Open Klaim: *Linguistic extensions for Hierarchical Structuring*

Rocco De Nicola
Dip. Sistemi e Informatica
Università di Firenze

denicola@dsi.unifi.it

Outline of this lecture

- Resumè on Klaim
- Klava
- Two programming example
- Open Nets
- Open Klaim
- Syntax and semantics
- An example

Open nets...

- These are highly reconfigurable
  - new nodes can get connected
  - existing nodes can disconnect
- Connection and disconnection can be
  - temporary
  - unexpected
- New connections can be established *on-the-fly*
- Current Scenarios: *Peer-to-Peer, Ad-hoc networking, …*

Connection Mode

- Tethered Mode
  - WAN connectivity is available
  - Information can be accessed from any point at any time
- Disconnected Mode
  - WAN connectivity is available
  - Users can work offline
  - When online a user works like in Tethered Mode
  - When a user goes online reconciliation and/or notification may be needed.
- Untethered Mode
  - WAN connectivity is unavailable
  - Communication:
    - is enabled by wireless devices
    - is limited to those devices that are in communication range
  - Users establish communication and share information

A new set of operators

- Designed for expressing dynamic evolution of open nets (BDP ACM SAC02)
- The proposed constructs are largely independent of a specific programming language
- Here they are put in concrete form by focusing on their integration within the Klaim framework.

