# Synchronised Hyperedge Replacement as a Model for Service Oriented Computing

Emilio Tuosto
University of Leicester

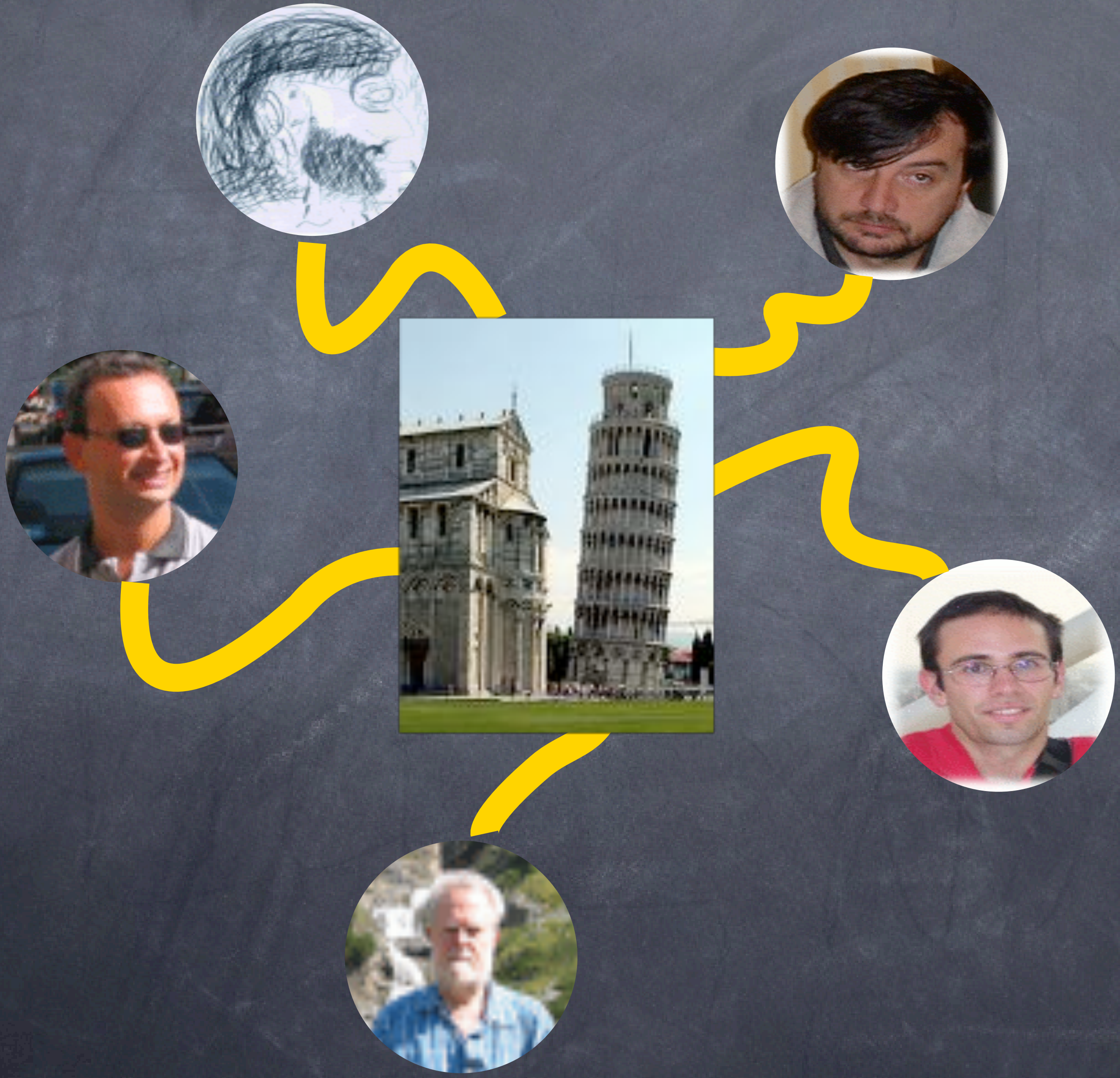
International PhD School in Theory and Practice of

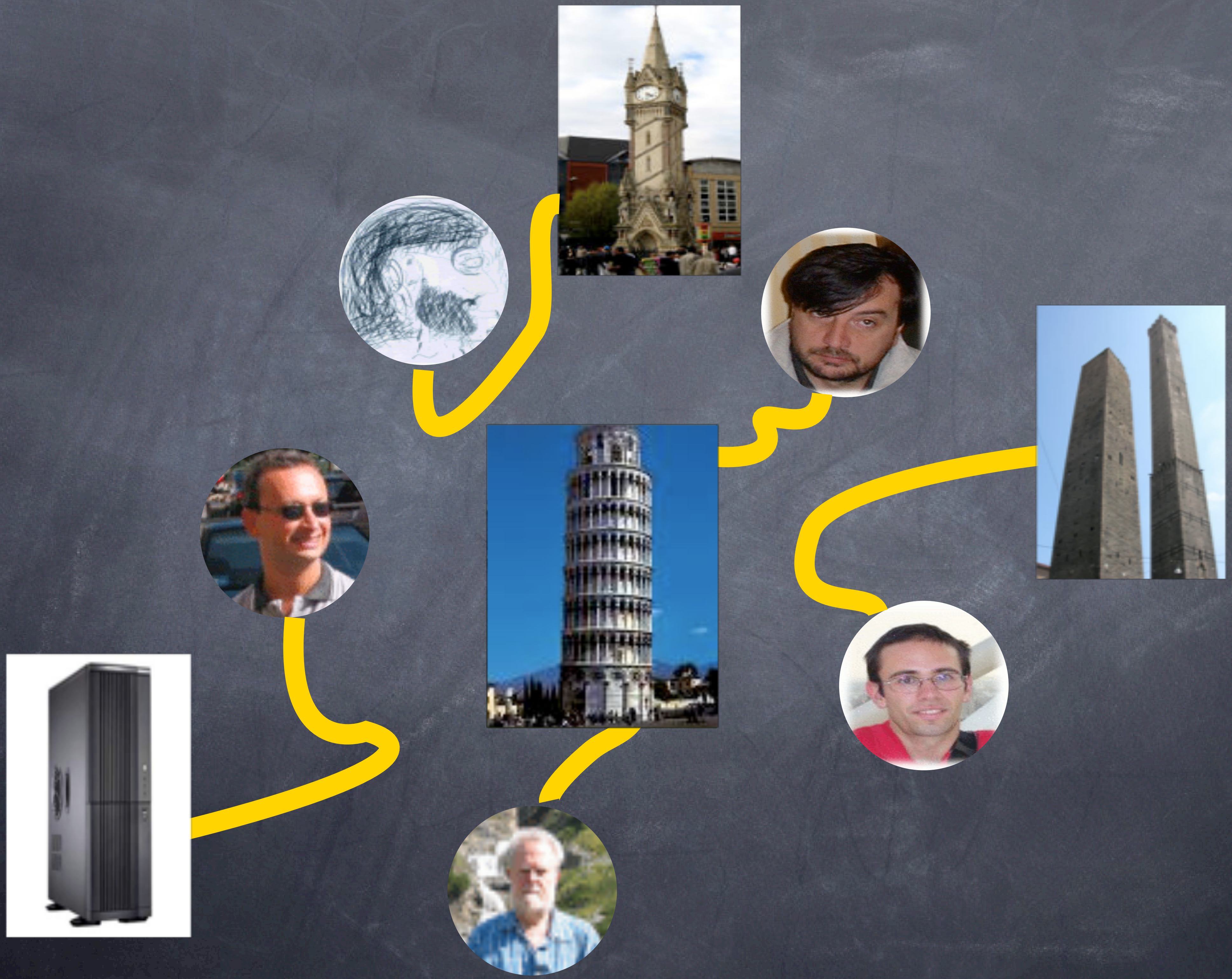
**Business Process Execution and Service Orientation**

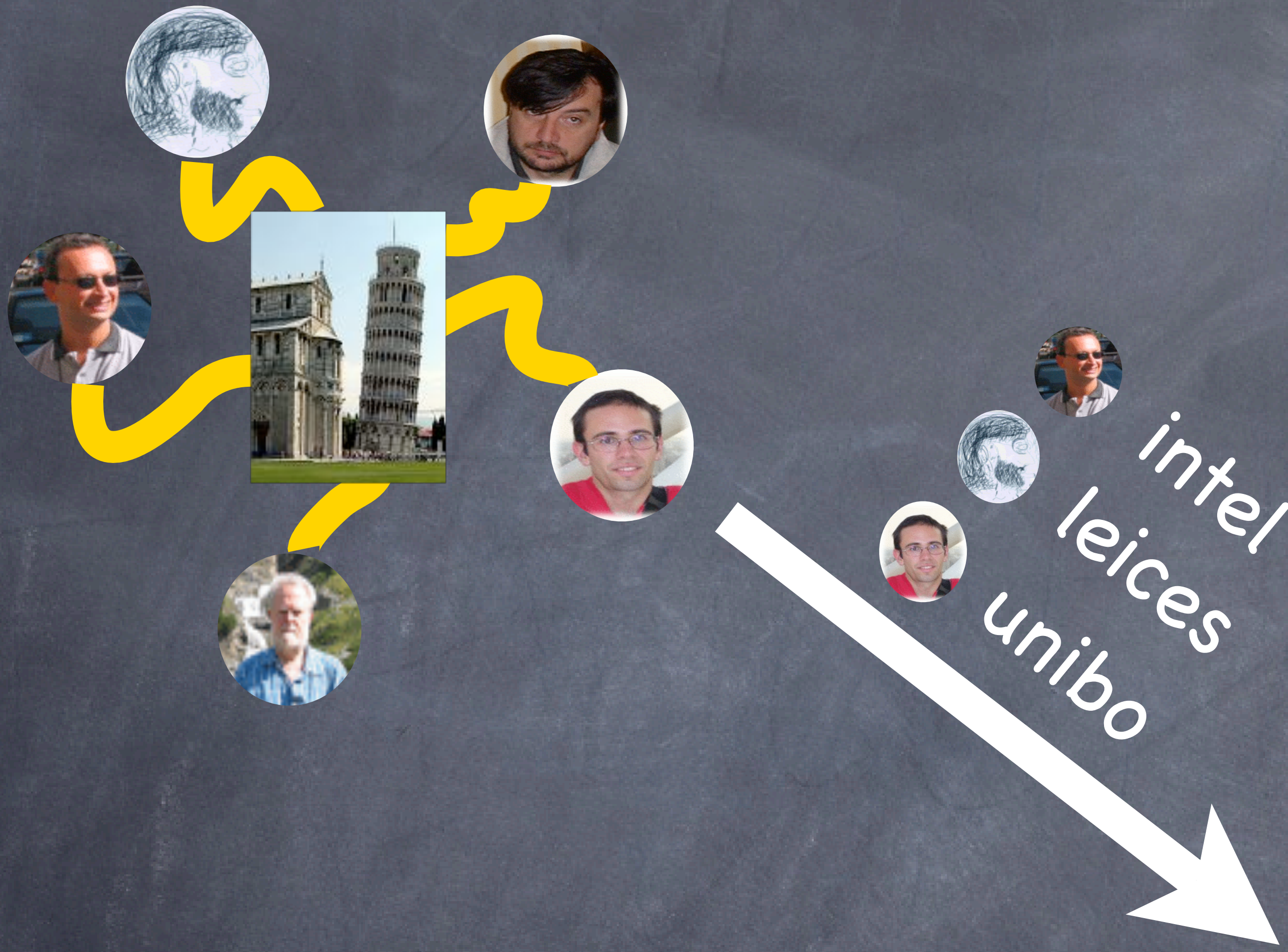
The IT University of Copenhagen, Denmark, October 16 - 19, 2007

# SHR framework

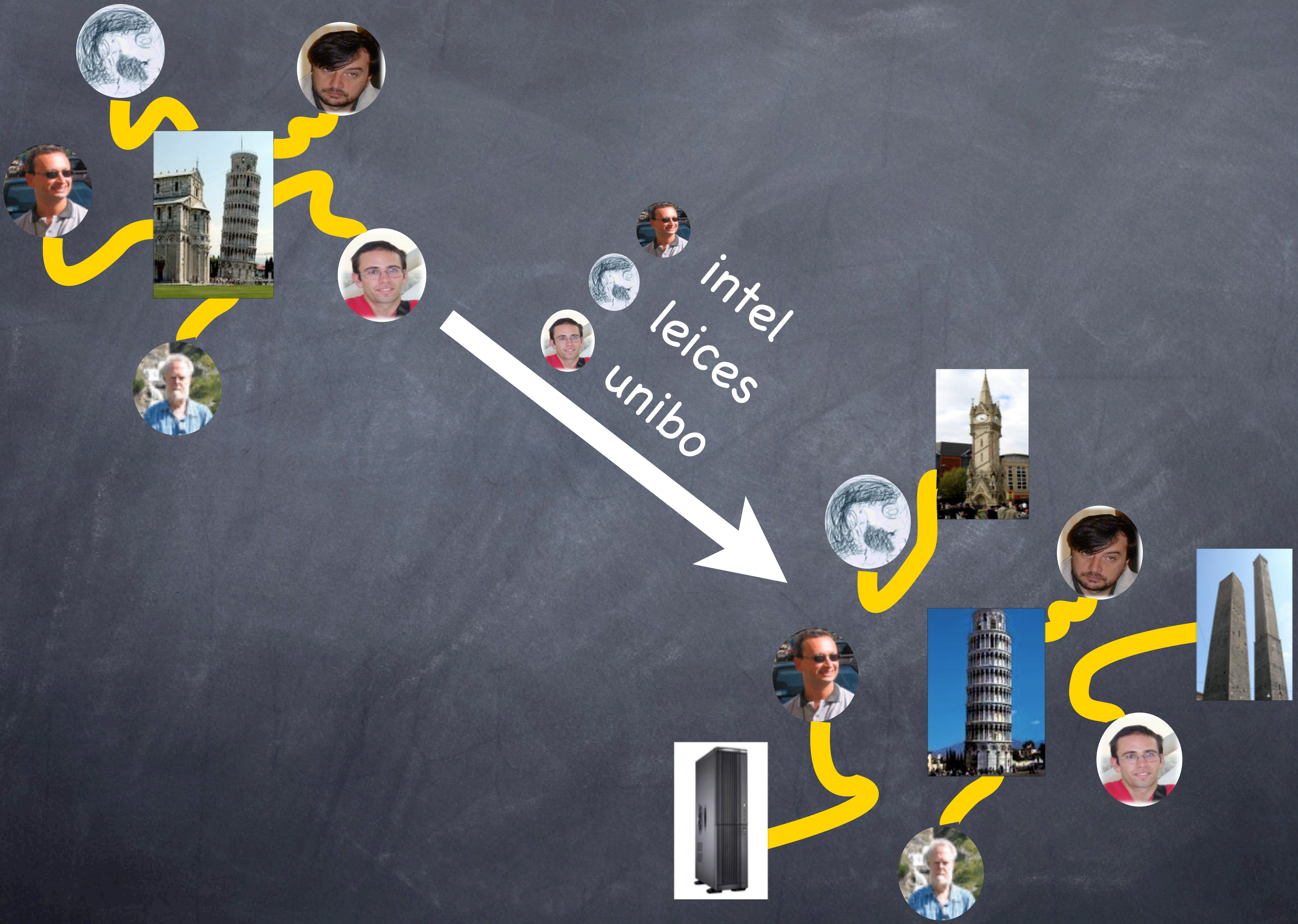
- SHR as a uniform framework for (non-)functional aspects of SOC
  - Context-free flavour
  - "SOC systems as Hypergraphs" & "SOC computations as SHR"
    - Components = hyperedges
    - Systems = bunches of hyperedges
    - Computing = rewrite hypergraphs...(distributed constraint solving)
    - ...using "some" (parameterisable) synchronisation policy







