PN^2: An Elementary Model for Design and Analysis of Multi-agent Systems

Kunihiko Hiraishi

School of Information Science
Japan Advanced Institute of Science and Technology

PN ^{2}: “Petri nets in a Petri net”

- Aim: To give a mathematical foundation for the design and analysis of *multi-agent systems* (MAS) by means of a Petri-net-based model.
- *PN* ^{2} is a formalisim based on place/transition nets (P/T nets), which is the elementary class of Petri nets.
- The main difference between *PN* ^{2}s and P/T nets is that each token, representing an agent, is also a P/T net.

Analysis of systems by Petri nets

- Mathematical analysis using static structure:
  - State equation and algebraic analysis
  - Reachability/liveness conditions for subclasses of Petri nets
  - Invariant analysis
  - Model reduction techniques
- Efficient techniques for state space exploration:
  - Partial order method
  - Reduced state space representation by unfolding
  - Efficient model checking techniques using BDDs
- Performance evaluation:
  - Stochastic Petri nets, GSPN, ...

Development of Petri-net models

Models	Meaning of a token
Place/Transition Nets	Existence of a resource
High-level Petri nets (e.g. *Coloured Petri Nets*)	A data value of an arbitrary complex data type
Object Petri Nets	An object (data values and procedures)

Object Petri nets

- OBJSA Nets [E. Battiston et al. 88],
- POTs, POPs [J. Engelfriet et al. 90],
- CO-OPN, CO-OPN2 [D. Buch et al. 93],
- PNTalk [M. Ceska et al. 94],
- Cooperative Nets [C. Sibertin-Blanc et al. 94],
- Object Petri Nets [C. Lakos 95],
- OB(PN)^2 [J. Lilius 96],
- Elementary Object Systems [R. Valk 88],
- Reference Nets [O. Kummar et al. 99], ...

Many of these models are based on high-level Petri nets.

Analysis and expressiveness

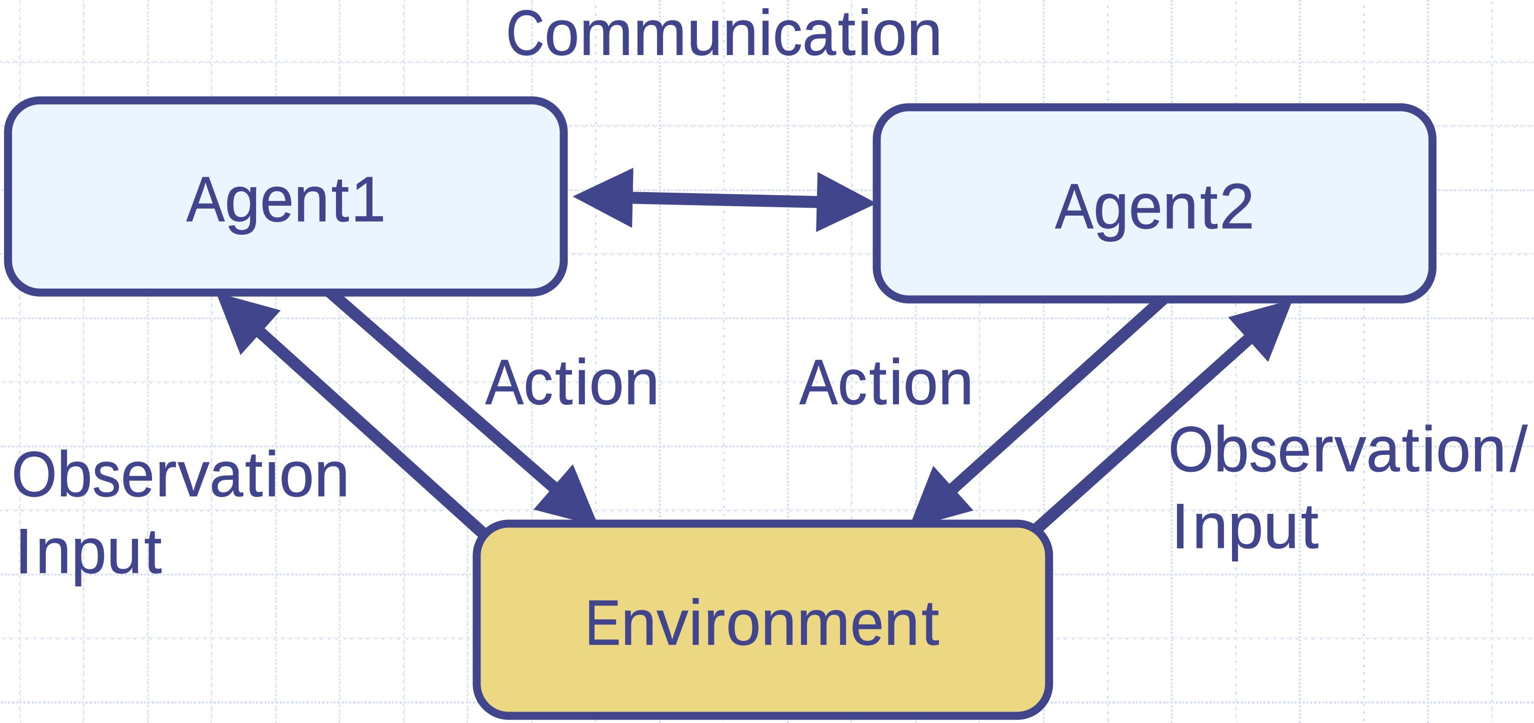
Models for mathematical analysis	*P/T nets*	*PN* ^{2}
	↕	↕
Description Language/ Simulation model	*Coloured Petri nets*	*Object Petri nets*
	Non-Object Systems	Object/Agent Systems

Concept of agents

- “ An agent is a computer system that is situated in some *environment*, and that is capable of *autonomous* action in this environment in order to meet its *design objectives* ”.

  (G. Weiss (ed.), “Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence”, The MIT Press, 1999).