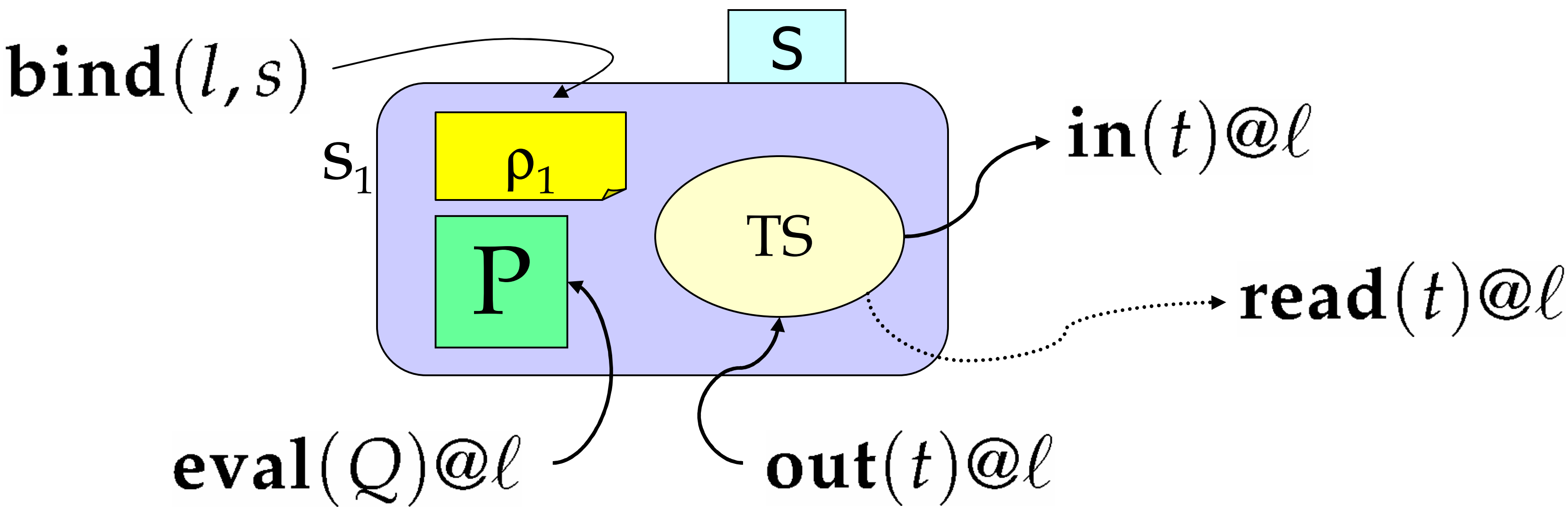
From Klaim to Open Klaim

Klaim is enriched

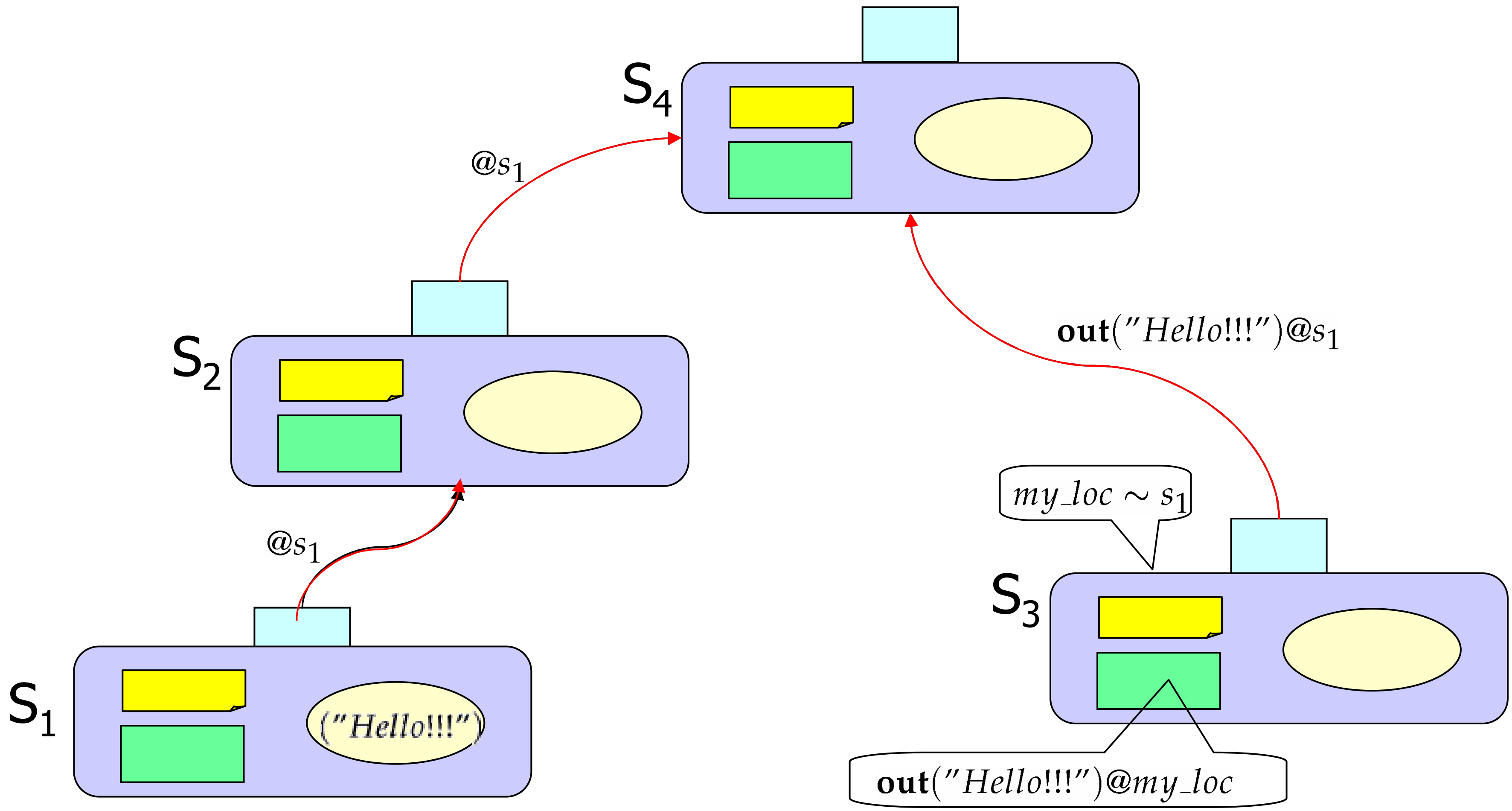
- with mechanisms to dynamically update allocation environments and handle node connectivity,
- with a new category of processes, called node coordinators or guardians that execute privileged operations to establish/accept new connections or remove existing ones