intel
leices
unibo



intel

leices

unibo

# Models

- Process calculi
  - CSP, CCS and π-calculus...
- Graph-based models
  - Synchronised Hyperedge Replacement (SHR)
  - Originally, SHR as a model of distributed systems and software architectures but
  - expressive enough to model many process calculi
- ...

# SHR features

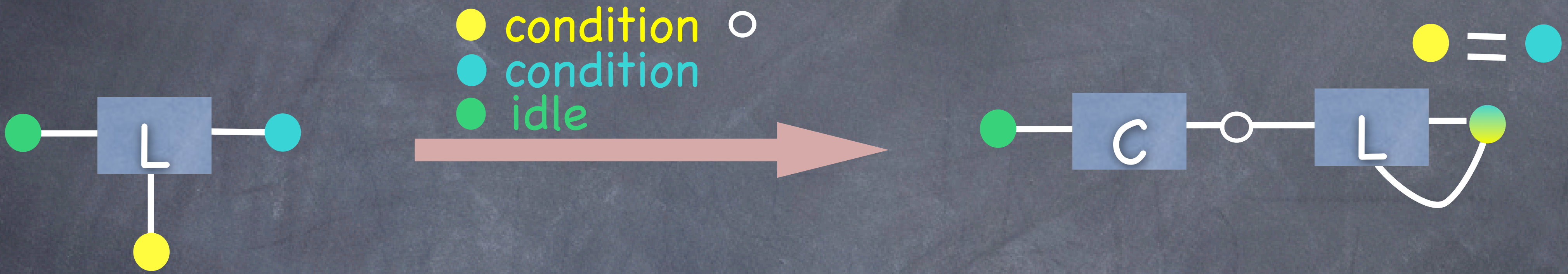
- can express many forms of synchronisation
- constraint satisfaction guide rewriting by synchronising “context-free productions”
    - components’ behaviour independently specified by productions
    - productions impose conditions on adjacent nodes
    - global transitions as application of “compatible” productions
- QoS mechanism for driving the rewritings

# SOC

- Modern distributed systems
  - complex and heterogeneous
  - many architectural levels
  - many communication infrastructures
  - geographically distributed
  - highly dynamic
- SOC as modelling paradigm

- Services are
  - independently specified/published
  - searched/discovered and dynamically assembled
  - dynamically reconfiguration
  - mobile and requiring complex synchronisations
  - “QoS aware”