Concept of agents



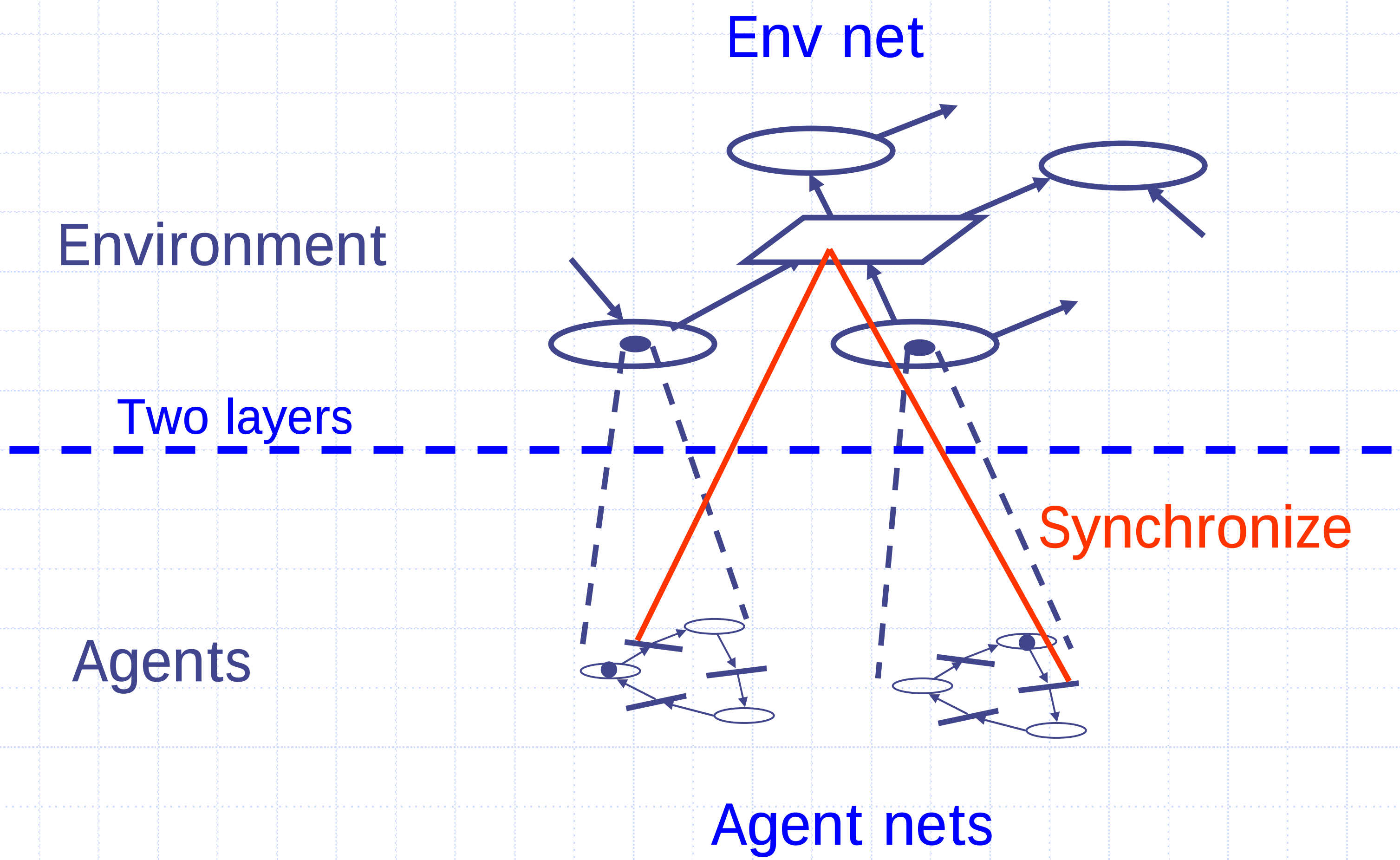
Environment

- An agent net has at best partial control over the environment.
- Static/dynamic environment.:
  - *A static environment* is one that can be assumed to remain unchanged except by the performance of actions by the agent.
  - *A dynamic environment* is one that has other processes operating on it, and which hence changes in ways beyond the agent's control. e.g. physical world (temperature, wind strength, …), existence of other AGVs in AGV system.
- Not all actions can be performed in all situations, i.e., an environment restricts the behavior of agents.

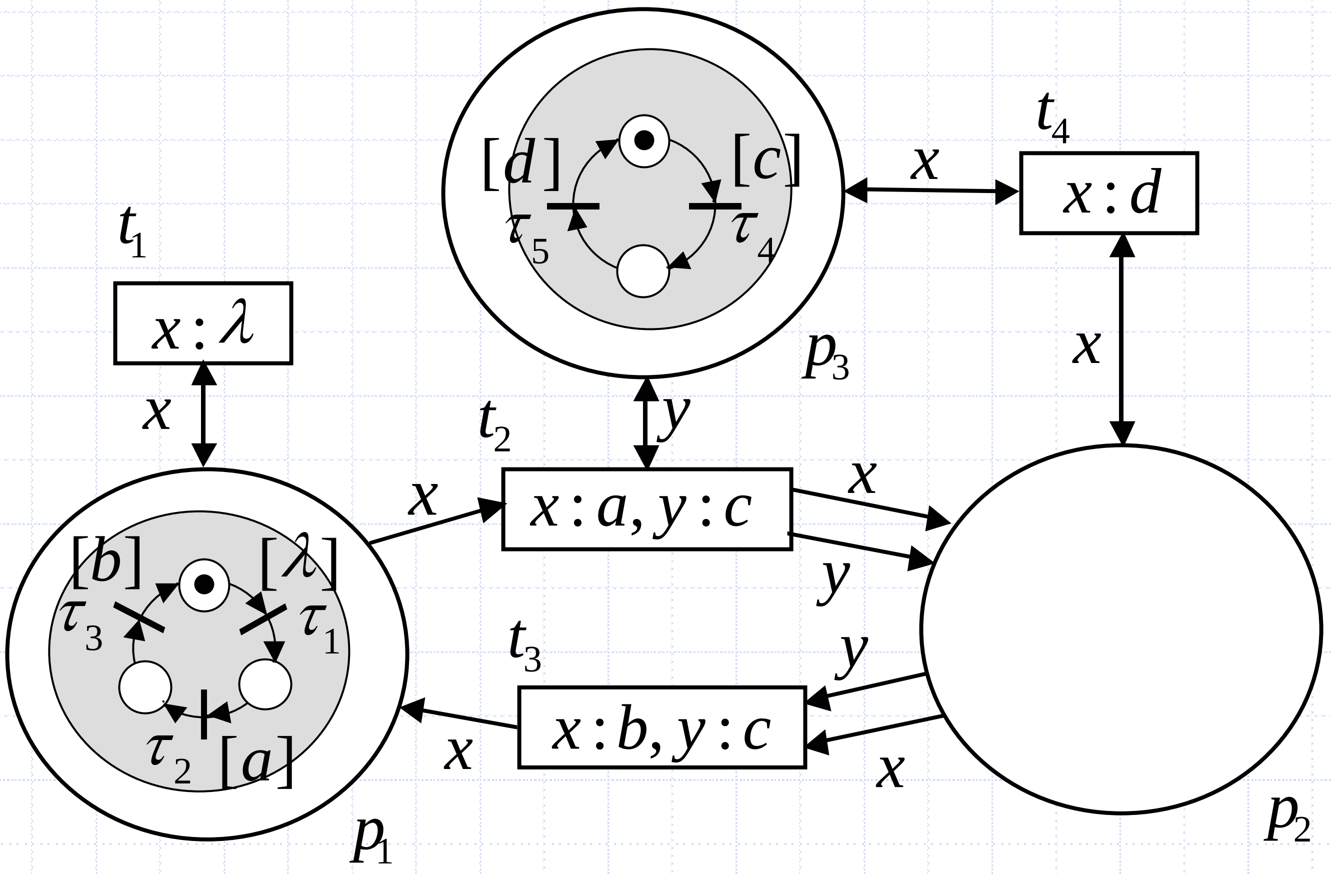
Essential properties of MAS that can be modeled in *PN*^{2}

- Environment in which agents behave
- Agent actions/observations restricted by the environment
- Mobility of agents (each agent can change its location in the environment.)
- Duplication of agents (each agent can be duplicated by its action.)
- Agent-agent communication
- Autonomous actions (actions performed without any interaction with outside)

