A graphical calculus for Klaim nets

Joint work

Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari,

Rosario Pugliese, *Emilio Tuosto*

{denicola,pugliese}@dsi.unifi.it, {giangi,ugo,etuosto}@di.unipi.it .

Dipartimento di Sistemi e Informatica[1]

Univerità di Firenze – Italy

Dipartimento di Informatica[2]

Univerità di Pisa – Italy

Plan of the talk

Plan of the talk

- Motivations

Plan of the talk

- Motivations
- Klaim overview

Plan of the talk

- Motivations
- Klaim overview
- Hypergraphs overview

Plan of the talk

- Motivations
- Klaim overview
- Hypergraphs overview
- An example

Motivations

Motivations: WAN Programming

Motivations: WAN Programming

- Client-Server metaphor is not enough

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow different capability

Motivations: WAN Programming

- Client-Server metaphor is not enough
- Mobility and dynamic linking of components (e.g. WEB services)
- Adaptability to different devices (e.g. PDA, laptop, mobile phones...)
- Location awareness
  - Applications are location dependent
  - Locations have different features
  - and allow different capability
- ***non-functional properties of systems***

Motivations: Programming QoS

Motivations: Programming QoS

- Connection parameters: depend on applications

Motivations: Programming QoS

- Connection parameters: depend on applications
- QoS as a measure of application's properties (security, reliability, efficiency...)

Motivations: Programming QoS

- Connection parameters: depend on applications
- QoS as a measure of application's properties (security, reliability, efficiency...)
- Need for expressing QoS *at application level*

Motivations: Programming QoS

- Connection parameters: depend on applications
- QoS as a measure of application's properties (security, reliability, efficiency...)
- Need for expressing QoS *at application level*
- Programming = declaring + connecting

Motivations: Programming QoS

- Connection parameters: depend on applications
- QoS as a measure of application's properties (security, reliability, efficiency...)
- Need for expressing QoS *at application level*
- Programming = declaring + connecting
- Executing = distributed constraint solving

Motivations: Our proposal

Motivations: Our proposal

- Klaim: rich calculus

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order
  - Coordination of connection management (SAC 02)

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order
  - Coordination of connection management (SAC 02)
- Connections equipped with “costs”...

Motivations: Our proposal

- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order
  - Coordination of connection management (SAC 02)
- Connections equipped with “costs”...
- ...negotiated by applications and coordinators

Motivations: Our proposal

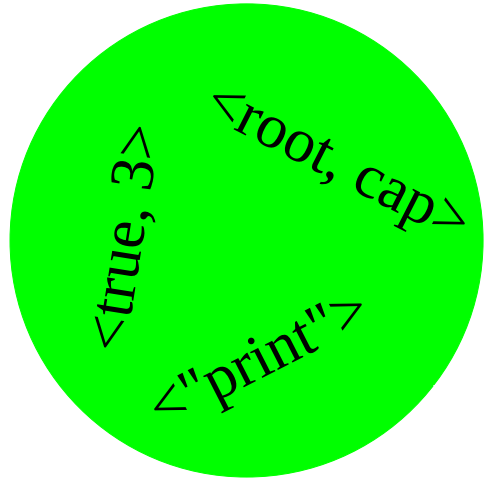
- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order
  - Coordination of connection management (SAC 02)
- Connections equipped with “costs”...
- ...negotiated by applications and coordinators
- Hypergraphs for routing and routing policies

Motivations: Our proposal

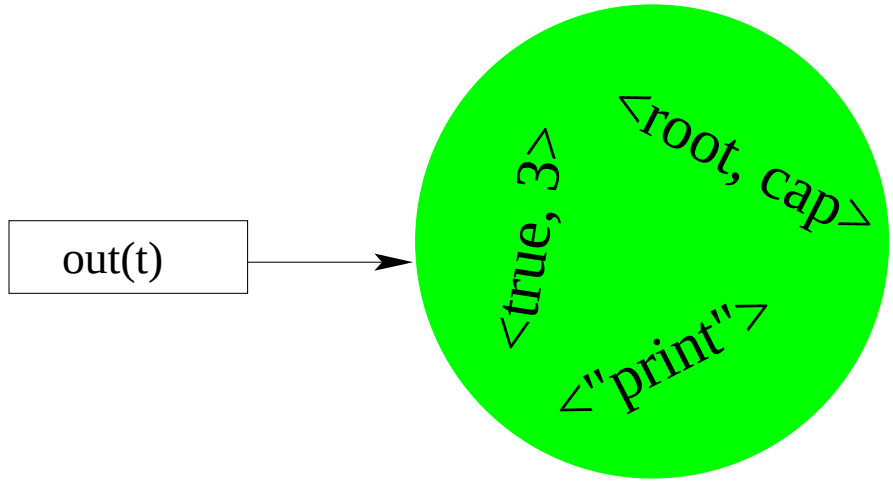
- Klaim: rich calculus
  - Intrinsically *peer-to-peer*
  - Explicit location handling
  - Higher order
  - Coordination of connection management (SAC 02)
- Connections equipped with “costs”...
- ...negotiated by applications and coordinators
- Hypergraphs for routing and routing policies