# SHR rewriting: in a nutshell

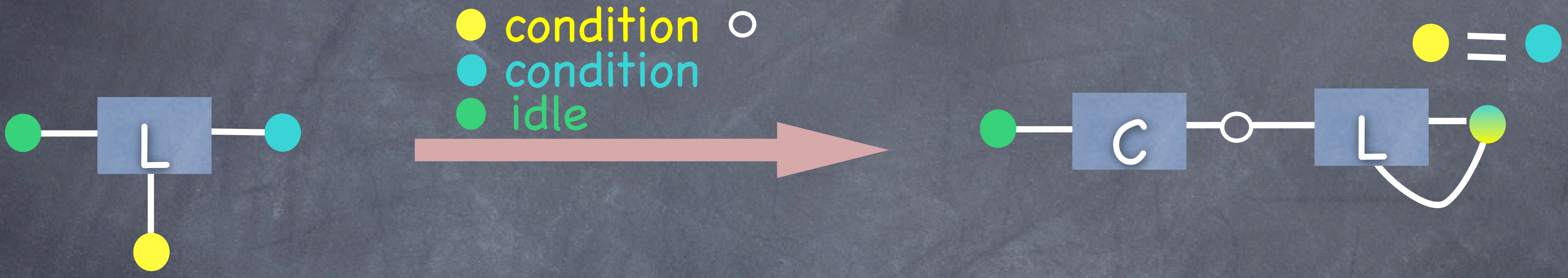


● condition ○

● condition

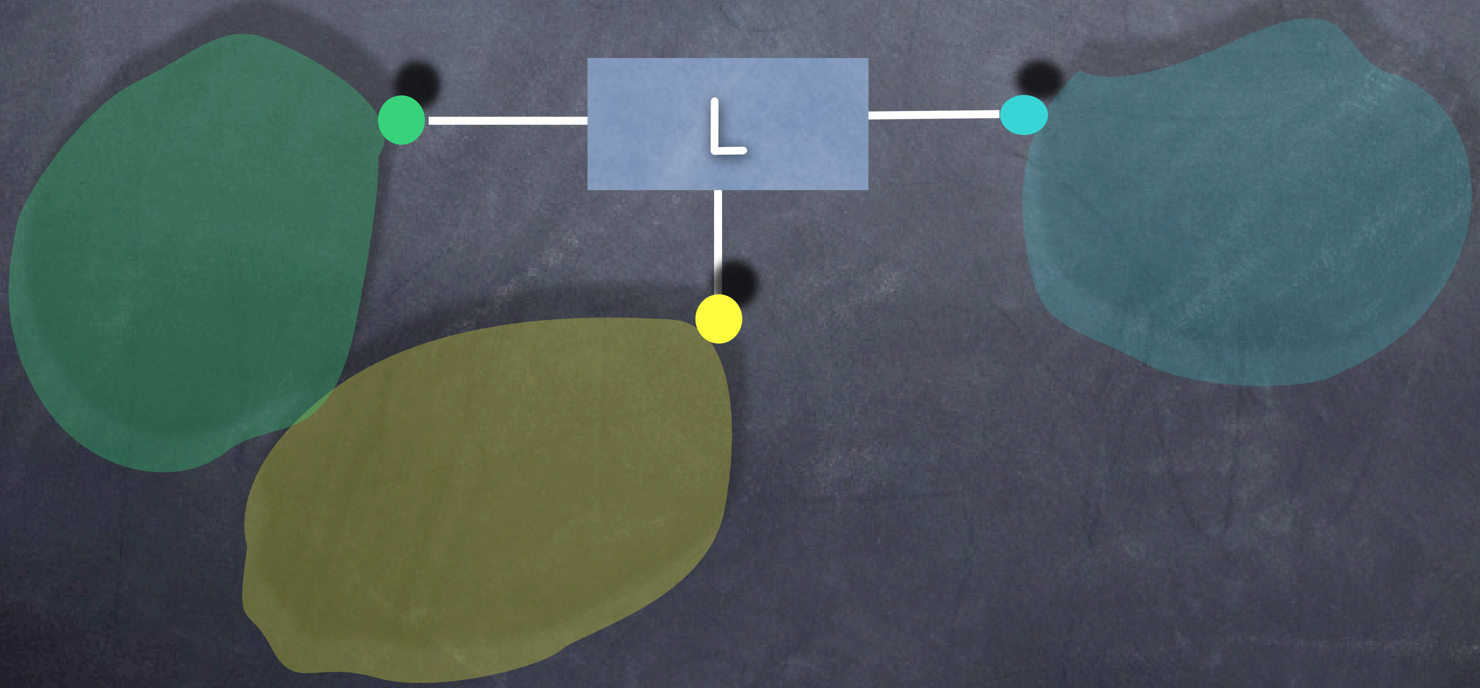
● idle

# SHR rewriting: in a nutshell

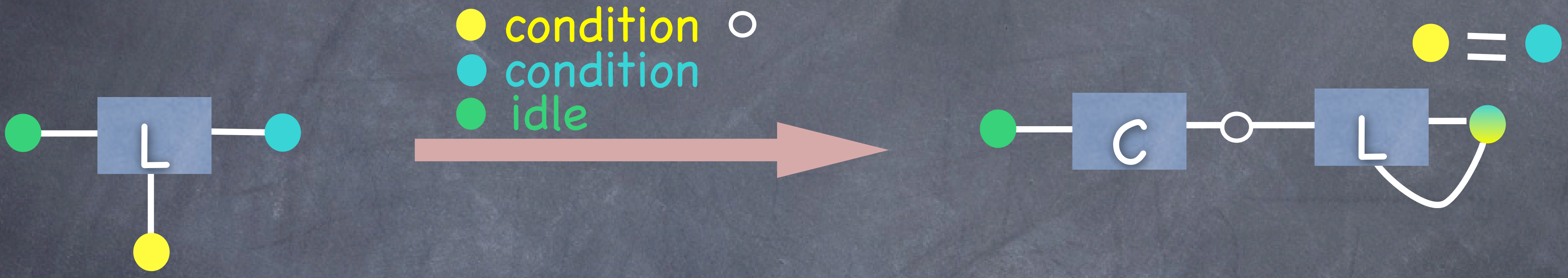


● condition ○
● condition
● idle

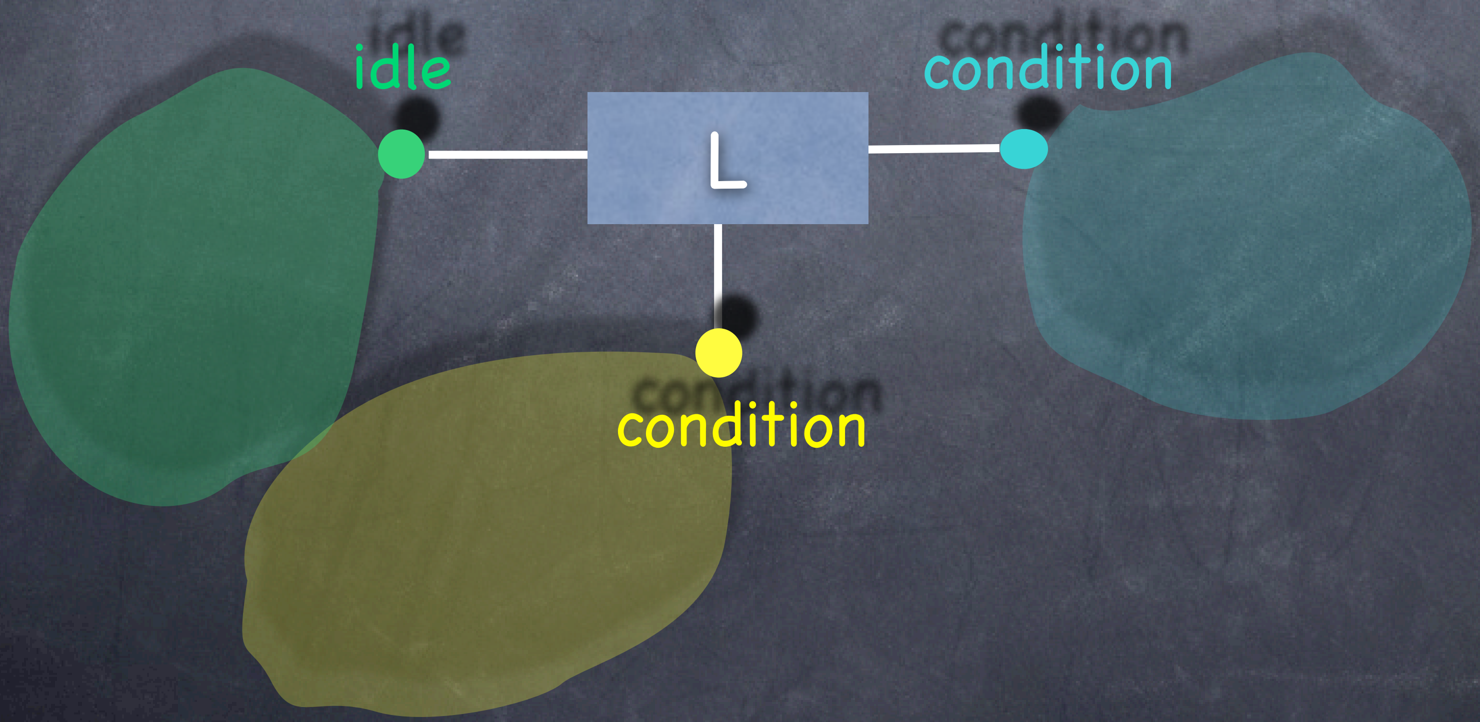
● = ●



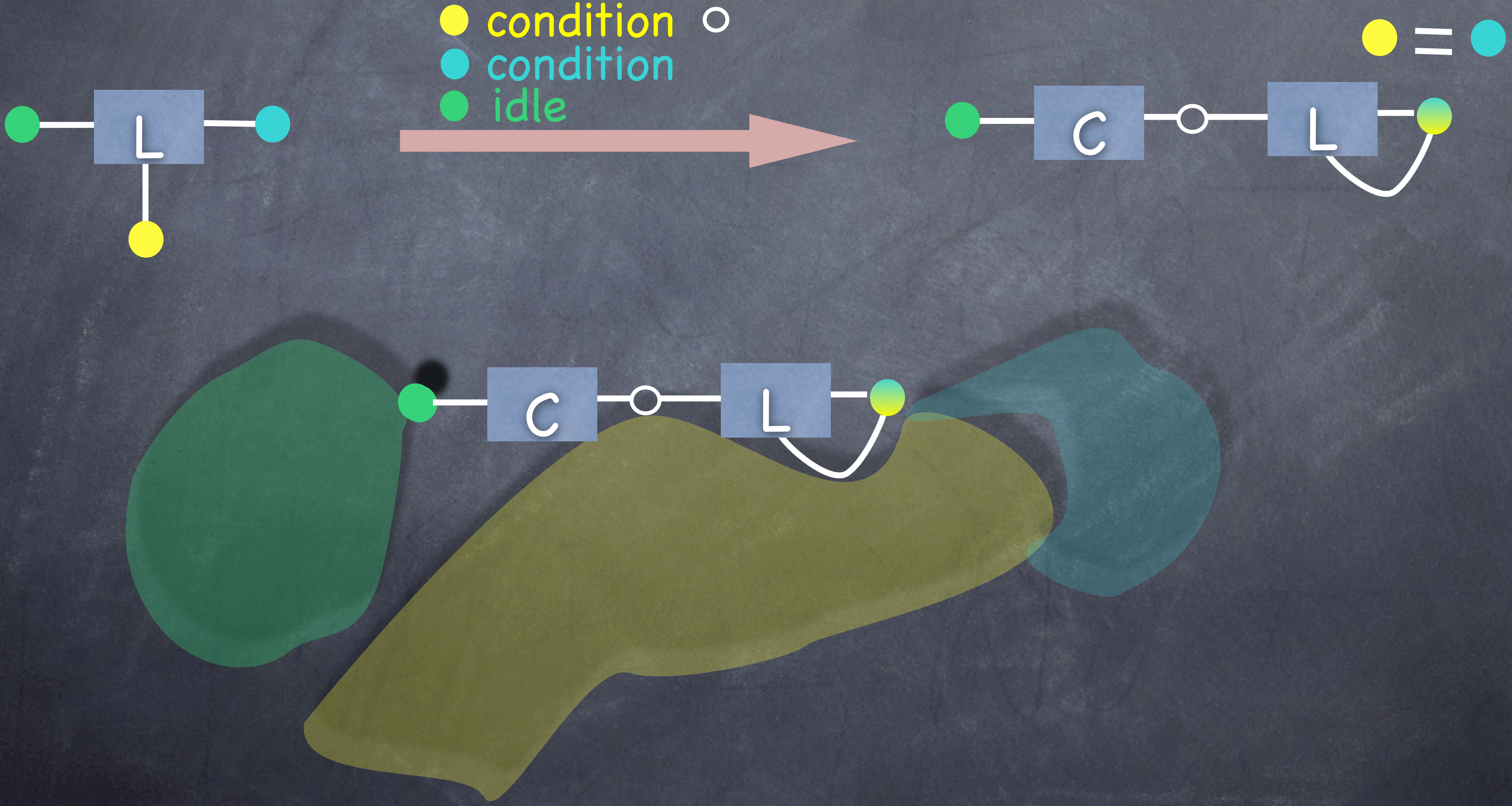
# SHR rewriting: in a nutshell



● condition ○
● condition
● idle



# SHR rewriting: in a nutshell



● condition ○
● condition
● idle

● = ●

# Why we deem SHR suitable for SOC

# Why we deem SHR suitable for SOC

- Edge replacement: “local”

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication

  ● = ● ⟼ ●

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication

  ● = ● ⟼ ●
- Expressive for

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication
  ● = ● ⟼ ●

- Expressive for
  - modelling process calculi

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication

  ● = ● ⟼ ●

- Expressive for
  - modelling process calculi
  - distributed coordination

# Why we deem SHR suitable for SOC

- Edge replacement: “local”
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication

  ● = ● ⟼ ●

- Expressive for
  - modelling process calculi
  - distributed coordination
  - application level QoS

# Why we deem SHR suitable for SOC

- Edge replacement: "local"
- Multi-party synchronisation
- New node creation
- Node fusion: model of mobility and communication
  ● = ● ⟼ ●

- Expressive for
  - modelling process calculi
  - distributed coordination
  - application level QoS
  - sophisticated synchronisations

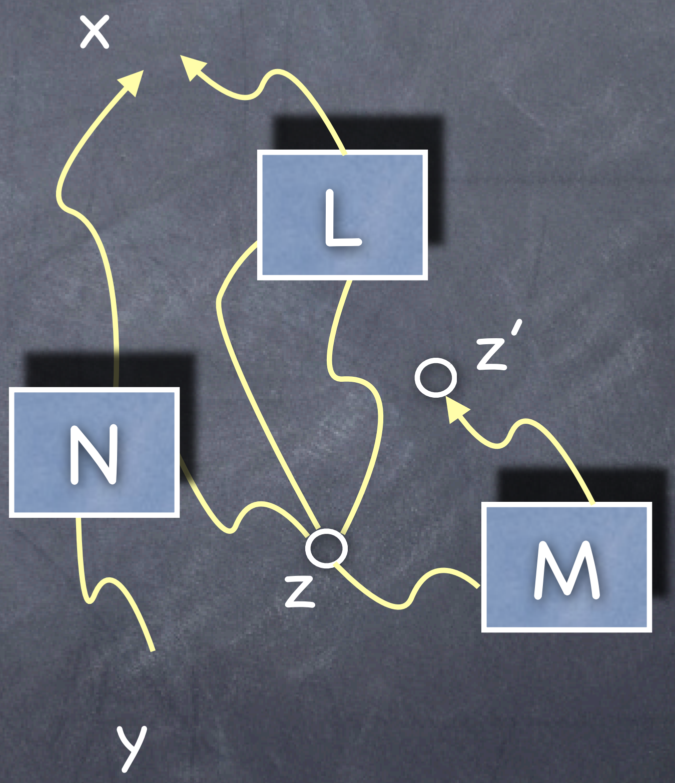
# Plan

- Give the basic definitions for SHR
- Analise 2 specific cases:
  - Milner synchronisation (with(out) mobility)
  - SHReQ
- ADR (if time allows)

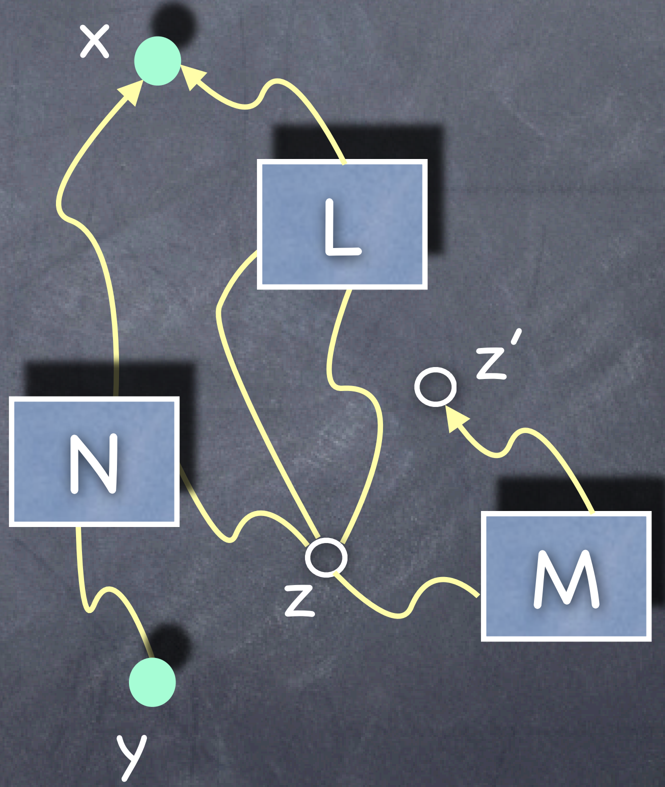
# Hypergraphs

Syntax

# Exercises

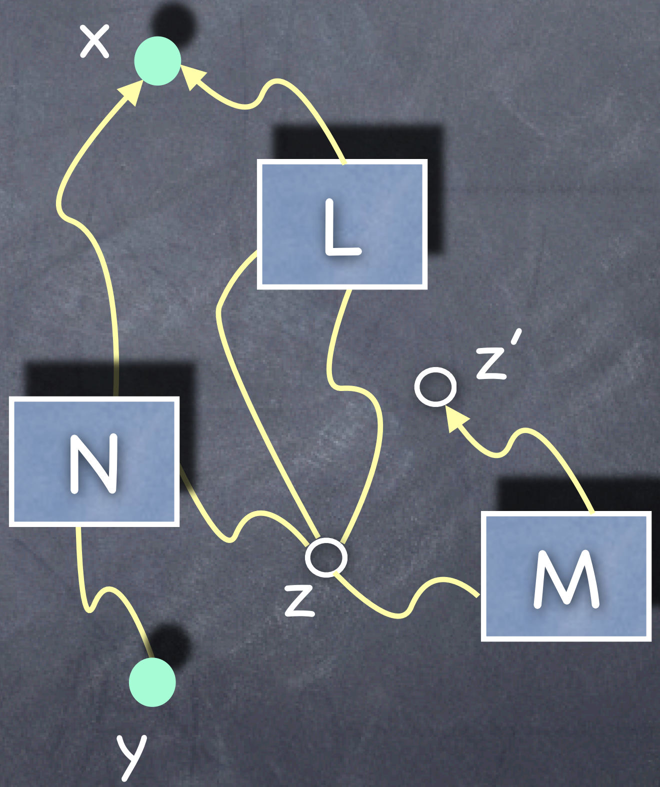


# Exercises



# Exercises

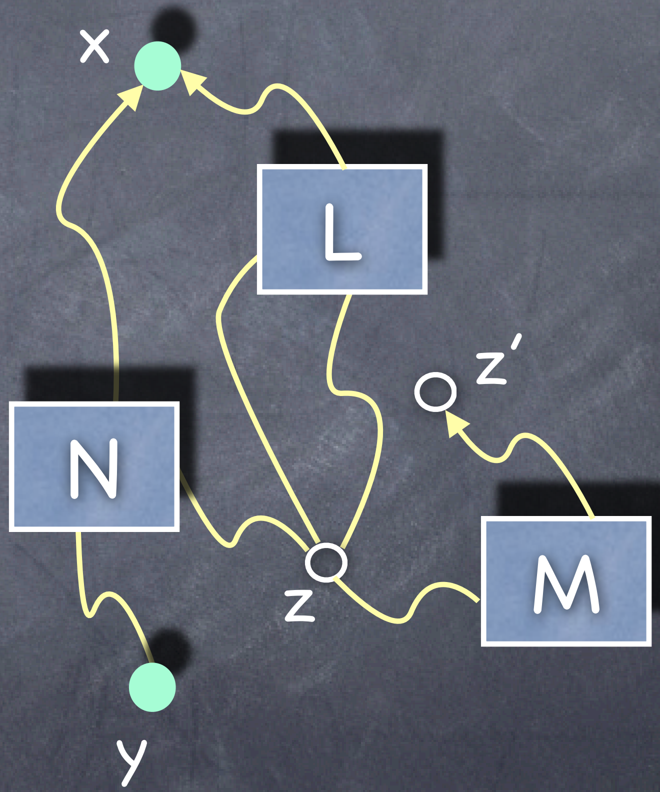
- Give the syntactic judgement of the hypergraph on the right



# Exercises

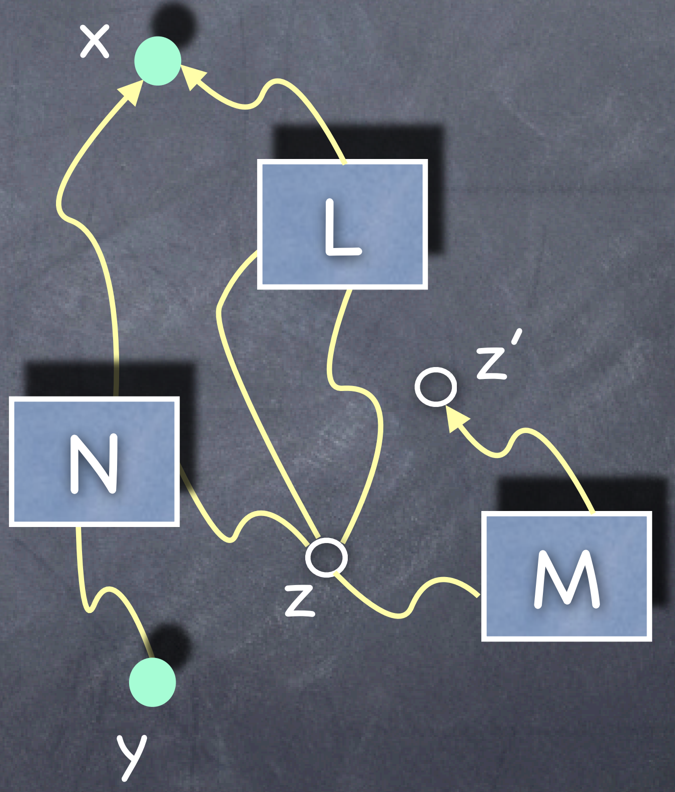
- Give the syntactic judgement of the hypergraph on the right

- Draw the graph of the following judgements:



