Discretionary Information Flow Control for Interaction-Oriented Specifications

Alberto Lluch Lafuente, Flemming Nielson, Hanne Riis-Nielson

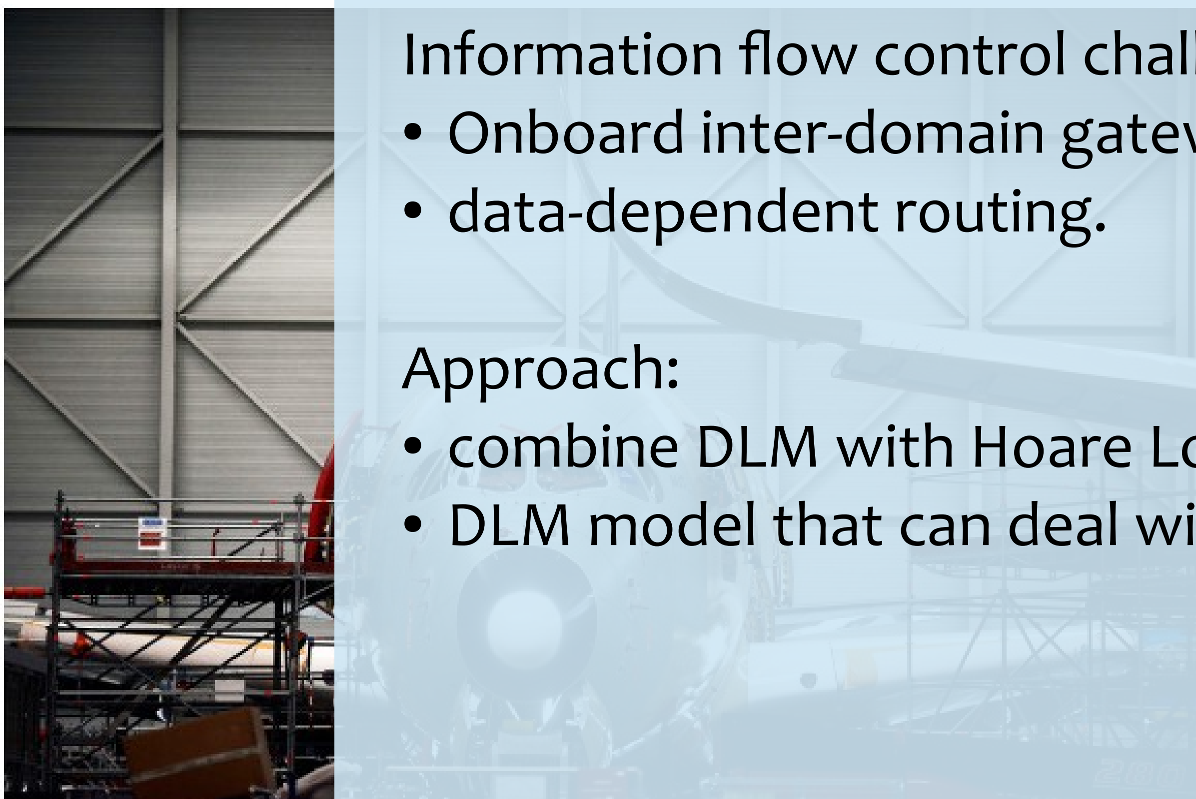
MOTIVATIONS

HACKERS COULD COMMANDEER NEW PLANES THROUGH PASSENGER WI-FI

An Airbus A350 on an assembly line, in Toulouse, France, April 11, 2015. REMY GABALDA/AFP/GETTY

WIRED Hackers Could Commandeer New Planes Through Passenger Wi-Fi

HACKERS COULD COMMANDEER NEW PLANES THROUGH PASSENGER WI-FI



An Airbus A350 on an assembly line, in Toulouse, France, April 11, 2015. REMY GABALDA/AFP/GETTY

Some activities at DTU/LBT

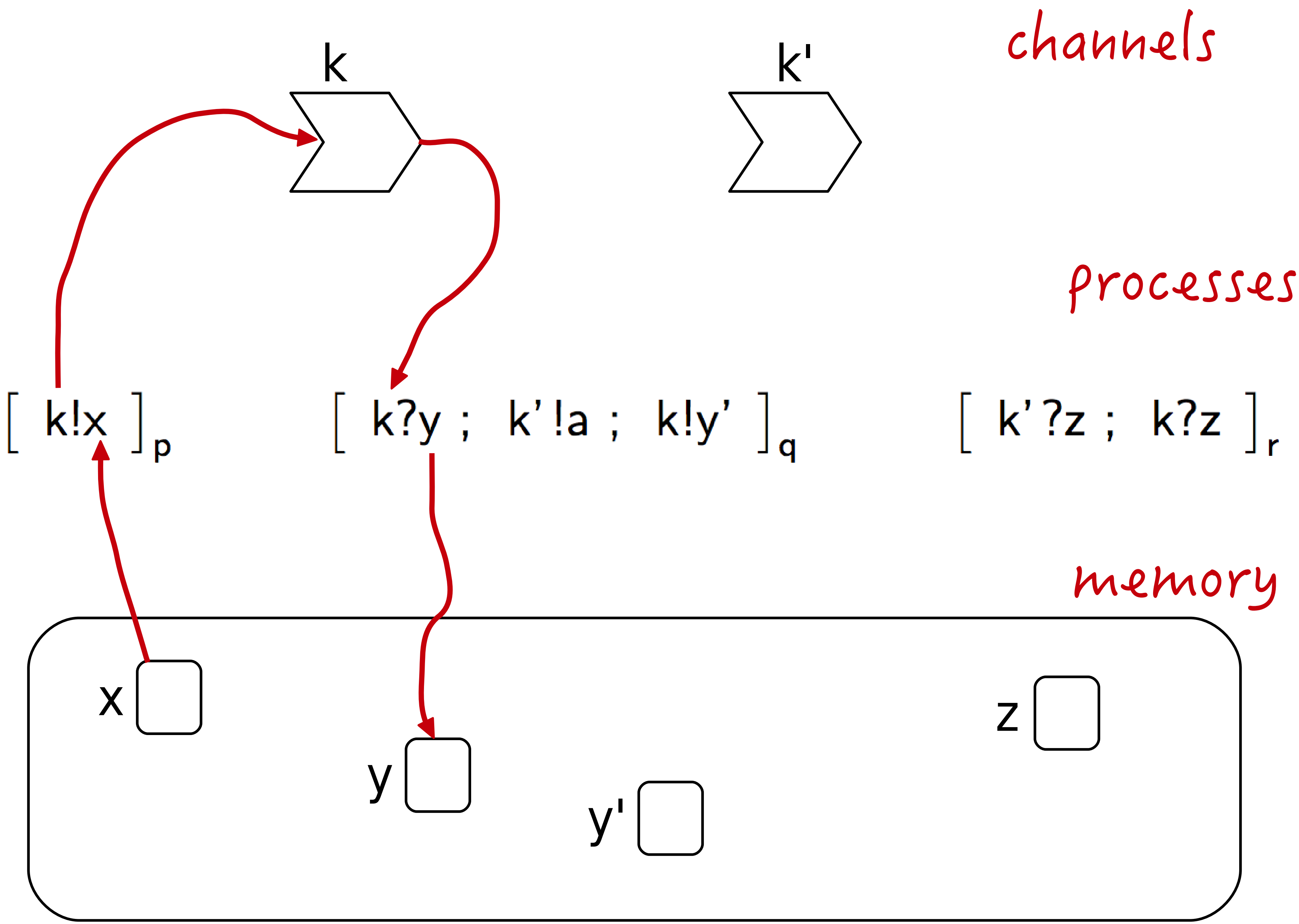
Adapting DLM for dealing Airbus needs.