Klaim & Gateways

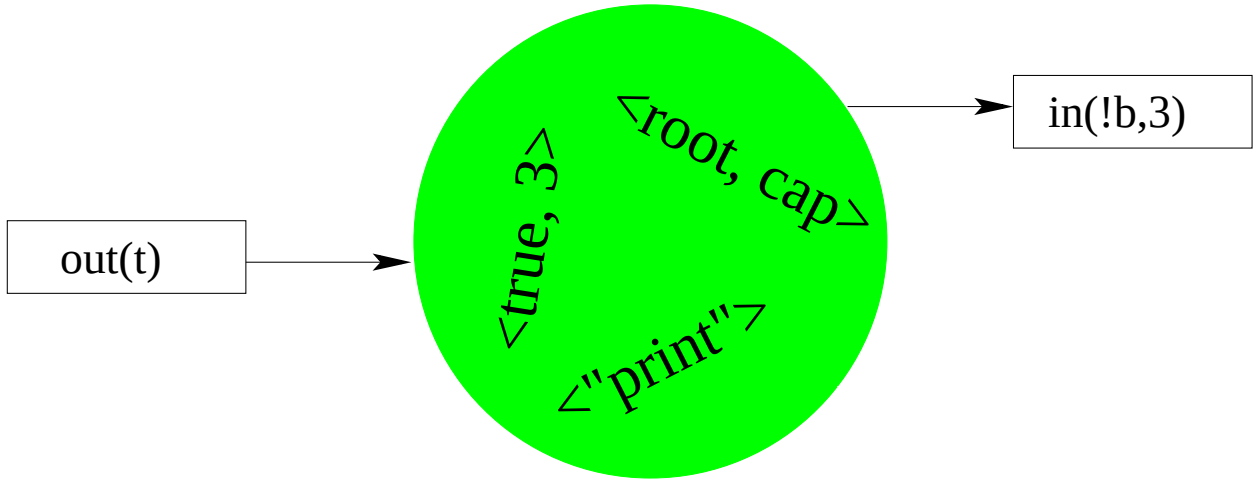
Klaim: Linda Model



Klaim: Linda Model

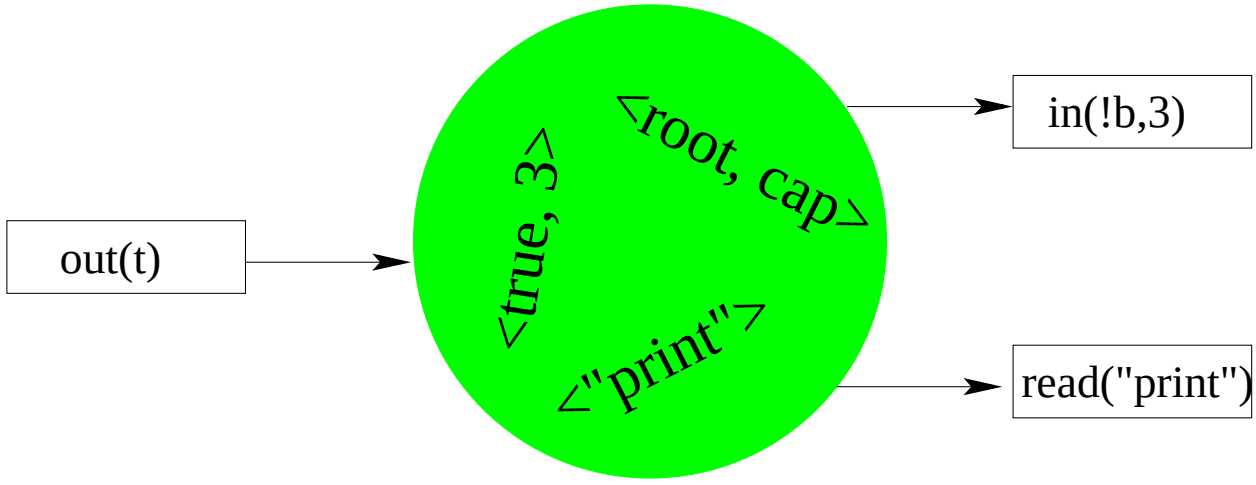


Klaim: Linda Model



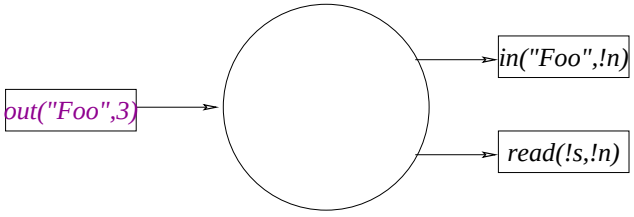
Tuple Space

Klaim: Linda Model

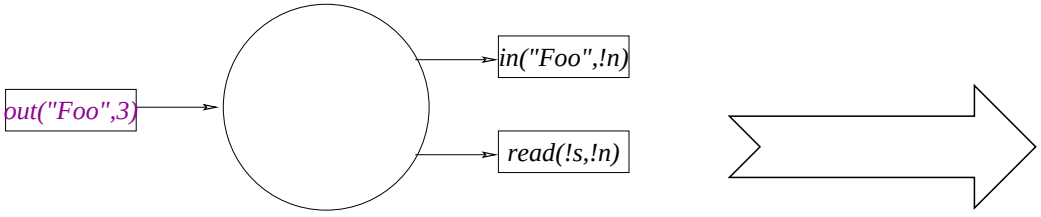


Tuple Space

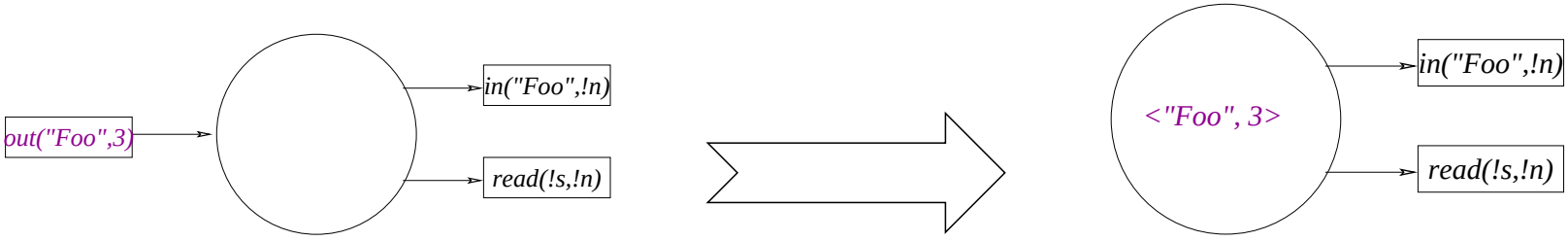
Linda semantics



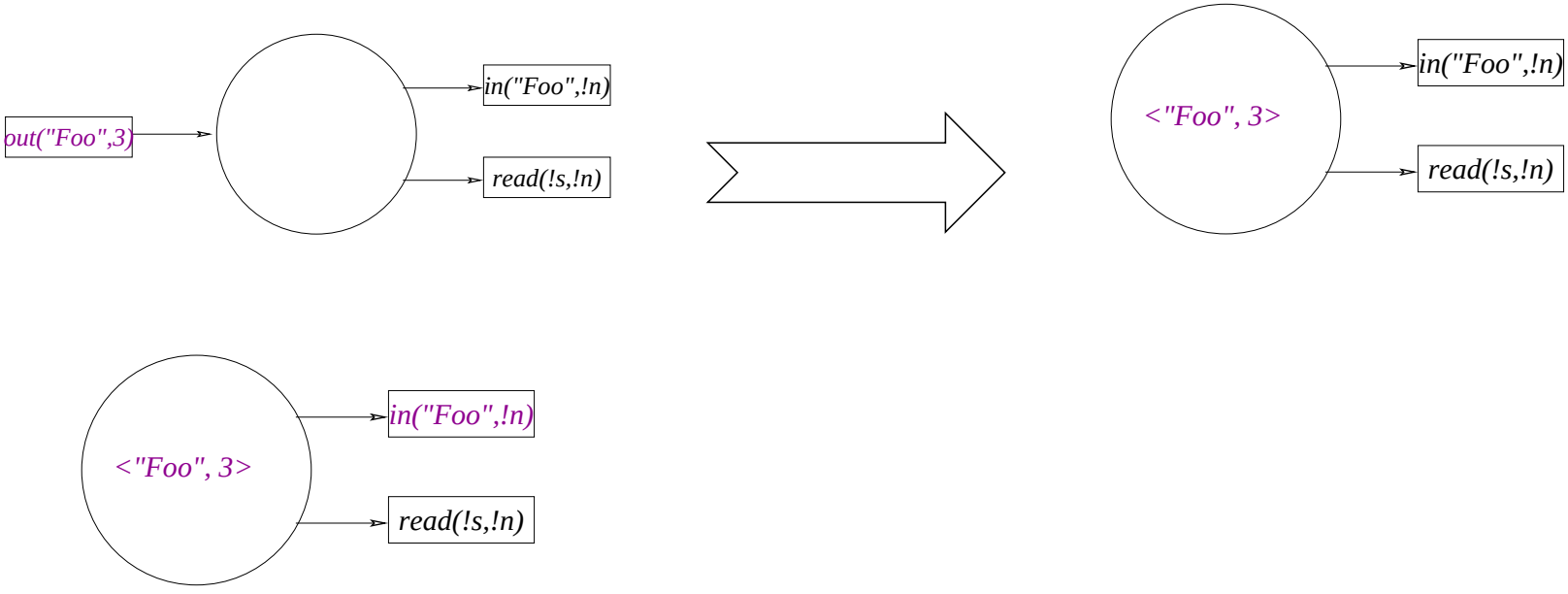
Linda semantics



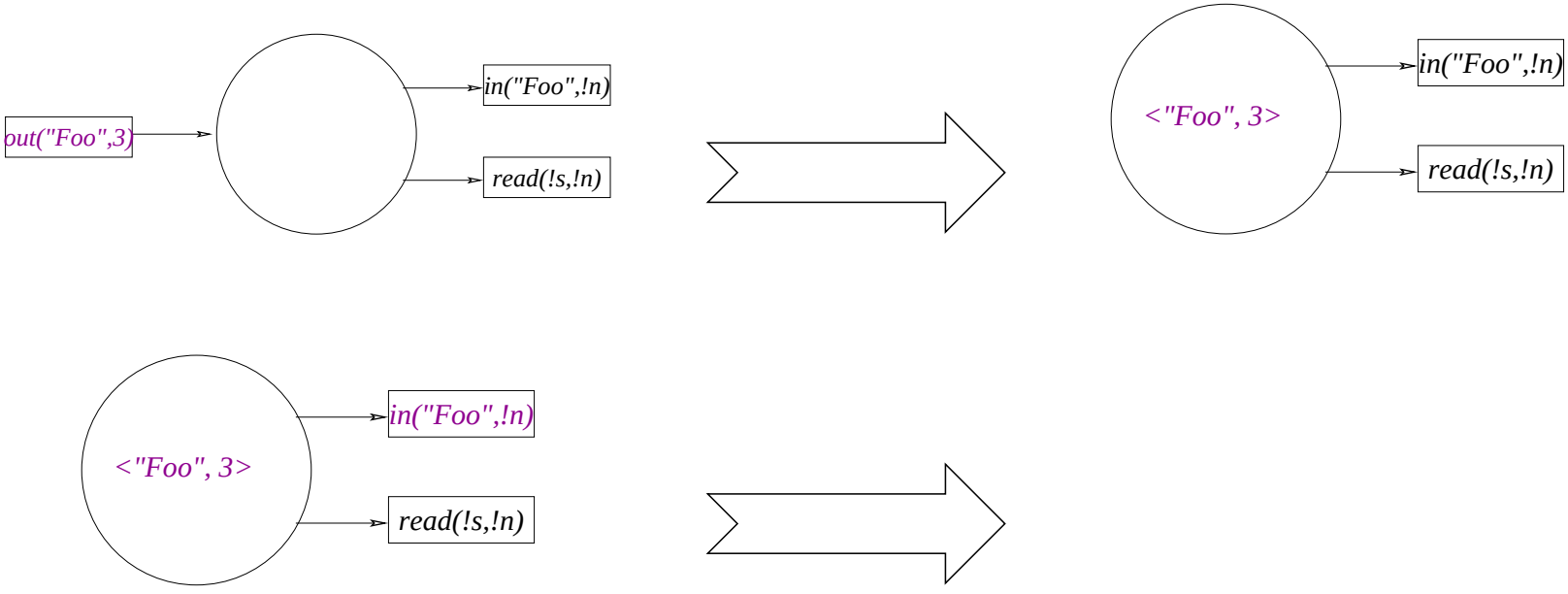
Linda semantics



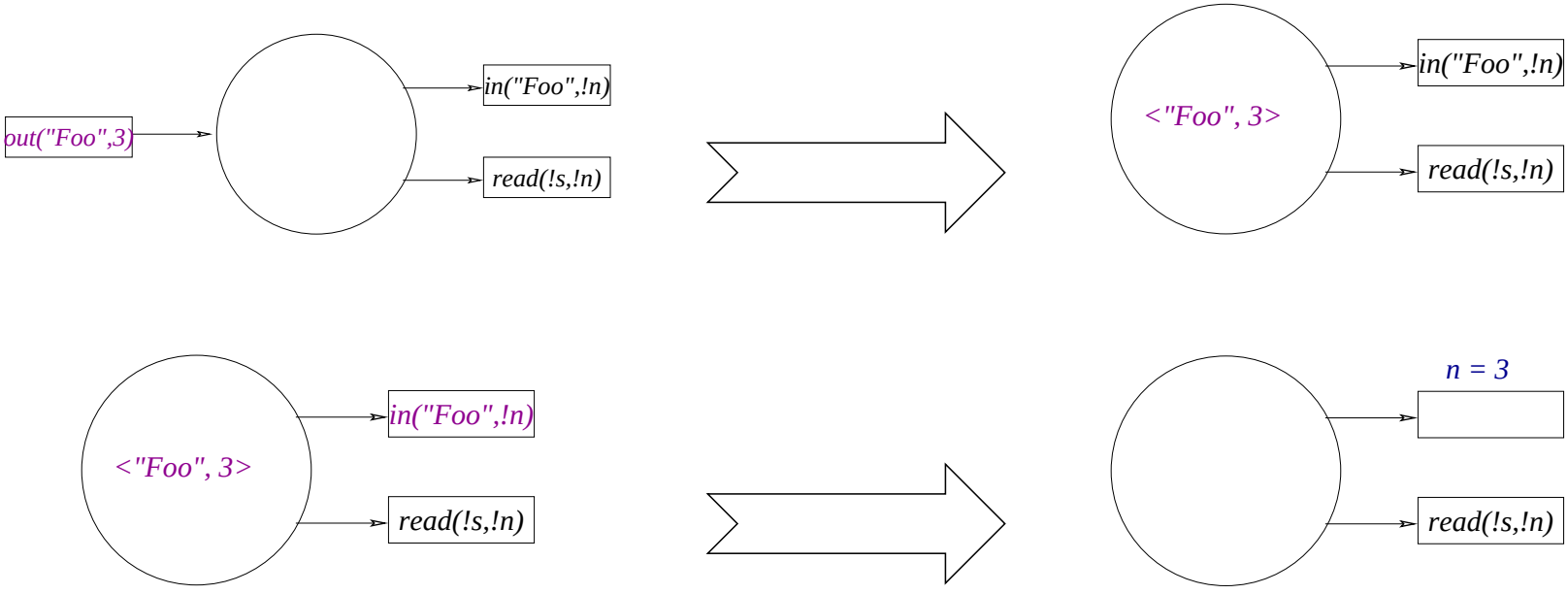
Linda semantics



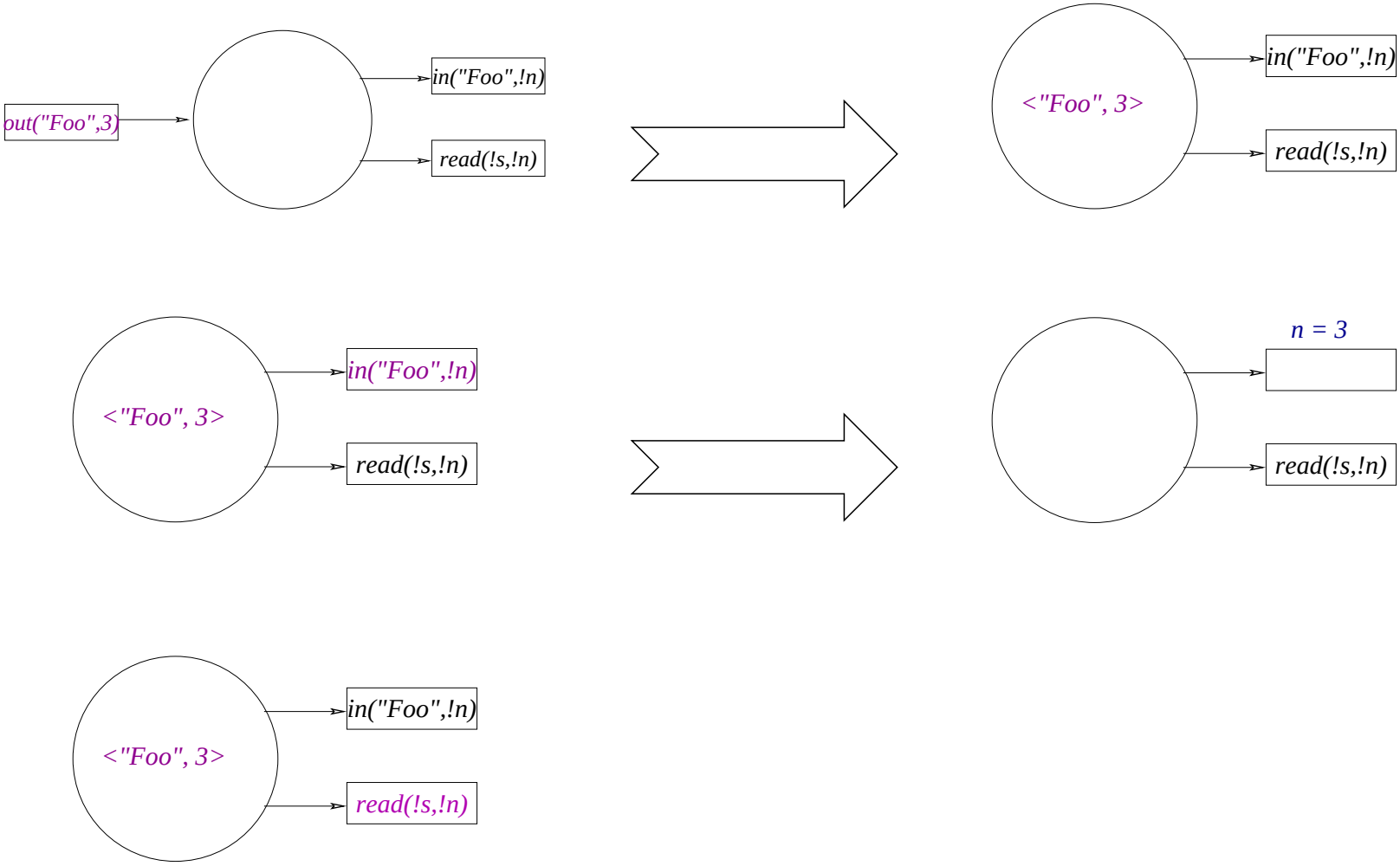
Linda semantics



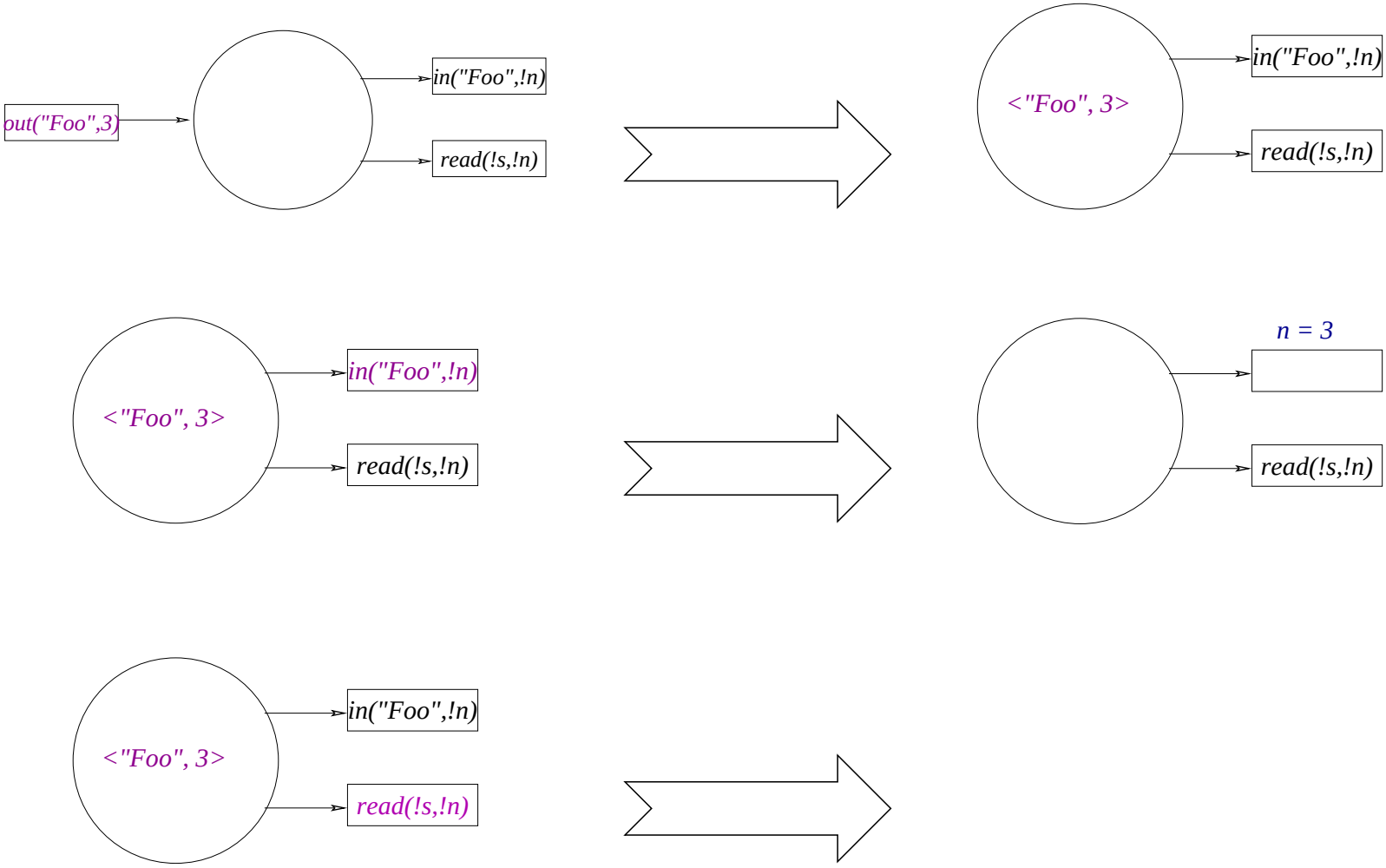
Linda semantics



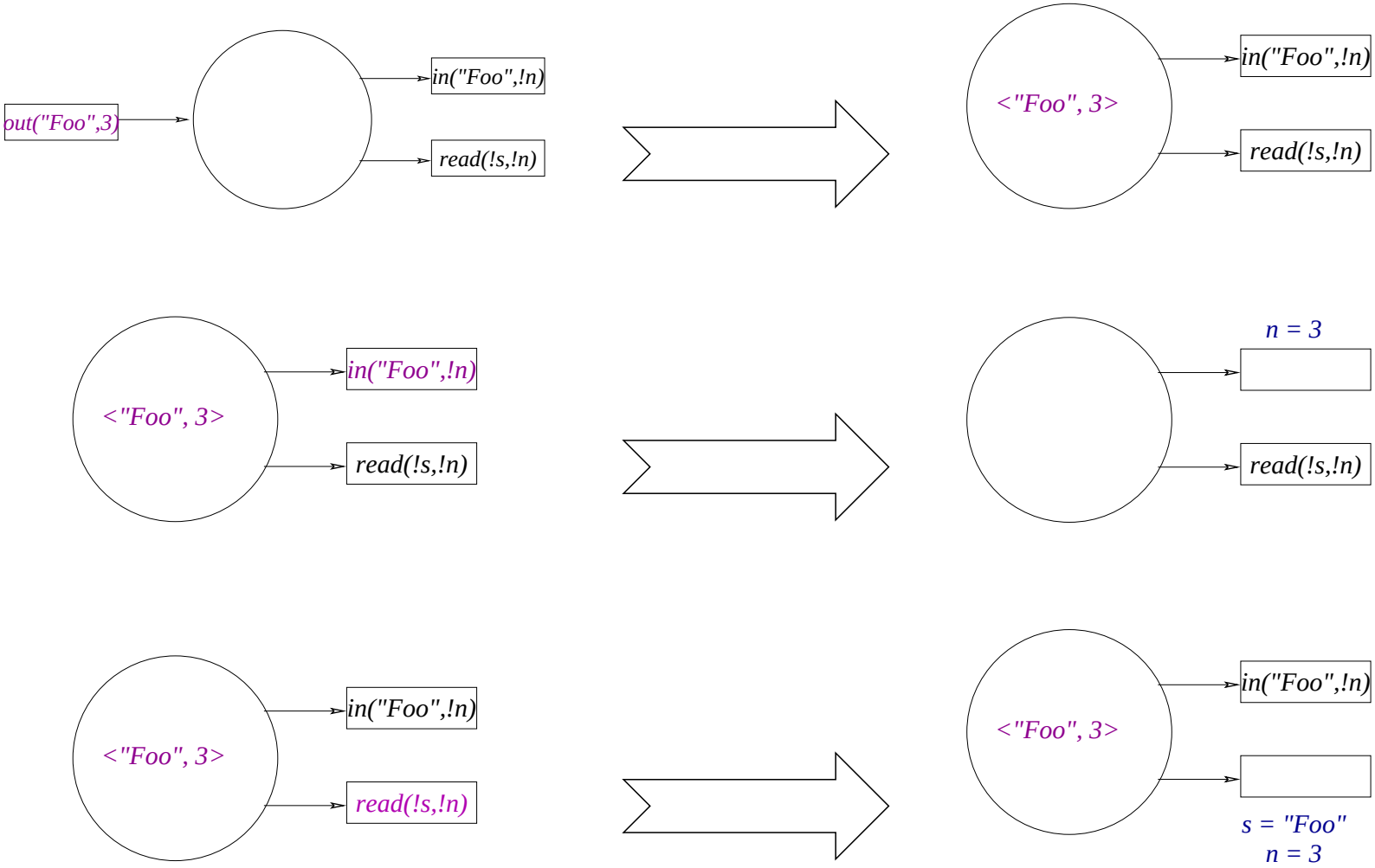
Linda semantics



Linda semantics



Linda semantics



Klaim vs klaim

Klaim vs klaim

- Only physic localities (no allocation environment or **bind**)