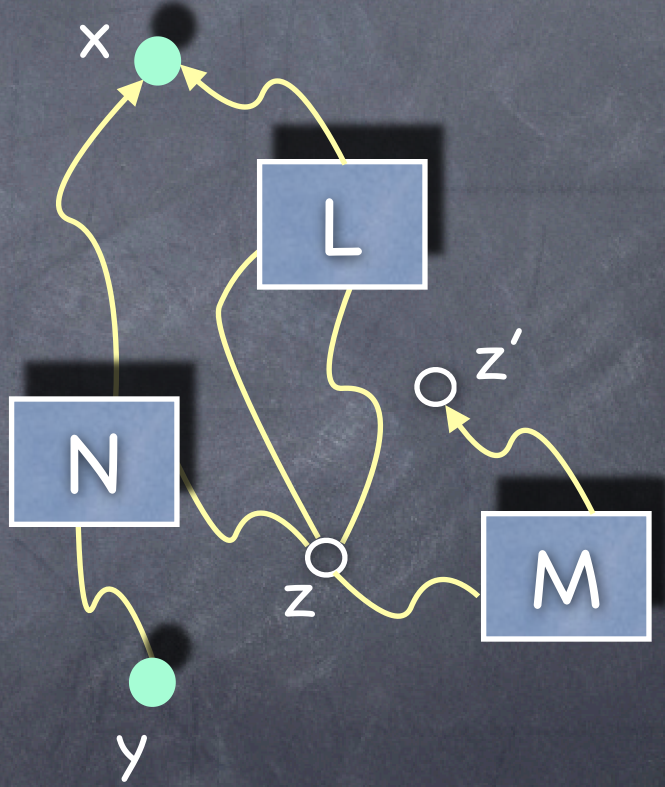
# Exercises

- Give the syntactic judgement of the hypergraph on the right
- Draw the graph of the following judgements:
    - $x,y \vdash L(x,y) \mid L(x) \mid M(y)$



# Exercises

- Give the syntactic judgement of the hypergraph on the right

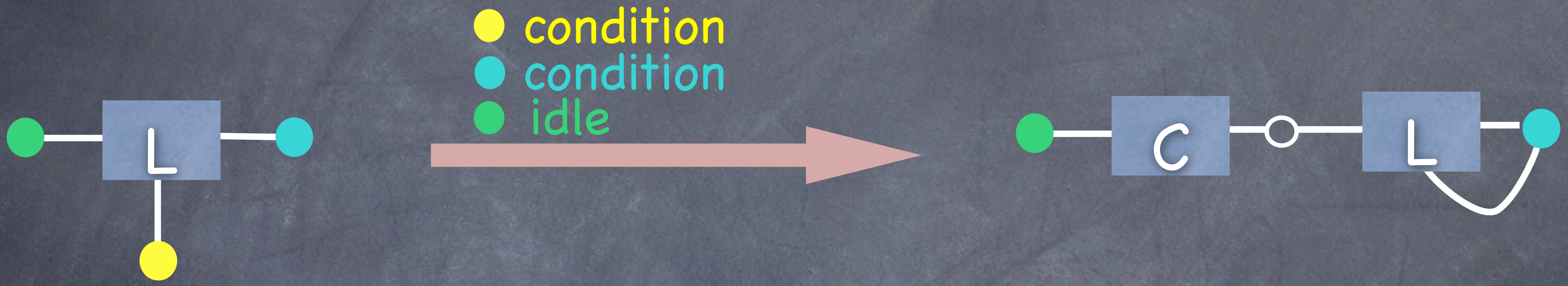


- Draw the graph of the following judgements:
  - x,y ⊢ L(x,y) | L(x) | M(y)
  - x,y ⊢ L(x,y) | L(x,z)

# The simplest SHR: Basic Milner SHR

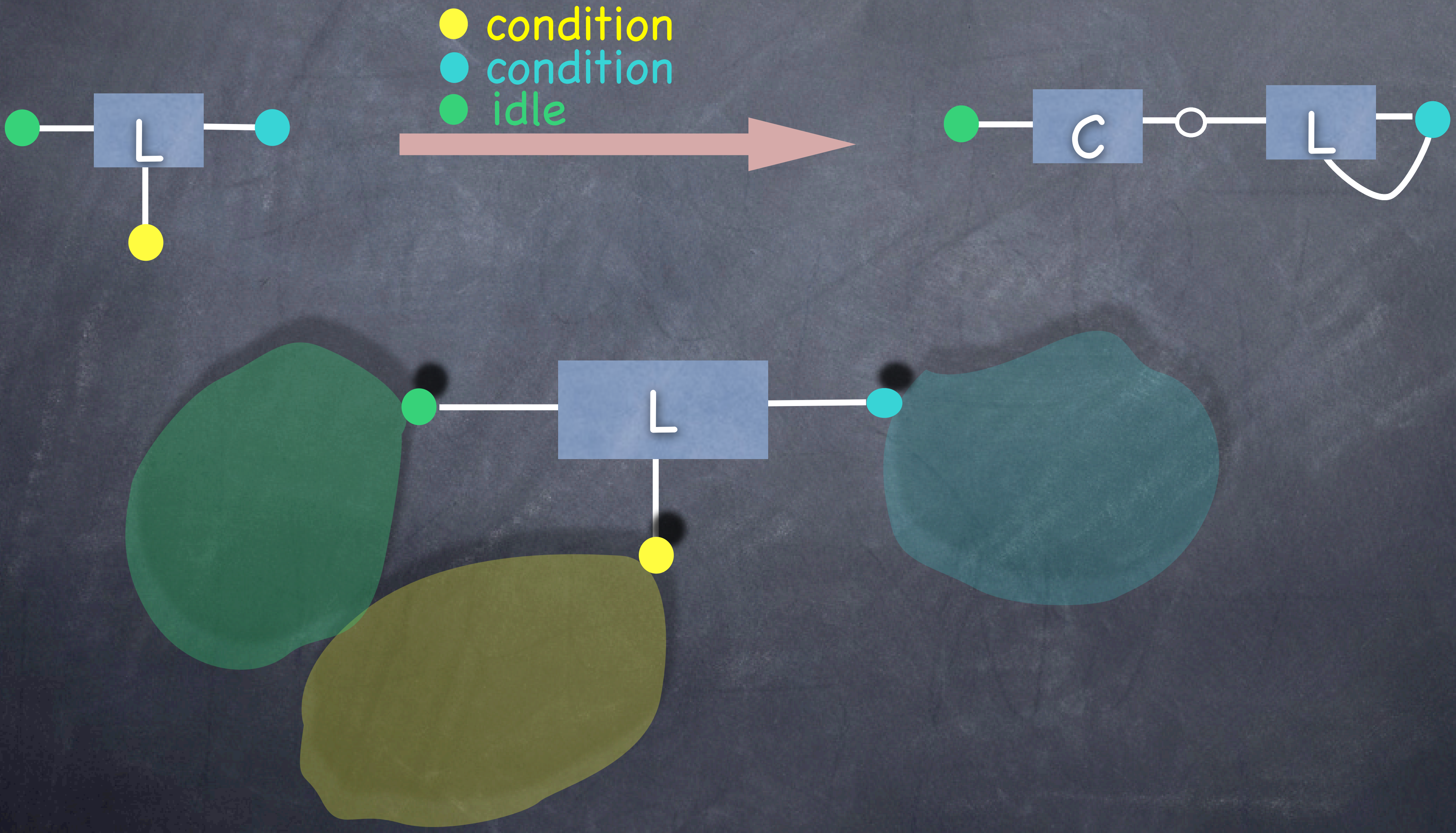
“Milner” synchronisation
without mobility [fhlmt05]