OpenKlaim nodes and operations

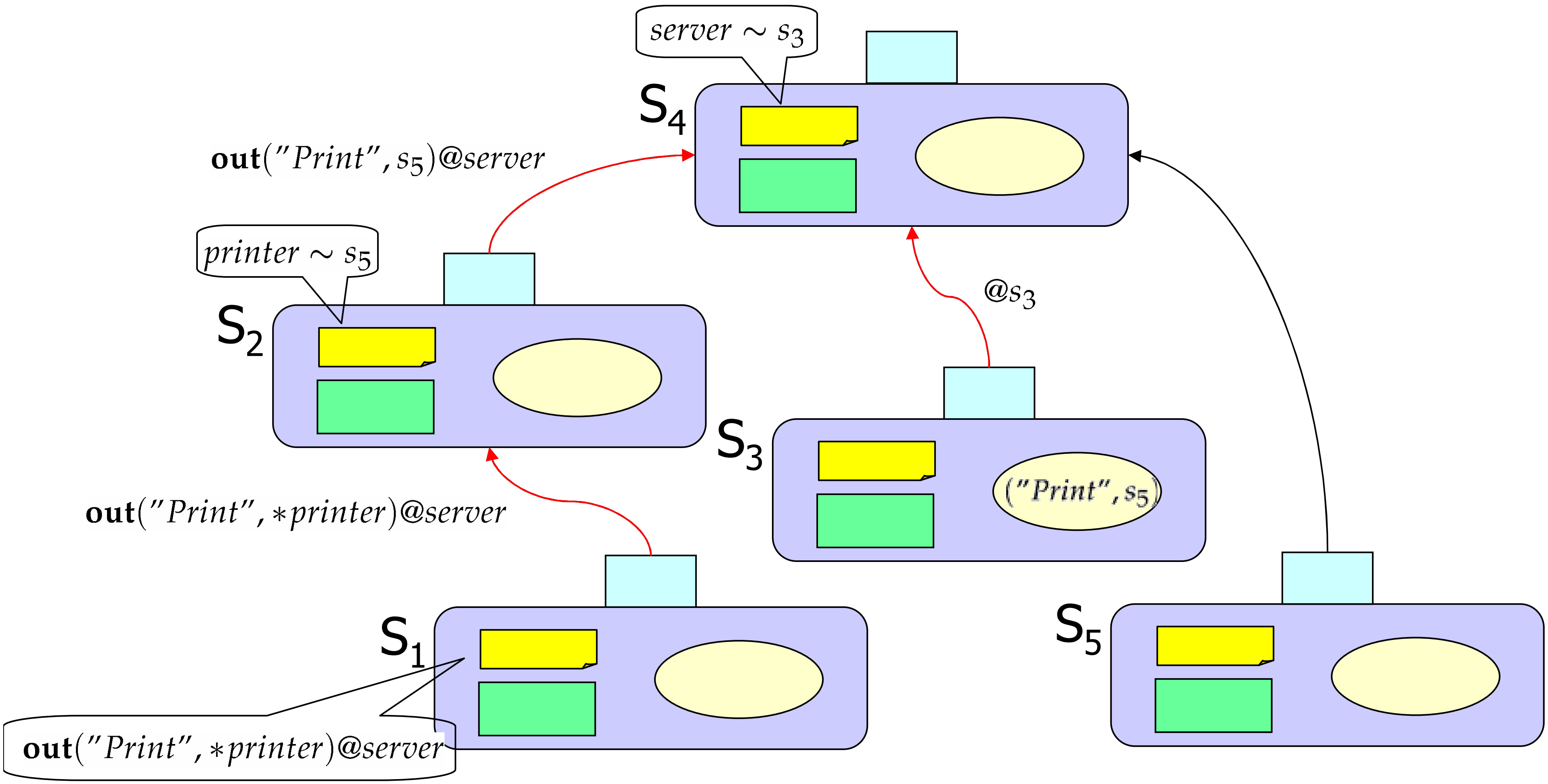
- Name (phys. loc.)
- Tuple space
- Gateways
- Enriched Processes
- Alloc. Env.