Information flow control challenges:

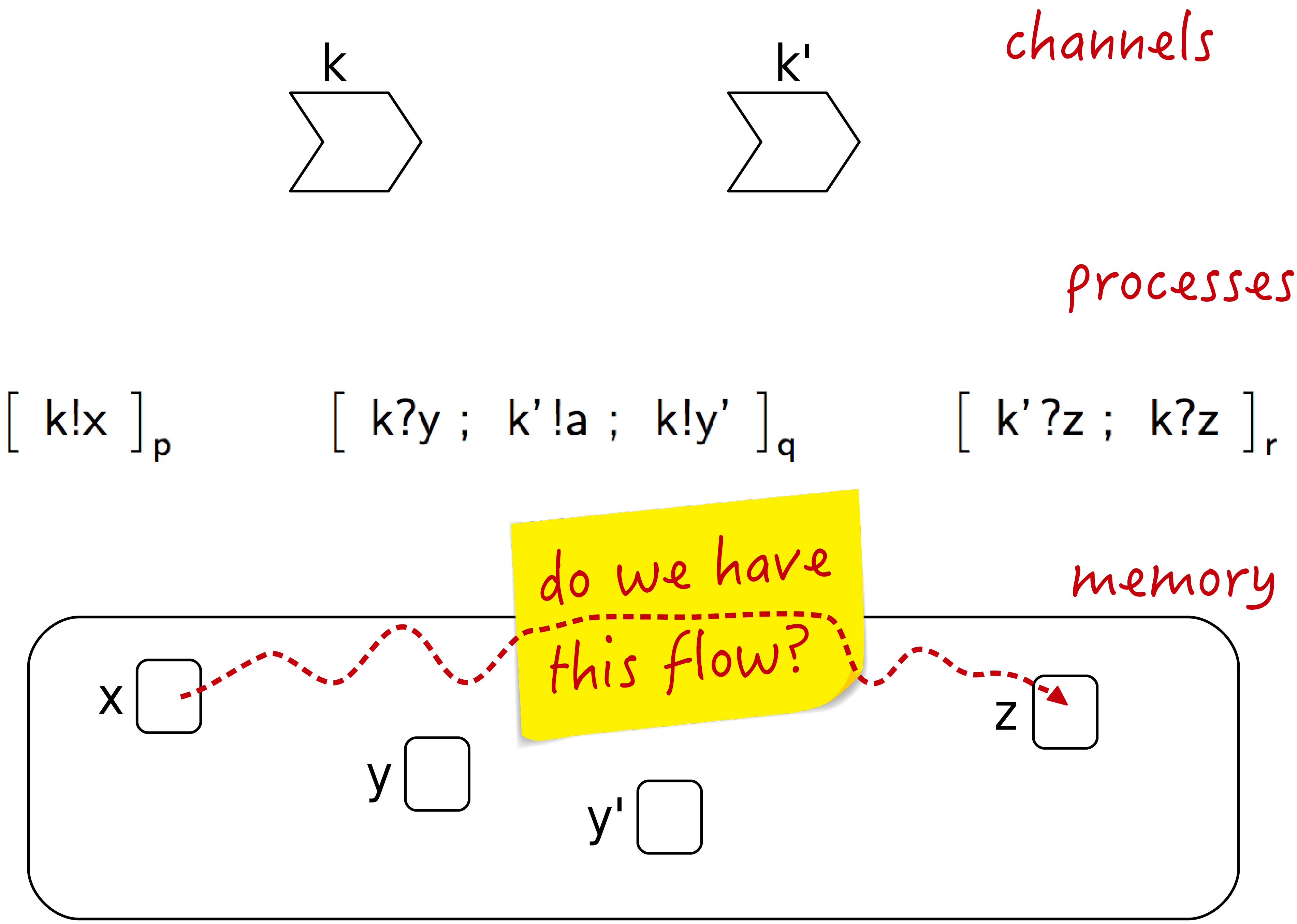
- Onboard inter-domain gateways;
- data-dependent routing.

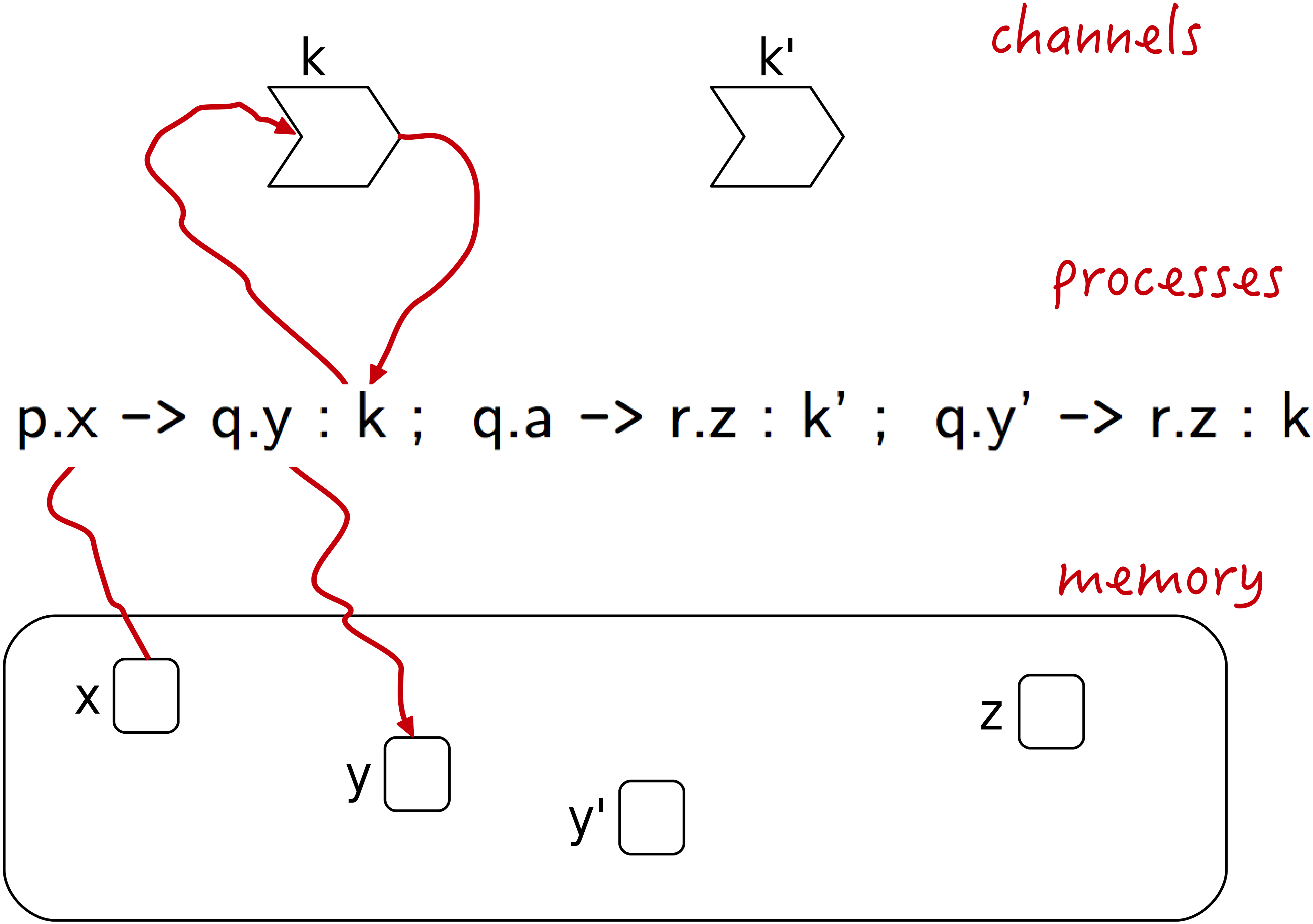
Approach:

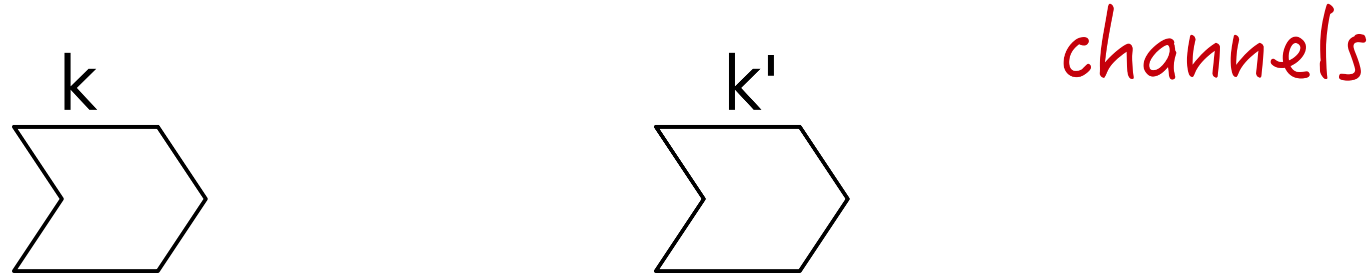
- combine DLM with Hoare Logics;
- DLM model that can deal with the Airbus gateway.



channels
k
k'
processes
[k!x]p
[k?y ; k' !a ; k!y']q
[k' ?z ; k?z]r
memory
x
y
y'
z

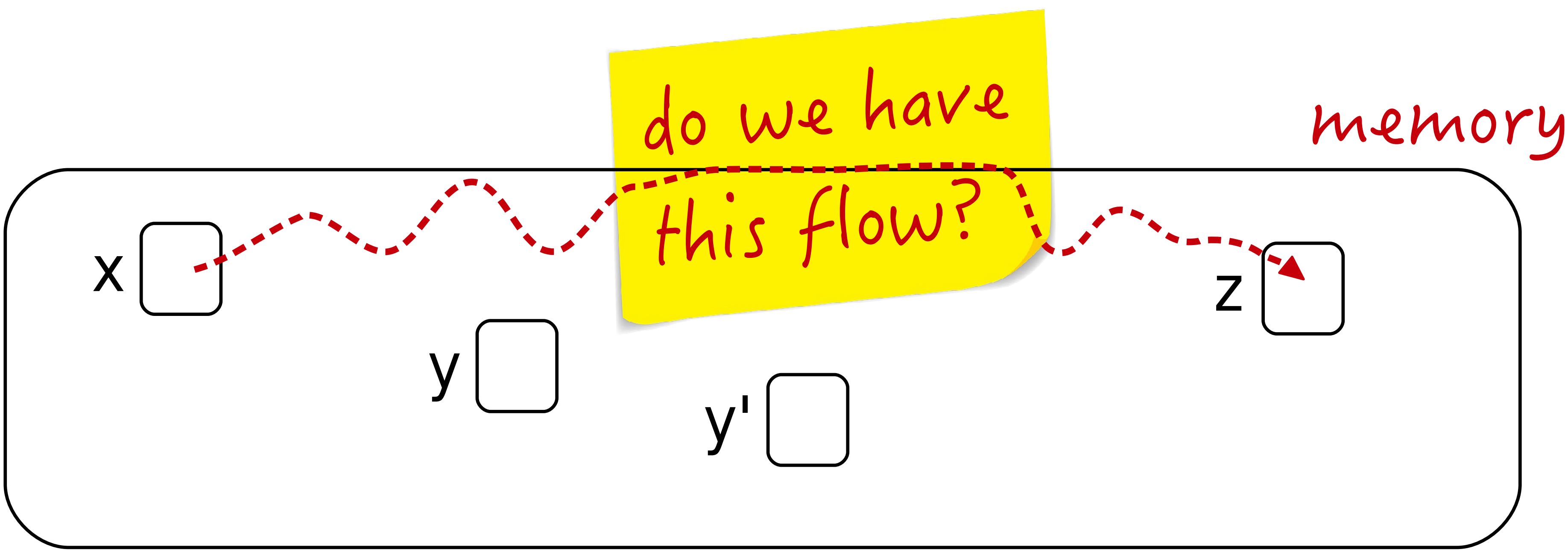

channels
k
k'
processes
[k!x]p
[k?y ; k'!a ; k!y']q
[k'?z ; k?z]r
memory
x
y
y'
z
do we have this flow?


channels
k
k'
processes
p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k
memory
x
y
y'
z



processes

p.x -> q.y : k ; q.a -> r.z : k’ ; q.y’ -> r.z : k



typical service/protocols choreographies

```
u.name -> rp.user : a ;
rp.user -> ip.id : b ;
u.my_pwd -> ip.pwd : a ;
if check(id,pwd)@ip then
    ip."ok" -> rp."ok" : b ;
    rp.class(user) -> s.class : c
else
    ip."fail" -> rp."fail" : b ;
    rp."na" -> s.class : c ;
( ip.rep(id) -> rp.report : b
| ( data := first(class) @ s;
    while data ≠ nil @s do
        s.data -> u.info : a ;
        data := data.next @ s
    then
        s."end" -> u.info : a ) )
```

```
[ a!name ;
  a!my_pwd ;
  loop
     a?info
     ⊕ (a?"end" ;
          break ) ]_u
```

```
[ b?id ;
  a?pwd ;
  if check(id,pwd) then
      b!"ok"
  else
      b!"fail" ;
  b!rep(id) ; ]_ip
```

```
[ a?user ;
  b!user ;
  ( (b?"ok" ;
     c!class(user))
     ⊕ (b?"fail" )
         c!"na") ;
  b?report ; ]_rp
```

```
[ c?class ;
  data := first(class) ;
  while data ≠ nil then
      a!data ;
      data := data.next ;
  then
      a!"end" ; ]_s
```

HPC choreographies?

map-reduce (2 processes)

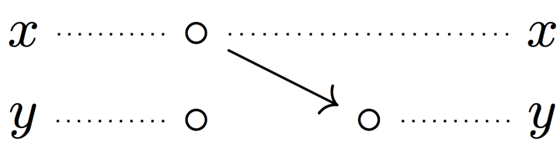
```
A.x2 -> B.y : k;
(
    x1 : = f(x1) @ A | y := f(y) @ B
);
B:y -> A.x2 : k;
z := aggregate(x1,x2) @ A;
```

map-reduce (3 processes)

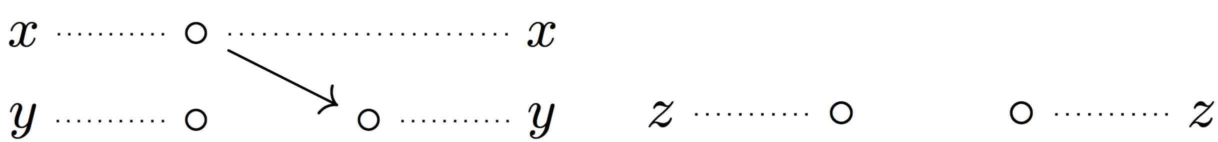
```
A1 -> A2 : k2 ; A1 -> A3 : k3;
(
    x1 : = f(x1) @ A1 | x2 : = f(x2) @ A2 | x3 : = f(x3) @ A3
);
A2 -> A1 : k2;
A3 -> A1 : k3;
z := aggregate(x1,x2,x3) @ A1;
```

p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k

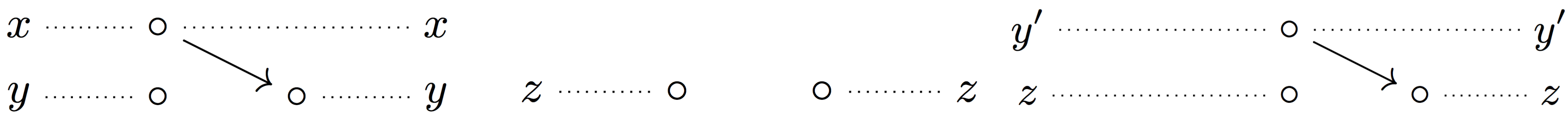
p.x -> q.y : k ;   q.a -> r.z : k' ;   q.y' -> r.z : k