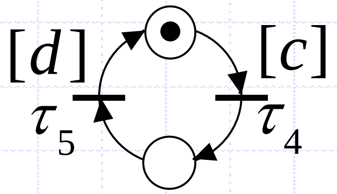
How to model MAS by PN^2

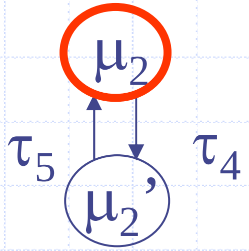


Synchronization Mechanism

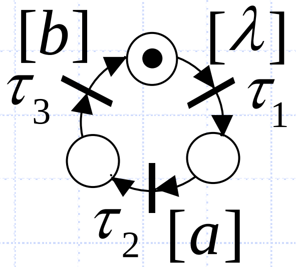


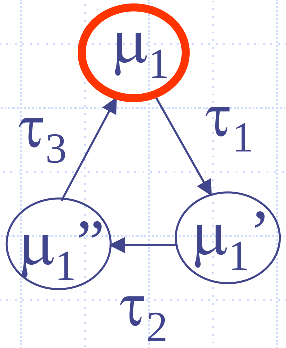
Synchronization Mechanism



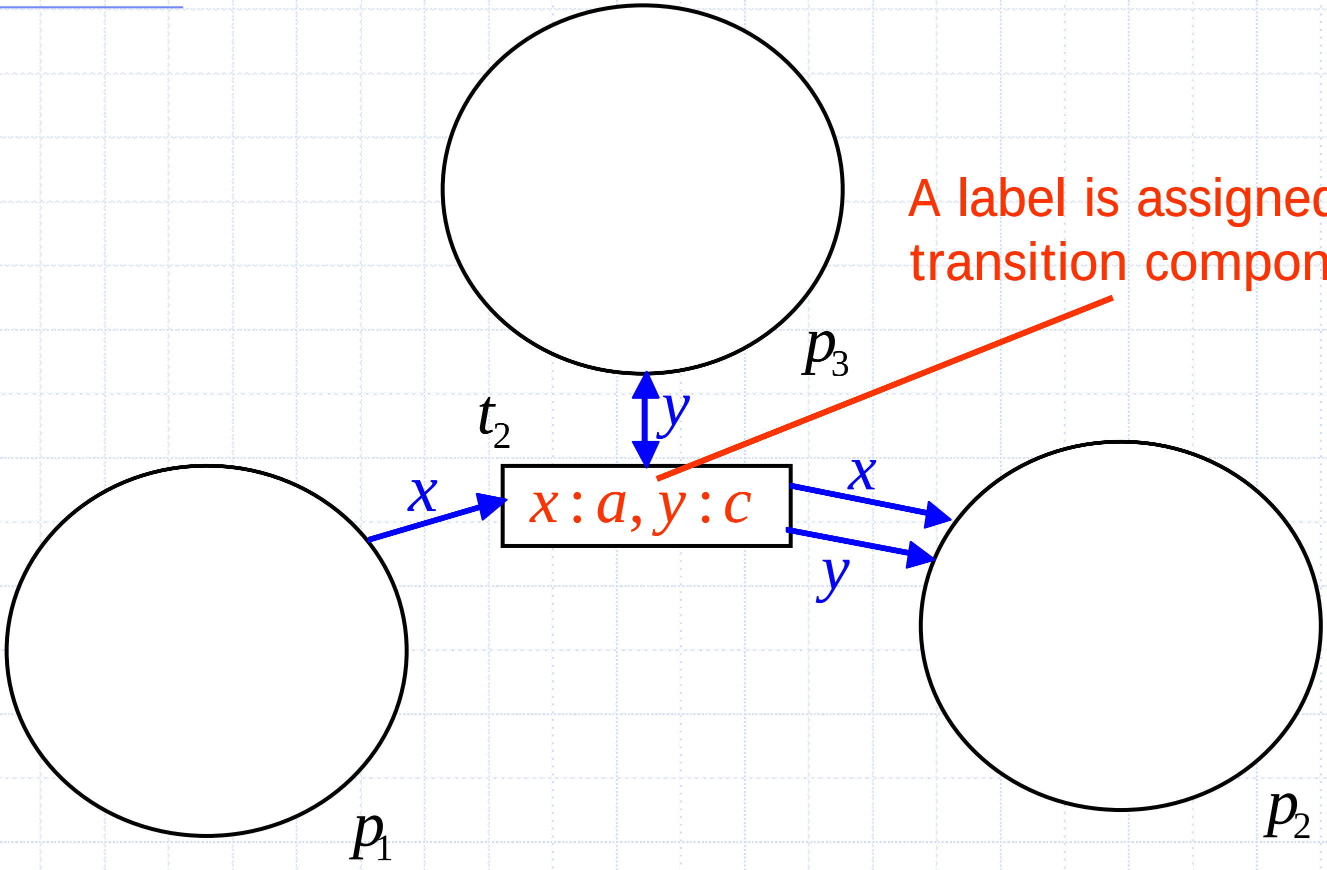


Each agent net is a labeled P/T net.





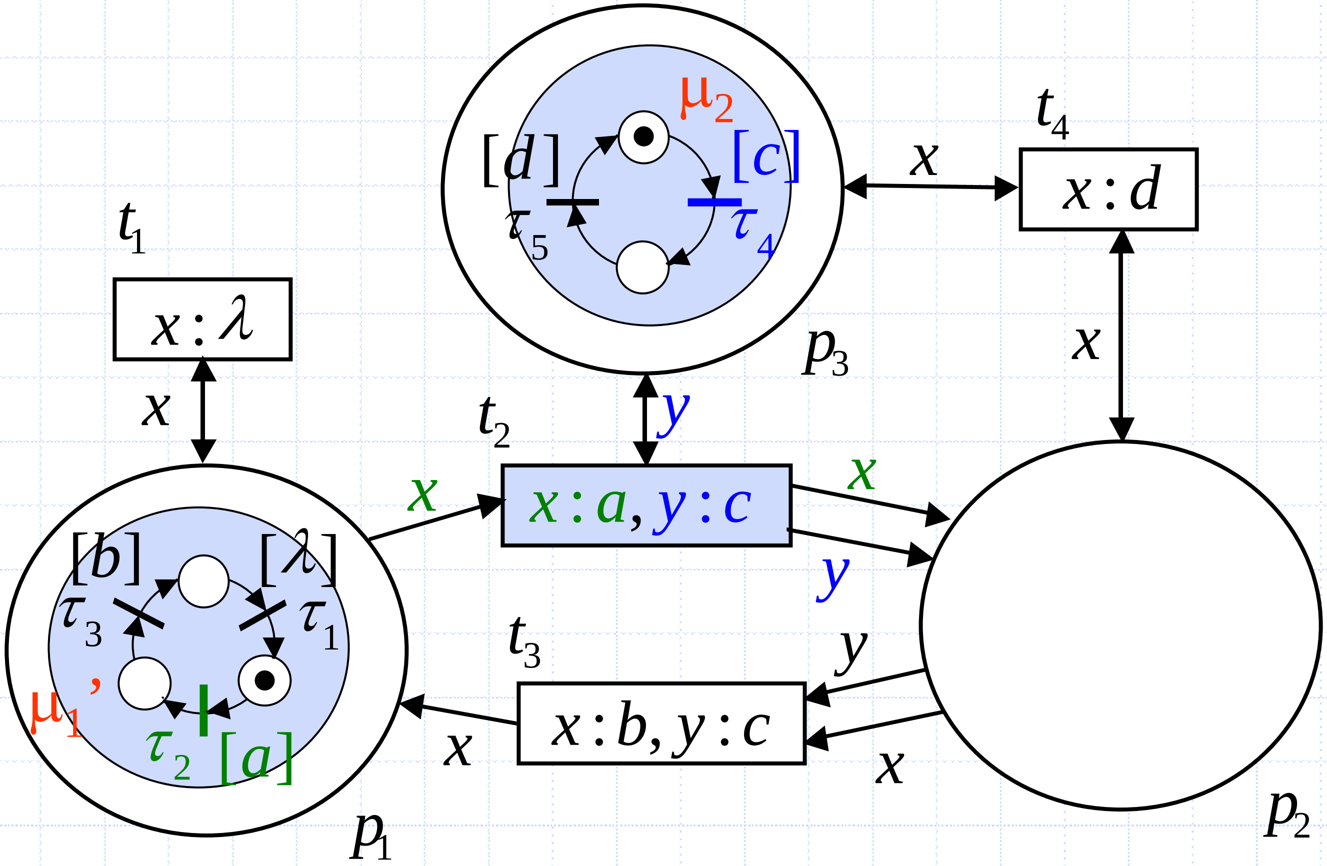
Synchronization Mechanism



A label is assigned to each transition component.

Transition t_2 consists of two *transition components* indicated by x and y.