Open Klaim Nets



Klaim Nets


server ~ s_3
S_4
out("Print", s_5)@server
printer ~ s_5
S_2
@s_3
S_3
("Print", s_5)
out("Print", ∗printer)@server
S_1
S_5
out("Print", ∗printer)@server

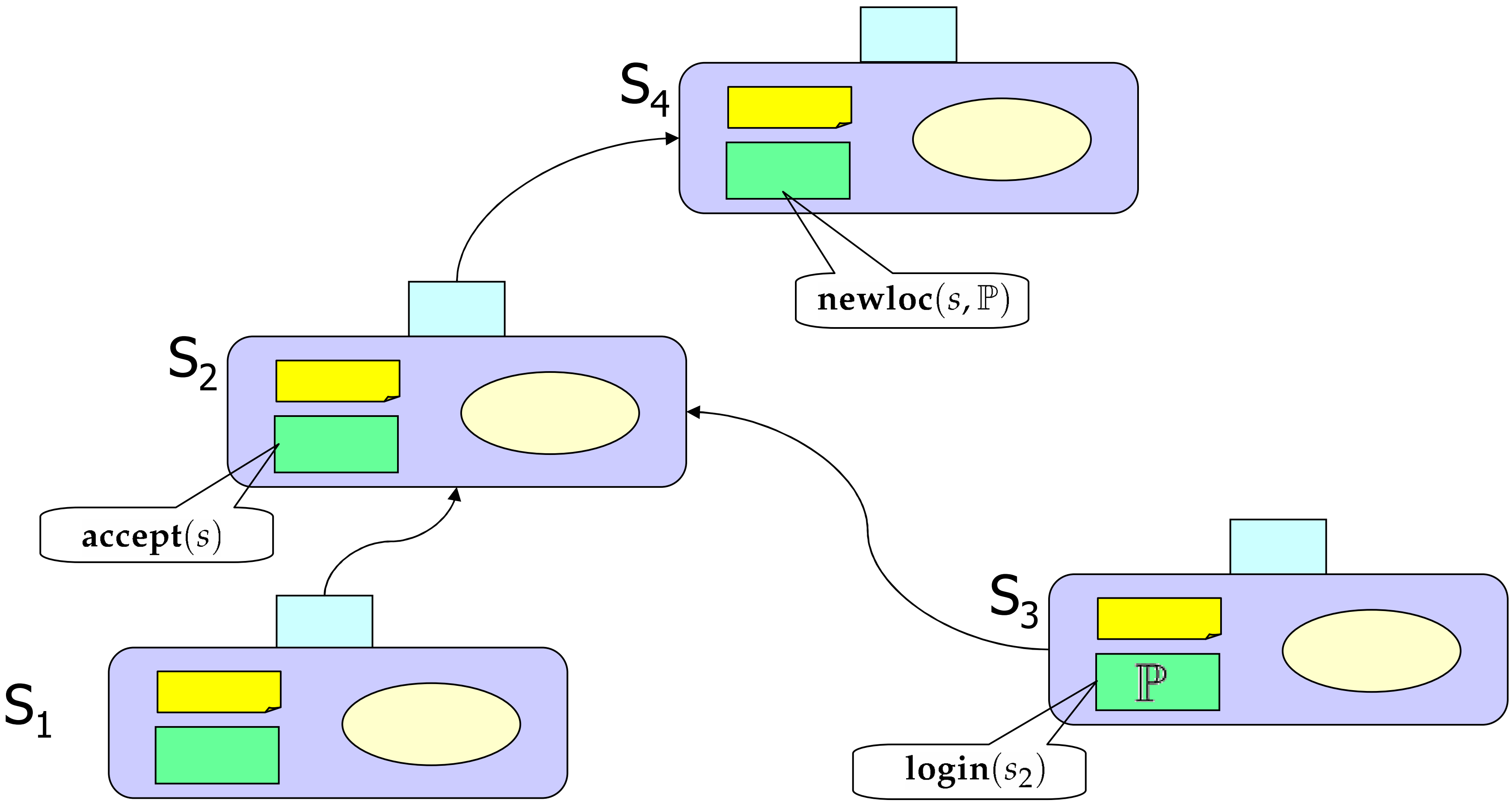