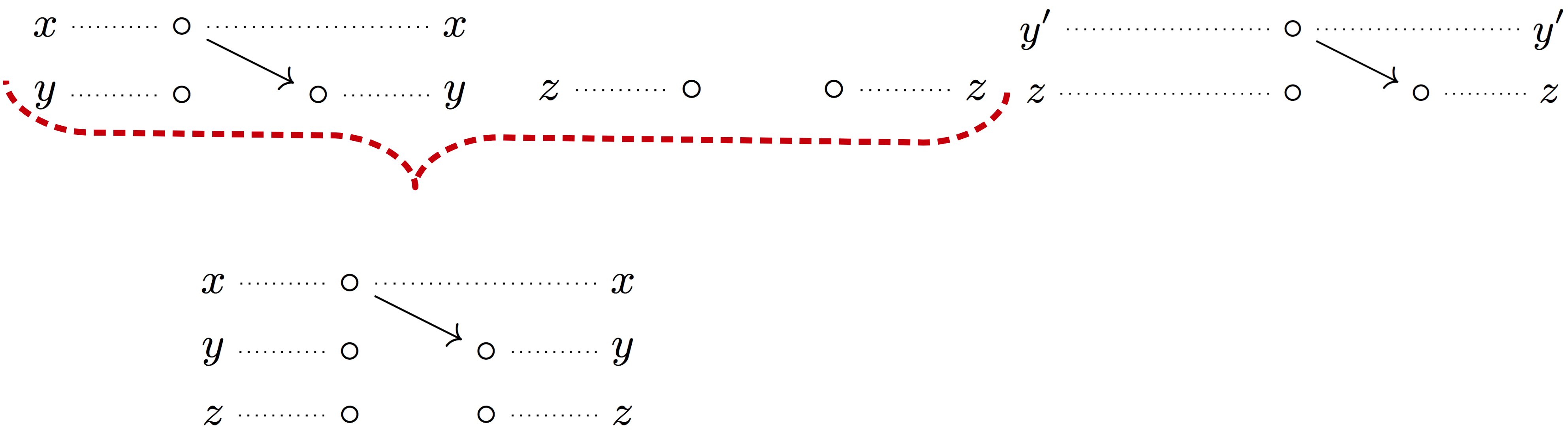
p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k



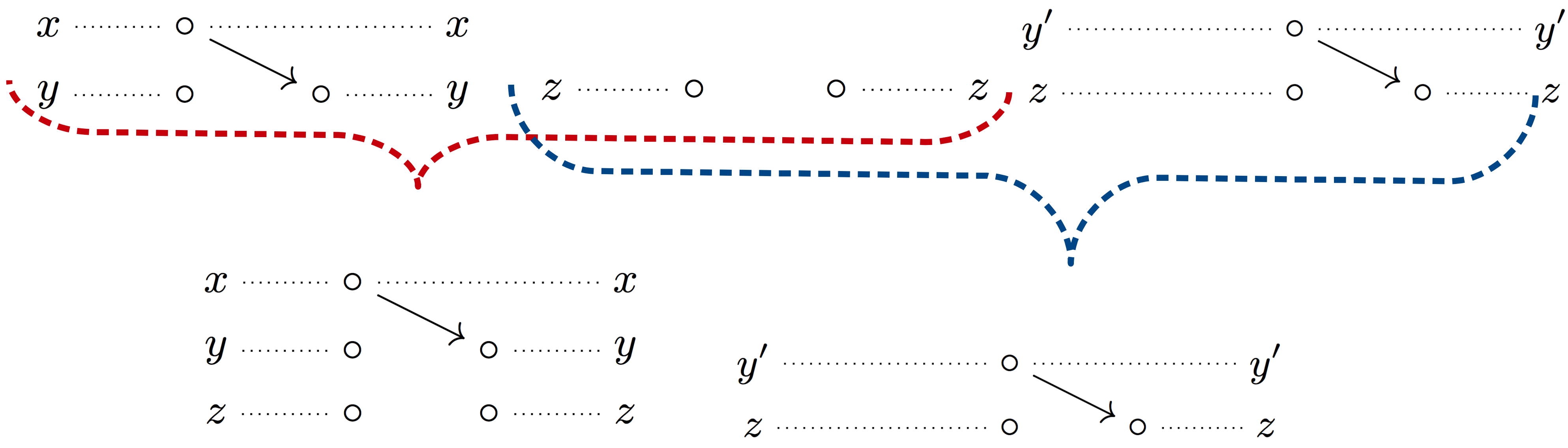
p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k



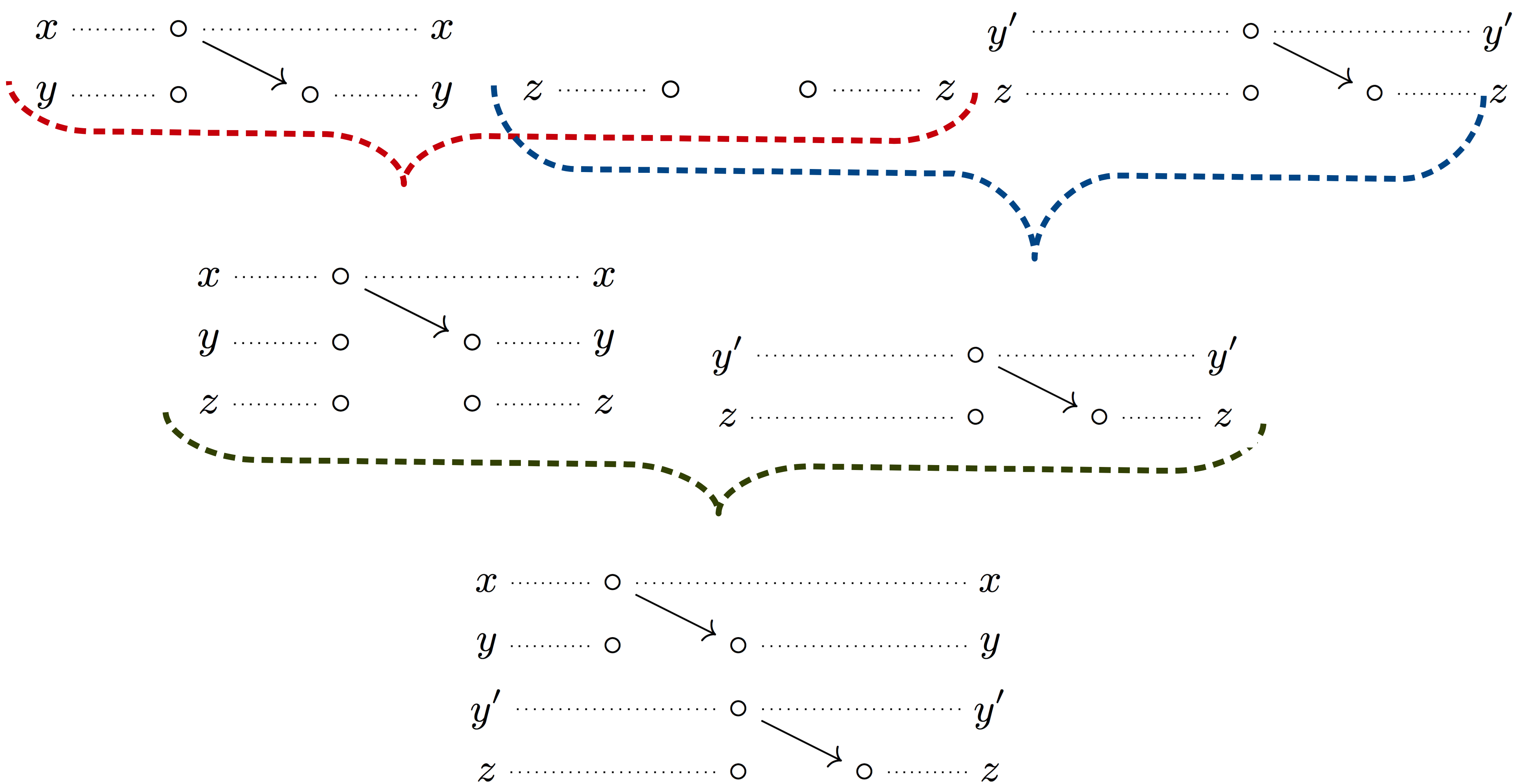
p.x -> q.y : k ;   q.a -> r.z : k' ;   q.y' -> r.z : k



