

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:

$$\begin{aligned} & \langle x = y \rangle \mid P \\ \equiv & \langle x = y \rangle \mid P\{y/x\} \\ \equiv & \langle x = y \rangle \mid P\{x/y\} \end{aligned}$$

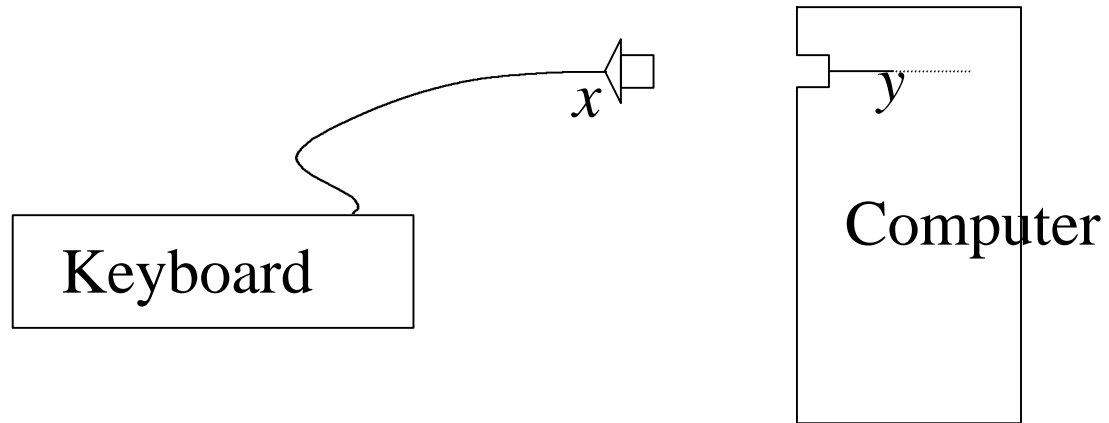
why explicit fusions?

- in real world
 - momentum of related work
 - simplify syntax
 - help implementation
-
- can add explicit fusions to different calculi. For example:

pi-F calculus

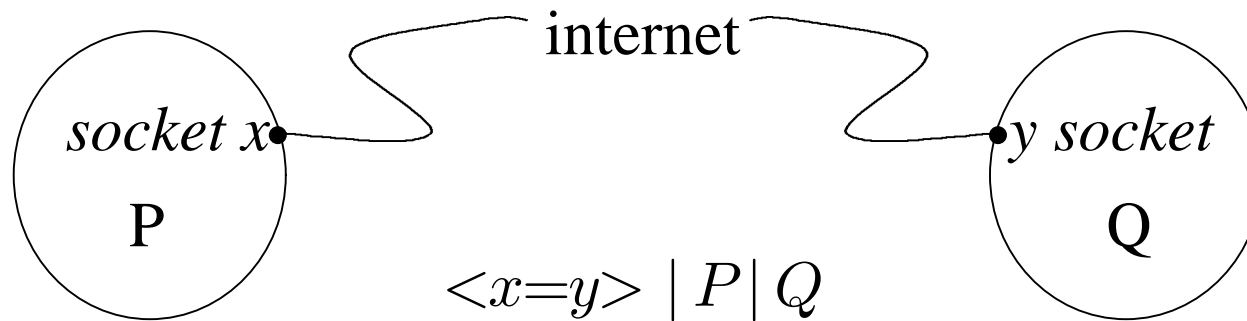
- like pi but with explicit fusions
- LTS, bisimulation
- embeds pi, fusion calculi
- full abstraction for fusion calc.

fusions in real world



$$(\langle x \rangle | K) @ (\langle y \rangle | C) \equiv \langle x=y \rangle | K | C$$

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

ccs



pi. [MPW89]

$\bar{u}.<y>P$	output
$u.(x)Q$	input
$(\nu x)P$	name-hiding

$$\bar{u}.<y>P \mid u.(x)Q \quad \searrow \quad P \mid Q\{\frac{y}{x}\}$$

cut-elim. as comm.

conc. constraints

action calculi

explicit fusions. [GW00]
 fusion calculus. [VP98]
 chi calculus. [FU97]

$\bar{u}.<y>P,$	$u.<x>Q$
-----------------	----------

$$\bar{u}.<y>P \mid u.<x>Q \quad \searrow \quad ???$$

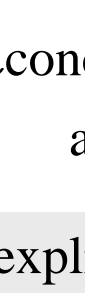
power of int. mob.

pi-I. [Sang96]

$\bar{u}.(y)P,$	$u.(x)Q$
-----------------	----------

$$\bar{u}.(y)P \mid u.(x)Q \quad \searrow \quad (\nu z)(P\{\frac{z}{y}\} \mid Q\{\frac{z}{x}\})$$

z fresh



pi-calculus

- comm. between output, input

$$\bar{u}\langle y \rangle.P \mid u(x).Q \quad \searrow \quad P \mid Q\{y/x\}$$

- mobile scope of names/channels

$$(\nu y)(\bar{u}\langle y \rangle \mid u(x).P)$$

$$\equiv (\nu y)(\bar{u}\langle y \rangle \mid u(x).P)$$

- dynamic creation of names/channels

concretions+abstractions [Mil99]