The coordinators language

- New class of processes ($\mathbb{P}$) that:
  - can perform new *special* actions:

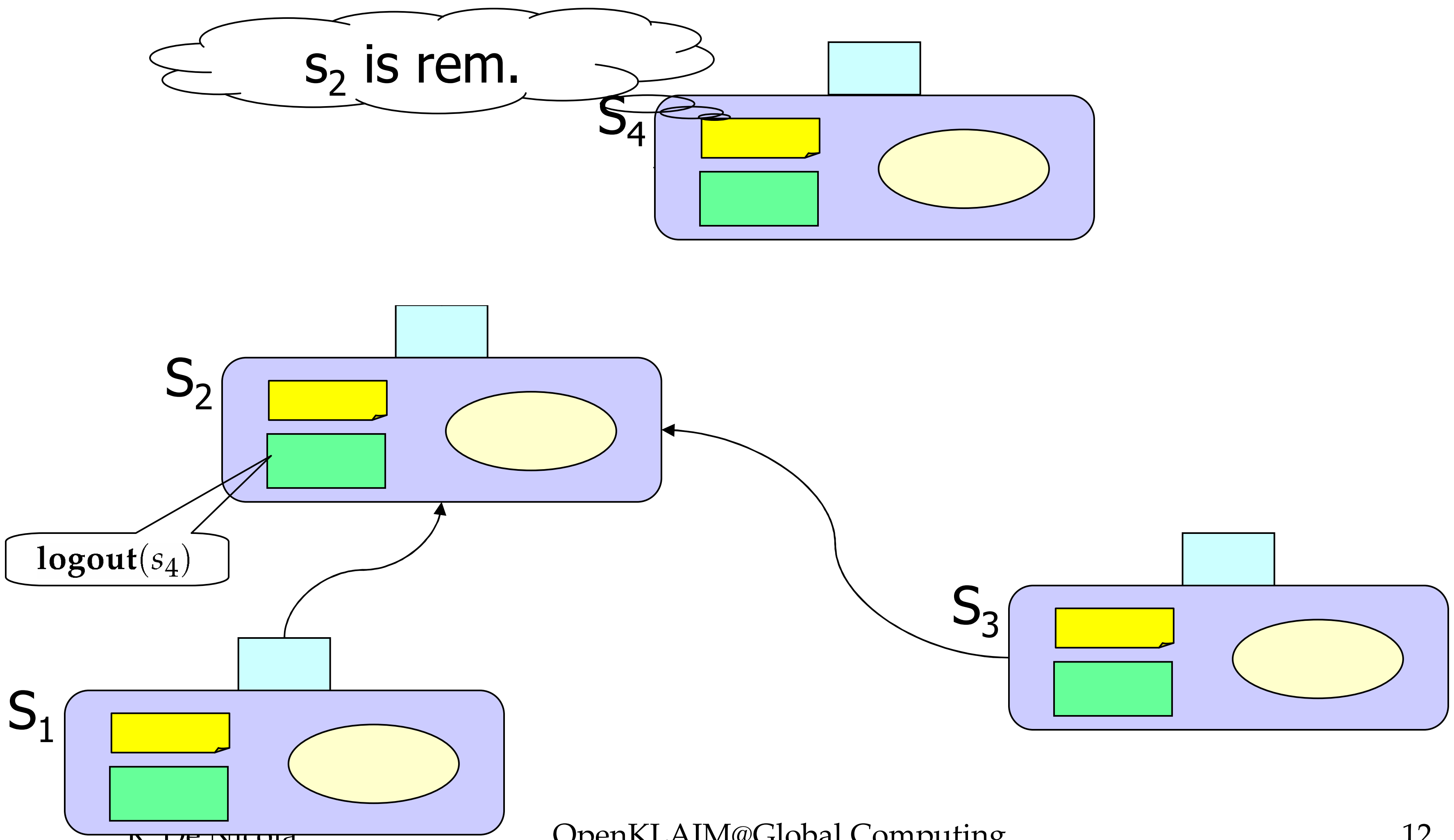
    $\mathbf{newloc}(s, \mathbb{P})$ $\quad$ $\mathbf{login}(\ell)$