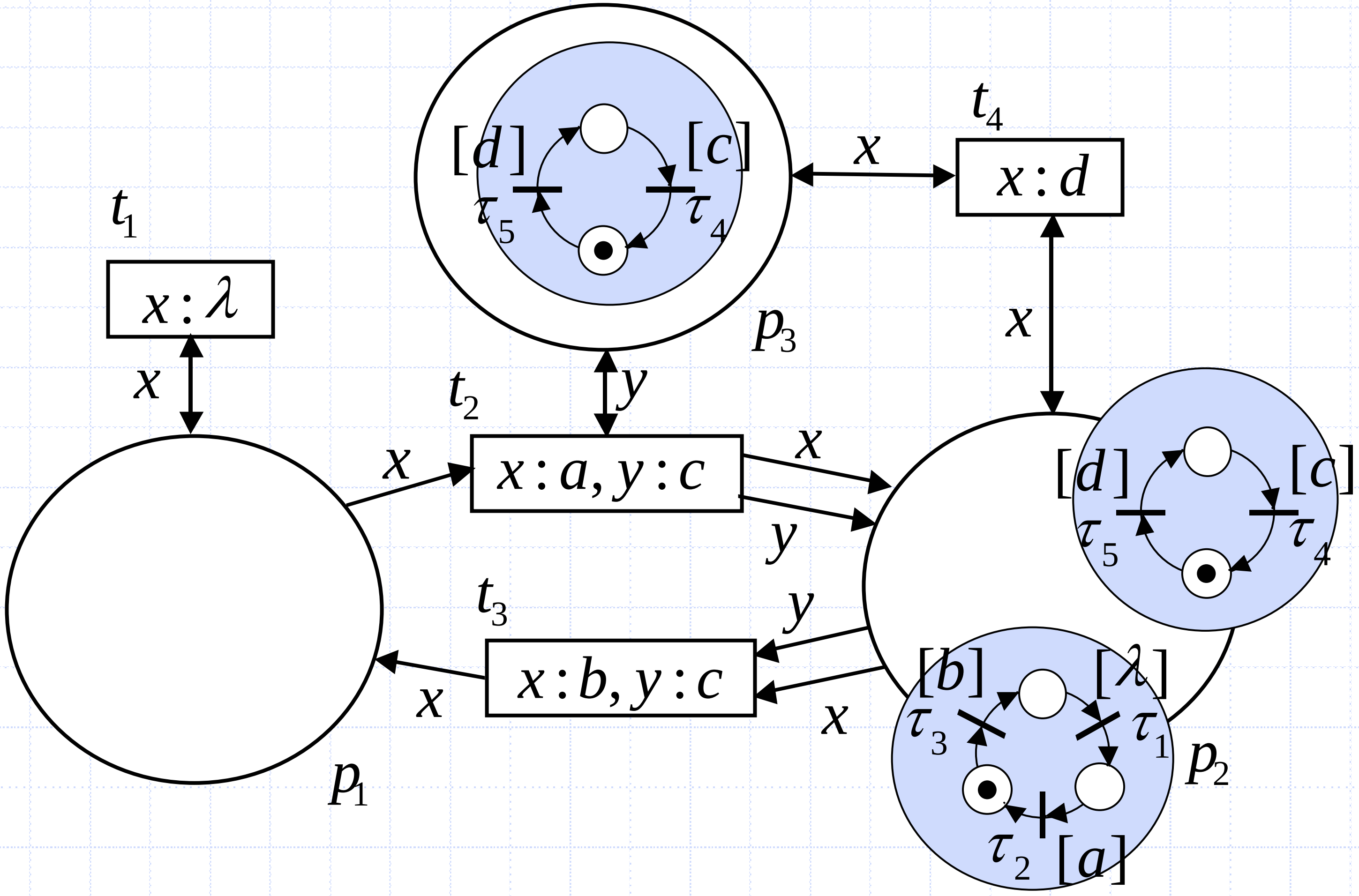
Synchronization Mechanism



Transition binding : $x \mapsto \tau_2$ in μ_1'; $y \mapsto \tau_4$ in μ_2

(τ_2 is enabled in μ_1'; τ_4 is enabled in μ_2)

Synchronization Mechanism



*PN*² is defined by sets and functions

- A *PN*² is a 9-tuple $(P, \Sigma, T, E, {}^{\bullet}{}', {}'^{\bullet}, {}'_{\Sigma}, {}'_{T}, s_0)$.
  - P is a finite set of places;
  - Σ is a finite set of symbols;
  - T is a finite set of transitions;
  - E is a finite set of transition components;
  - ${}^{\bullet}{}' : E \rightarrow \mathbf{M}_P$, ${}'^{\bullet} : \mathrm{E} \rightarrow \mathbf{M}_P$ (inputs and outputs of each transition component);
  - ${}'_{\Sigma} : E \rightarrow \Sigma$ (the label assigned to each transition component);
  - ${}'_{T} : E \rightarrow T$ (the transition that each transition component belongs to);
  - s_0 is the initial configuration, where a configuration is a mapping $s : P \rightarrow \mathbf{M}_{PT\Sigma}$.

PN^2 is defined by sets and functions

- A transition binding (b, w) is enabled in state s if for each place $p \in P$:

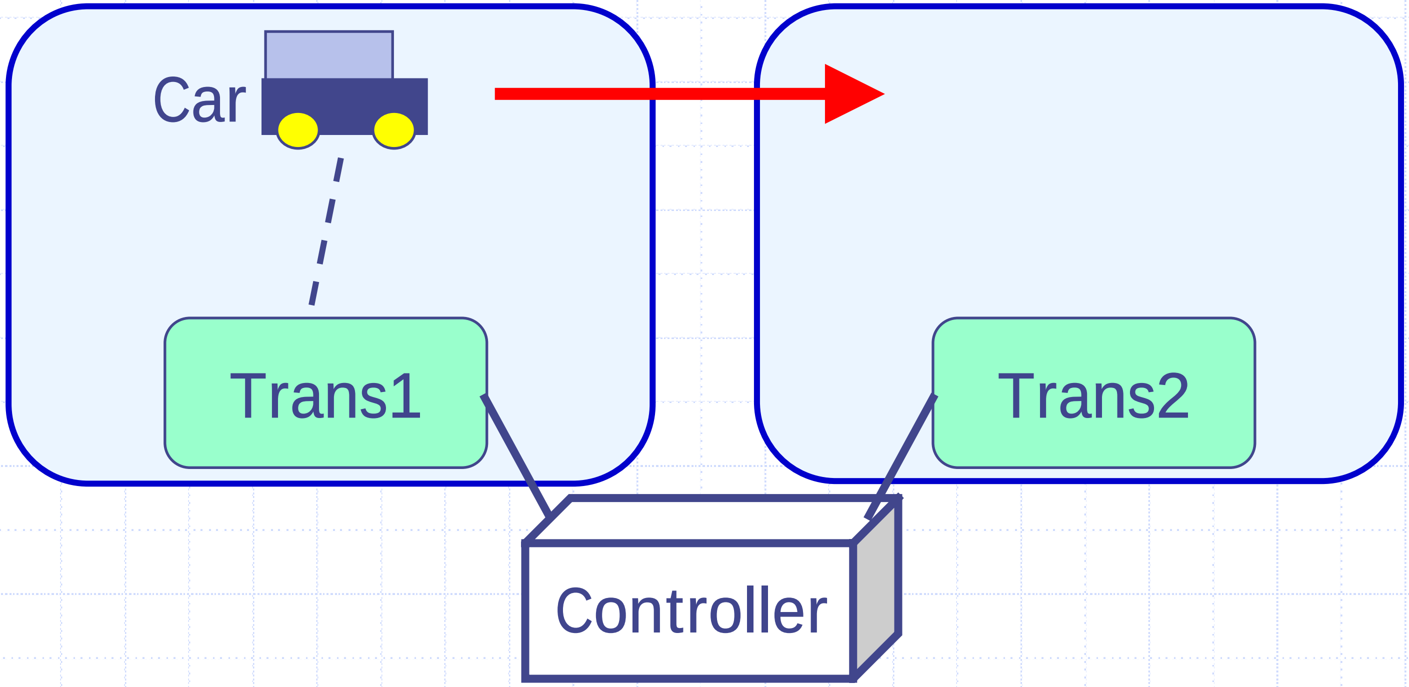
$$\sum_{e \in E_t} \left({}^{\bullet}e\right)(p) \# b(e) \subseteq s(p)$$

- An occurrence of a transition binding (b, w) changes the state to s' such that for each place $p \in P$:

$$s'(p) = s(p) - \sum_{e \in E_t} \left({}^{\bullet}e\right)(p) \# b(e) + \sum_{e \in E_t} \left(e^{\bullet}\right)(p) \# \delta(b(e), w(e))$$

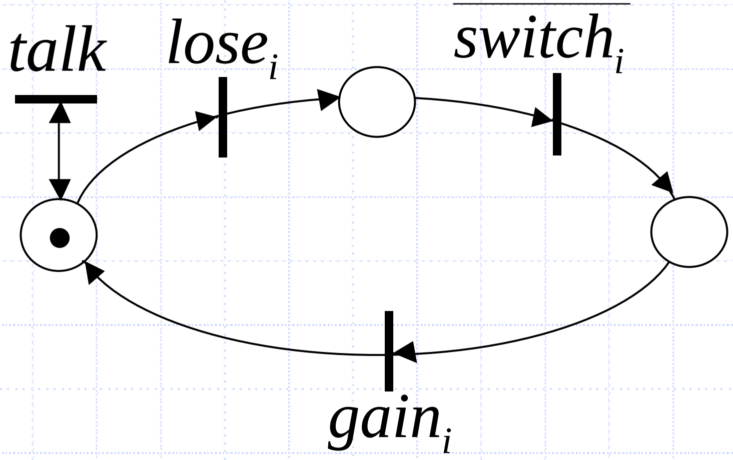
Example: π-calculus and *PN* 2

- There are vehicles on the move, each connected by a unique wavelength to a single transmitter. The transmitters all have fixed connections to a central control. On some event (e.g., signal fading) a vehicle may be switched to another transmitter.