Klaim vs klaim

- Only physic localities (no allocation environment or **bind**)
- **out** is always *local*

Klaim vs klaim

- Only physic localities (no allocation environment or **bind**)
- **out** is always *local*
- Incoming gateways are added

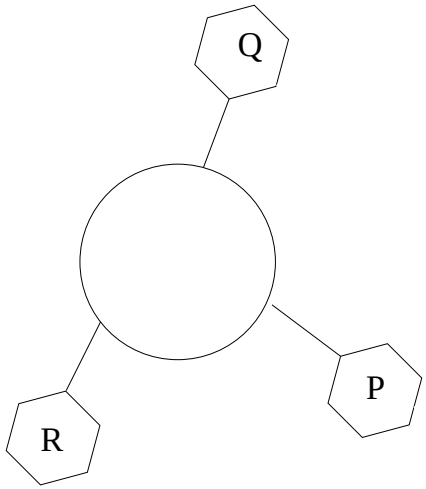
Klaim vs klaim

- Only physic localities (no allocation environment or **bind**)
- **out** is always *local*
- Incoming gateways are added
- Gateways have costs

Klaim vs klaim

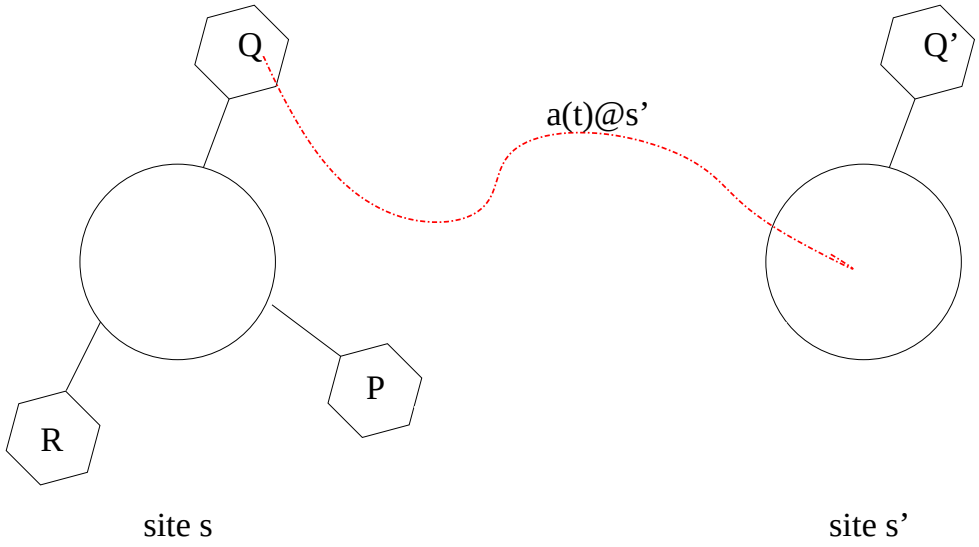
- Only physic localities (no allocation environment or **bind**)
- **out** is always *local*
- Incoming gateways are added
- Gateways have costs
- **disconnect**

Klaim: standard processes



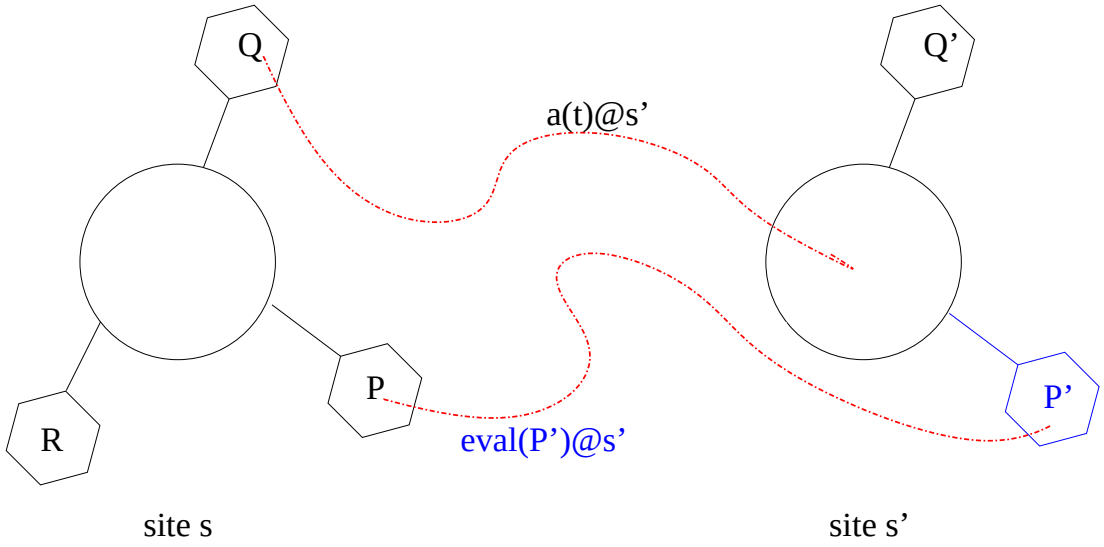
$$
\begin{array}{lcl}
P & ::= & \mathbf{out}(t) \\
 & | & a@s.P \\
a & ::= & \mathbf{in}(T) \\
 & | & \mathbf{read}(T) \\
 & | & \mathbf{eval}(P)
\end{array}
$$

Klaim: standard processes



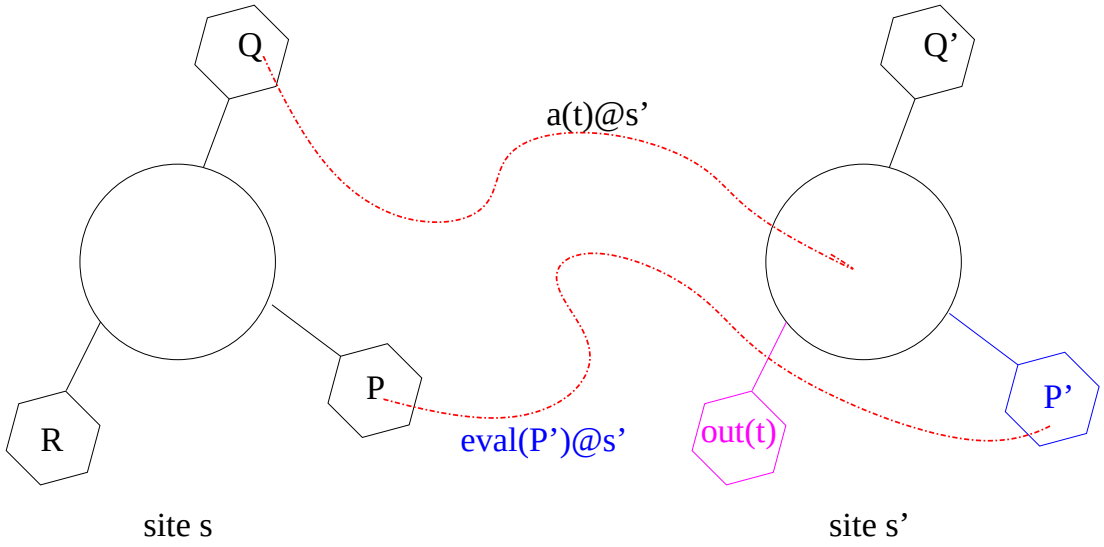
$$
\begin{array}{lcl}
P & ::= & \mathbf{out}(t) \\
 & | & a@s.P \\
a & ::= & \mathbf{in}(T) \\
 & | & \mathbf{read}(T) \\
 & | & \mathbf{eval}(P)
\end{array}
$$

Klaim: standard processes



$$
\begin{array}{rcl}
P & ::= & \mathbf{out}(t) \\
 & | & a@s.P \\
a & ::= & \mathbf{in}(T) \\
 & | & \mathbf{read}(T) \\
 & | & \mathbf{eval}(P)
\end{array}
$$

Klaim: standard processes



$$
\begin{array}{lcl}
P & ::= & \mathbf{out}(t) \\
 & | & a@s.P \\
a & ::= & \mathbf{in}(T) \\
 & | & \mathbf{read}(T) \\
 & | & \mathbf{eval}(P)
\end{array}
$$

Klaim: gateways

Klaim: gateways

- Coordinators (super processes)

Klaim: gateways

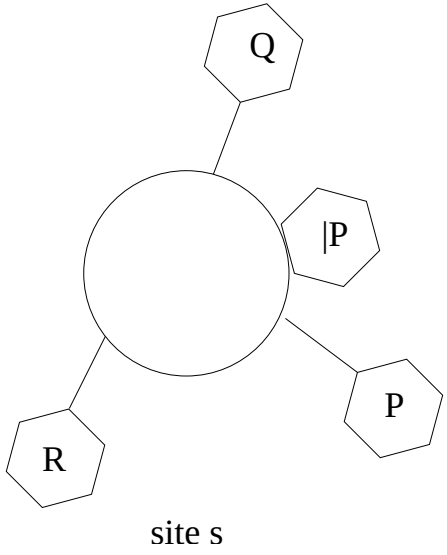
- Coordinators (super processes)
- Dynamic site creation

Klaim: gateways

- Coordinators (super processes)
- Dynamic site creation
- Gateway connection management

Klaim: gateways

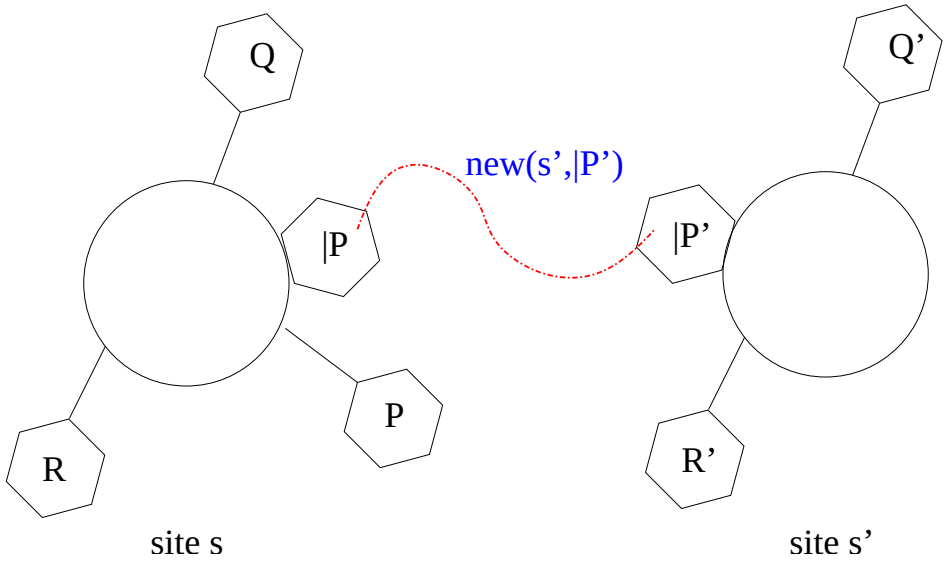
- Coordinators (super processes)
- Dynamic site creation
- Gateway connection management



$$
\begin{array}{lcl}
\mathbb{P} & ::= & \gamma.\mathbb{P} \mid \mathbb{P}_1 \mid \mathbb{P}_2 \\
\gamma & ::= & \alpha \\
 & \mid & \mathbf{new}(s, \mathbb{P}) \\
 & \mid & \mathbf{login}(s, \kappa) \\
 & \mid & \mathbf{accept}(s, \kappa) \\
 & \mid & \mathbf{logout}(s, \kappa) \\
 & \mid & \mathbf{disconnect}(s, \kappa) \\
N & ::= & s ::^{I,O} \mathbb{P} \mid N_1 \parallel N_2
\end{array}
$$

Klaim: gateways

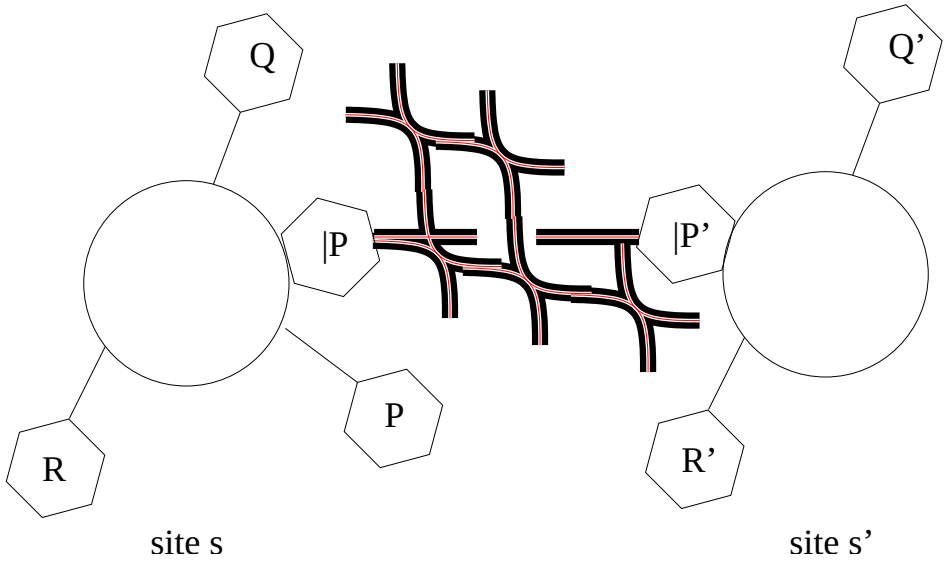
- Coordinators (super processes)
- Dynamic site creation
- Gateway connection management



$$
\begin{array}{lcl}
\mathbb{P} & ::= & \gamma.\mathbb{P} \mid \mathbb{P}_1 \mid \mathbb{P}_2 \\
\gamma & ::= & \alpha \\
 & \mid & \mathbf{new}(s, \mathbb{P}) \\
 & \mid & \mathbf{login}(s, \kappa) \\
 & \mid & \mathbf{accept}(s, \kappa) \\
 & \mid & \mathbf{logout}(s, \kappa) \\
 & \mid & \mathbf{disconnect}(s, \kappa) \\
N & ::= & s ::^{I,O} \mathbb{P} \mid N_1 \parallel N_2
\end{array}
$$