- syntactic change: $\bar{u}.\langle y \rangle P$

- makes a *commitment* \bar{u} , u

- effects sorted out by @

$$\bar{u}.\langle y \rangle P \mid u.(x)Q \quad \searrow \quad \langle y \rangle P @ (x)Q$$

related work 2

Categorical

Framework

Graphs

Fusions

Explicit

Symmetric ACs [GW]

Expl. Subs

Fusion Systems [G00]

pi-F [GW00]

Equators [H95]

Permutations [H]

Process Graphs [Y94]

Tiles [GM96]

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

$P ::=$	nil	empty process
	$P P$	parallel composition
	$(\nu x)P$	name-hiding
	$\langle x \rangle$	datum
	$\langle x=y \rangle$	explicit fusion
	$x.P$	input
	$\bar{x}.P$	output

datums

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

$\langle x \rangle | \langle y \rangle | P \dots \langle xy \rangle P$

explicit fusion

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

pi-F reaction

$$\bar{u}.P \mid u.Q \searrow P @ Q$$

$$\begin{aligned} \bar{u}.\langle y \rangle \mid P \mid u.(\nu x)\langle x \rangle \mid Q &\searrow (\langle y \rangle \mid P) @ (\nu x)\langle x \rangle \mid Q \\ &\equiv (\nu x)\langle x=y \rangle \mid P \mid Q \\ &\equiv P\{y/x\} \mid Q\{y/x\} \end{aligned}$$

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. $\langle x \rangle @ \langle y \rangle \equiv \langle x=y \rangle$

encode bound input $u.(x)Q$

encode lazy input $(\nu x)(u.\langle x \rangle \mid Q)$

struct.cong. rules for fusions

$$\begin{aligned}
 & (\nu x)(\bar{x}.\mathbf{nil}) \\
 \equiv & (\nu x)(\nu y)(\langle x=y \rangle | \bar{x}.\mathbf{nil}) && \text{create fresh bound name } y \text{ as alias for } x \\
 \equiv & (\nu x)(\nu y)(\langle x=y \rangle | \bar{y}.\mathbf{nil}) && \text{substitute } y \text{ for } x \\
 \equiv & (\nu y)(\bar{y}.\mathbf{nil}) && \text{remove the now-unused bound name } x
 \end{aligned}$$

$$\begin{aligned}
 \langle x=y \rangle | \langle y=z \rangle & \equiv \langle x=y \rangle | \langle x=z \rangle && \text{transitivity} \\
 \langle x=y \rangle & \equiv \langle y=x \rangle && \text{symmetry} \\
 \langle x=x \rangle & \equiv \mathbf{nil} && \text{reflexivity}
 \end{aligned}$$



$$\begin{aligned}
 \langle x=y \rangle | \bar{z}.P & \equiv \langle x=y \rangle | \bar{z}.(\langle x=y \rangle | P) \\
 \langle x=y \rangle | \bar{x}.P & \equiv \langle x=y \rangle | \bar{y}.P && \text{and other small-step substitutions}
 \end{aligned}$$



$$(\nu x)(\langle x=y \rangle) \equiv \mathbf{nil} \quad \text{restriction delimits scope of fusion}$$

standard presentation

- reaction relation ↘
labelled transitions
bisimulation

labelled transitions

- “commitment” labels $\xrightarrow{\bar{u}}$ \xrightarrow{u} $\xrightarrow{\tau}$
- $\xrightarrow{\tau}$ corresponds to ↘

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

$$\bar{u}.P \xrightarrow{\bar{u}} P \qquad u.P \xrightarrow{u} P \qquad \bar{u}.P | u.Q \xrightarrow{\tau} P @ Q$$

Bisimulation is standard, augmented for fusion contexts:

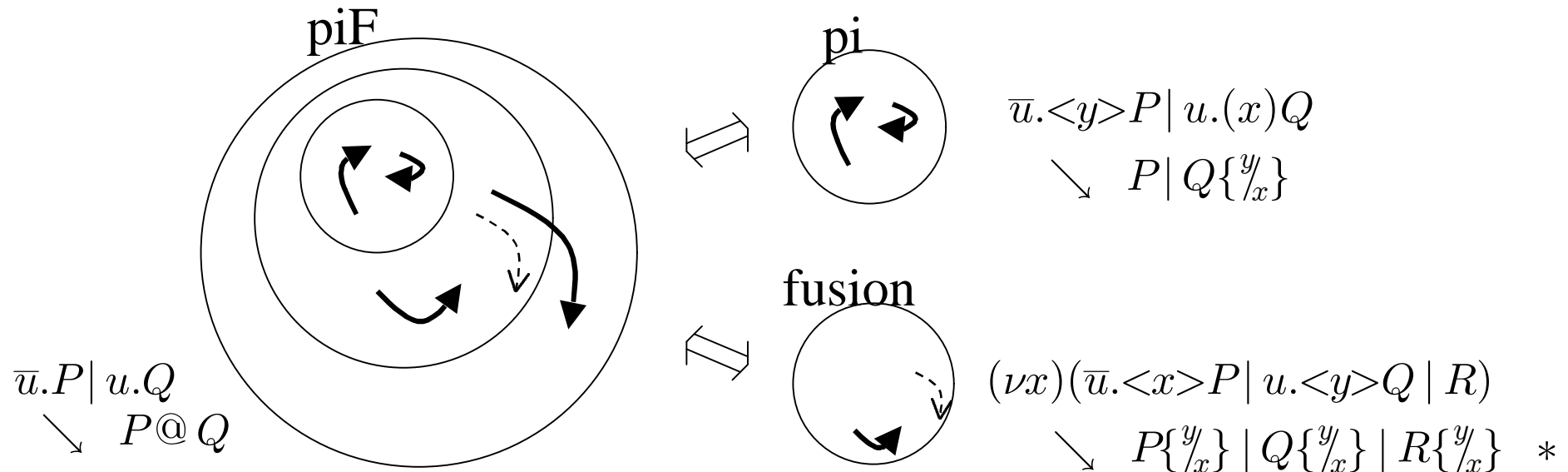
a relation S is an *open bisimulation* iff whenever $P S Q$ then