p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k

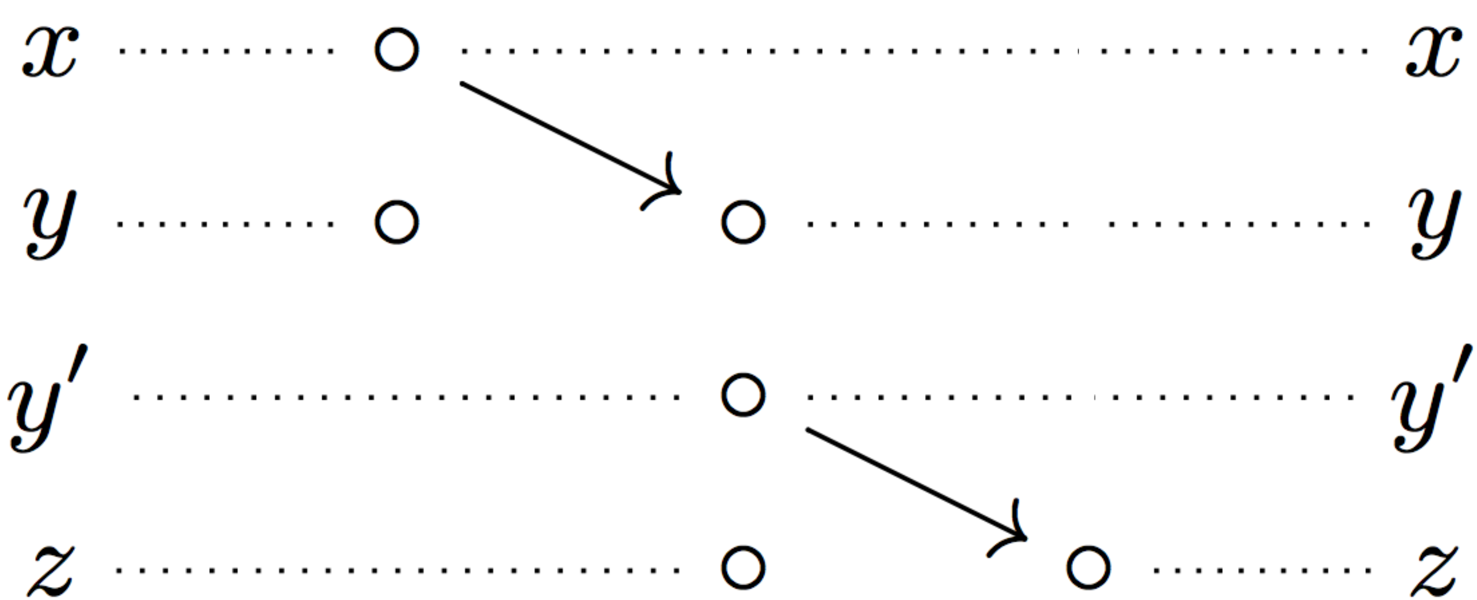


p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k



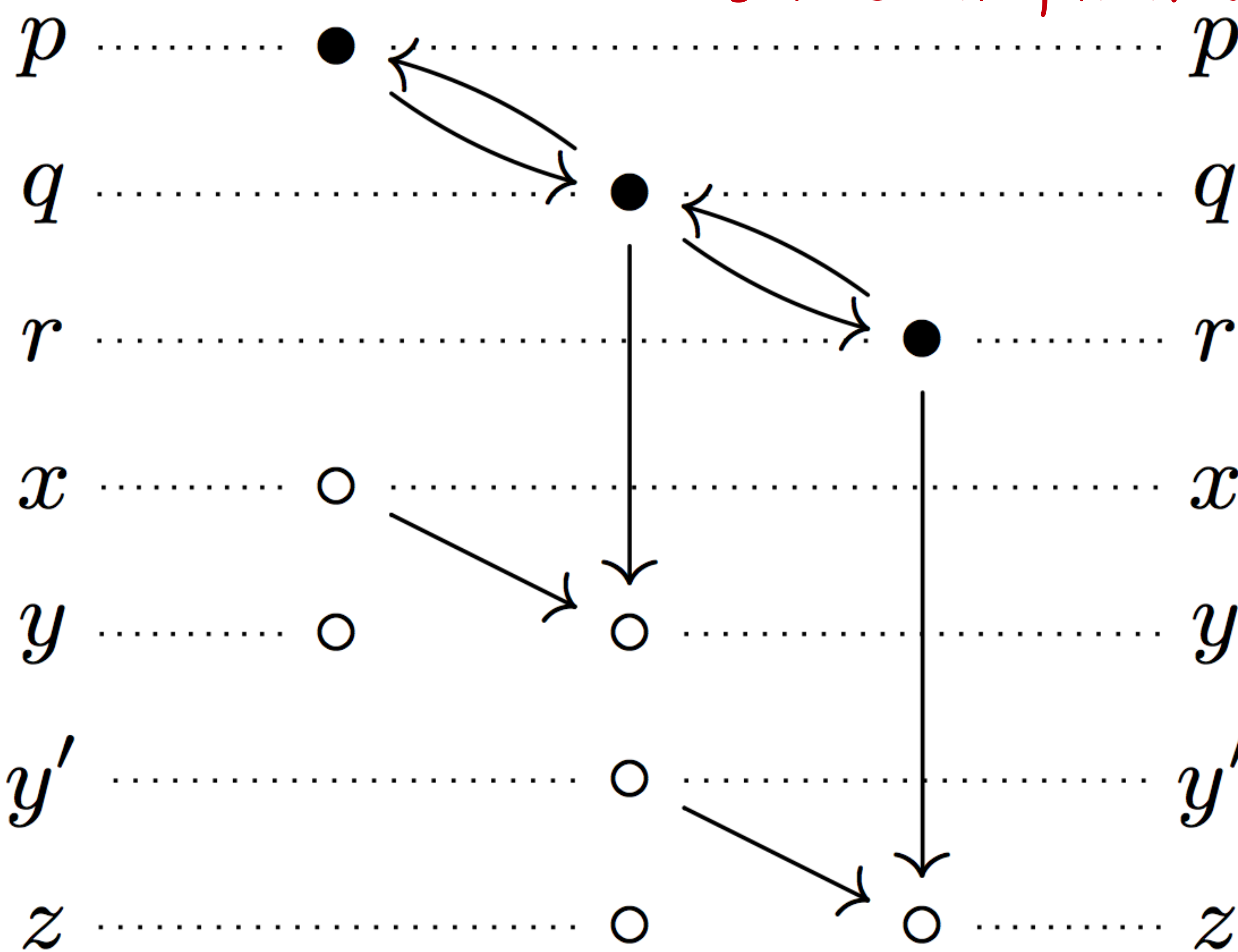
```
p.x -> q.y : k ;  q.a -> r.z : k' ;  q.y' -> r.z : k
```