Klaim: gateways

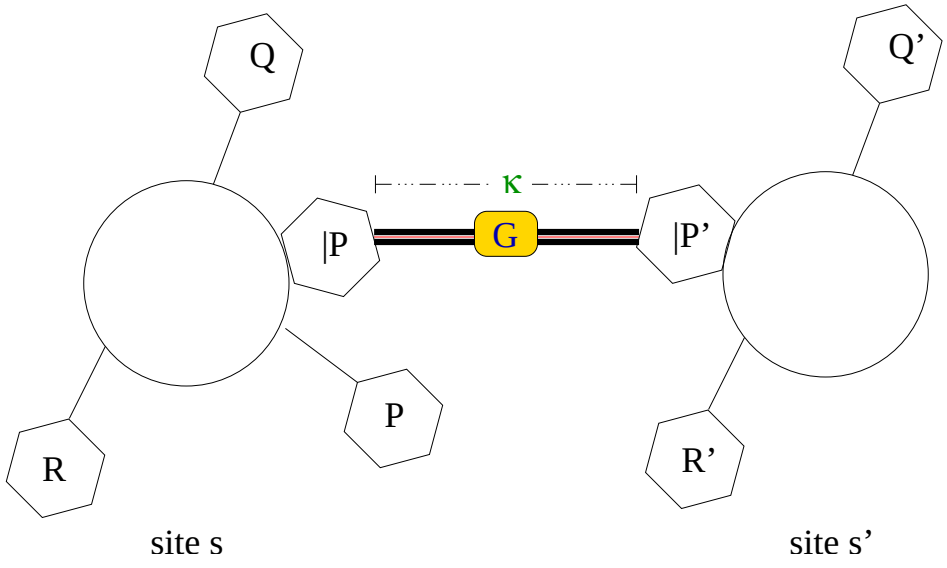
- Coordinators (super processes)
- Dynamic site creation
- Gateway connection management



$$
\begin{array}{lcl}
\mathbb{P} & ::= & \gamma.\mathbb{P} \mid \mathbb{P}_1 \mid \mathbb{P}_2 \\
\gamma & ::= & \alpha \\
 & \mid & \mathbf{new}(s, \mathbb{P}) \\
 & \mid & \mathbf{login}(s, \kappa) \\
 & \mid & \mathbf{accept}(s, \kappa) \\
 & \mid & \mathbf{logout}(s, \kappa) \\
 & \mid & \mathbf{disconnect}(s, \kappa) \\
N & ::= & s ::^{I,O} \mathbb{P} \mid N_1 \parallel N_2
\end{array}
$$

Klaim: gateways

- Coordinators (super processes)
- Dynamic site creation
- Gateway connection management



$$
\begin{array}{lcl}
\mathbb{P} & ::= & \gamma.\mathbb{P} \mid \mathbb{P}_1 \mid \mathbb{P}_2 \\
\gamma & ::= & \alpha \\
 & \mid & \mathbf{new}(s, \mathbb{P}) \\
 & \mid & \mathbf{login}(s, \kappa) \\
 & \mid & \mathbf{accept}(s, \kappa) \\
 & \mid & \mathbf{logout}(s, \kappa) \\
 & \mid & \mathbf{disconnect}(s, \kappa) \\
N & ::= & s ::^{I,O} \mathbb{P} \mid N_1 \parallel N_2
\end{array}
$$

Connection costs κ

κ: abstraction for charatestics and cost of a connection.

Connection costs κ

κ: abstraction for charatestics and cost of a connection.

For instance:
$\kappa = \langle c, \pi \rangle$: cost (bandwidth, latency,...) and access rights

Connection costs κ

κ: abstraction for charatestics and cost of a connection.

For instance:
$\kappa = \langle c, \pi \rangle$: cost (bandwidth, latency,...) and access rights

Operating with costs:

$$\langle c_1, \pi_1 \rangle \otimes \langle c_2, \pi_2 \rangle = \langle c_1 + c_2, \pi_1 \cap \pi_2 \rangle$$

$$\langle c, \pi \rangle \models \mathbf{a} \iff \mathbf{a} \in \pi \qquad (\mathbf{a} \in \{\mathbf{o}, \mathbf{e}, \mathbf{i}, \mathbf{r}\})$$

Klaim Semantics (1)

$$s ::^{I,O} \mathbf{eval}(P)@s'.\mathbb{P} \xrightarrow[s,\top]{\mathbf{e}(s,P,s')} s ::^{I,O} \mathbb{P} \qquad \text{(EVAL)}$$

$$s ::^{I,O} \mathbb{P} \xrightarrow[s,\top]{s::^{I,O}\mathbb{P}} \mathbf{0} \qquad \text{(SITE)}$$

$$\frac{N_1 \xrightarrow[s_1,\kappa]{a} N_1' \quad N_2 \xrightarrow[s_2,\kappa_2]{s_2::^{I\sqcup\langle s_1,\kappa_1\rangle,O}\mathbb{P}} N_2'}{N_1 \parallel N_2 \xrightarrow[s_2,\kappa\otimes\kappa_1]{a} N_1' \parallel N_2' \parallel s_2 ::^{I\sqcup\langle s_1,\kappa_1\rangle,O} \mathbb{P}} \qquad \text{(ENV)}$$

$$\frac{N_1 \xrightarrow[s,\kappa_1]{\mathbf{e}(s_1,P,s_2)} N_1' \qquad N_1' \xrightarrow[s,\kappa_2]{s_2::^{I,O}\mathbb{P}} N_2 \qquad \kappa_1 \otimes \kappa_2 \models \mathbf{e}}{N_1 \rightarrowtail N_2 \parallel s_2 ::^{I,O} P|\mathbb{P}} \qquad \text{(NETEVAL)}$$

Klaim Semantics (2)

$$s_1 ::^{I,O} \mathbf{login}(s_2, \kappa).\mathbb{P} \xrightarrow{\mathbf{lin}(s_1, \kappa, s_2)} s_1 ::^{I, O \sqcup \langle s_2, \kappa \rangle} \mathbb{P} \qquad \text{(LOGIN)}$$

$$s_2 ::^{I,O} \mathbf{accept}(s, \kappa).\mathbb{P} \xrightarrow{\mathbf{acc}(s_2, \kappa, s_1)} s_2 ::^{I \sqcup \langle s_1, \kappa \rangle, O} \mathbb{P}[s_1/s] \qquad \text{(ACCEPT)}$$

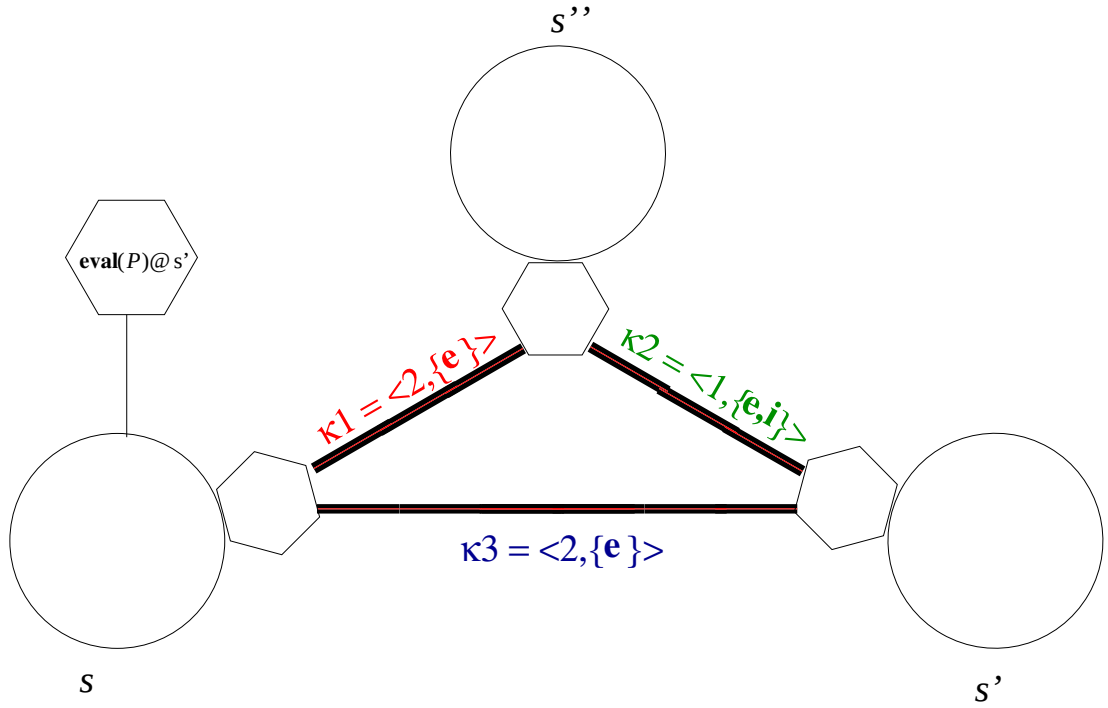
$$\frac{N_1 \xrightarrow{\mathbf{lin}(s_1, \kappa, s_2)} N_1' \quad N_1' \xrightarrow{\mathbf{acc}(s_2, \kappa, s_1)} N_2}{N_1 \rightarrowtail N_2} \qquad \text{(NETCONNECT}$$

Klaim: Example

$$
\begin{array}{ll}
s & ::\{\langle s',\{\kappa 3\}\rangle\},\{\langle s'',\{\kappa 1\}\rangle\} \quad \mathbf{eval}(P)@s' \\
\| & \\
s'' & ::\{\langle s',\{\kappa 2\}\rangle\},\{\langle s,\{\kappa 1\}\rangle\} \quad \ldots \\
\| & \\
s' & ::\{\langle s'',\{\kappa 2\}\rangle\},\{\langle s,\{\kappa 3\}\rangle\} \quad \ldots
\end{array}
$$

Klaim: Example

$$
\begin{array}{l}
s \;::\{\langle s',\{\kappa 3\}\rangle\},\{\langle s'',\{\kappa 1\}\rangle\}\ \mathbf{eval}(P)@s' \\
\quad \| \\
s'' \;::\{\langle s',\{\kappa 2\}\rangle\},\{\langle s,\{\kappa 1\}\rangle\}\ \ldots \\
\quad \| \\
s' \;::\{\langle s'',\{\kappa 2\}\rangle\},\{\langle s,\{\kappa 3\}\rangle\}\ \ldots
\end{array}
$$



Hypergraphs

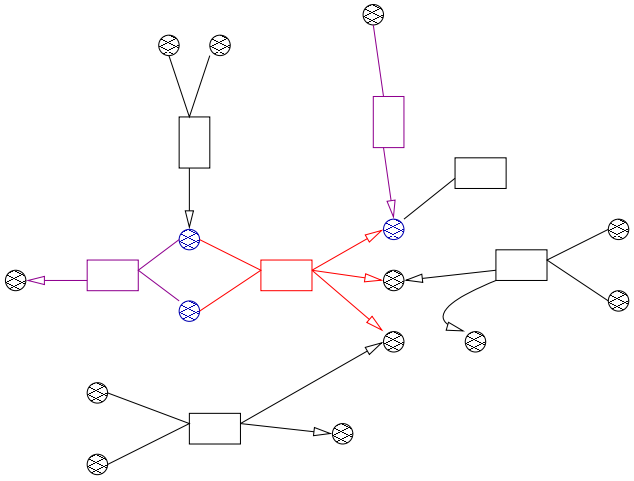
Constrained synchronizations

Constrained synchronizations

(Hyper)Graphical calculus: edge replacement

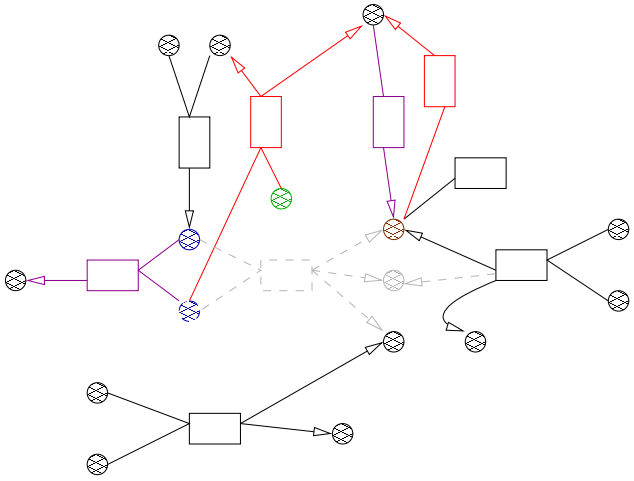
Constrained synchronizations

(Hyper)Graphical calculus: edge replacement



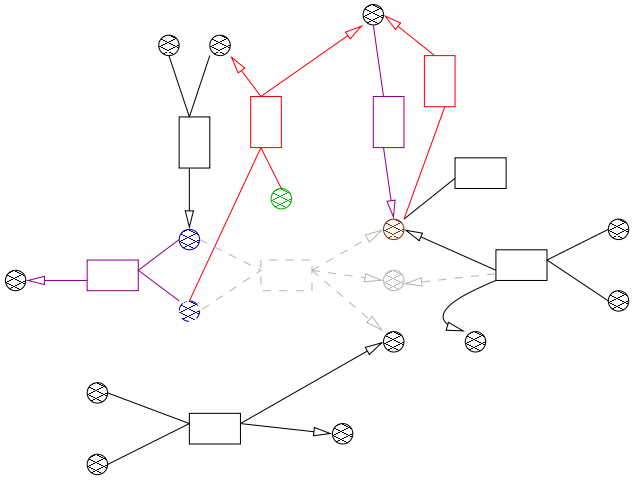
Constrained synchronizations

(Hyper)Graphical calculus: edge replacement



Constrained synchronizations

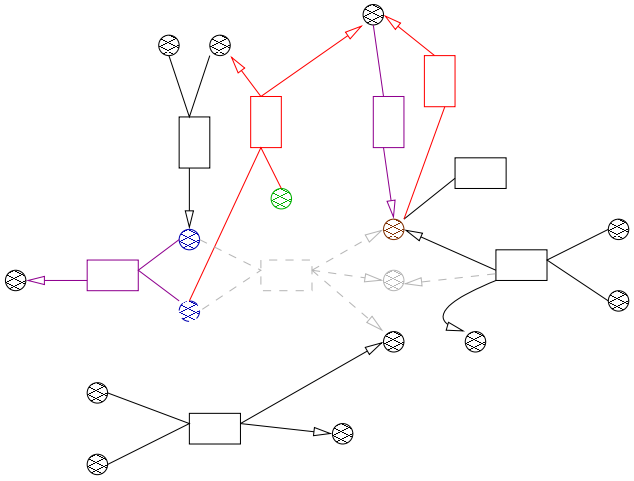
(Hyper)Graphical calculus: edge replacement



- Edge replacement: local
- Synchronization as distributed constraint solving
- New node creation
- Node fusion: mobility model

Constrained synchronizations

(Hyper)Graphical calculus: edge replacement



- Edge replacement: local
- Synchronization as distributed constraint solving
- New node creation
- Node fusion: mobility model