# bMSHR: in a nutshell



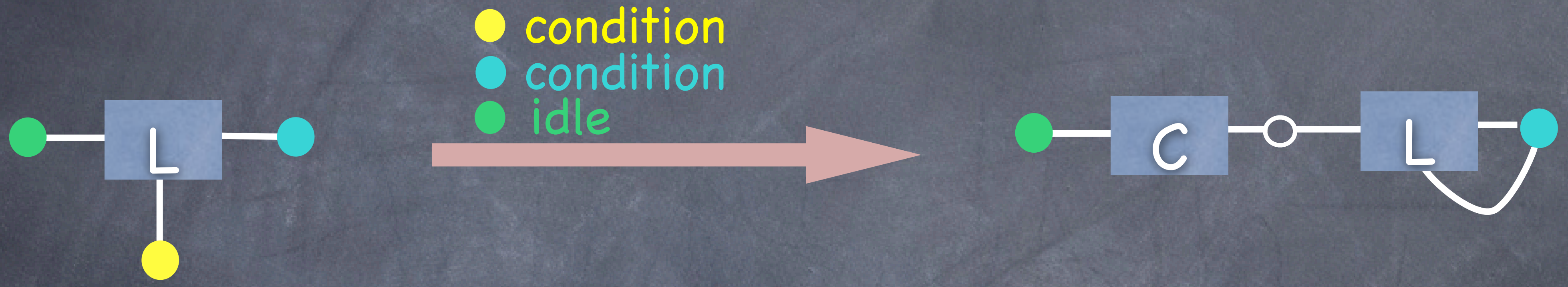
● condition
● condition
● idle

# bMSHR: in a nutshell

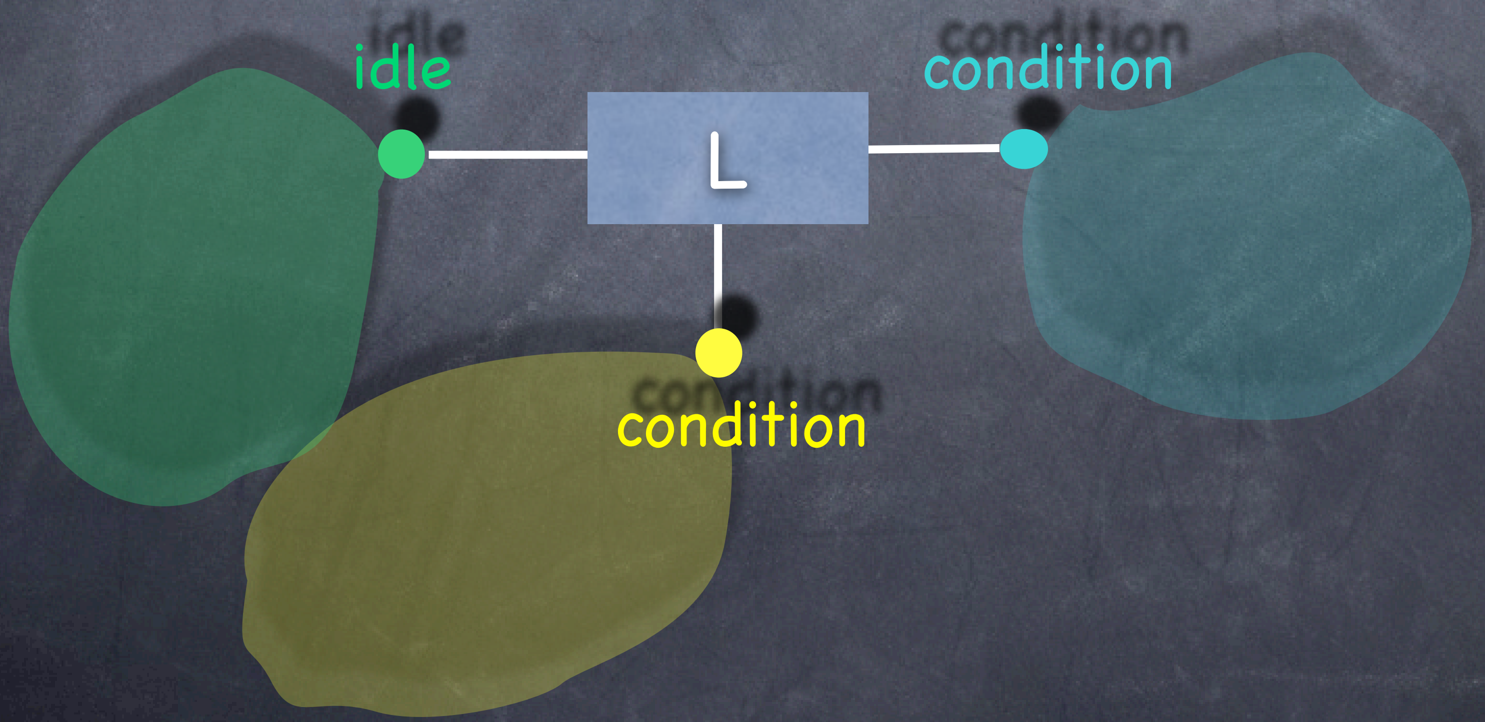


● condition
● condition
● idle

# bMSHR: in a nutshell



● condition
● condition
● idle

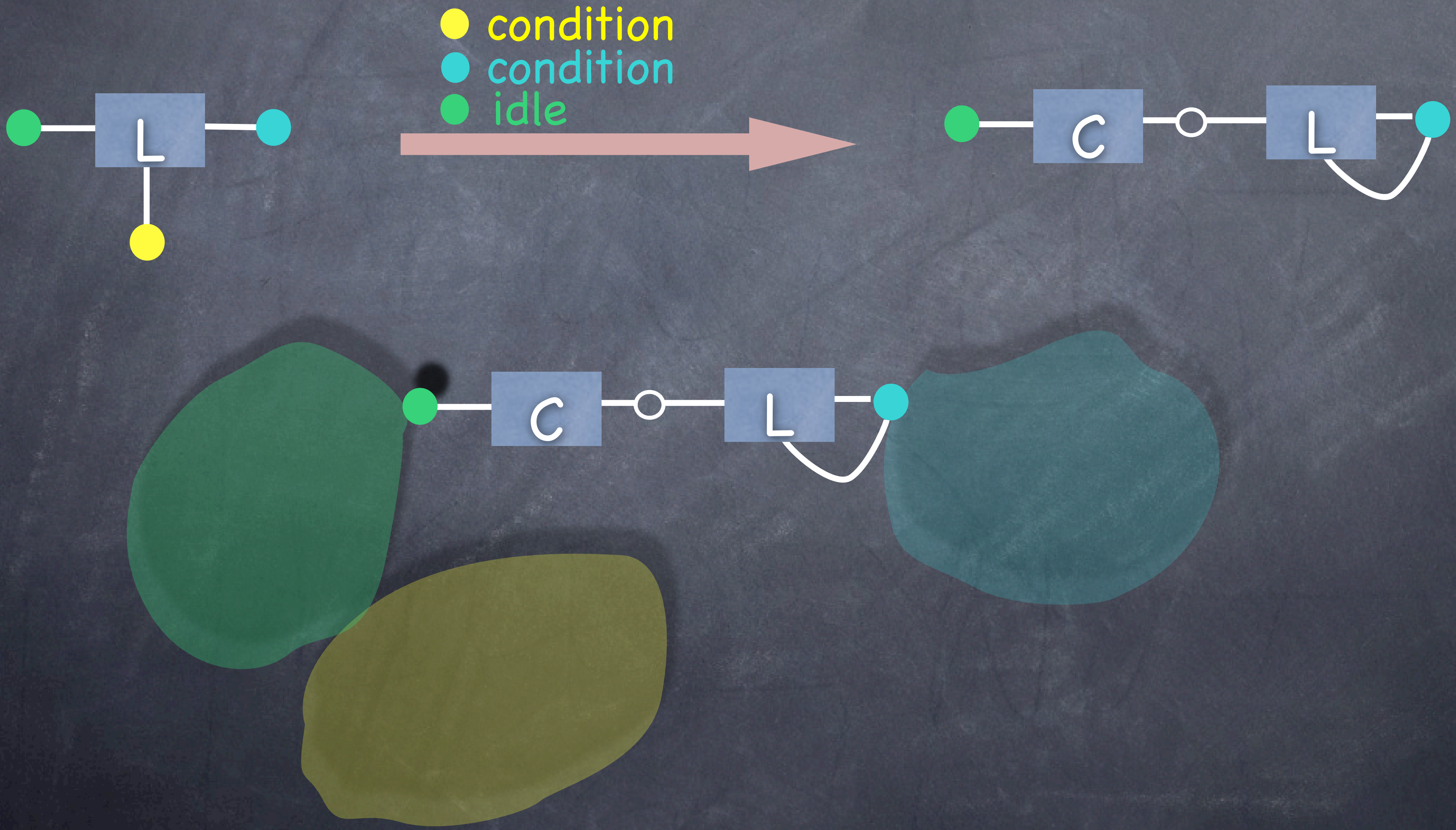


idle

condition

condition

# bMSHR: in a nutshell



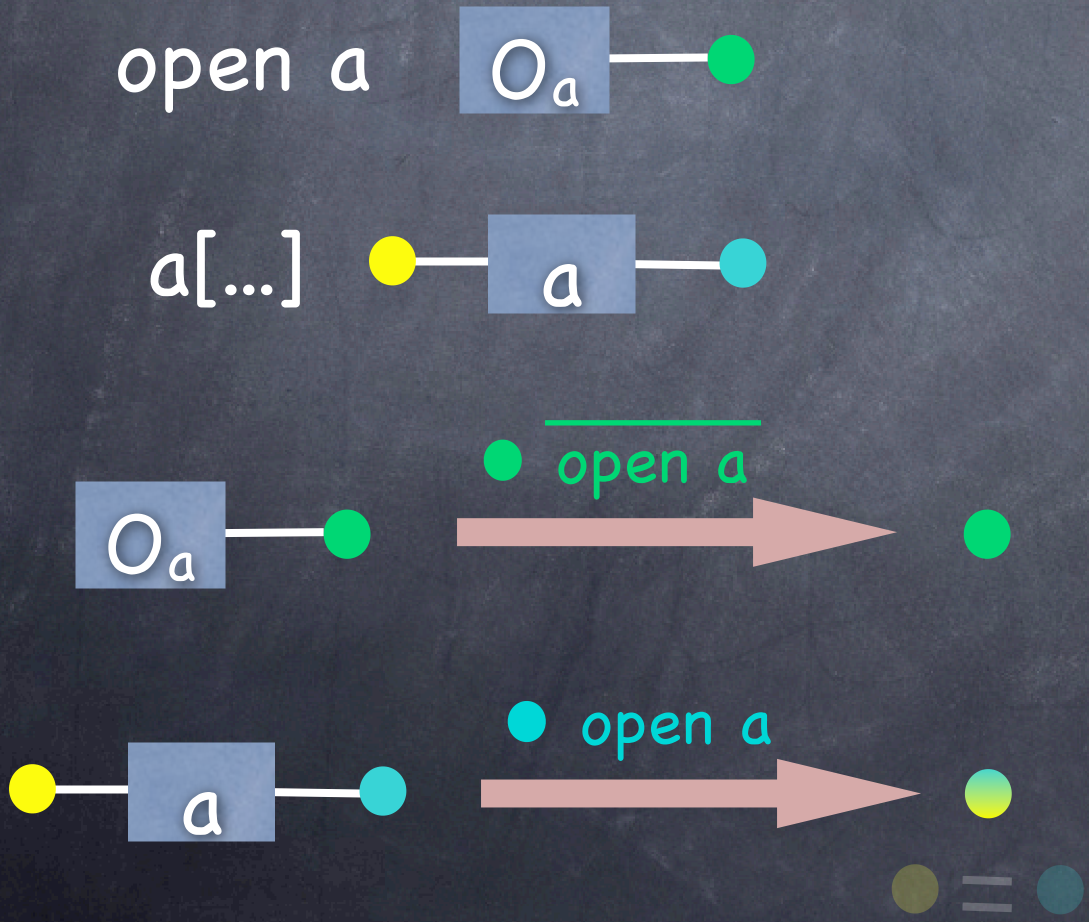
● condition
● condition
● idle

# SHR & mobility

“Milner” synchronisation
mobility [fmt01]

# Quest for mobility...

In Ambient: open a | a[...] → ...



open a $O_a$

a[...] a

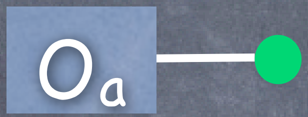
$O_a$ $\overline{\text{open } a}$

a open a

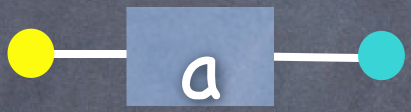
# Quest for mobility...

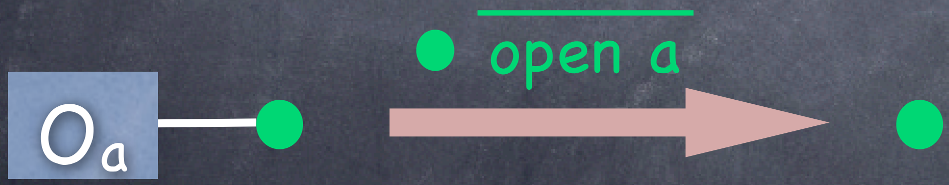
In Ambient: open a | a[...] → ...

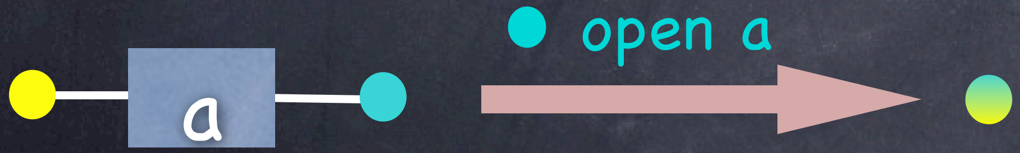
open a

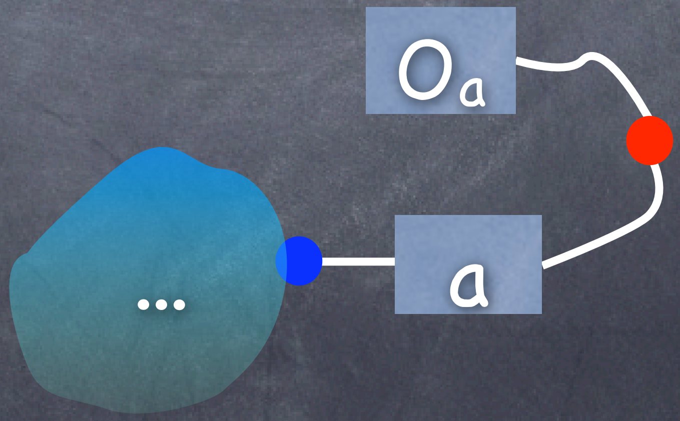


a[...]



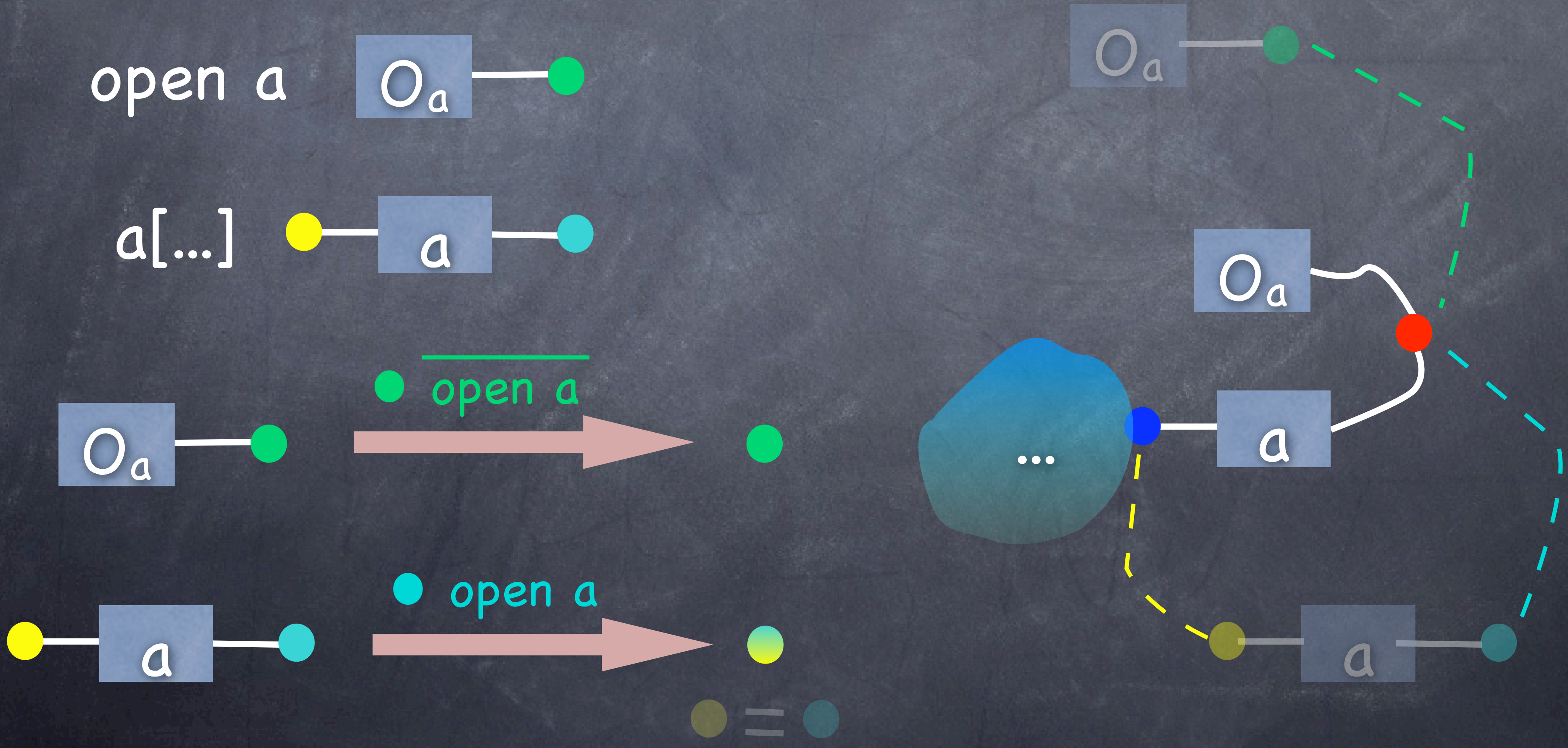






# Quest for mobility...

In Ambient: open a | a[...] → ...