Explicit dataflow



p.x -> q.y : k ; q.a -> r.z : k' ; q.y' -> r.z : k

"some" implicit dataflow



INTERACTION-ORIENTED CHOREOGRAPHIES

Trace-based semantics

$$
\begin{aligned}
\mathit{Traces}(C_1; C_2) &= \mathit{Traces}(C_1)\, \mathit{Traces}(C_2) \\
\mathit{Traces}(C_1 \mid C_2) &= \mathit{Traces}(C_1) \bowtie \mathit{Traces}(C_2) \\
\mathit{Traces}(\textsf{\textit{if}}\ e@p\ \textsf{\textit{then}}\ C_1\ \textsf{\textit{else}}\ C_2) &= e@p\ (\mathit{Traces}(C_1) \cup \mathit{Traces}(C_2)) \\
\mathit{Traces}(\textsf{\textit{while}}\ e@p\ \textsf{\textit{do}}\ C_1\ \textsf{\textit{then}}\ C_2) &= e@p\ (\mathit{Traces}(C_1)\ e@p)^*\ \mathit{Traces}(C_2) \\
\mathit{Traces}(A) &= \{A\}
\end{aligned}
$$

$$\mathit{Traces} : \mathcal{C} \rightarrow 2^{\mathcal{A}^*}$$

$$\mathcal{A} = A \cup \{e@p \mid e \in \mathbf{Expr}, p \in \mathbf{Prin}\}$$

Well-formedness criteria for choreographies

C is *well-formed* if

1. every occurrence of $C_1 \mid C_2$ in C should be such that
   $$en(C_1) \cap en(C_2) = \emptyset;$$

2. all traces $\sigma \in \mathit{Traces}(C)$ satisfy the following condition:
   If $\sigma = \sigma' \alpha \beta \sigma''$, with $\alpha, \beta \in \mathcal{A}$ then
   $$pn(\alpha) \cap pn(\beta) \neq \emptyset$$
   or
   $$\sigma' \beta \alpha \sigma'' \in \mathit{Traces}(C).$$

INFORMATION FLOWS

Danmarks Tekniske Universitet

DTU Compute
Department of Applied Mathematics and Computer Science

Language-Based Technology

Model Analyse Realise

$$f(x+\Delta x)=\sum_{i=0}^{\infty}\frac{(\Delta x)^i}{i!}f^{(i)}(x)$$

Graph-based flows and their composition

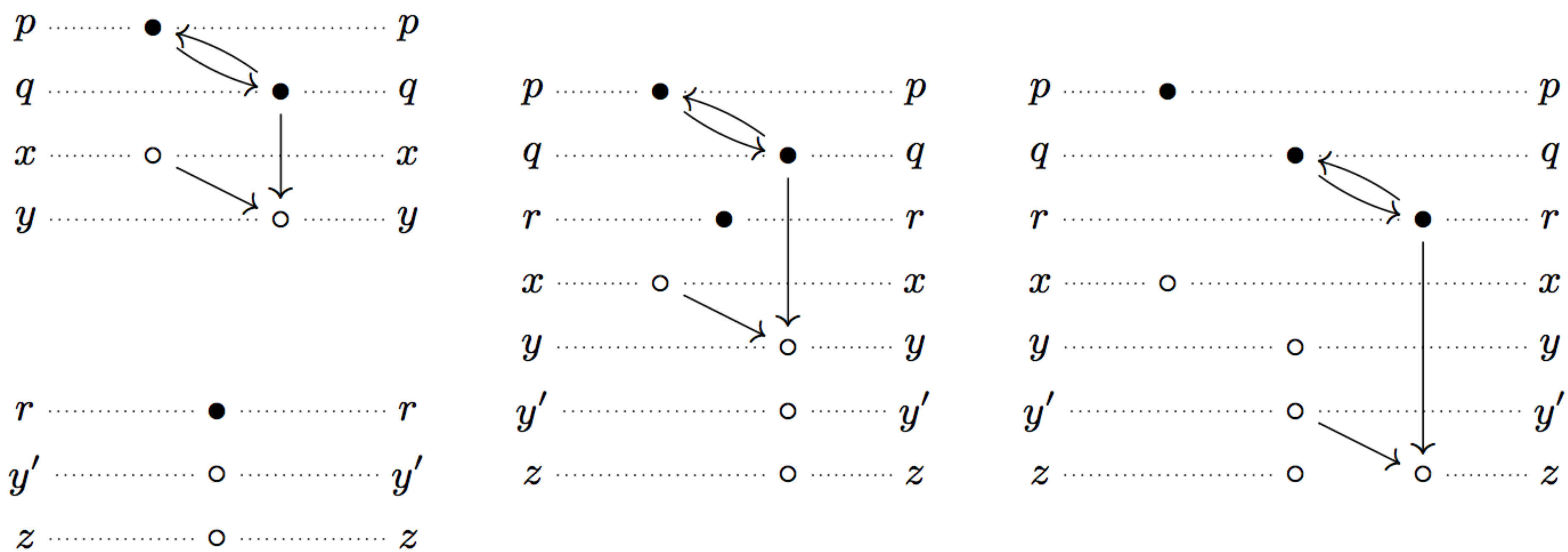
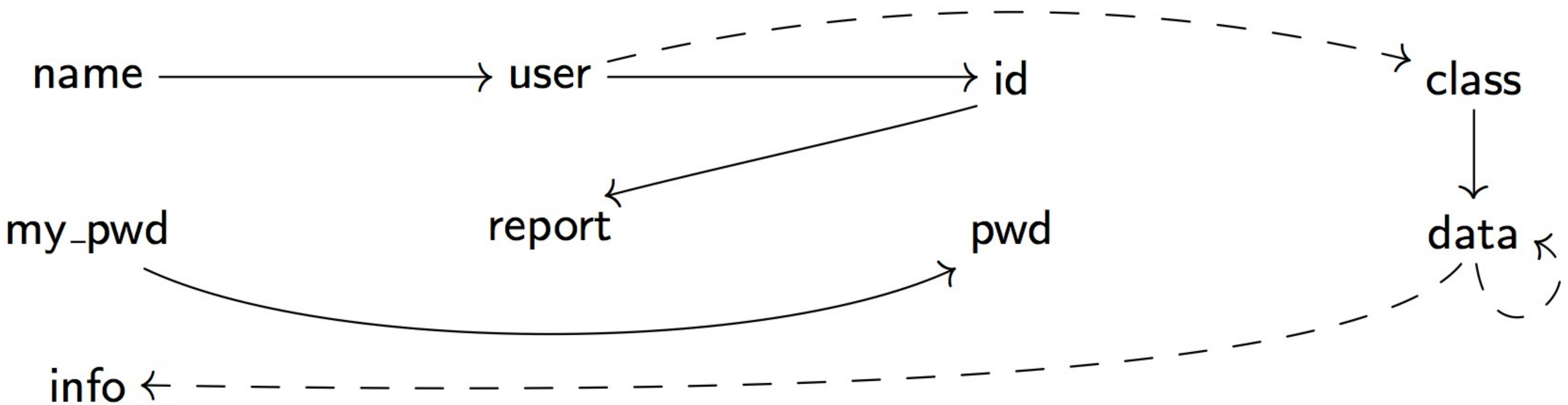