Benefits:

- Powerful model of system composition (π, π-I, fusion)
- LTS for Ambient (ICTCS 2001)...
- ...and for Klaim
- and its extension to gateways with costs

Graphs: Syntax

Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v}$$

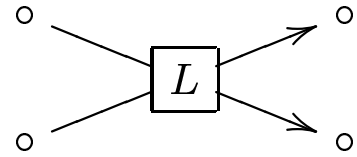
Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v} \qquad\qquad L : 2 \to 2$$

Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v}$$

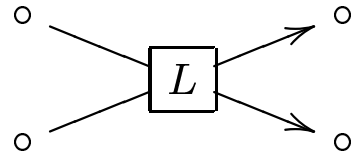
$$L : 2 \to 2$$



Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v}$$

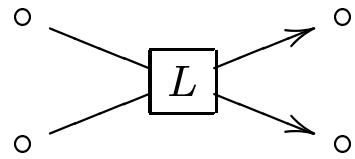
$$L : 2 \to 2$$



$$G ::= L(\vec{x}) \mid G|G \mid \nu\, y.G \mid nil$$

Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v} \qquad L : 2 \to 2$$

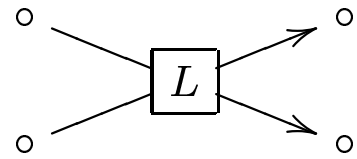


$$G ::= L(\vec{x}) \mid G|G \mid \nu\, y.G \mid nil$$

Syntactic Judgment	$\Gamma \vdash G$	$fn(G) \subseteq \Gamma$

Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v} \qquad L : 2 \to 2$$



$$G ::= L(\vec{x}) \mid G|G \mid \nu\, y.G \mid nil$$

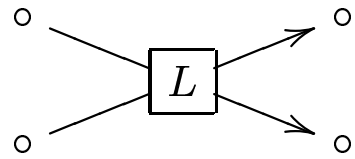
Syntactic Judgment	$\Gamma \vdash G$	$fn(G) \subseteq \Gamma$

An example:

$$L : 2 \to 1, \quad M : 1 \to 1$$
$$x, y \vdash \nu\, z.(L(y, z, x)|M(x, z))$$

Graphs: Syntax

$$\mathcal{L} = \bigcup_{u,v \in \omega} \mathcal{L}_{u \to v} \qquad L : 2 \to 2$$

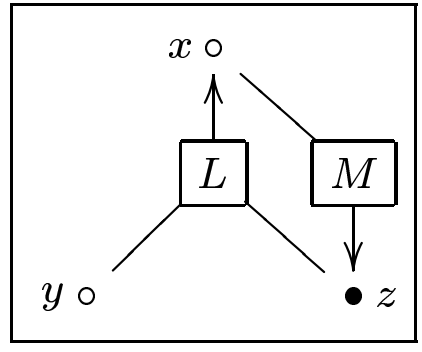


$$G ::= L(\vec{x}) \mid G|G \mid \nu\, y.G \mid nil$$

Syntactic Judgment	$\Gamma \vdash G$	$fn(G) \subseteq \Gamma$

An example:

$$L : 2 \to 1, \quad M : 1 \to 1$$
$$x, y \vdash \nu\, z.(L(y, z, x)|M(x, z))$$



Graphs: Transitions and Productions

Graphs: Transitions and Productions

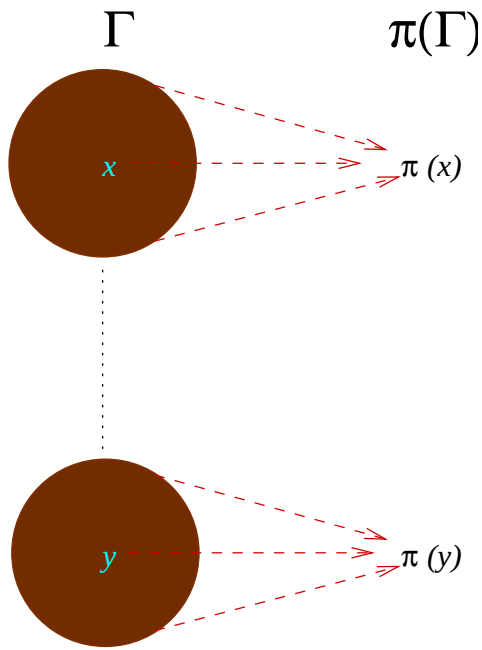
Transitions $\quad \Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2$

$$\begin{cases} \Lambda : \Gamma \multimap (Act \times \mathcal{N}^*) \\ \pi : \Gamma \to \Gamma, \quad \pi^{-1}(x) \neq \emptyset \Rightarrow x \in \pi^{-1}(x) \\ n(\Lambda) \cap \Gamma \subseteq \pi(\Gamma) \\ \phi = \pi(\Gamma) \cup (n(\Lambda) - \Gamma) \end{cases}$$

Graphs: Transitions and Productions

Transitions $\quad \Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2$

$$\begin{cases} \Lambda : \Gamma \multimap (Act \times \mathcal{N}^*) \\ \pi : \Gamma \to \Gamma, \quad \pi^{-1}(x) \neq \emptyset \Rightarrow x \in \pi^{-1}(x) \\ n(\Lambda) \cap \Gamma \subseteq \pi(\Gamma) \\ \phi = \pi(\Gamma) \cup (n(\Lambda) - \Gamma) \end{cases}$$



Graphs: Transitions and Productions

Transitions $\quad \Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2$

$$\begin{cases} \Lambda : \Gamma \multimap (Act \times \mathcal{N}^*) \\ \pi : \Gamma \to \Gamma, \quad \pi^{-1}(x) \neq \emptyset \Rightarrow x \in \pi^{-1}(x) \\ n(\Lambda) \cap \Gamma \subseteq \pi(\Gamma) \\ \phi = \pi(\Gamma) \cup (n(\Lambda) - \Gamma) \end{cases}$$

Production $\quad set(\vec{x}) \vdash L(\vec{x}) \xrightarrow{\Lambda,\pi} \phi \vdash G$

Graphs: Transitions and Productions

Transitions $\quad \Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2$

$$\begin{cases} \Lambda : \Gamma \multimap (Act \times \mathcal{N}^*) \\ \pi : \Gamma \to \Gamma, \quad \pi^{-1}(x) \neq \emptyset \Rightarrow x \in \pi^{-1}(x) \\ n(\Lambda) \cap \Gamma \subseteq \pi(\Gamma) \\ \phi = \pi(\Gamma) \cup (n(\Lambda) - \Gamma) \end{cases}$$

Production $\quad set(\vec{x}) \vdash L(\vec{x}) \xrightarrow{\Lambda,\pi} \phi \vdash G$

$$x, y, z \vdash L(y, z, x) \xrightarrow{\Lambda, id} x, y, z, u, v \vdash L'(x, y) | L''(u, z, v) \qquad \Lambda = \left\{ \begin{array}{c} (x,\ \overline{send},\ \langle u \rangle), \\ (y,\ rec,\ \langle v \rangle) \end{array} \right\}$$

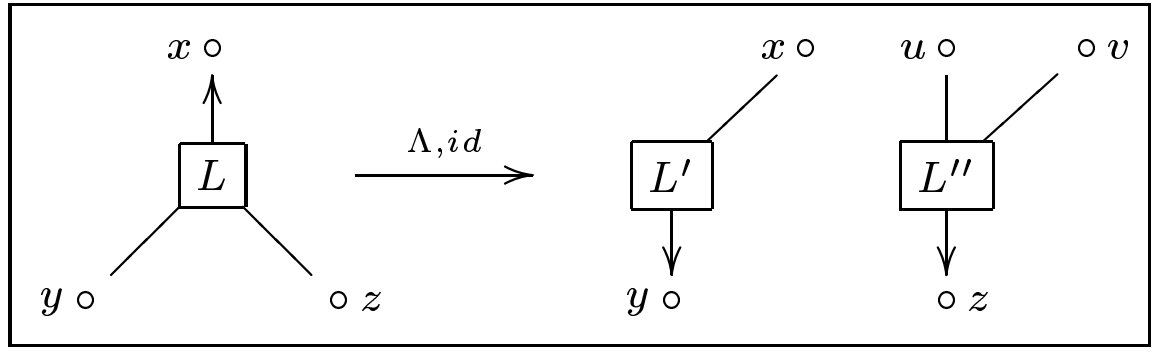
Graphs: Transitions and Productions

Transitions $\quad \Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2$

$$\begin{cases} \Lambda : \Gamma \multimap (Act \times \mathcal{N}^*) \\ \pi : \Gamma \to \Gamma, \quad \pi^{-1}(x) \neq \emptyset \Rightarrow x \in \pi^{-1}(x) \\ n(\Lambda) \cap \Gamma \subseteq \pi(\Gamma) \\ \phi = \pi(\Gamma) \cup (n(\Lambda) - \Gamma) \end{cases}$$

Production $\quad set(\vec{x}) \vdash L(\vec{x}) \xrightarrow{\Lambda,\pi} \phi \vdash G$

$$x, y, z \vdash L(y, z, x) \xrightarrow{\Lambda, id} x, y, z, u, v \vdash L'(x, y) | L''(u, z, v) \qquad \Lambda = \left\{ \begin{array}{c} (x,\ \overline{send},\ \langle u \rangle), \\ (y,\ rec,\ \langle v \rangle) \end{array} \right\}$$



Graphs: Semantics (1)

$$(par)\ \frac{\Gamma \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2 \qquad \Gamma' \vdash G_1' \xrightarrow{\Lambda',\pi'} \phi' \vdash G_2'}{\Gamma, \Gamma' \vdash G_1 | G_1' \xrightarrow{\Lambda \cup \Lambda', \pi \cup \pi'} \phi, \phi' \vdash G_2 | G_2'} \qquad \Gamma \cap \Gamma' = \emptyset$$

$$(res)\ \frac{\Gamma, x \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2}{\Gamma \vdash \nu\ x.G_1 \xrightarrow{\Lambda', \pi|_\Gamma} \phi' \vdash \nu\ \vec{z}.G_2}$$

$$\begin{cases} (\pi(x) = \pi(y) \wedge x \neq y) \Rightarrow \pi(x) \neq x \\ \Lambda(x) \uparrow \text{ or } \Lambda(x) = (\tau, \langle\rangle), \\ \Lambda' = \Lambda - (x, \tau, \langle\rangle) \\ \vec{z} = \phi - \phi' \end{cases}$$

Graphs: Semantics (2)

$$(merge1)\ \frac{\Gamma, x, y \vdash G_1 \xrightarrow{\Lambda,\pi} \phi \vdash G_2}{\Gamma, x \vdash \sigma G_1 \xrightarrow{\Lambda', \pi;\rho|_{\Gamma,x}} \phi' \vdash \nu\,\vec{u}.\rho G_2}$$

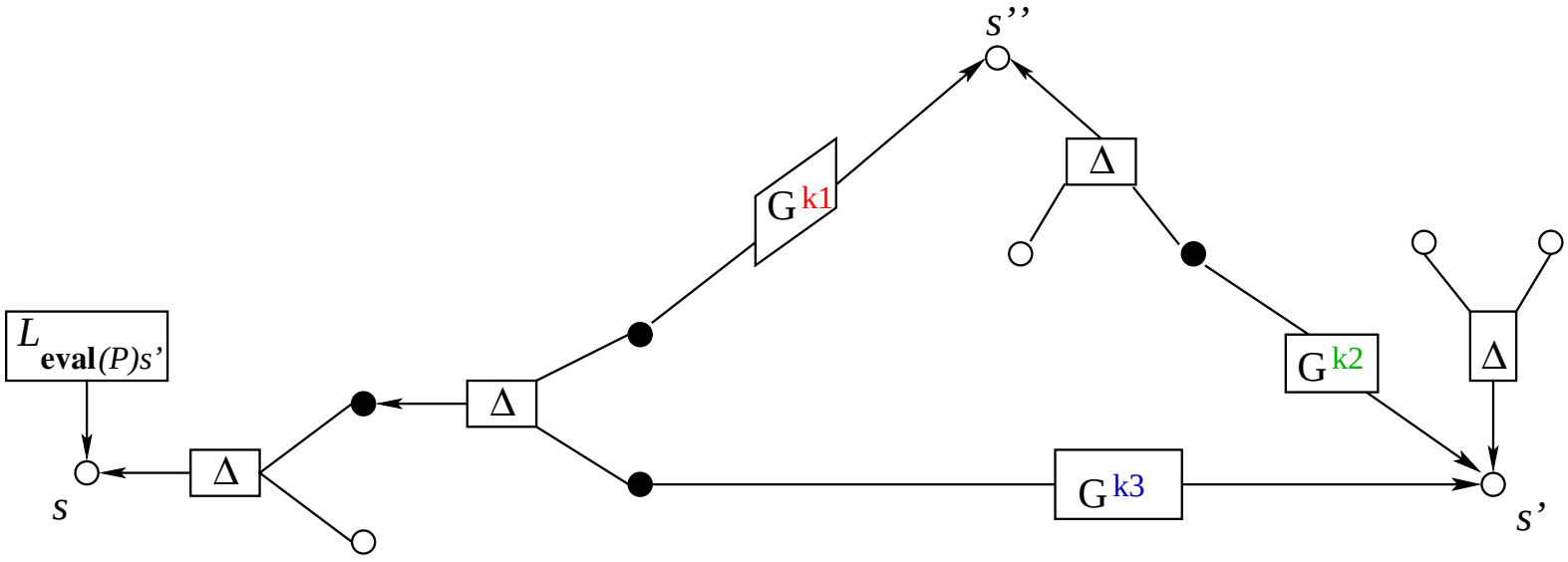
$$\begin{cases} \Lambda(y) \uparrow \\ \sigma = (y \mapsto x) \\ \rho = (\pi(y) \mapsto \pi(x)) \\ \Lambda' = \lambda z.\rho(\Lambda(z)) \\ \vec{u} = \rho(\sigma(\phi)) - \phi' \end{cases}$$

$$(merge2)\ \frac{\Gamma, x, y \vdash G_1 \xrightarrow{\Lambda \cup \{(x,a,\vec{v}),(y,\overline{a},\vec{w})\},\pi} \phi \vdash G_2}{\Gamma, x \vdash \sigma G_1 \xrightarrow{\Lambda' \cup \{(x,\tau,\langle\rangle)\},\pi;\rho|_{\Gamma,x}} \phi' \vdash \nu\,\vec{u}.\rho G_2}$$

