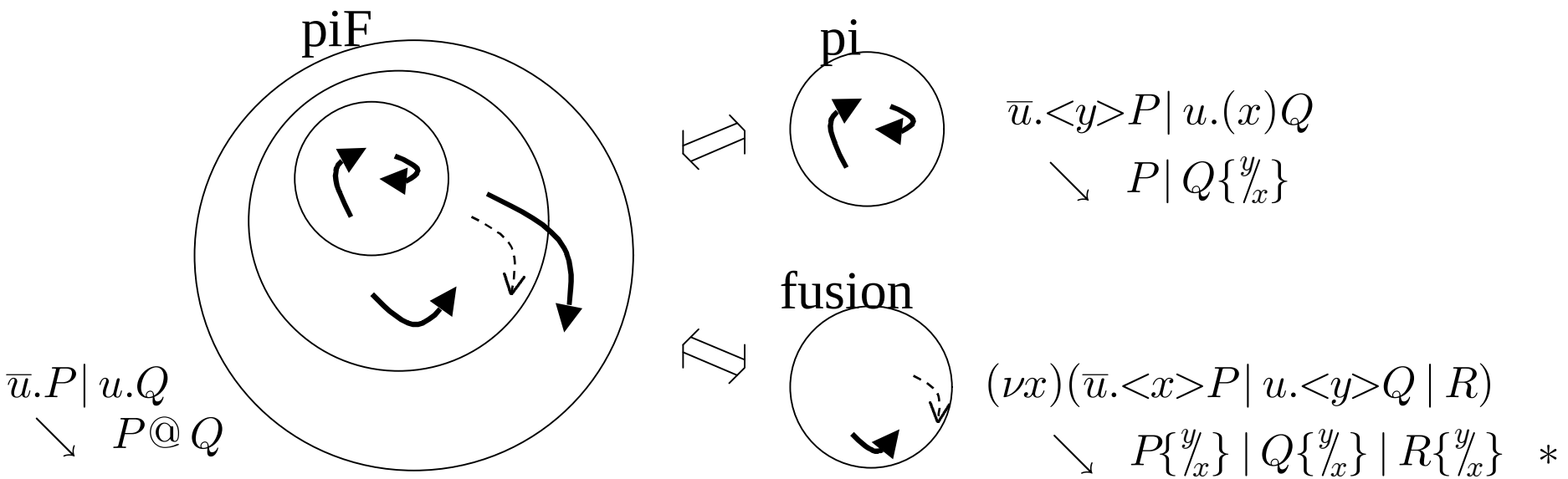
a relation *S* is an *open bisimulation* iff whenever *PSQ* then

- P, Q have same interface (same datums, fusions, restrictions)
- for all x, y, if $<x{=}y> \mid P \xrightarrow{\alpha} P_1$ then $<x{=}y> \mid Q \xrightarrow{\alpha} Q_1$ and $P_1 \mathcal{S} Q_1$
- and vice versa

To avoid the quantification, we add a *fusion transition*

$$\overline{u}.P \mid v.Q \xrightarrow{?u=v} P@Q$$

embedding pi and fusion



- Translations straightforward. (using bound-input encoding)
- Both embeddings preserve structural congruence
- Pi embedding preserves reaction relation
- Fusion embedding weakly preserves reaction relation
  (must further constrain it, with same constraint *)

results

pi-F bisimulation is a congruence.