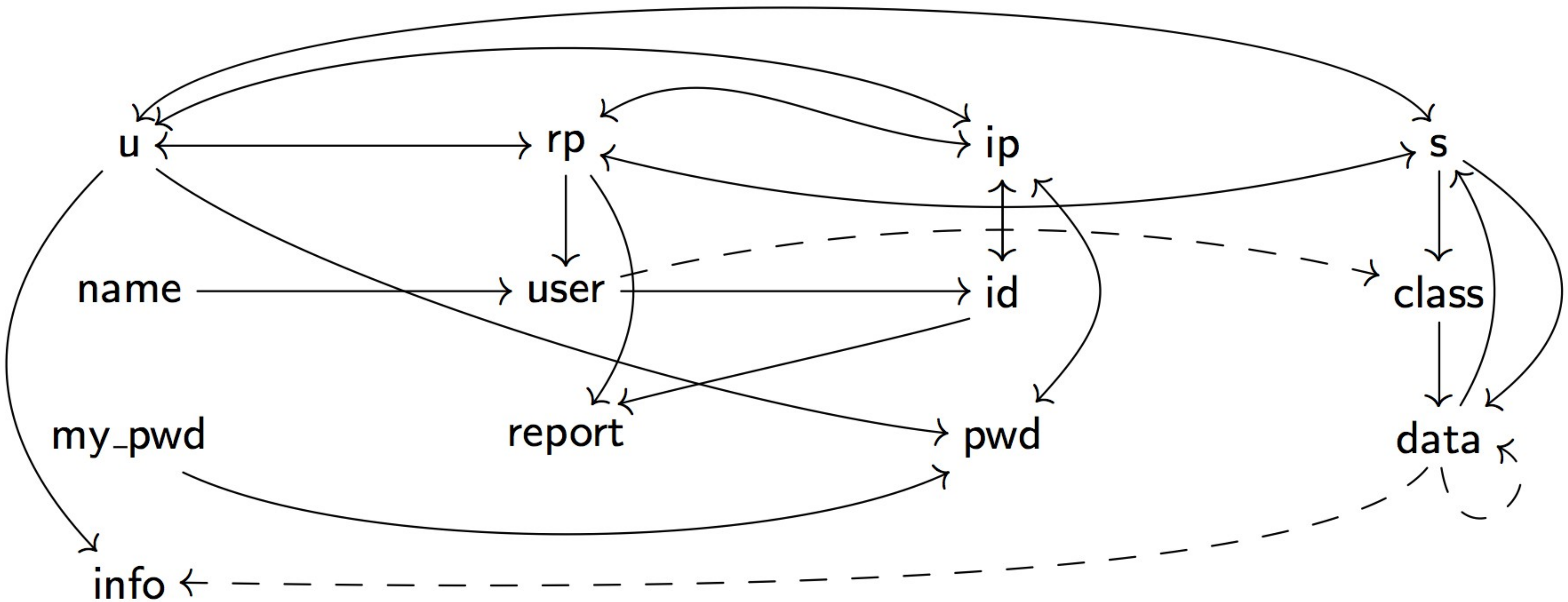
Fig. 5. Flows F_1 (top left), $\mathbf{Id}_{\{r,y',z\}}$ (bottom left), $F_1 \otimes \mathbf{Id}_{\{r,y',z\}}$ (mid) and $F2$ (right).

flow graphs as terms, e.g. $(F_1 \otimes \mathbf{Id}_{\{r,y',z\}}) \circ F2$

A big (policy) flow graph (simplified)

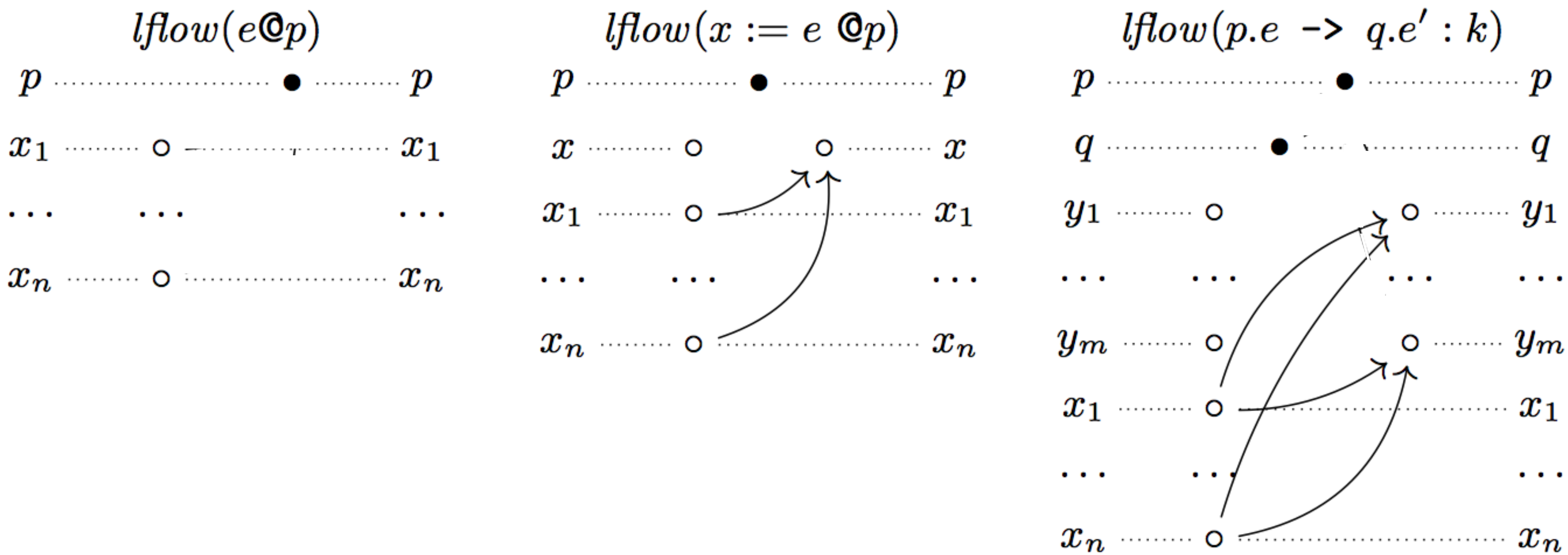


A big (policy) flow graph (simplified)



Examples of flow annotations (i)
- explicit data flows only -

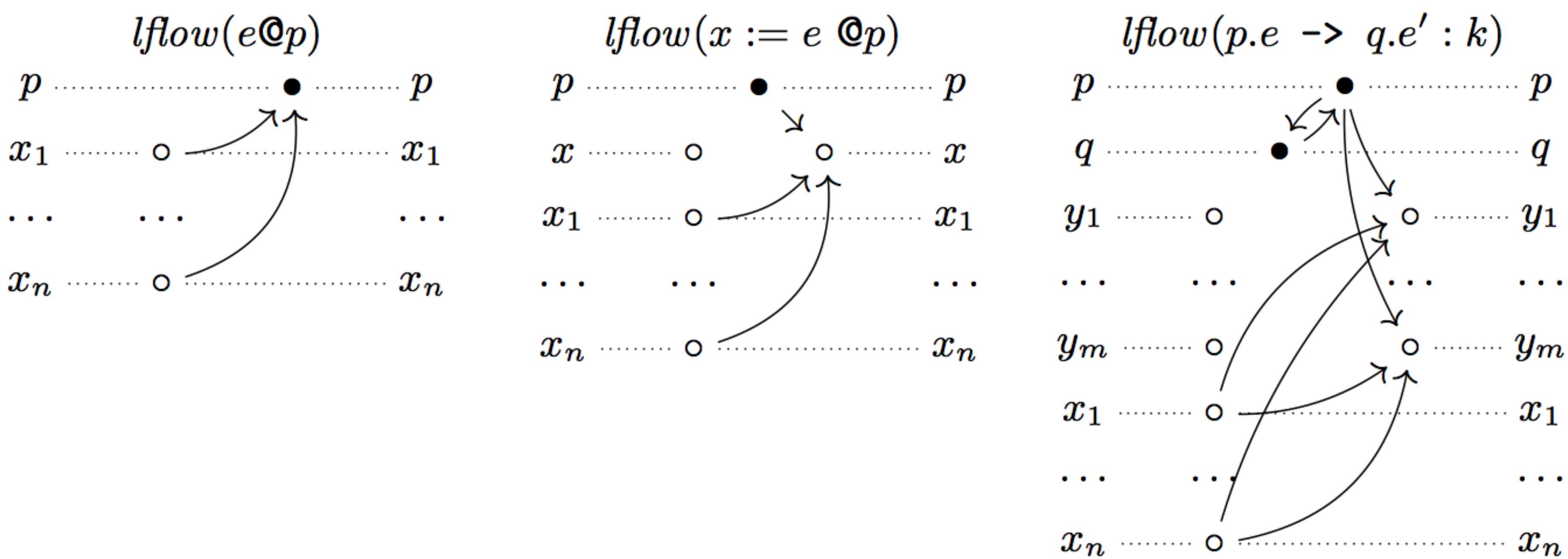
$$flow(\alpha) = lflow(\alpha) \otimes \mathbf{I}_{\mathbf{Ent} \setminus en(\alpha)} \qquad lflow(\mathsf{skip}) = \mathbf{0}$$



$$fn(e) = \{x_1, \ldots, x_n\}$$
$$fn(e') = \{y_1, \ldots, y_m\}$$

Examples of flow annotations (ii)
- some implicit flows included -

$$flow(\alpha) = lflow(\alpha) \otimes \mathbf{I}_{\mathbf{Ent} \setminus en(\alpha)} \qquad lflow(\mathsf{skip}) = \mathbf{0}$$