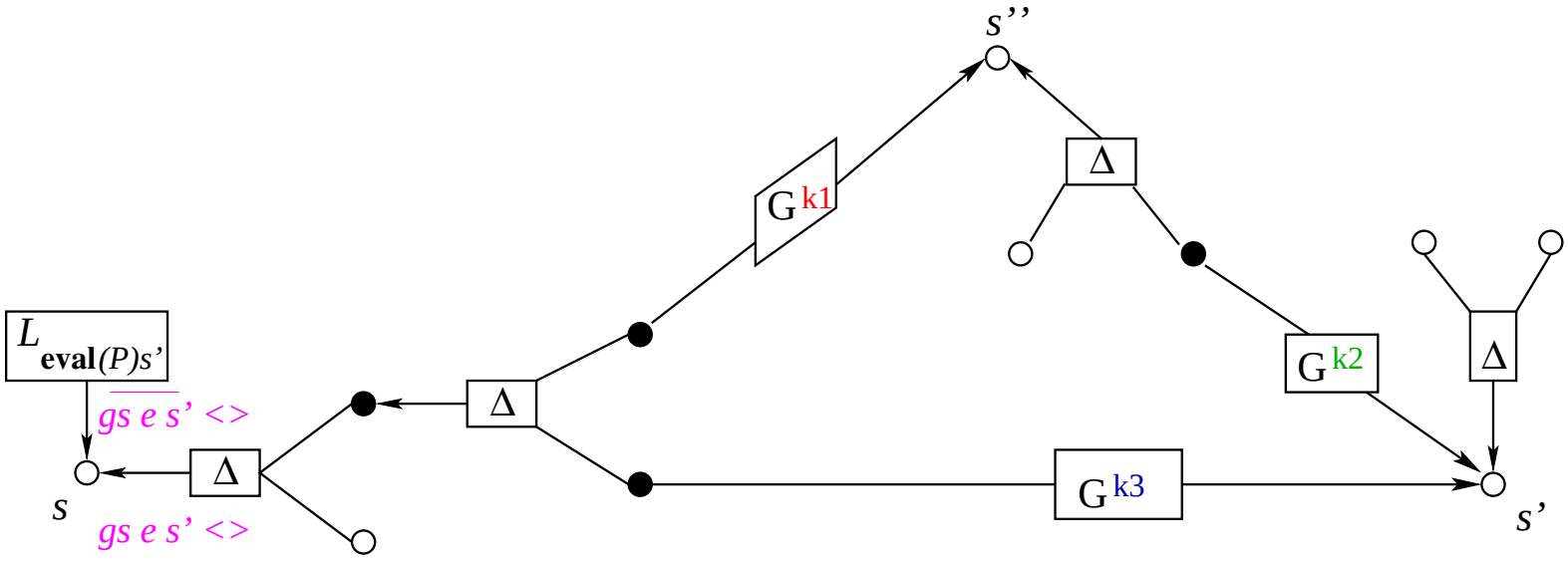
$$\begin{cases} \sigma = (y \mapsto x) \\ \rho = mgu(\sigma(\vec{v}) \mapsto \sigma(\vec{w})) \\ \Lambda' = \lambda z.\rho(\Lambda(z)) \\ \vec{u} = \rho(\sigma(\phi)) - \phi' \end{cases}$$

An Example:
Graphs for klaim and routing

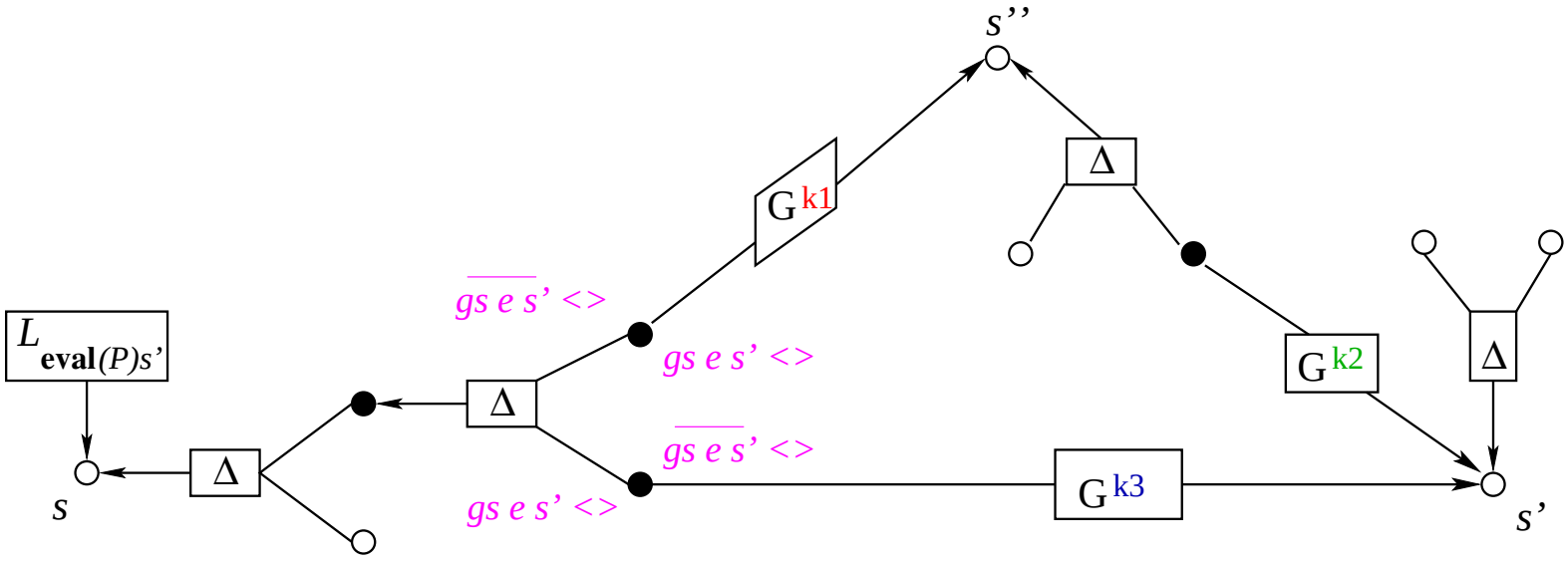
abc



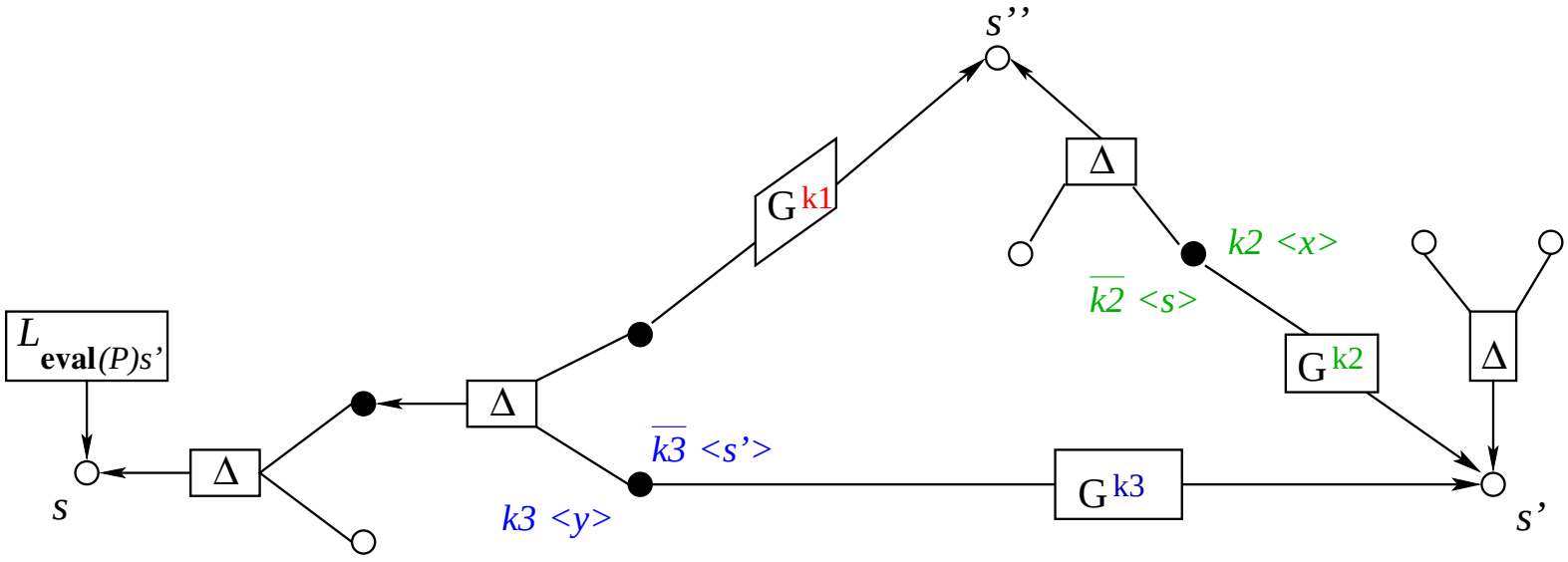
abc



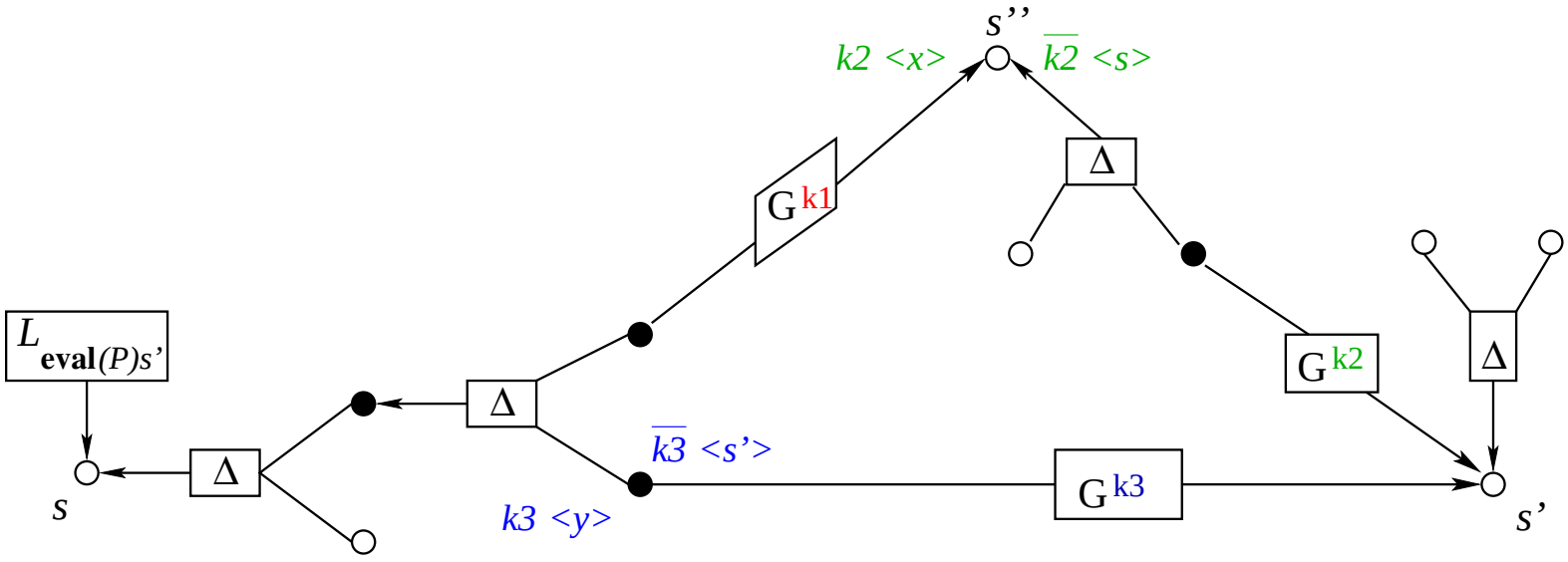
abc


s''
G k1
Δ
$\overline{gs\ e\ s'\ <>}$
$L_{\mathbf{eval}(P)s'}$
gs e s' <>
G k2
Δ
Δ
Δ
$\overline{gs\ e\ s'\ <>}$
s
gs e s' <>
G k3
s'