open a $O_a$

a[...] a

$O_a$ $\overline{\text{open } a}$

a open a

$O_a$

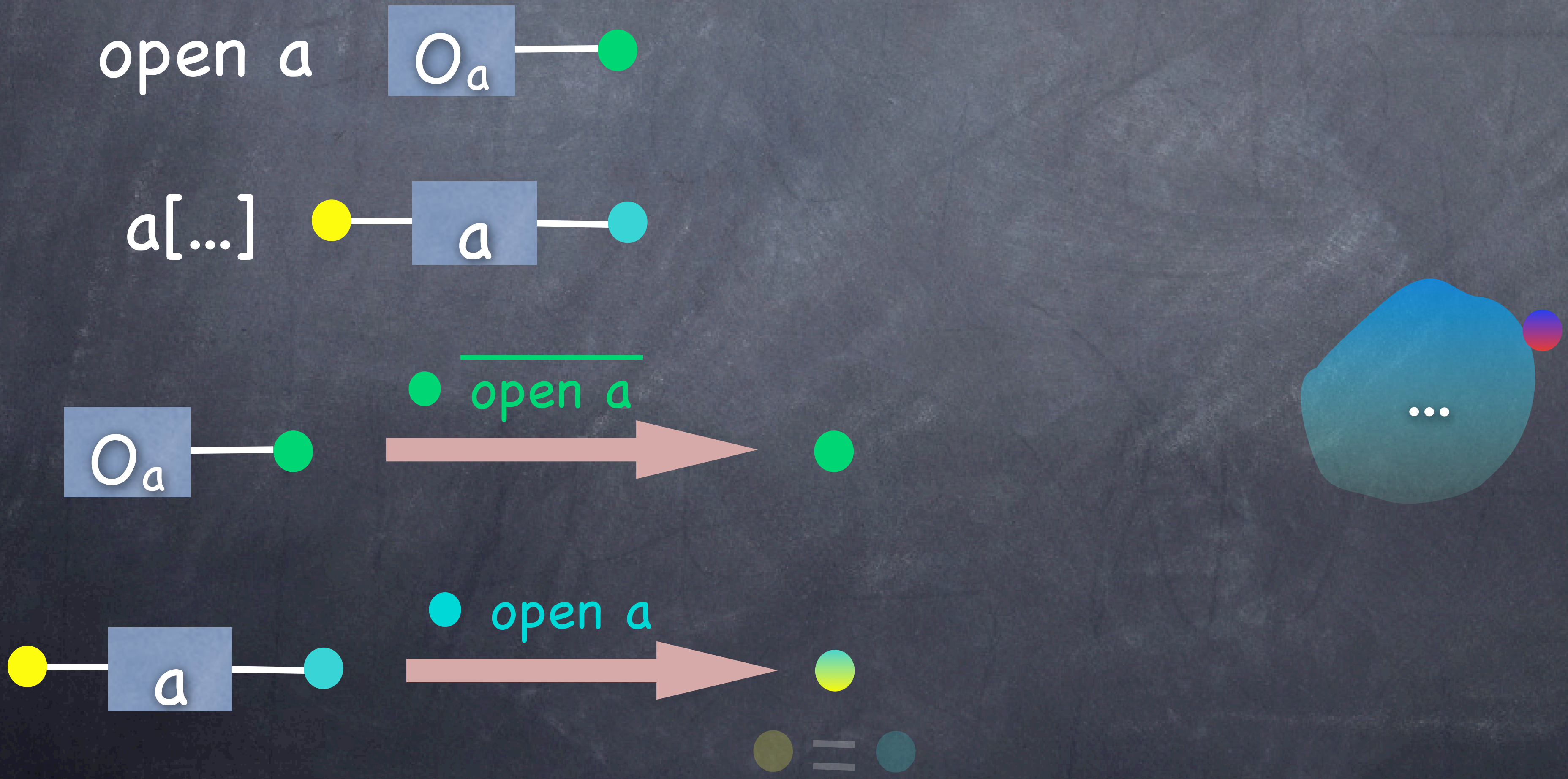
$O_a$

... a

a

# Quest for mobility...

In Ambient: open a | a[...] → ...

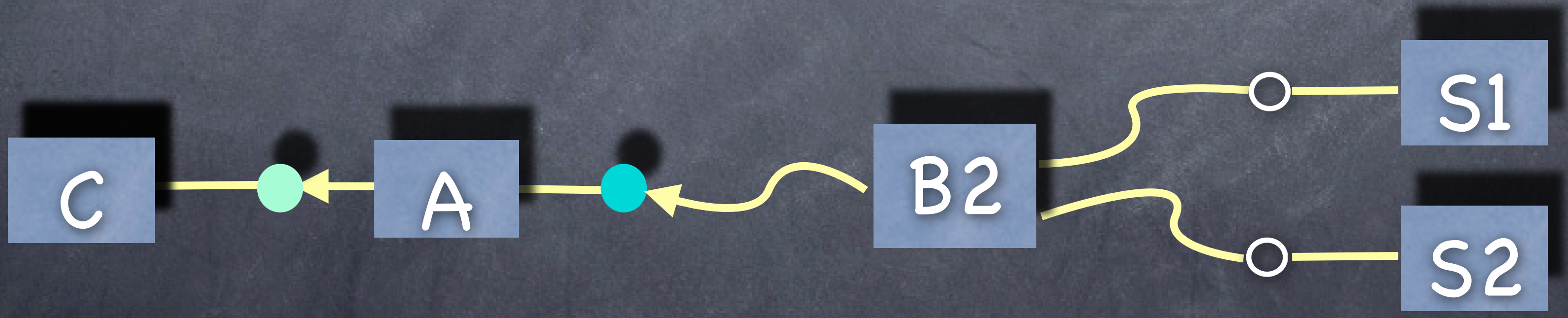


open a

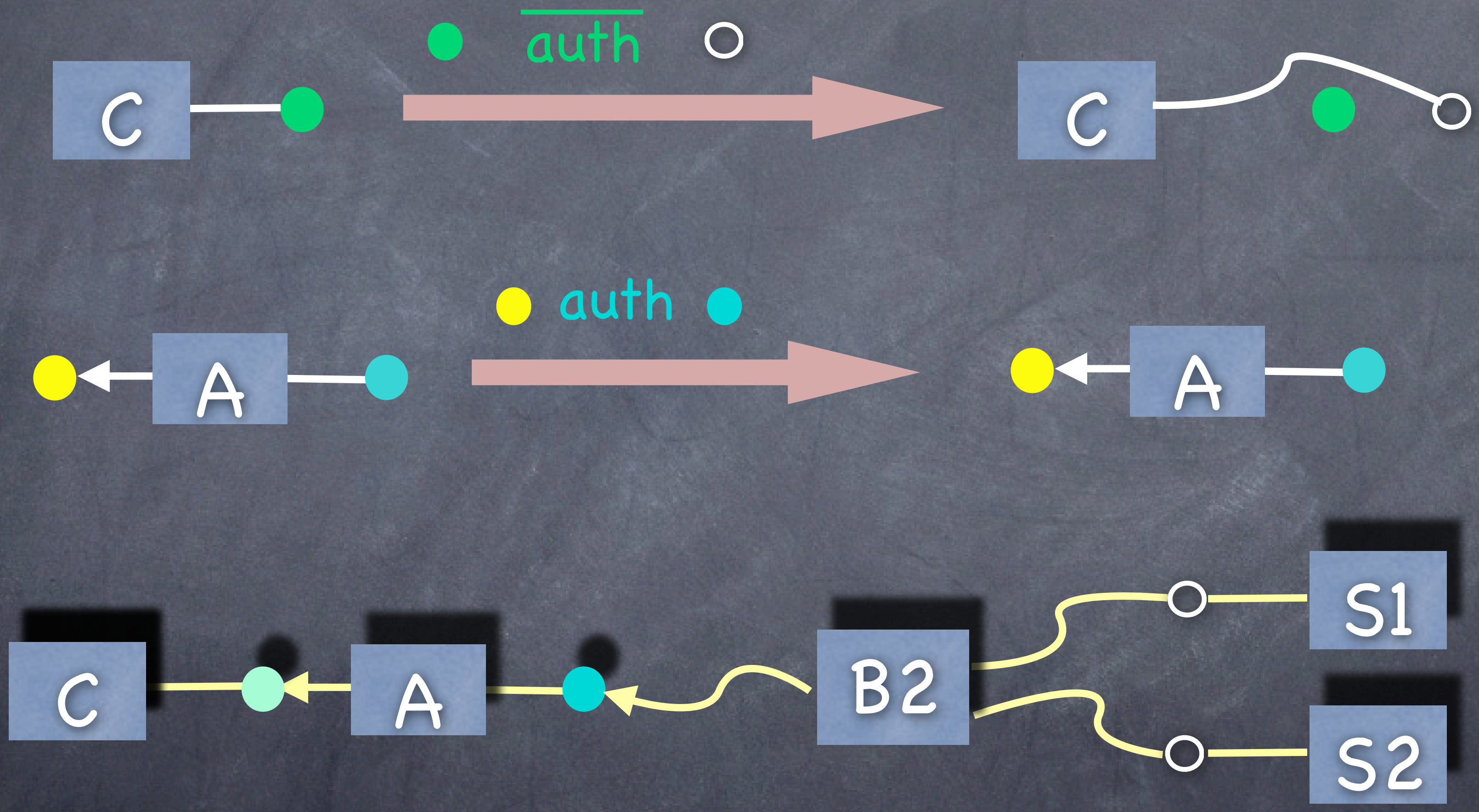
a[...]