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

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

$$\bar{u}.P | v.Q \xrightarrow{?u=v} P @ Q$$

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

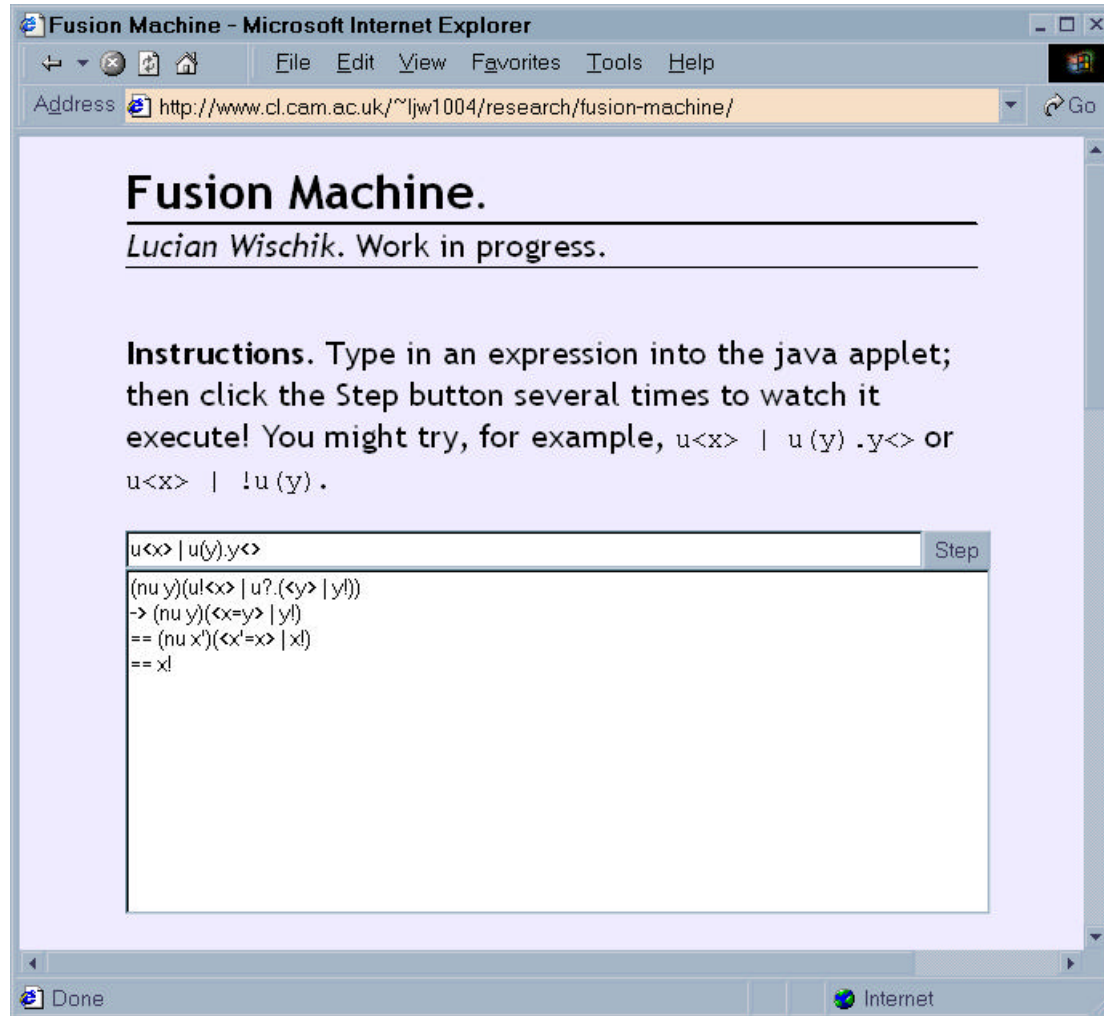
- proof: fusion hyperequivalence is just pi-F bisim. without interfaces

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

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

$$(\nu xy)(\bar{u}. \langle xy \rangle \mid P)$$

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”)