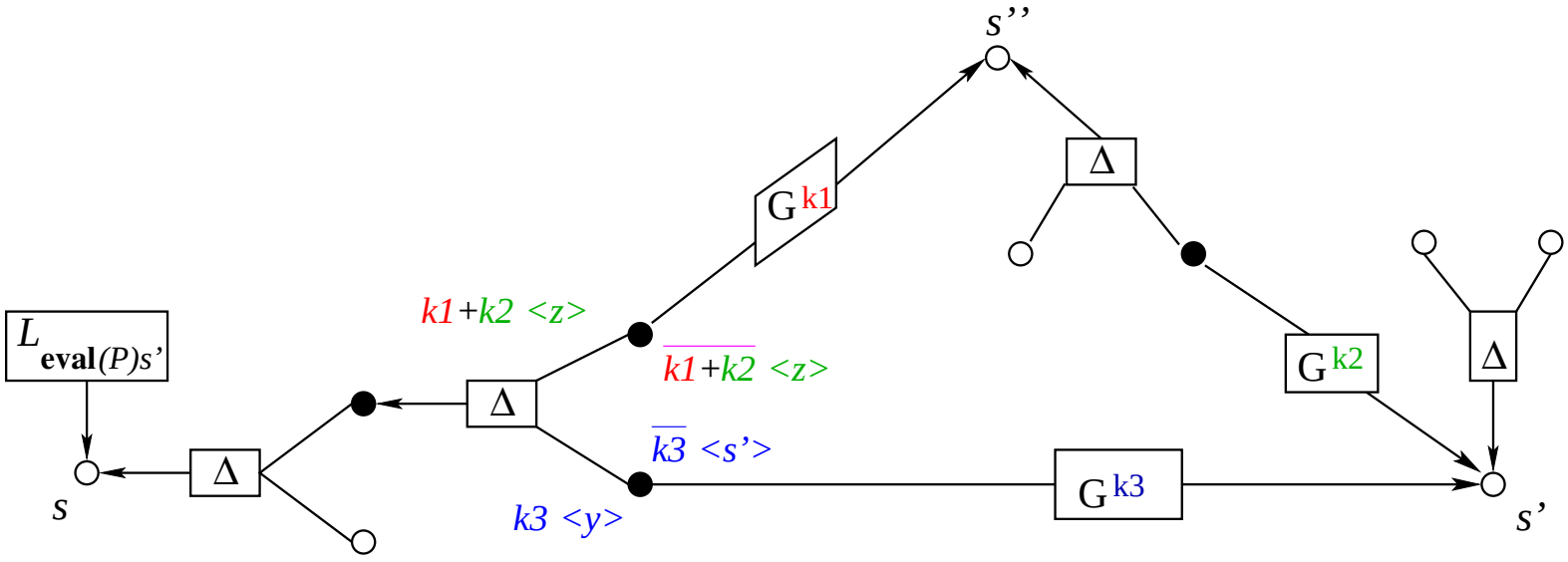
abc



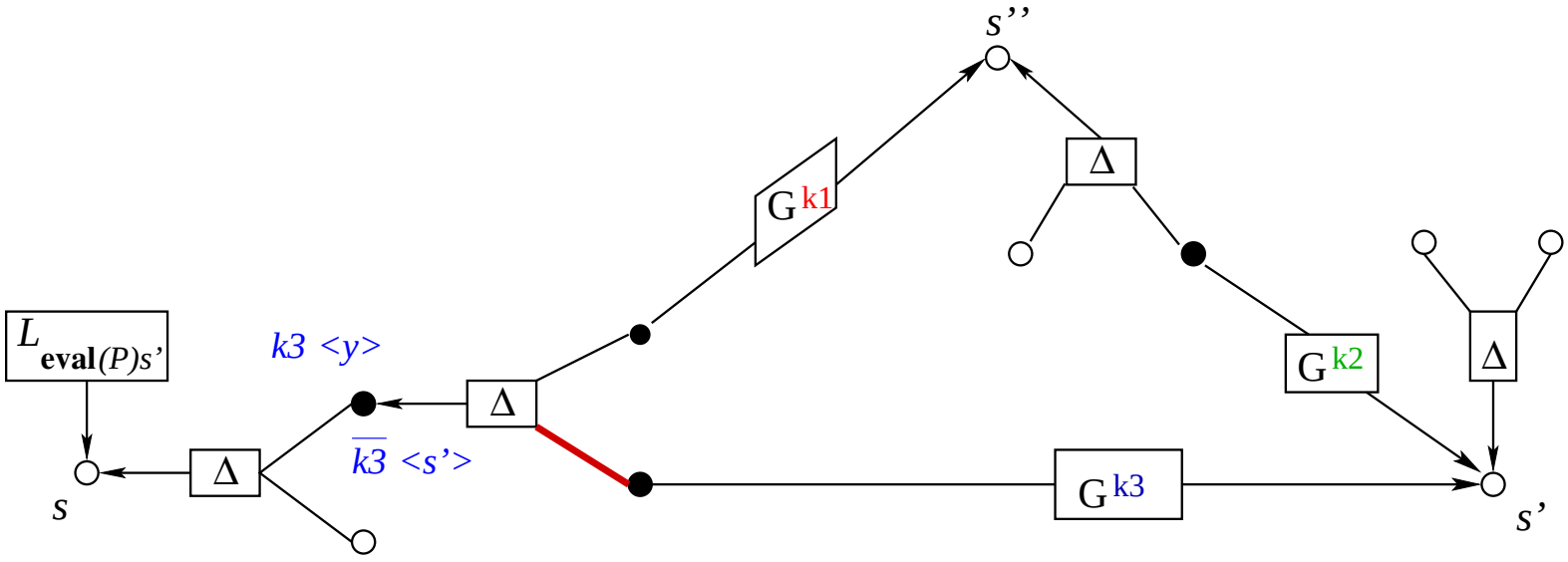
abc



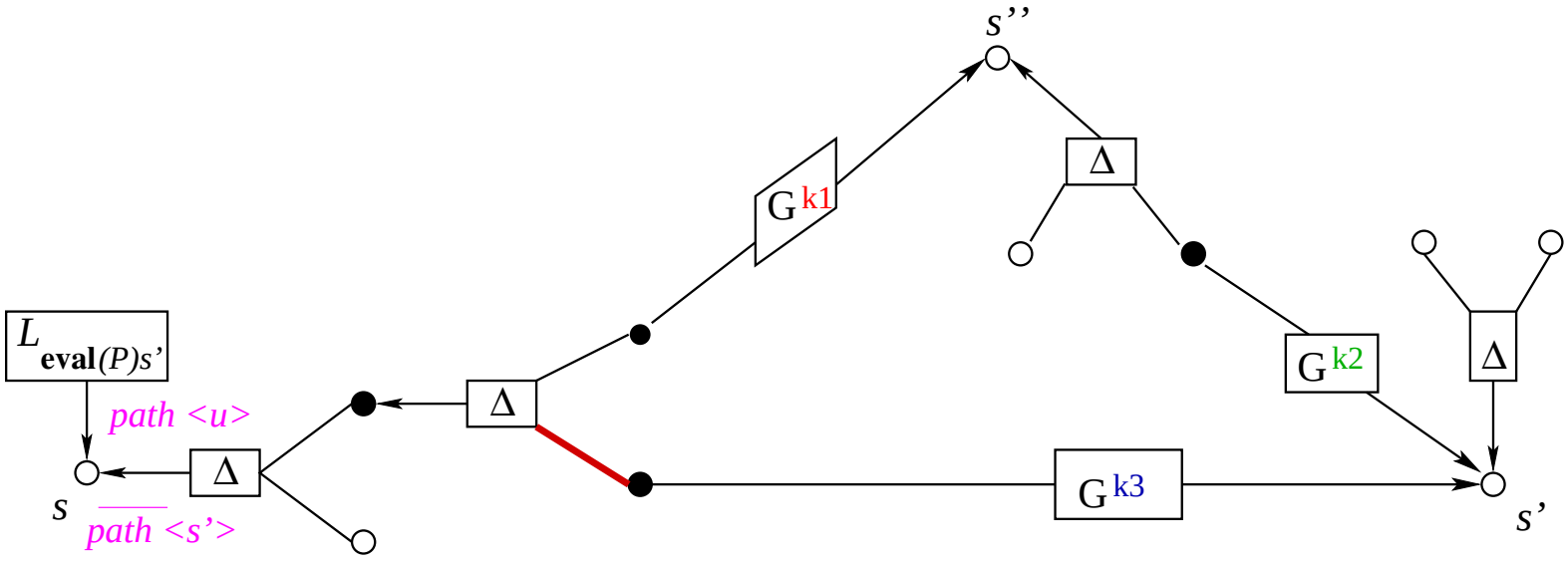
abc



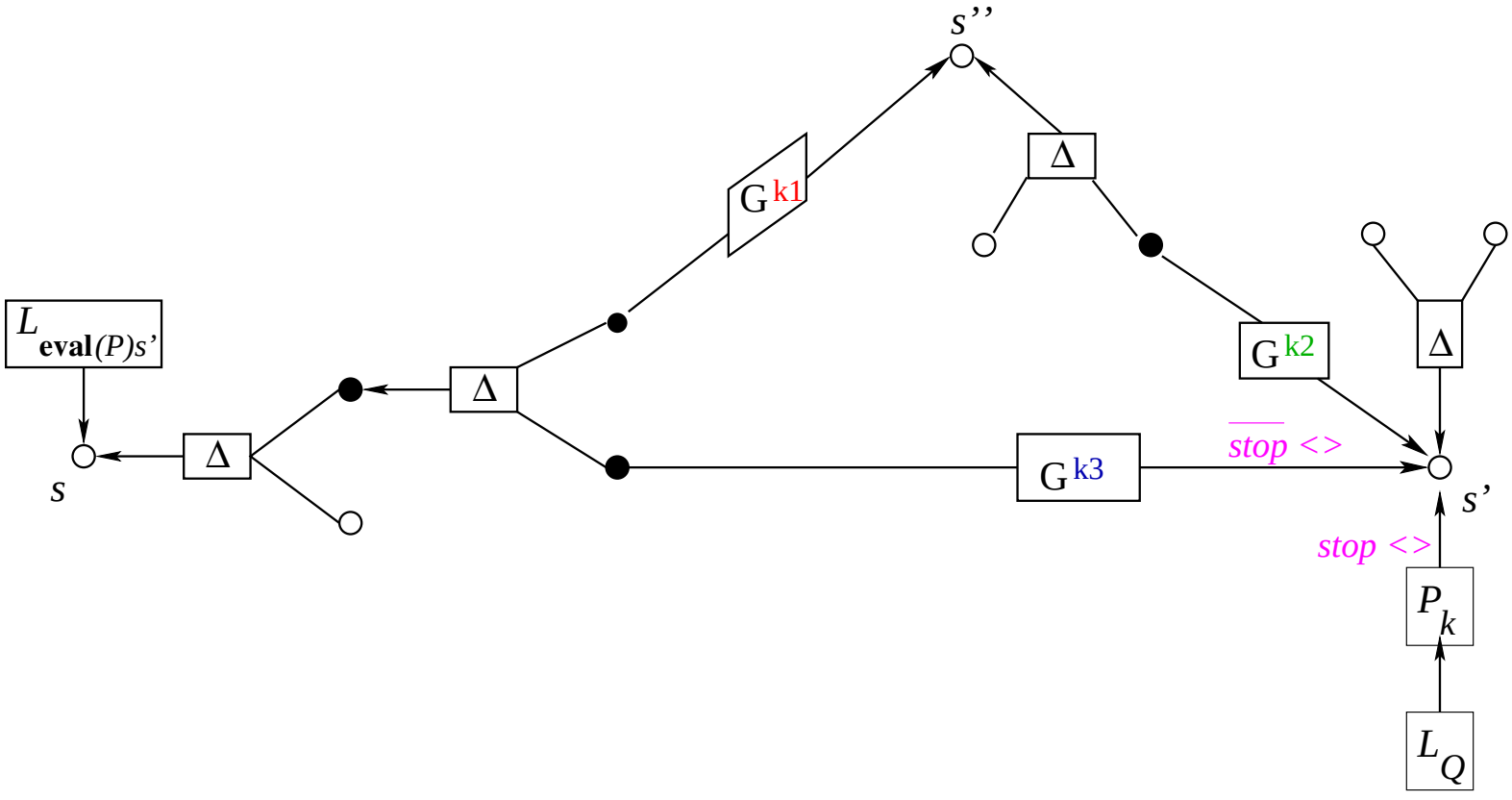
abc



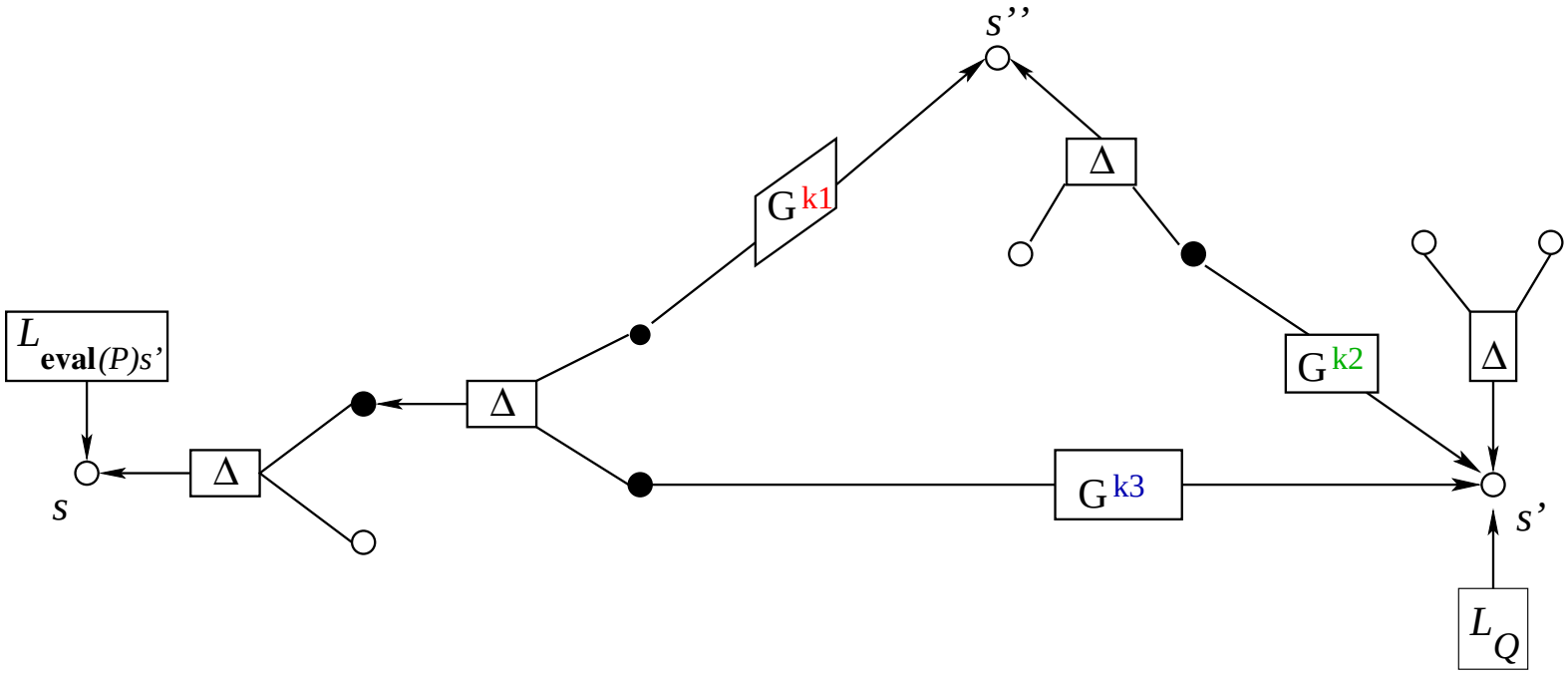
abc



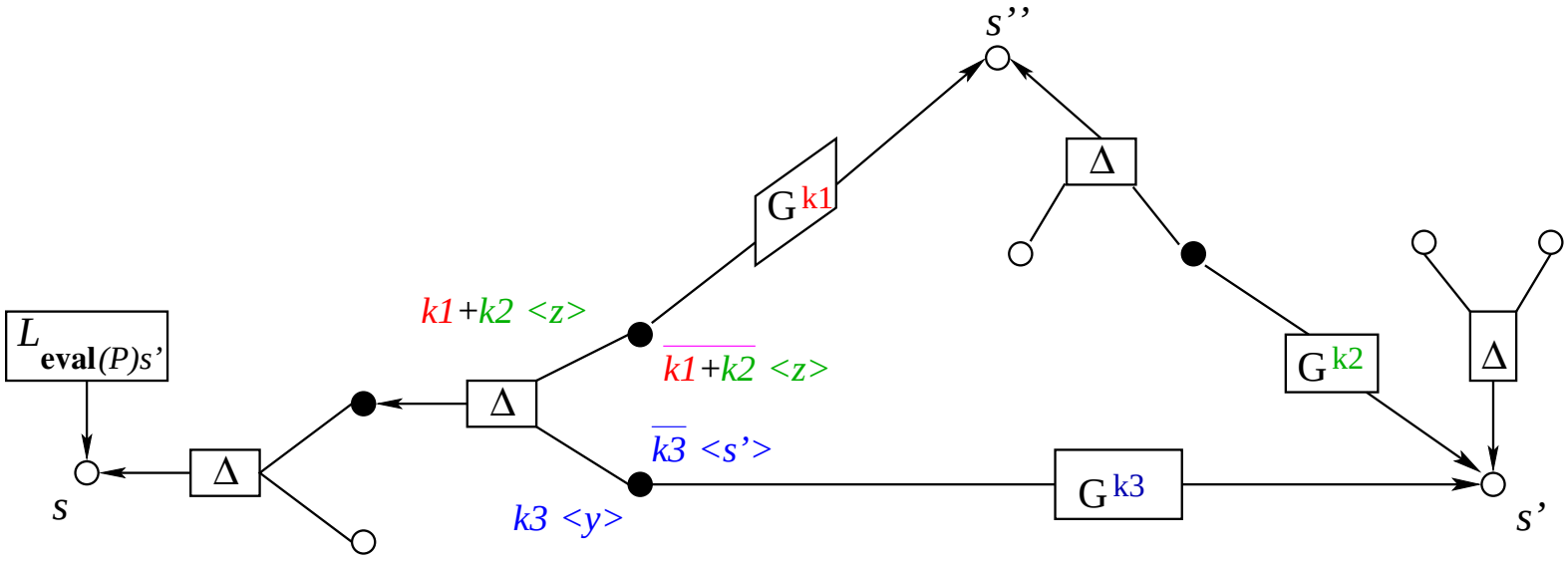