    $\mathbf{logout}(\ell)$ $\quad$ $\mathbf{accept}(s)$
  - do not move
  - model the *network-interface* of operating system

Dynamic evolution.... (1)



Dynamic evolution.... (2)



Open Klaim Syntax

Six syntactic categories:

- Tuples
- Actions
- Privileged actions
- Processes (execute actions)
- Coordinators (execute actions and privileged action)
- Nets (evolve)

Actions and Privileged Actions

$$
\begin{array}{llll}
a & ::= & & \text{Actions} \\
 & & \mathbf{out}(t)@\ell & \textit{output} \\
 & \mid & \mathbf{in}(T)@\ell & \textit{input} \\
 & \mid & \mathbf{read}(T)@\ell & \textit{read} \\
 & \mid & \mathbf{eval}(P)@\ell & \textit{migration} \\
 & \mid & \mathbf{bind}(u,l) & \textit{bind} \\
pa & ::= & & \text{Privileged Actions} \\
 & & a & \textit{(standard) action} \\
 & \mid & \mathbf{newloc}(u,\mathbb{P}) & \textit{creation} \\
 & \mid & \mathbf{login}(\ell) & \textit{login} \\
 & \mid & \mathbf{logout}(\ell) & \textit{logout} \\
 & \mid & \mathbf{accept}(u) & \textit{accept}
\end{array}
$$

Super Processes and Nets

f	$+=$		Tuple Fields
		$*l$	*Dereferentiation*
$\mathbb{P}$	$::=$		Node Coordinators
		P	*(standard) process*
	$\mid$	$pa.\mathbb{P}$	*action prefixing*
	$\mid$	$\mathbb{P}_1 \mid \mathbb{P}_2$	*parallel composition*
	$\mid$	$\mathbb{A}$	*NodeCoordinator invocation*
N	$::=$		Nets
		$\mathbf{0}$	*empty net*
	$\mid$	$l ::_{\rho}^{S} \mathbb{P}$	*single node*
	$\mid$	$l :: \langle et \rangle$	*located tuple*
	$\mid$	$N_1 \parallel N_2$	*net composition*

Transition Labels…

- indicate the *gateway* used to perform the operation *and* reflect:
  - The information transmitted over the net:

$$\xrightarrow[s]{\mathbf{i}(s_1, t, s_2)}$$

  - The resources available and the involved processes:

$$\xrightarrow[s]{et@s_1} \qquad \xrightarrow[s]{s_1 ::_{\rho}^{S} P}$$

Operational semantics

We introduce

1. The action that signal presence of tuples and of structured processes in the net and the possibility of interaction by changing gateway.
2. Processes willingness to execute in/out operation (these actions are made possible only if there are the appropriate gateways)
3. The coordinators actions