# An example with mobility

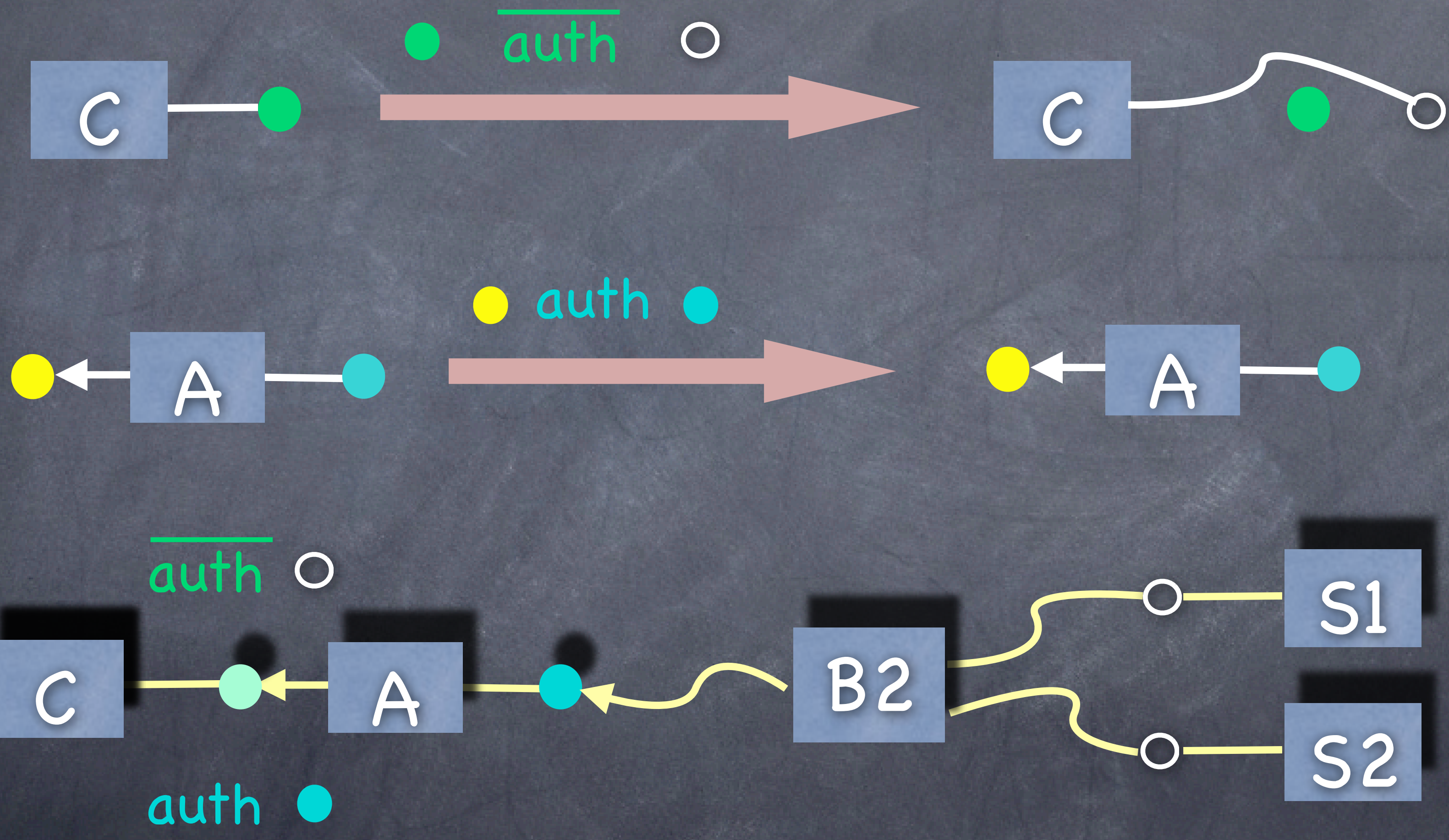
# An example with mobility



# An example with mobility

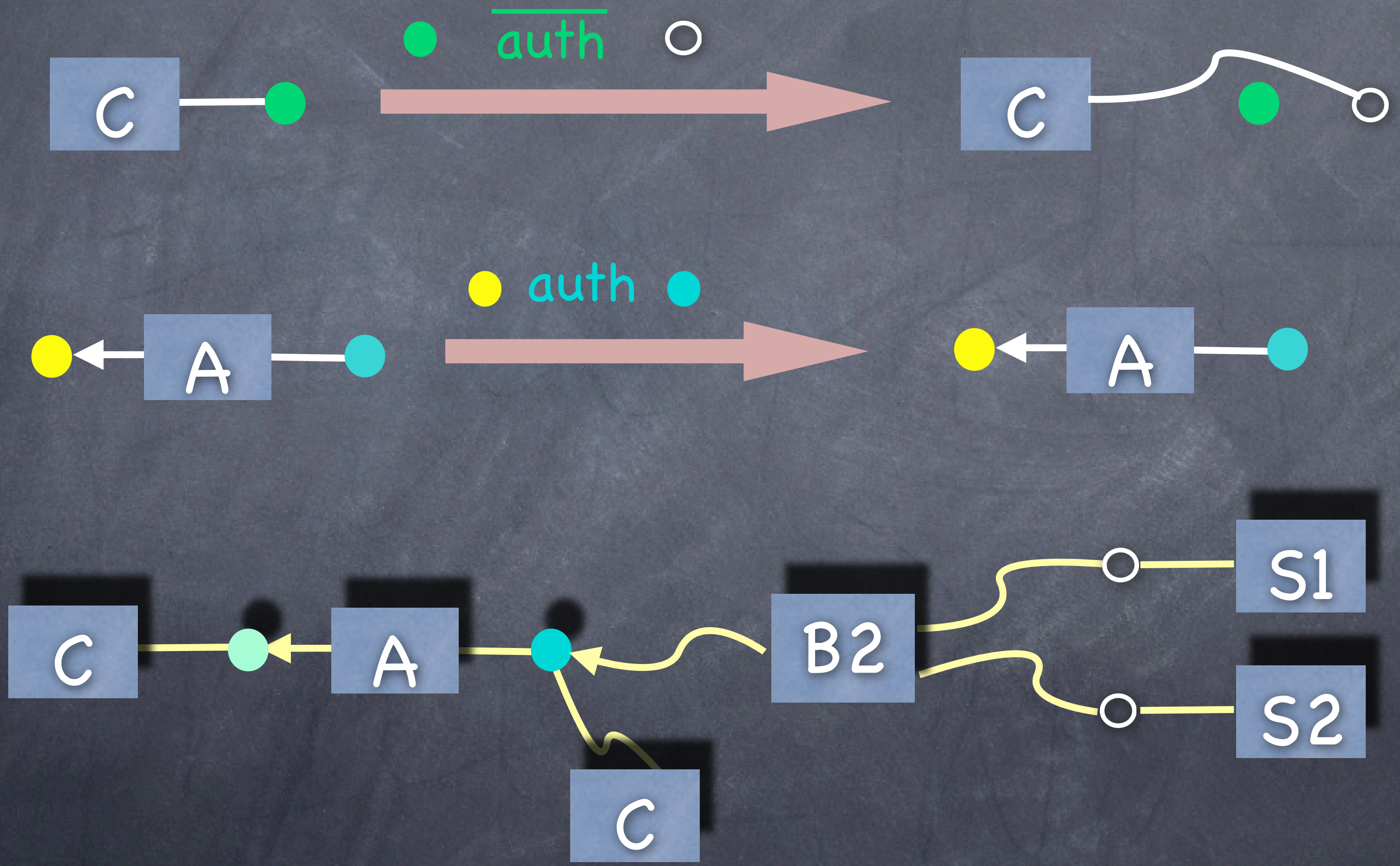


# An example with mobility



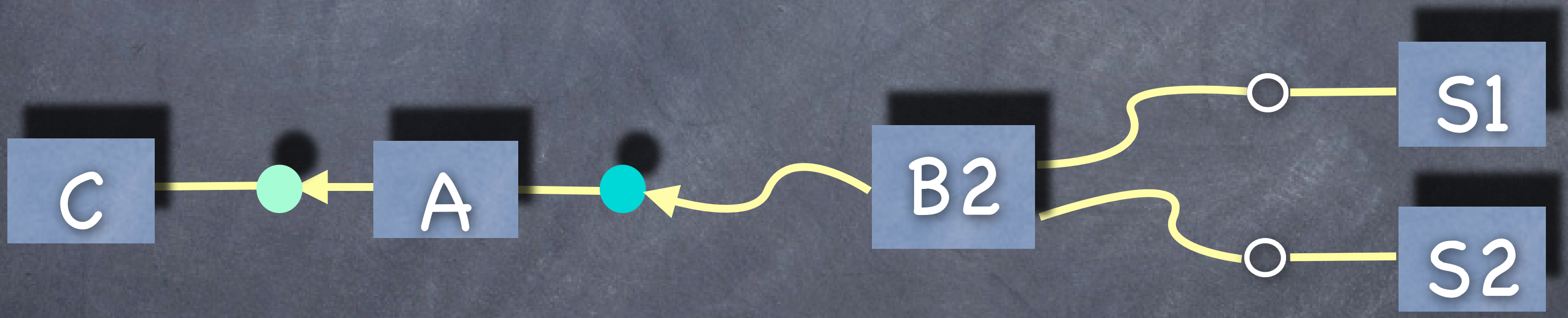
mgu( ● = ○ ) yields { ●/○ }

# An example with mobility

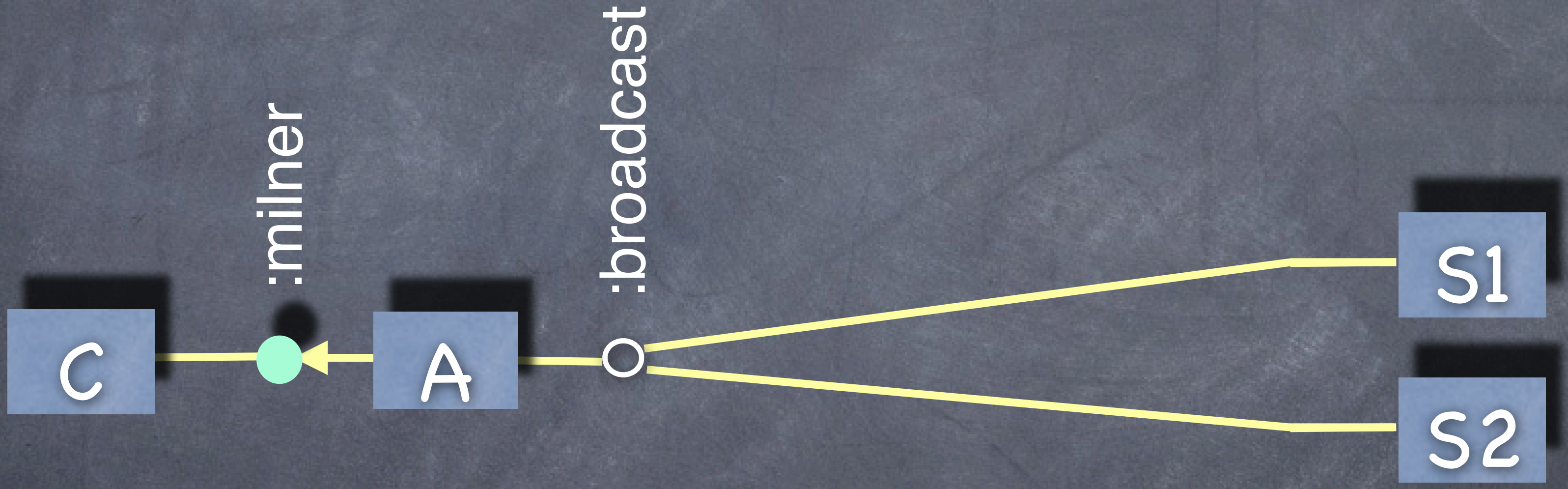


# Synchronisation algebras with mobility

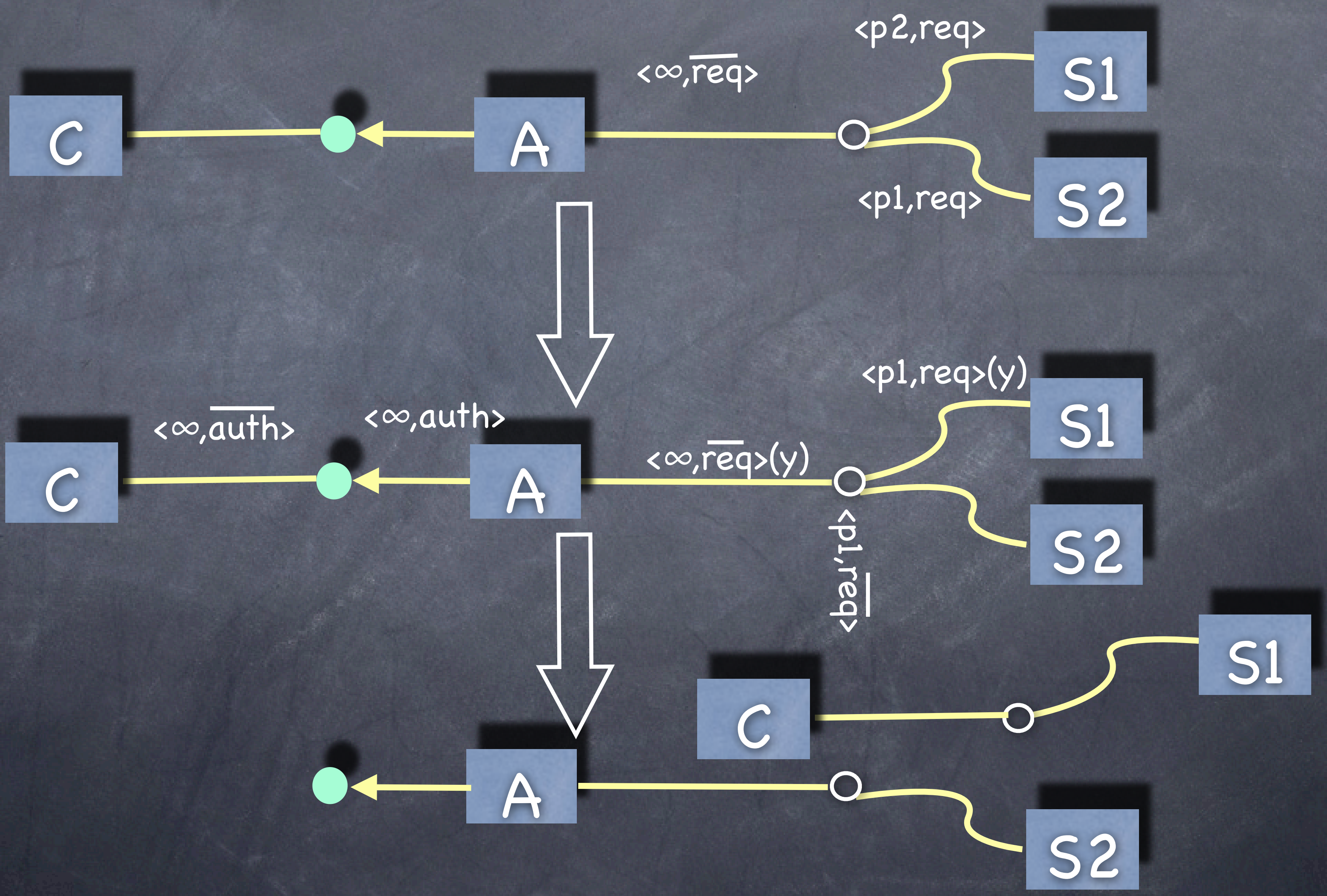
# Synchronisation algebras with mobility



# Synchronisation algebras with mobility



# SHReQ



C

A

$\langle\infty,\overline{\text{req}}\rangle$

<p2,req>

S1

S2

<p1,req>

$\langle\infty,\overline{\text{auth}}\rangle$

<∞,auth>

C

A

<p1,req>(y)

S1

$\langle\infty,\overline{\text{req}}\rangle(y)$

S2

$\langle p1,\overline{\text{req}}\rangle$

C

S1

A

S2

# Dealing with quality

# Watch...

...for more fun :)

# Service Oriented Architectural Design

with
R. Bruni, A. Lluch Lafuente, and U. Montanari
Dipartimento di Informatica
Universita' di Pisa

# Motivations

SEnSOria aims to develop an approach for engineering SOCs

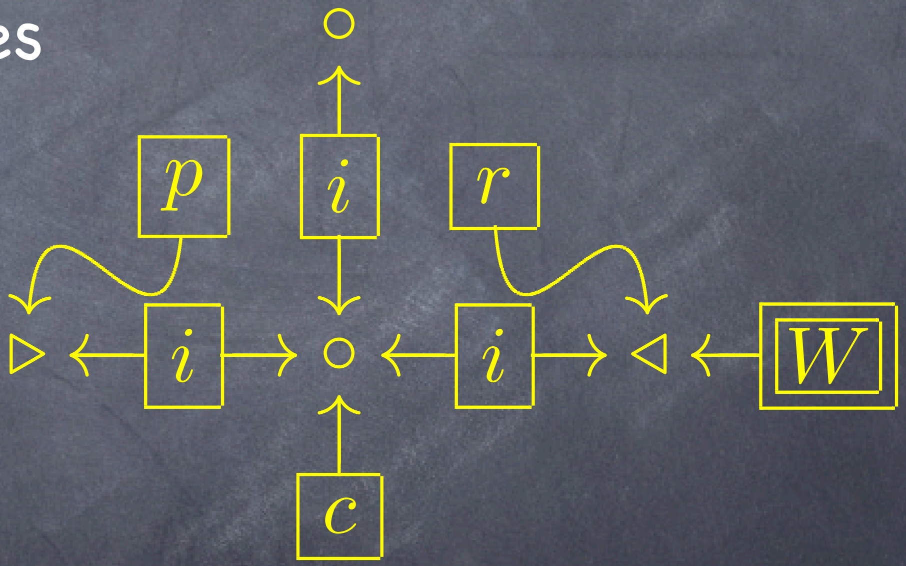
- Key issues of service-based architectures:
  - design
  - reconfiguration
- Styles for reusing existing design patterns
- Run-time changes (e.g., dynamic binding)
  - require reconfigurations of architectures
  - complement their static reconfigurations
  - driven by architectural information specified during design
- Often, architectural styles must be preserved or consistently changed

# ADR principles

- Architectures are modelled as suitable graphs
- Hierarchical architectural designs
  - style preserving rules (not original)
  - algebraic presentation (original)
- Reconfigurations defined over style proofs instead of actual architectures
  - exploits the algebraic presentation
  - straightforward definition of hierarchical and inductive reconfigurations (ordinary term rewriting and SOS)
  - only valid contexts considered (not all concrete designs)
  - matching is simpler during reconfigurations (design driven)