abc



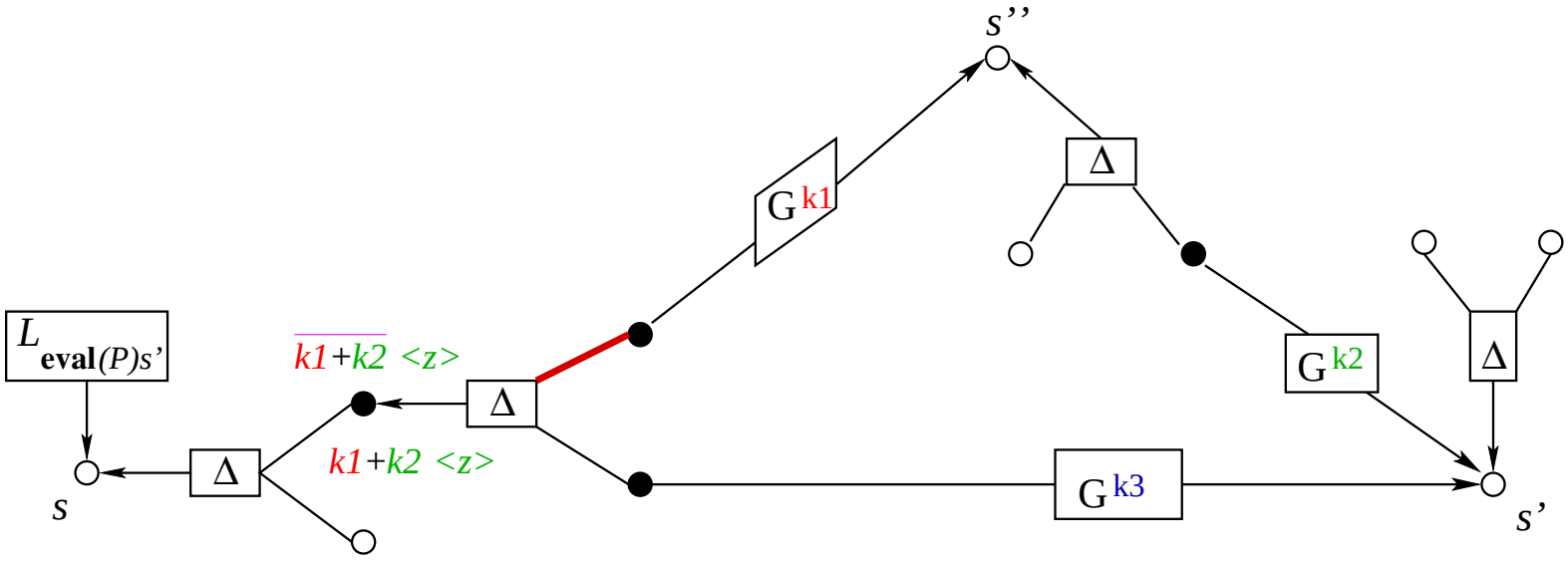
abc



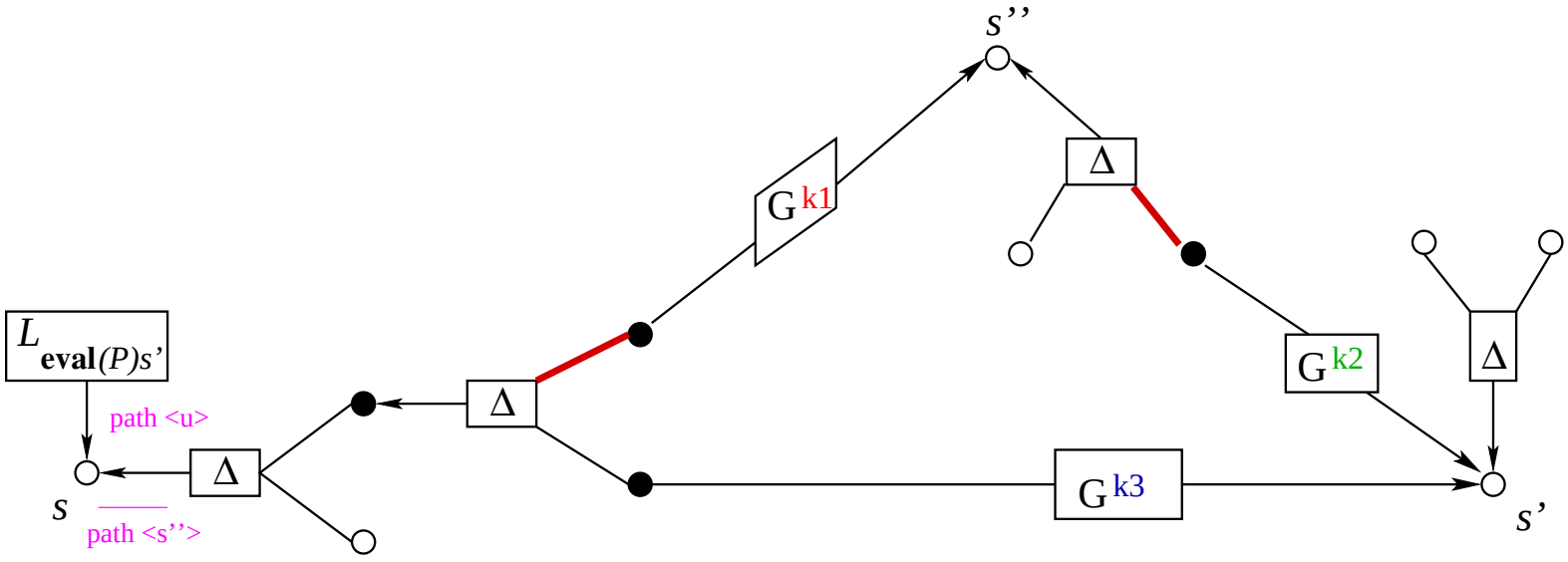
abc



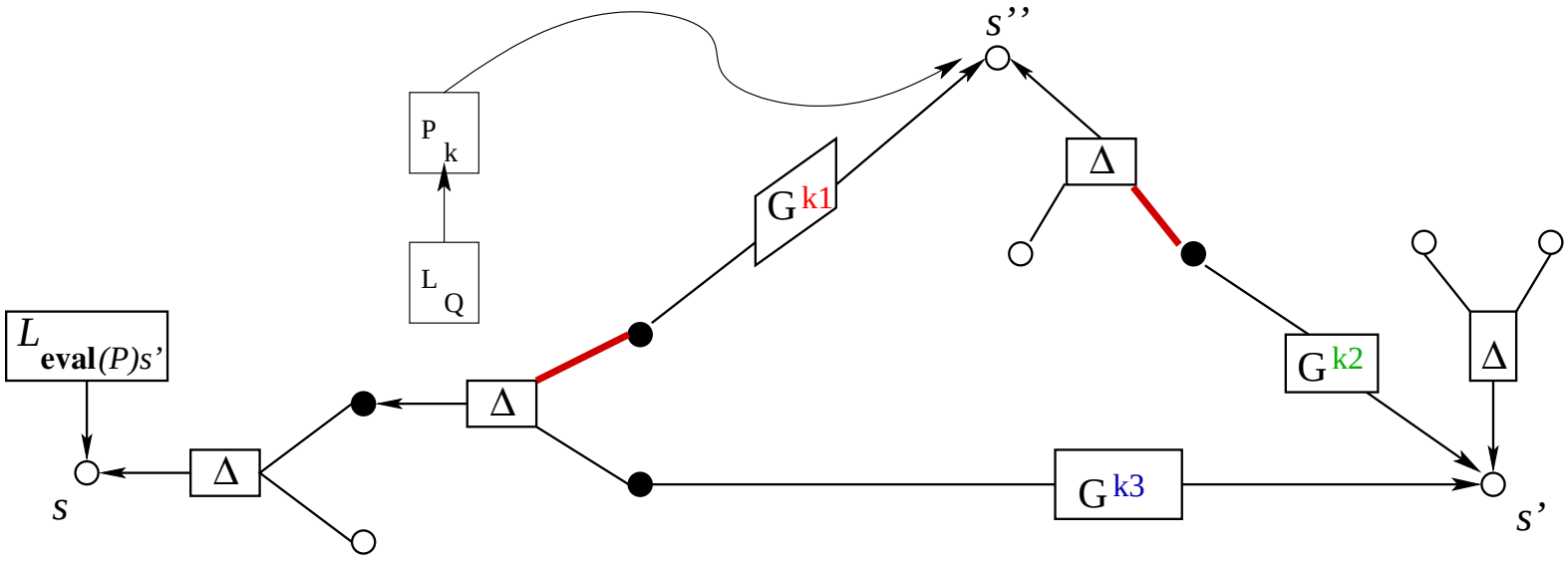
abc



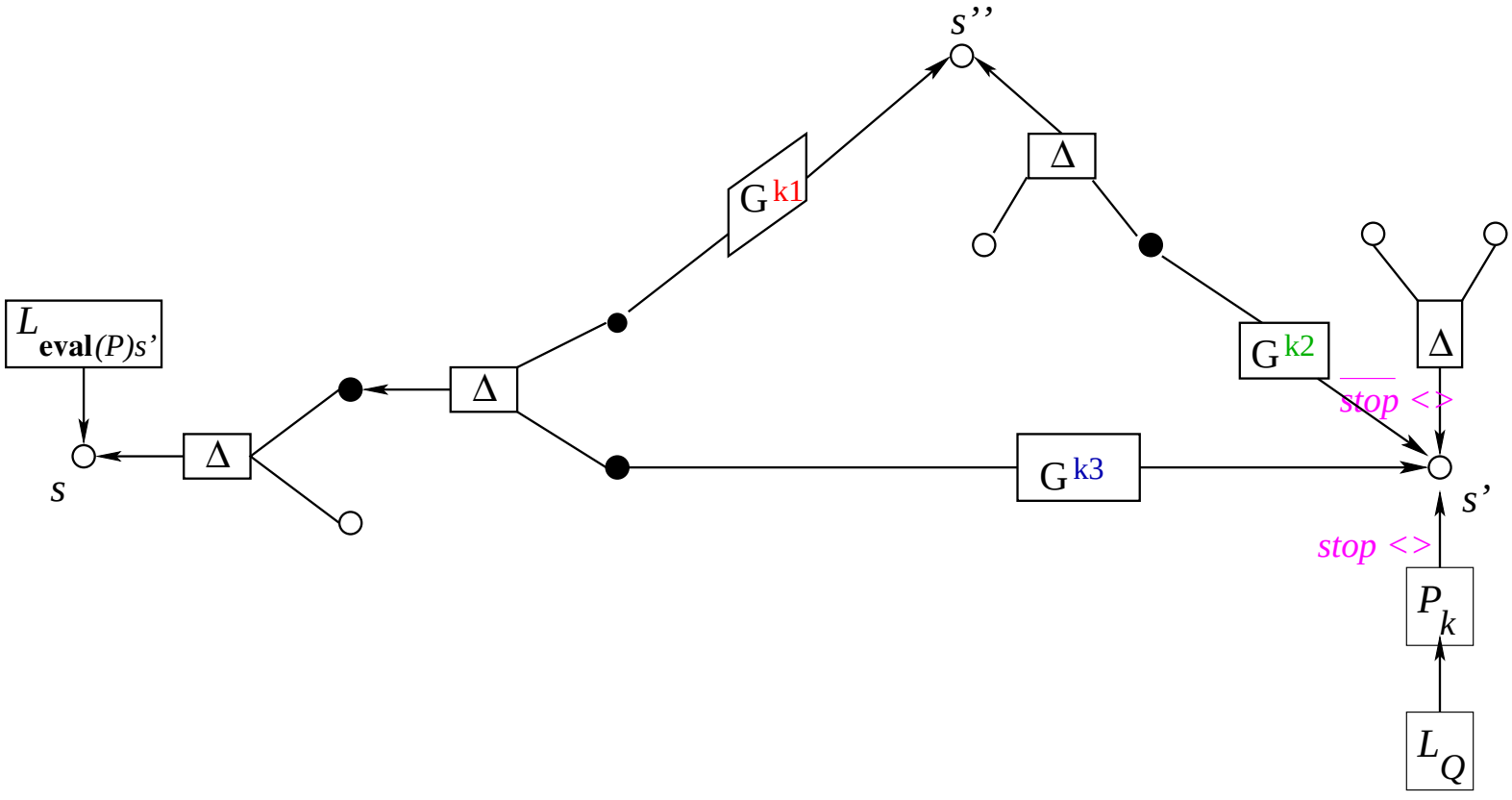
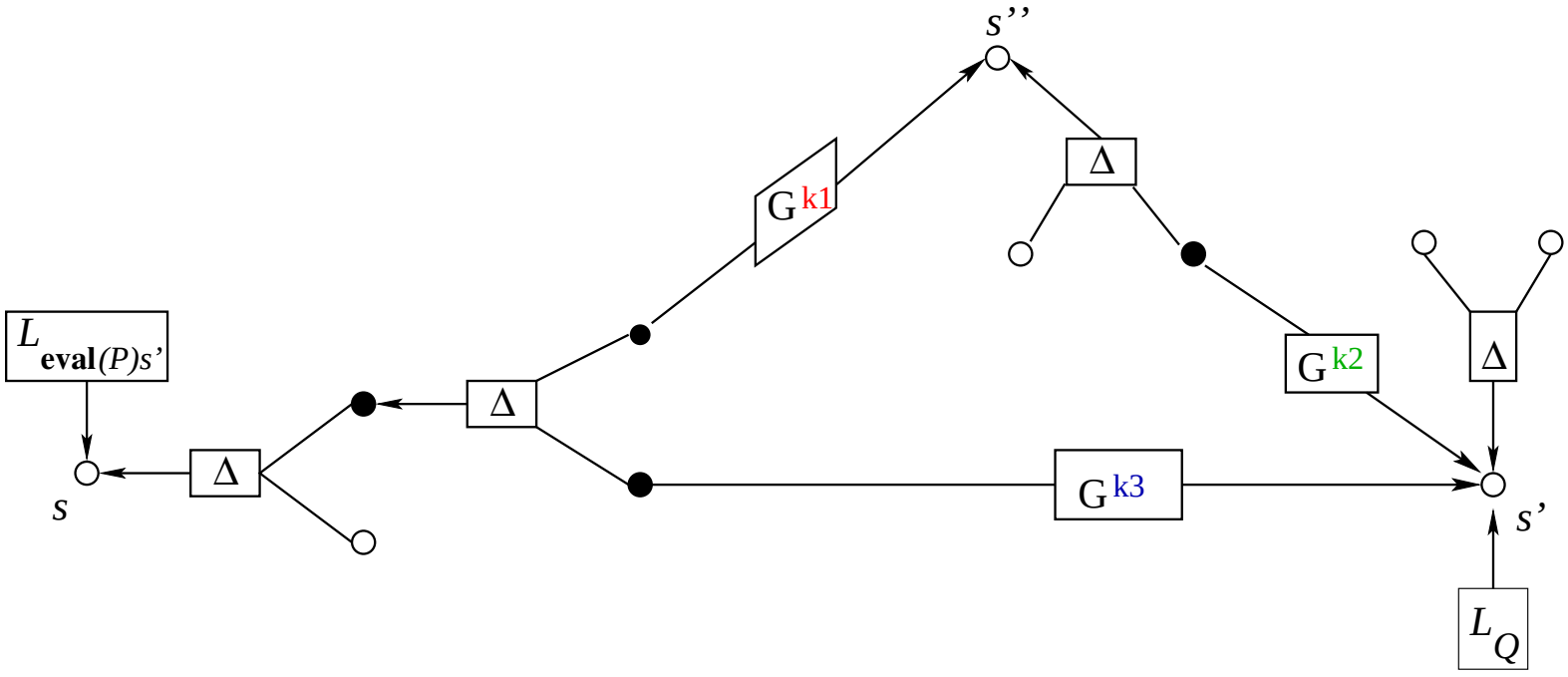
abc



abc



abc



abc