Nodes, tuples and gateways

$$l :: \langle et \rangle \xrightarrow[l]{\langle et \rangle @ l} \mathbf{0} \quad \text{(Tuple)}$$

$$l ::_{\rho}^{S} \mathbb{P} \xrightarrow[l]{l ::_{\rho}^{S} \mathbb{P}} \mathbf{0} \quad \text{(Node)}$$

$$\frac{N_1 \xrightarrow[l_1]{\lambda} N_1' \quad N_2 \xrightarrow[l_2]{l_2 ::_{\rho}^{\{l_1\} \cup S} \mathbb{P}} N_2'}{N_1 \parallel N_2 \xrightarrow[l_2]{\lambda\{\rho\}} N_1' \parallel N_2' \parallel l_2 ::_{\rho}^{\{l_1\} \cup S} \mathbb{P}} \quad \text{(Env)}$$

Process Actions/Intentions

$$l ::_{\rho}^{S} \mathbf{bind}(u, l_1).\mathbb{P} \xrightarrow[l]{\mathbf{b}(l,u,l_1)} l ::_{\rho[l_1/u]}^{S} \mathbb{P}$$

$$\textit{if } \rho(u) \textit{ is undefined} \quad \text{(Bind)}$$

$$l ::_{\rho}^{S} \mathbf{out}(t)@\ell.\mathbb{P} \xrightarrow[l]{\mathbf{o}(l,[\![\, t \,]\!]_{\rho},\rho(\ell))} l ::_{\rho}^{S} \mathbb{P} \quad \text{(Out)}$$

$$l ::_{\rho}^{S} \mathbf{in}(T)@\ell.\mathbb{P} \xrightarrow[l]{\mathbf{i}(l,[\![\, T \,]\!]_{\rho},\rho(\ell))} l ::_{\rho}^{S} \mathbb{P} \quad \text{(In)}$$

Actions by the coordinator

$$\frac{l_2 \notin L}{L \vdash l_1 ::_\rho^S \mathbf{newloc}(u, \mathbb{P}).\mathbb{P}' \xrightarrow{\mathbf{n}(l_1,\mathbb{P},l_2)} L \cup \{l_2\} \vdash l_1 ::_\rho^S \mathbb{P}'[l_2/u]}$$

$$l_1 ::_\rho^S \mathbf{login}(l_2).\mathbb{P} \xrightarrow{\mathbf{lin}(l_1,-,l_2)} l_1 ::_\rho^S \mathbb{P}$$

$$l_1 ::_\rho^S \mathbf{logout}(l_2).\mathbb{P} \xrightarrow{\mathbf{lout}(l_1,-,l_2)} l_1 ::_\rho^S \mathbb{P}$$

$$l_1 ::_\rho^S \mathbf{accept}(u).\mathbb{P} \xrightarrow{\mathbf{acc}(l_1,-,l_2)} l_1 ::_\rho^{S \cup \{l_2\}} \mathbb{P}[l_2/u]$$

Open Klaim – Operational Semantics

- The intentions of the processes/ coordinators are taken into account to describe the overall evolution of Nets.
- We have a net transition in correspondence of each kind of action performed by the process or the coordinator.
- Communication action take place only if there is a commom gateway.

Open Klaim – Op. Sem.

$$\frac{N_1 \xrightarrow[l]{\mathbf{b}(l_2,u,l_1)} N_2}{N_1 \rightarrowtail N_2}$$

$$\frac{N_1 \xrightarrow[l]{\mathbf{o}(l_1,et,l_2)} N_1' \quad N_1' \xrightarrow[l]{l_2::^S_\rho P} N_2}{N_1 \rightarrowtail N_2 \parallel l_2::^S_\rho \langle et \rangle | P}$$