Example: π-calculus and *PN* 2

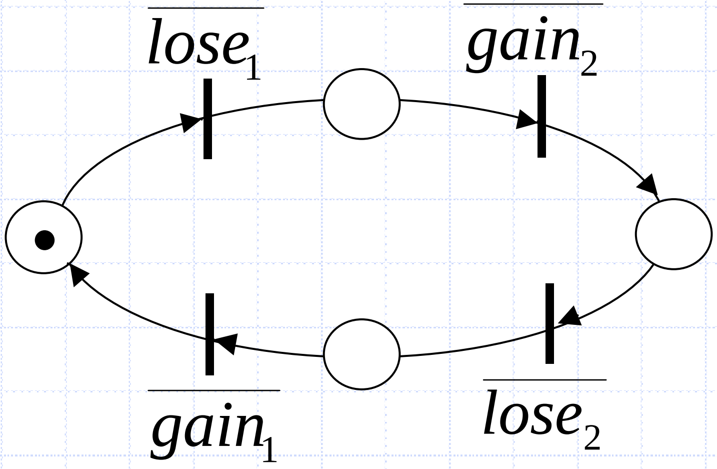
$$Trans(talk, switch, gain, lose) \equiv$$
$$talk.Trans\langle talk, switch, gain, lose\rangle$$
$$+\, lose(t,s).\overline{switch}\langle t,s\rangle.Idtrans\langle gain, lose\rangle.$$
$$Idtrans(gain, lose) \equiv gain(t,s).Trans\langle t,s,gain,lose\rangle.$$



Agent net *Trans$_i$*

Example: π-calculus and *PN* ^{2}

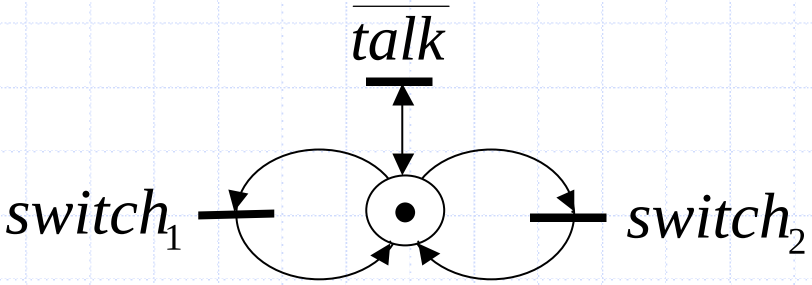
$$Control_1 \equiv \overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle.Control_2.$$
$$Control_2 \equiv \overline{lose_2}\langle talk_1, switch_1\rangle.\overline{gain_1}\langle talk_1, switch_1\rangle.Control_1.$$



Agent net *Control*

Example: π-calculus and *PN*²

$$Car(talk, switch) \equiv \overline{talk}.Car\langle talk, switch\rangle + switch(t,s).Car\langle t,s\rangle.$$

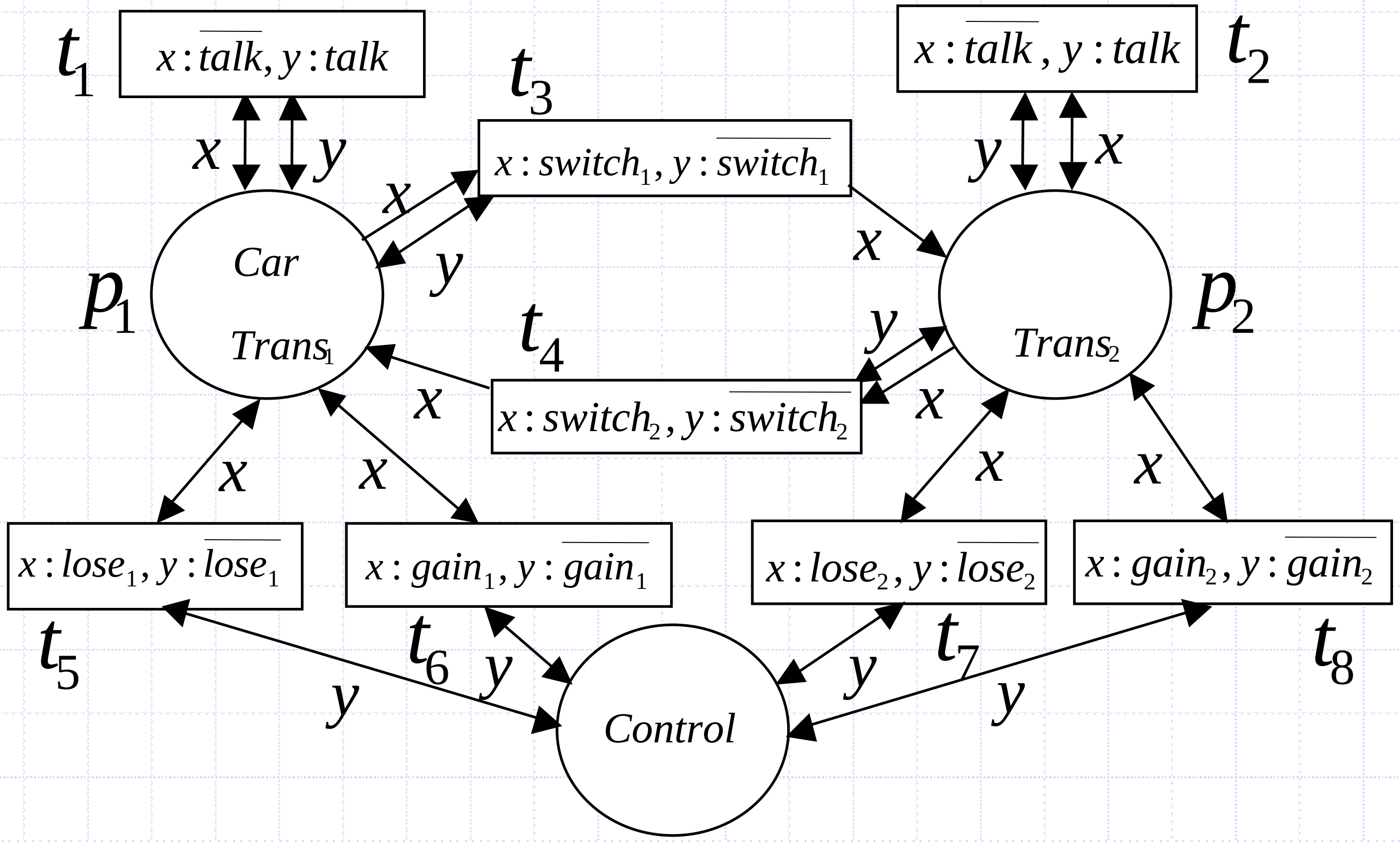


Agent net *Car*

Example: π-calculus and *PN* 2

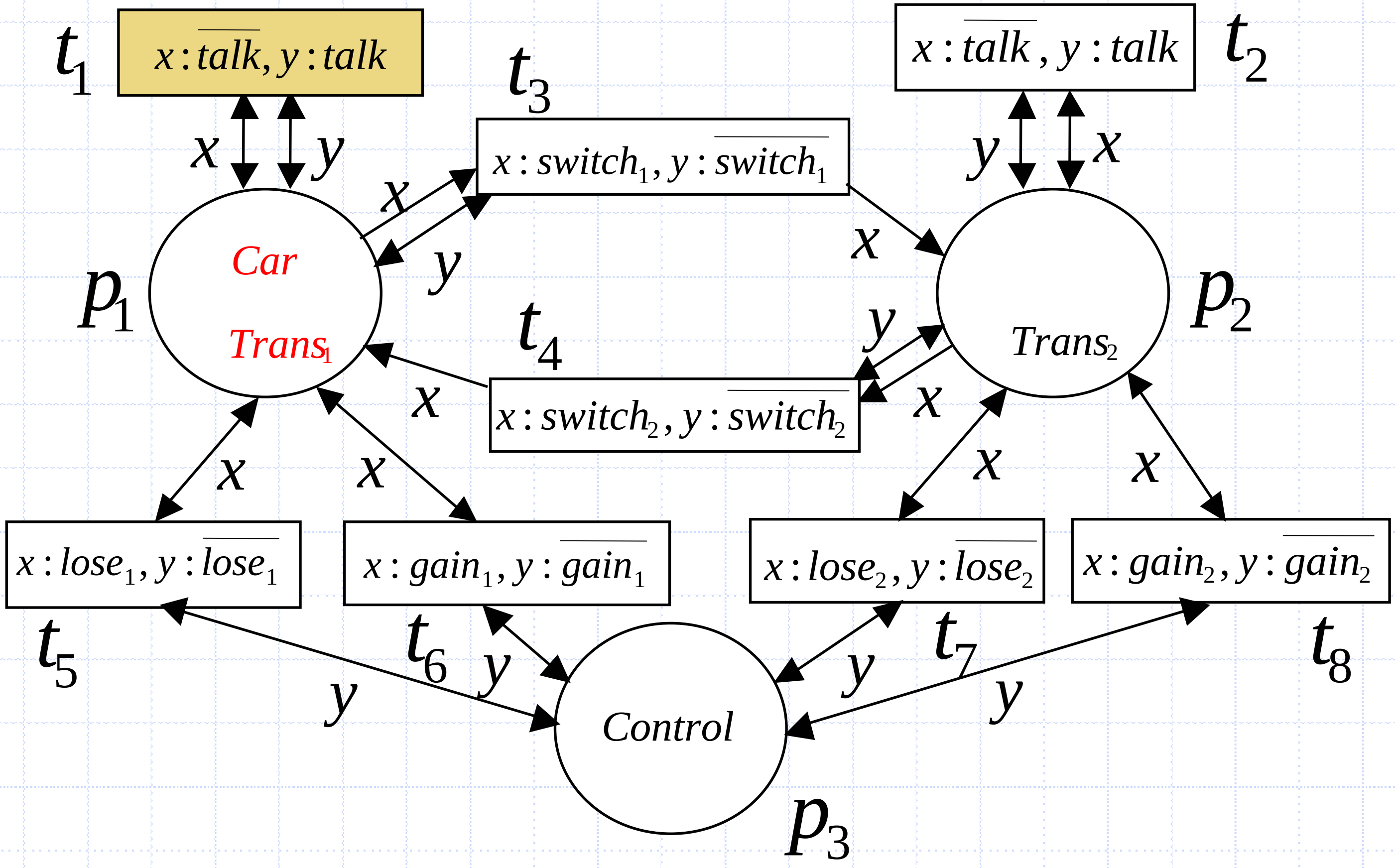
$$
\begin{aligned}
System \equiv \mathbf{new}\ & talk_1, switch_1, gain_1, lose_1, \\
& talk_2, switch_2, gain_2, lose_2 \\
& (Car\langle talk_1, switch_1\rangle \mid \\
& \ Trans\langle talk_1, switch_1, gain_1, lose_1\rangle \mid \\
& \ Idtrans\langle gain_2, lose_2\rangle \mid \\
& \ Control_1).
\end{aligned}
$$