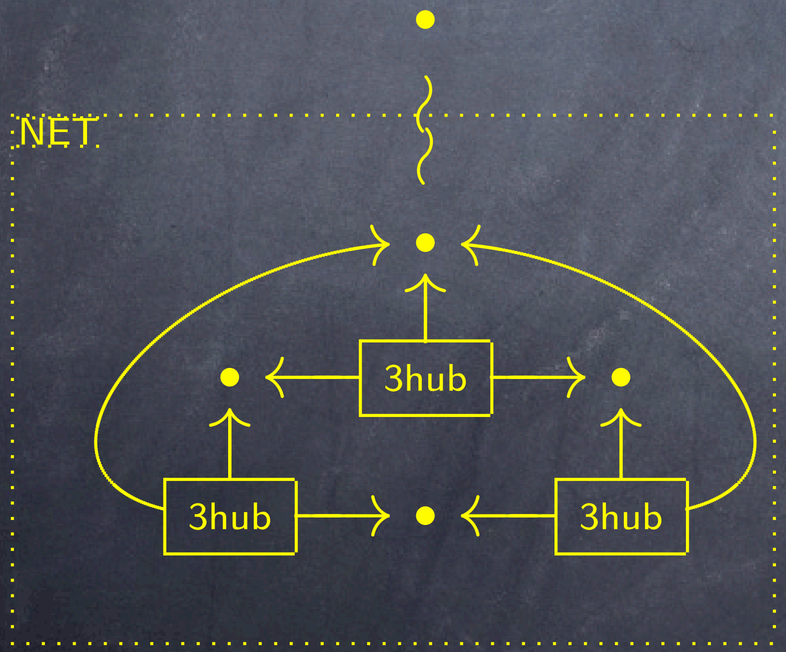
# ADR ingredients

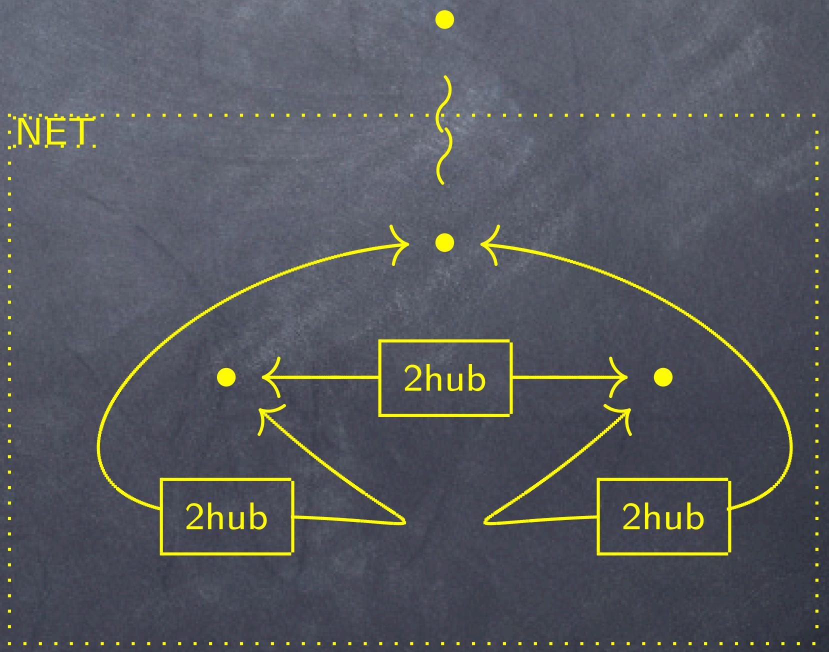
- Hypergraphs
  - edges model components: can be terminal and non-terminal edges
  - nodes model connecting ports
- Type-(hyper)graphs
- Productions
  - rules like L ::= R
  - specify how non-terminals should be replaced



# ADR by example

- A local networking architecture
- 2 styles where each network hub has degree of connectivity 2 or 3
- Connections between hubs are also driven by the style

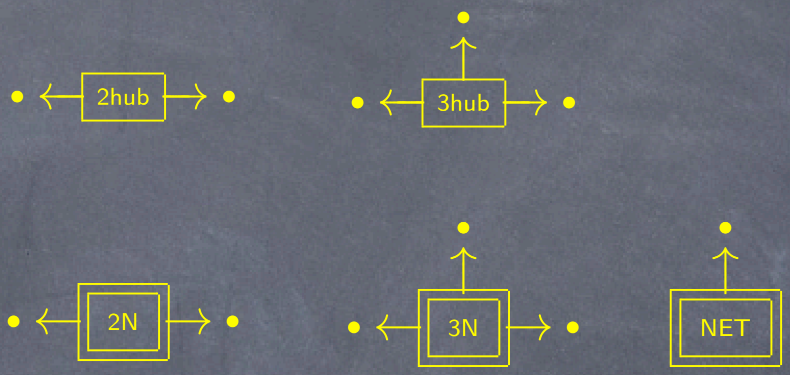




# Designs and productions

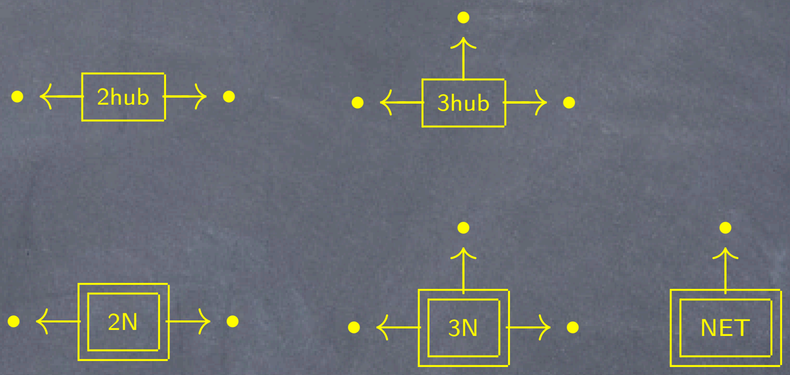
# Designs and productions

- Edges for the network example



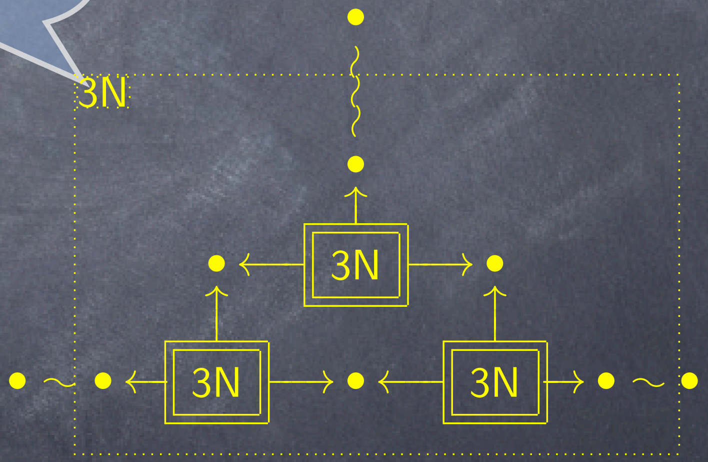
# Designs and productions

- Edges for the network example



- A design consists of
  - a lhs L which is a graph made of a single non-terminal edge
  - a rhs R graph possibly containing non-terminal edges
  - a map from the nodes of L to the nodes of R
- A production is a design where the occurrences of non-terminal are distinguished

represents the abstract class of the component



$3N ::= \texttt{link3}(3N, 3N, 3N)$

$\texttt{link3} : 3N \times 3N \times 3N \to 3N$

type of the production

# ADR metaphor

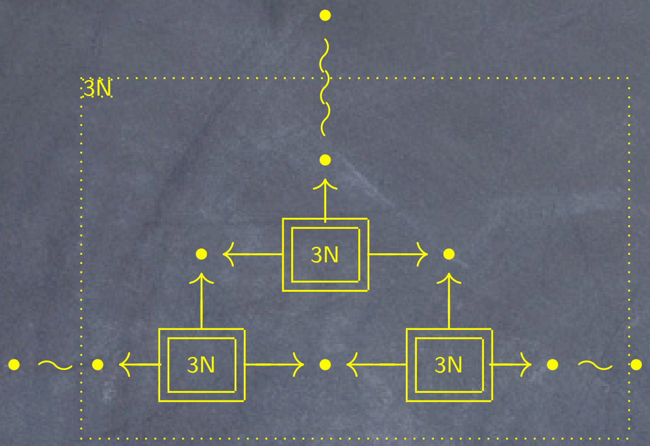
- A term of a grammar is an instance of a design
- Terms with variables are partial designs
- Replacing variables corresponds to refinement
- Replacing subterms with variables corresponds to abstraction
- Replacements are driven by term rewriting rules, namely reconfiguration rules t -> t'
  - style is preserved if t and t' have the same abstract class
  - otherwise styles change...in a consistent way

# Design rewritings

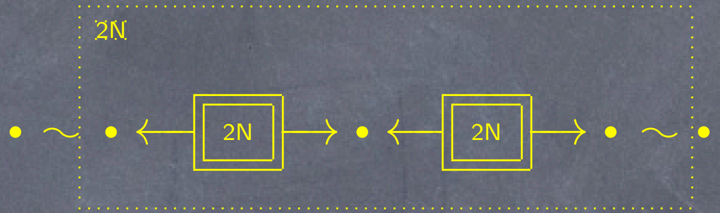
$$\texttt{link3to2}: \frac{x_1 \xrightarrow{\texttt{3to2}} x'_1 \qquad x_2 \xrightarrow{\texttt{3to2}} x'_2 \qquad x_3 \xrightarrow{\texttt{3to2}} x'_3}{\texttt{link3}(x_1, x_2, x_3) \xrightarrow{\texttt{3to2}} \texttt{link2}(\texttt{link2}(x'_2, x'_1), x'_3)}$$

# Design rewritings

$\texttt{link3} : \mathsf{3N} \times \mathsf{3N} \times \mathsf{3N} \to \mathsf{3N}$



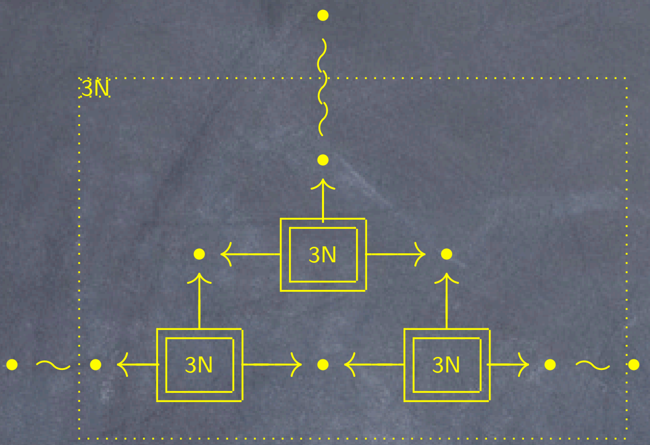
$\texttt{link2} : \mathsf{2N} \times \mathsf{2N} \to \mathsf{2N}$



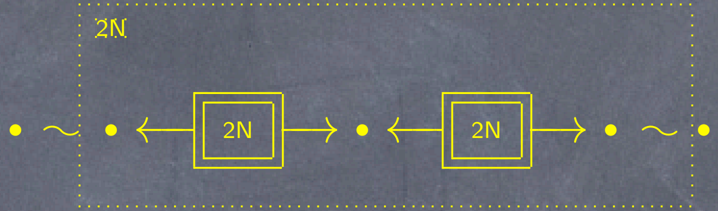
$$\texttt{link3to2} : \frac{x_1 \xrightarrow{\texttt{3to2}} x_1' \qquad x_2 \xrightarrow{\texttt{3to2}} x_2' \qquad x_3 \xrightarrow{\texttt{3to2}} x_3'}{\texttt{link3}(x_1, x_2, x_3) \xrightarrow{\texttt{3to2}} \texttt{link2}(\texttt{link2}(x_2', x_1'), x_3')}$$

# Design rewritings

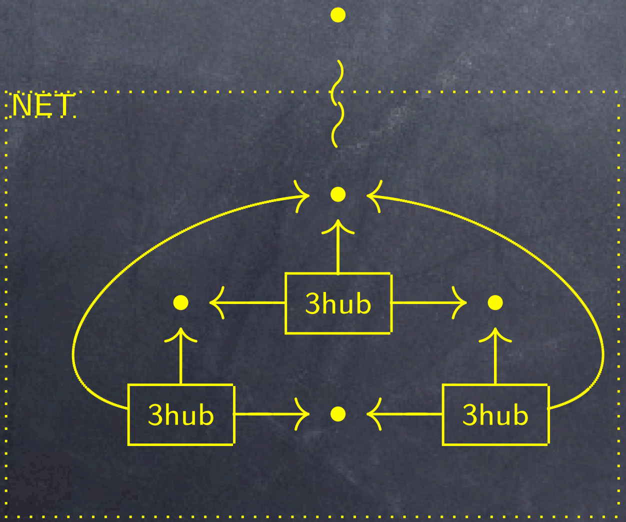
$\mathtt{link3} : 3\mathsf{N} \times 3\mathsf{N} \times 3\mathsf{N} \to 3\mathsf{N}$

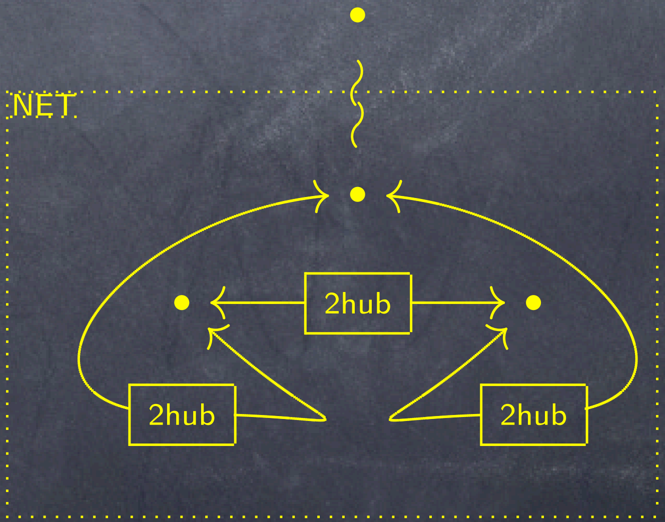


$\mathtt{link2} : 2\mathsf{N} \times 2\mathsf{N} \to 2\mathsf{N}$



$$\mathtt{link3to2} : \frac{x_1 \xrightarrow{\mathtt{3to2}} x_1' \qquad x_2 \xrightarrow{\mathtt{3to2}} x_2' \qquad x_3 \xrightarrow{\mathtt{3to2}} x_3'}{\mathtt{link3}(x_1, x_2, x_3) \xrightarrow{\mathtt{3to2}} \mathtt{link2}(\mathtt{link2}(x_2', x_1'), x_3')}$$


NET
3hub
3hub
3hub


NET
2hub
2hub
2hub