- proof: standard. create another LTS without struct. cong. rule

  $P \sim Q \;\; \Rightarrow \;\; \forall C.\, C[P] \sim C[Q]$

pi-F bisimulation equals fusion hyperequivalence

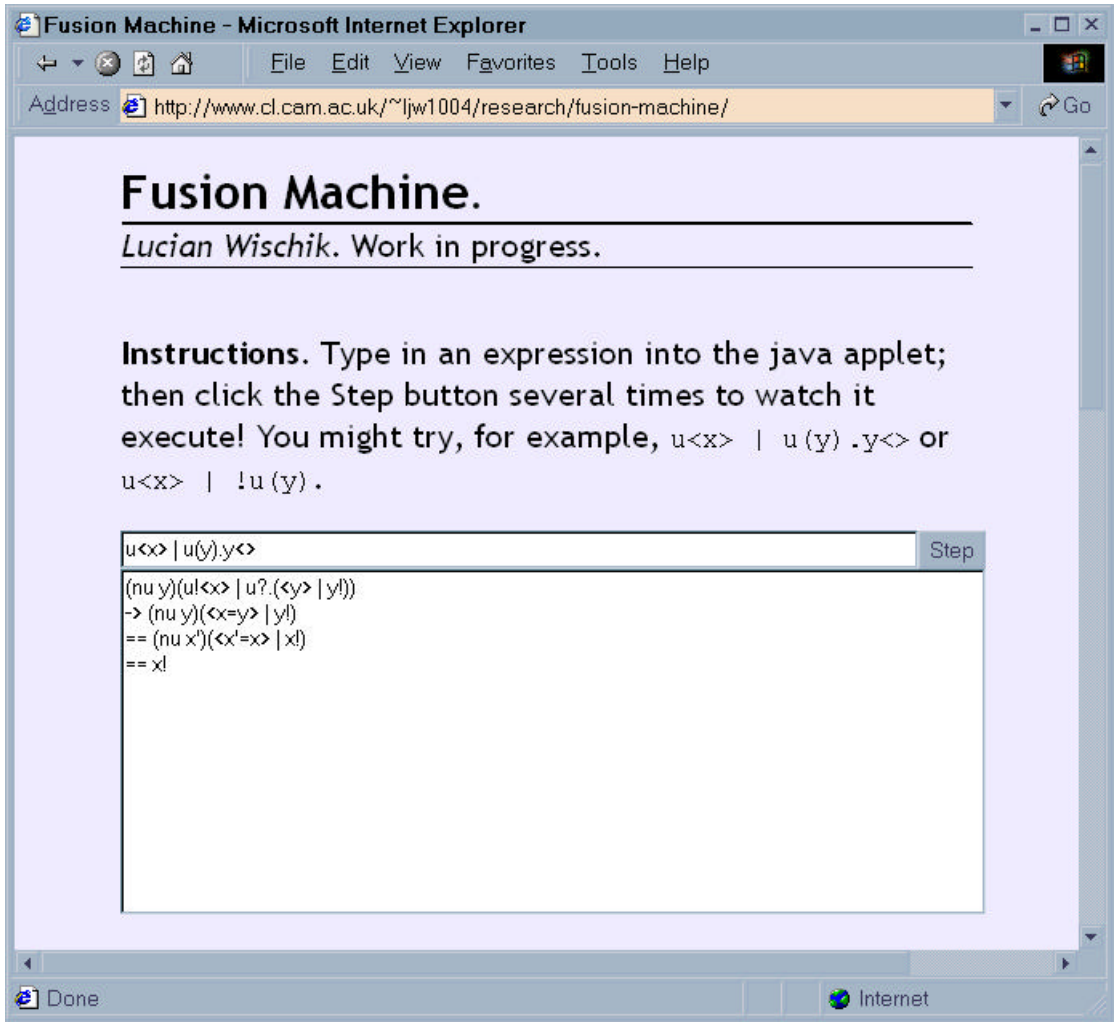
- proof: fusion hyperequivalence is just pi-F bisim. without interfaces

pi-F bisimulation is stronger than pi-open-bisimulation

- “open” bisimulation [S96]: one that’s closed w.r.t subsitutions (or fusions)
- proof: in example below, no pi context can make *x*=*y*; but pi-F can.

  $(\nu xy)(\overline{u}.<xy>|\, P)$

ongoing work



- (replication)
- implementation based on explicit fusions
- lambda calculus encoding, using datums to represent lambda variables
- add explicit fusions to other calculi (“fusion systems”)