Example: π-calculus and PN^2

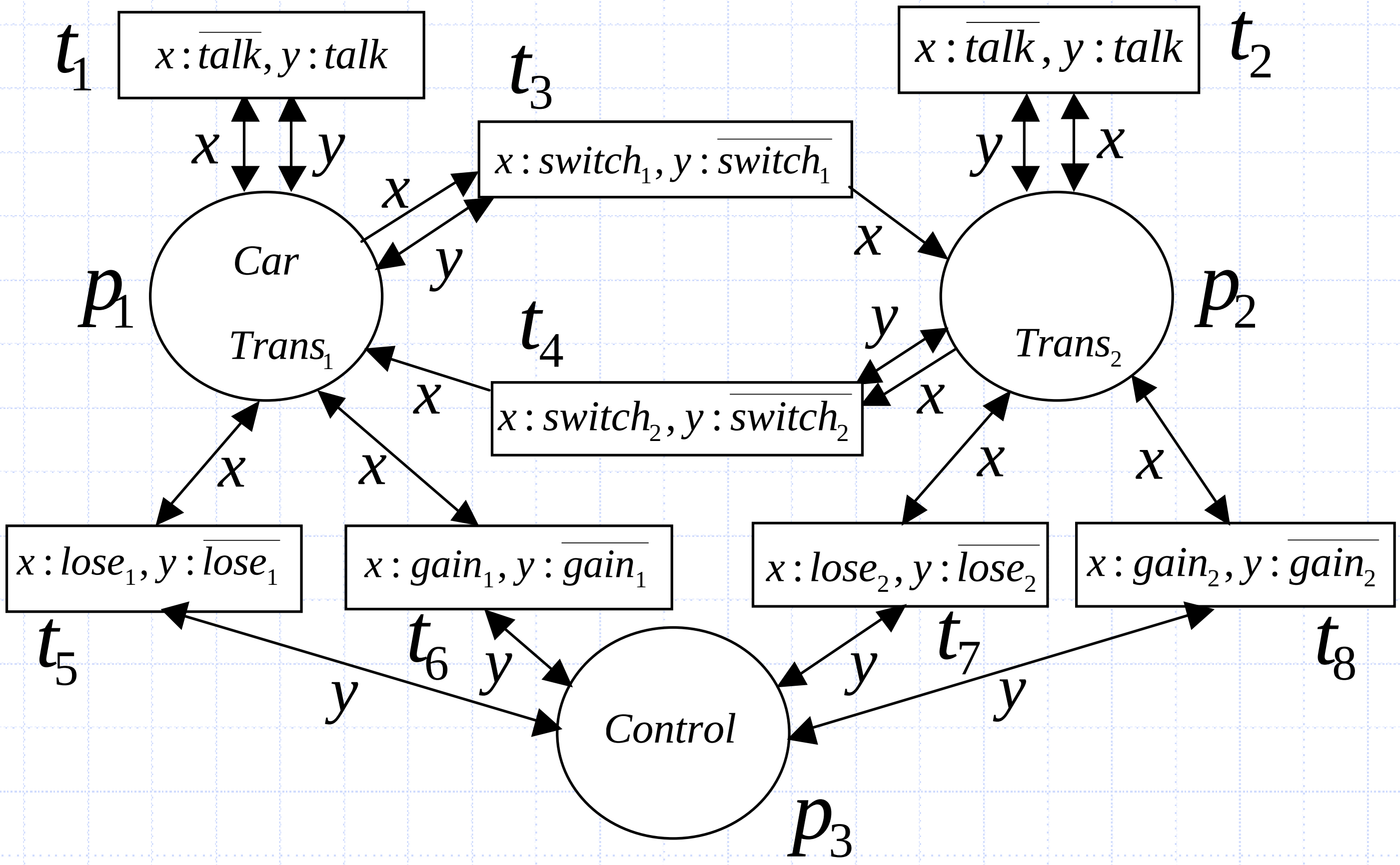


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
Car
$Trans_1$
p_2
$Trans_2$
$x:lose_1, y:\overline{lose_1}$
$x:gain_1, y:\overline{gain_1}$
$x:lose_2, y:\overline{lose_2}$
$x:gain_2, y:\overline{gain_2}$
t_5
t_6
t_7
t_8
Control
x
y