$$match([\![T]\!]_\rho, et) = \sigma \qquad and$$

$$\frac{N_1 \xrightarrow[l]{\langle et \rangle @ l_2} N_1' \quad N_1' \xrightarrow[l]{\mathbf{i}(l_1,[\![T]\!]_\rho,l_2)} N_2}{N_1 \rightarrowtail N_2\sigma}$$

$$\frac{N_1 \xrightarrow{\mathbf{n}(l_1,\mathbb{P},l_2)} N_2}{N_1 \succ\!\!\longrightarrow N_2 \parallel l_2::^{\emptyset}_{[l_2/\mathtt{self}]} \mathbb{P}}$$

$$\frac{N_1 \xrightarrow{\mathbf{lin}(l_1,-,l_2)} N_1' \quad N_1' \xrightarrow{\mathbf{acc}(l_2,-,l_1)} N_2}{N_1 \succ\!\!\longrightarrow N_2}$$

$$\frac{N_1 \xrightarrow{\mathbf{lout}(l_1,-,l_2)} N_1' \quad N_1' \xrightarrow[l_2]{l_2::^{\{l_1\}\cup S}_{\rho} \mathbb{P}} N_2 \quad \rho' = \rho \setminus l_1}{N_1 \succ\!\!\longrightarrow N_2 \parallel l_2::^{S}_{\rho'} \mathbb{P}}$$

An example: a chat system

- The chat system is made of:
  - a server that
    - dispatches messages
    - accepts connections
  - and clients connected to it
    - logical localities are used for *nicknames*

Basic functionality...

$\textbf{subscribe}(s, l)[\mathbb{P}_1, \mathbb{P}_2] \triangleq$
$\quad \mathbf{login}(s).$
$\quad \mathbf{out}(\text{“register”}, l)@s.$
$\quad \mathbf{in}(l, !ok)@s.$
$\quad$ **if** ok **then**
$\qquad \mathbb{P}_1$
$\quad$ **else**
$\qquad \mathbf{logout}(s).\mathbb{P}_2$
$\quad$ **endif**

$\textbf{register}(s, l)[\mathbb{P}_1, \mathbb{P}_2] \triangleq$
$\quad \mathbf{accept}(s).$
$\quad \mathbf{in}(\text{“register”}, !l)@\mathtt{self}.$
$\quad$ **if** l not already registered **then**
$\qquad \mathbf{out}(l, \textbf{true})@\mathtt{self}.$
$\qquad \mathbf{bind}(l, s).\mathbb{P}_1$
$\quad$ **else**
$\qquad \mathbf{out}(l, \textbf{false})@\mathtt{self}.\mathbb{P}_2$
$\quad$ **endif**

Send & receive messages...

$\mathit{ReceiveMessages}() \stackrel{\mathit{def}}{=}$
while true do
$\mathbf{in}(!\mathit{msg}, !\mathit{from})@\mathtt{self}.$
print the message *msg* on the screen
enddo

$\mathit{SendMessage}(\mathit{server}, \mathit{nickname}) \stackrel{\mathit{def}}{=}$
while true do
$\mathbf{read}(\text{“connected”}, \textbf{true})@\mathtt{self}.$
input the message *msg*
$\mathbf{out}(\text{“message”}, \mathit{msg}, \mathit{nickname})@\mathit{server}$
enddo

$\mathit{BroadcastMessages}() \stackrel{\mathit{def}}{=}$
while true do
$\mathbf{in}(\text{“message”}, !\mathit{message}, !\mathit{from})@\mathtt{self}.$
for every l in the list of clients
$\mathbf{out}(\mathit{message}, \mathit{from})@l$
enddo

Possible extensions...

- Unidirectional connections
  - For modelling wireless-network
- QoS primitives
  - Work in progress (De Nicola, Ferrari, Montanari, Pugliese, Tuosto)
- Routing functions
  - To drive the selection of the path

Klaim site

http://music.dsi.unifi.it

- A few papers
- Current Implementation:
  - X-Klaim has the new primitives