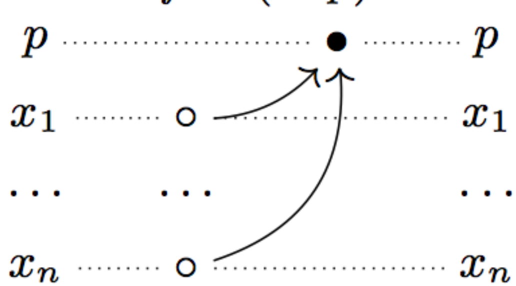


$$fn(e) = \{x_1, \ldots, x_n\}$$
$$fn(e') = \{y_1, \ldots, y_m\}$$

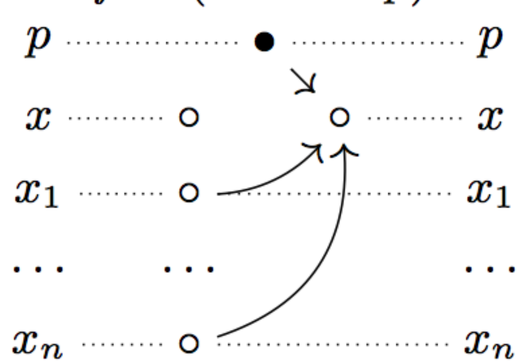
Examples of flow annotations (ii)
- implicit flows included -

$flow(\alpha) = lflow(\alpha) \otimes \mathbf{I}_{\mathbf{Ent} \setminus en(\alpha)}$ $\qquad$ $lflow(\mathsf{skip}) = \mathbf{0}$

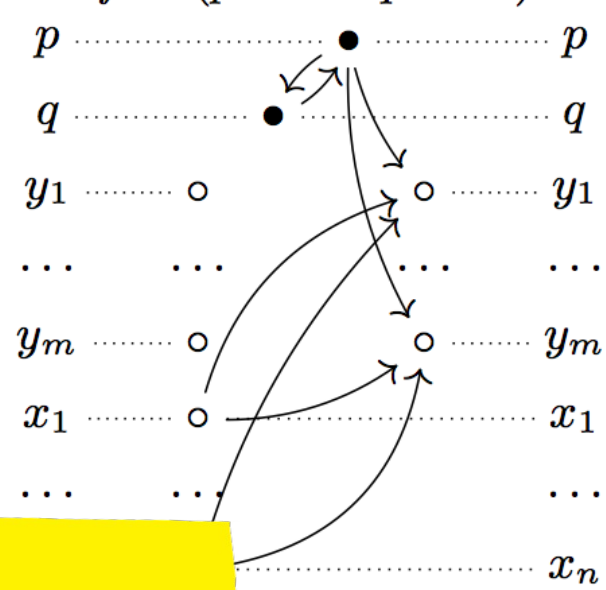
$lflow(e @ p)$



$lflow(x := e\ @ p)$



$lflow(p.e \texttt{ -> } q.e' : k)$



Well-formed annotations

$flows : \mathcal{A} \rightarrow \mathcal{F}$ *is* well-formed *iff*

$\forall \alpha \in \mathcal{A} : flows(\alpha) = F \otimes \mathbf{I}_{\mathbf{Ent} \setminus en(F)} \wedge en(F) \subseteq en(\alpha)$

$fn(e) = \{x_1, \ldots, x_n\}$

$fn(e') = \{y_1, \ldots, y_m\}$

CHECKING INFORMATION FLOW POLICIES

When is a policy satisfied?

$$
\begin{array}{rcl}
C \models \Pi & \textit{iff} & \mathit{Traces}(C) \models \Pi \\
T \models \Pi & \textit{iff} & \forall \sigma \in T : \sigma \models \Pi \\
\sigma \models \Pi & \textit{iff} & \sigma = \sigma''\sigma'\sigma''' \Rightarrow \sigma' \vdash \Pi \\
\sigma \vdash \Pi & \textit{iff} & \mathit{flows}(\sigma) \models \Pi \\
F \models \Pi & \textit{iff} & \forall \eta, \eta' \in en(\Pi) : \\
& & \quad i_F(\eta) \rightarrow^*_{G_F} o_F(\eta') \;\Rightarrow\; i_\Pi(\eta) \rightarrow^*_{G_\Pi} o_\Pi(\eta')
\end{array}
$$

In words: no sub-trace can have a flow not allowed by the policy.

Checking types/policies

$$\text{en(Ent)}\ \frac{\vdash C_1 : \Pi \qquad \vdash C_2 : \Pi}{\vdash C_1 ; C_2 : \Pi} \qquad \frac{\vdash C_1 : \Pi \qquad \vdash C_2 : \Pi}{\vdash C_1 \mid C_2 : \Pi} \qquad \frac{A \models \Pi}{\vdash A : \Pi}$$

$$\frac{e@p \models \Pi \qquad \vdash C_1 : \Pi \qquad \vdash C_2 : \Pi}{\vdash \textsf{if } e@p \textsf{ then } C_1 \textsf{ else } C_2 : \Pi} \qquad \frac{e@p \models \Pi \qquad \vdash C_1 : \Pi \qquad \vdash C_2 : \Pi}{\vdash \textsf{while } e@p \textsf{ do } C_1 \textsf{ then } C_2 : \Pi}$$

Main result

$$\vdash C : \Pi \ \textit{then}\ C \models \Pi$$

In practice: just check all actions and conditions.

inferring types/over-approximating flows

$$\text{en(Ent)}\ \frac{\vdash C_1 : F_1 \qquad \vdash C_2 : F_2}{\vdash C_1 ; C_2 : F_1 \odot F_2} \qquad \frac{\vdash C_1 : F_1 \qquad \vdash C_2 : F_2}{\vdash C_1 \mid C_2 : F_1 \otimes F_2} \qquad \frac{}{\vdash A : \mathit{flow}(F)}$$

$$\frac{\vdash C_1 : F_1 \qquad \vdash C_2 : F_2}{\vdash \textsf{if } e@p \textsf{ then } C_1 \textsf{ else } C_2 : \mathit{flows}(\mathsf{e@p}) \odot (F_1 \otimes F_2))}$$

$$\frac{\vdash C_1 : F_1 \qquad \vdash C_2 : F_2}{\vdash \textsf{while } e@p \textsf{ do } C_1 \textsf{ then } C_2 : \mathit{flows}(e@p) \odot (F_1 \odot \mathit{flows}(e@p))^{\infty} \odot F_2}$$

$$\text{where } F \odot G = F \otimes G \otimes (F \circ G)$$

Conjecture

$$\vdash C : F \text{ then } C \models F$$

$C \models \Pi$ could be checked by checking $F \models \Pi$

CONCLUDING REMARKS

Summary

1. Simple choreography description language to provide interaction-oriented specifications of concurrent systems

2. Graph-based information flow specifications
   (i) semantics (flow annotations for events)
   (ii) policies

3. Sound type system based on over-approximation of the flows in a specification

Future work?

Some issues/extensions worth considering:

- Over- and Under-approximations
- Type inference
- Non-interference
- Intransitive policies
- Compositionality and Dynamicity
- HPC primitives (e.g. MPI-like scatter/agg.)
- Projections (distributed implementation)

THANKS!

DTU Compute
Department of Applied Mathematics and Computer Science

Questions?

albl@dtu.dk
albertolluch.com