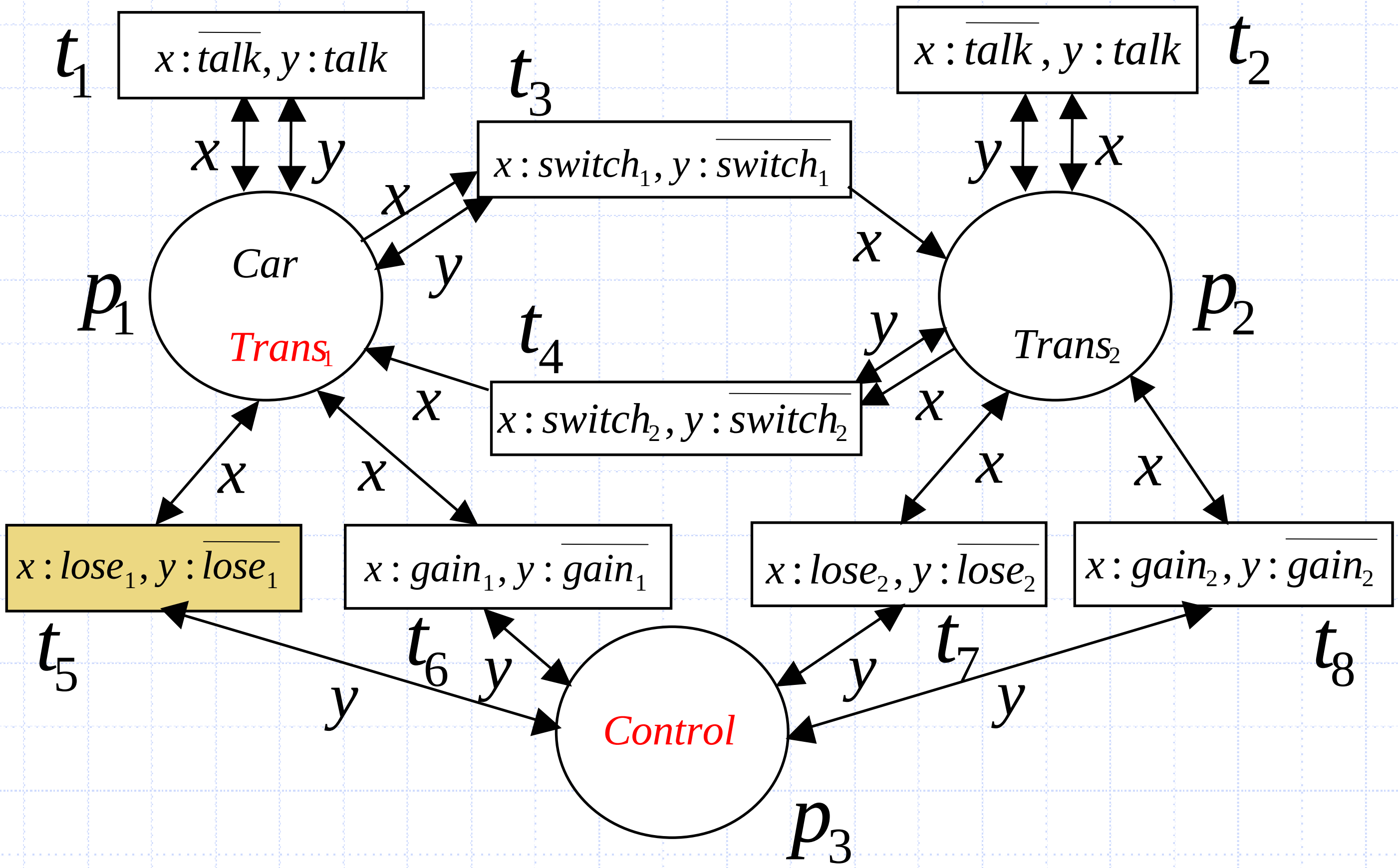
Env net and agent nets

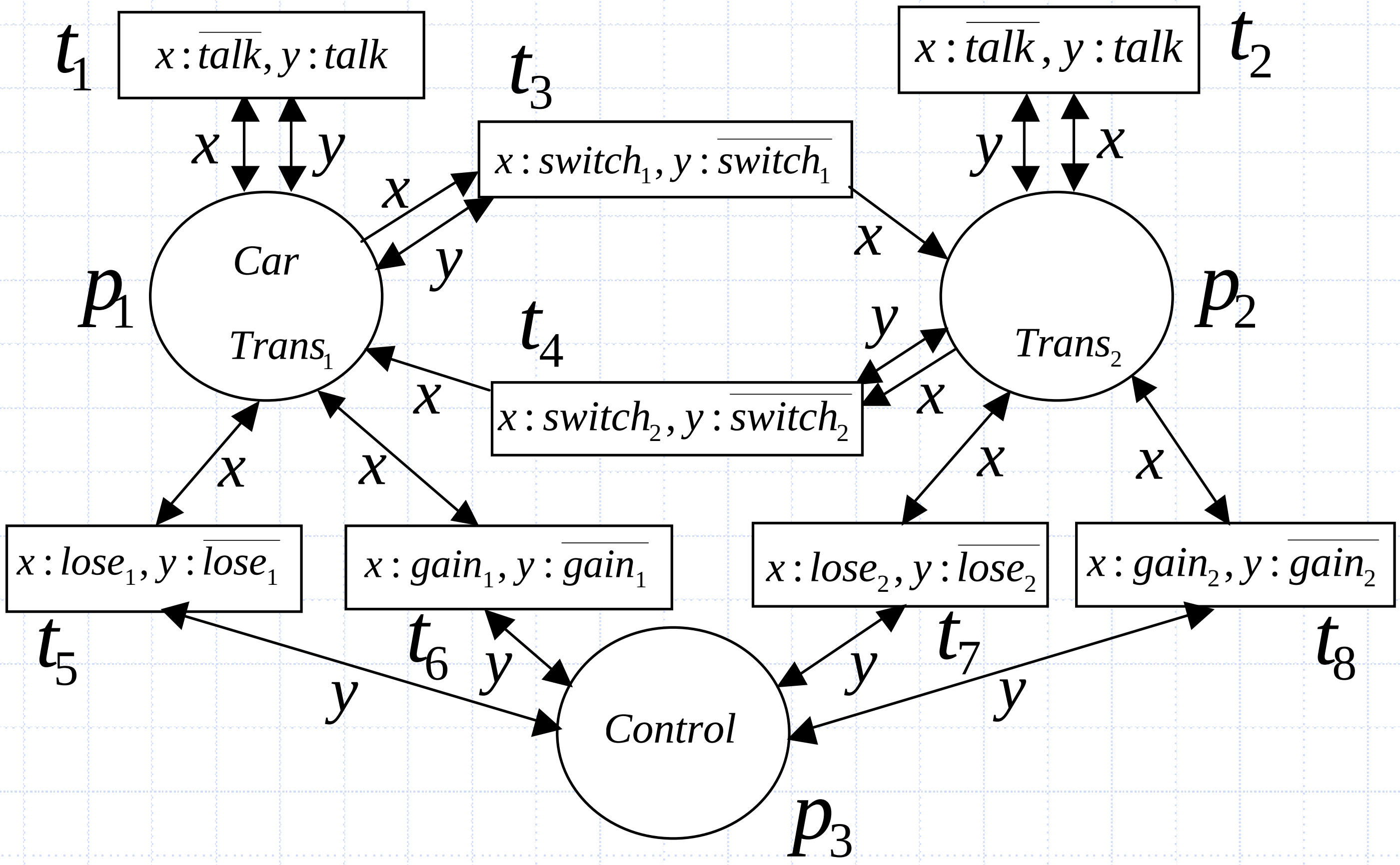


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
Car
$Trans_1$
p_2
$Trans_2$
p_3
Control
t_5
$x:lose_1, y:\overline{lose_1}$
t_6
$x:gain_1, y:\overline{gain_1}$
t_7
$x:lose_2, y:\overline{lose_2}$
t_8
$x:gain_2, y:\overline{gain_2}$
x
y

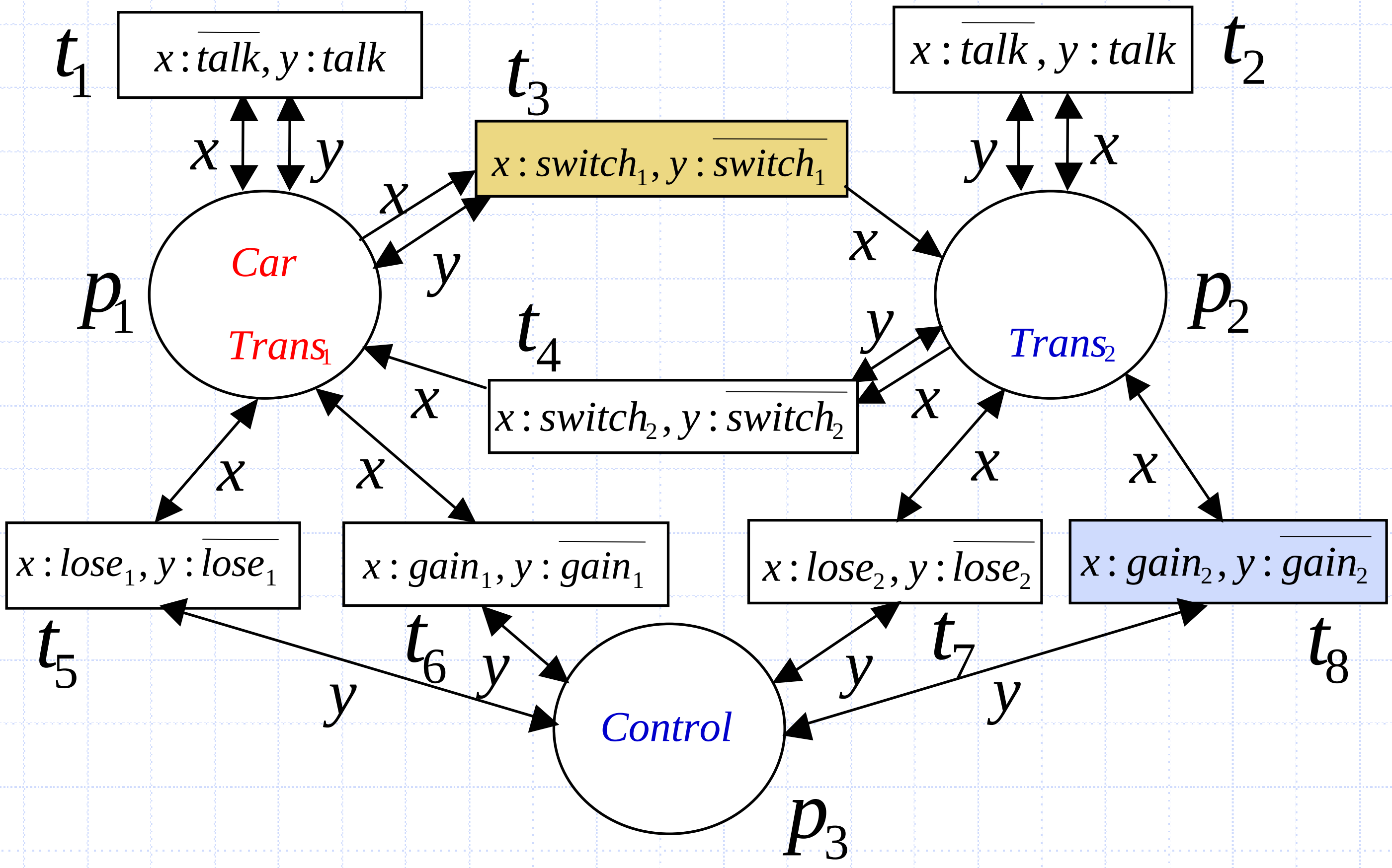


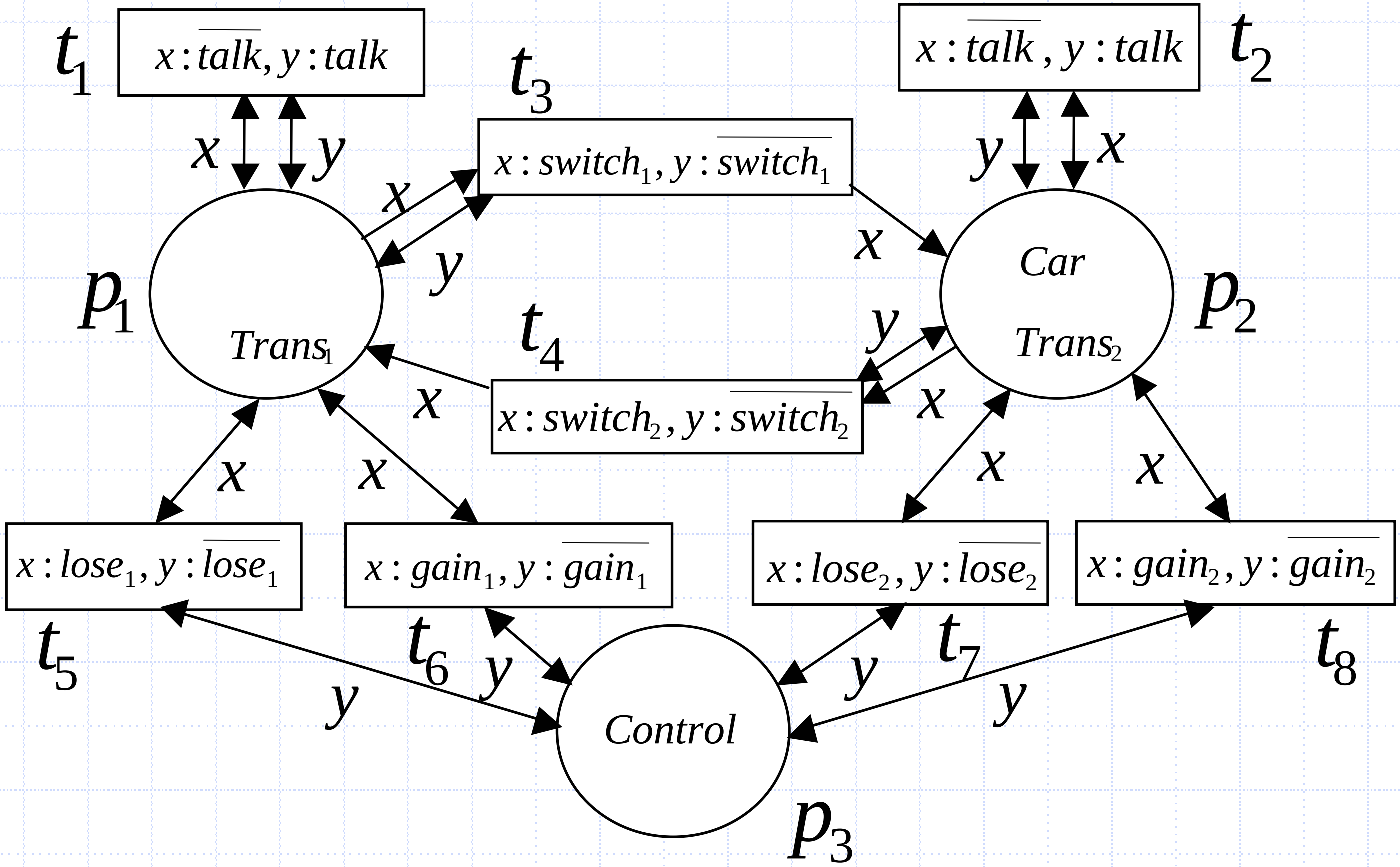
t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
Car
$Trans_1$
p_2
$Trans_2$
p_3
Control
$x:lose_1, y:\overline{lose_1}$
t_5
$x:gain_1, y:\overline{gain_1}$
t_6
$x:lose_2, y:\overline{lose_2}$
t_7
$x:gain_2, y:\overline{gain_2}$
t_8
x
y

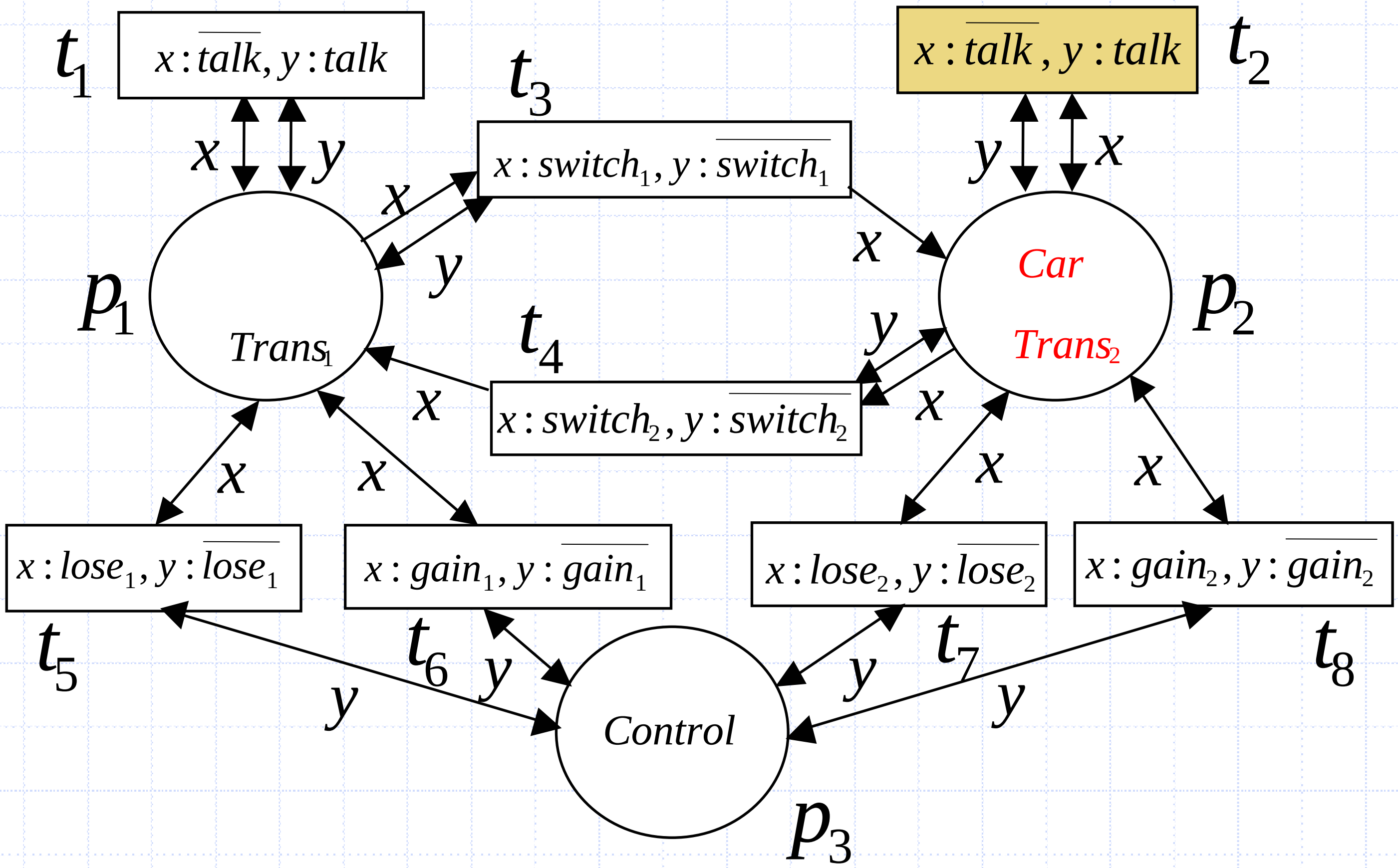

t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
Car
Trans$_1$
p_2
Trans$_2$
p_3
Control
$x:lose_1, y:\overline{lose_1}$
t_5
$x:gain_1, y:\overline{gain_1}$
t_6
$x:lose_2, y:\overline{lose_2}$
t_7
$x:gain_2, y:\overline{gain_2}$
t_8
x
y

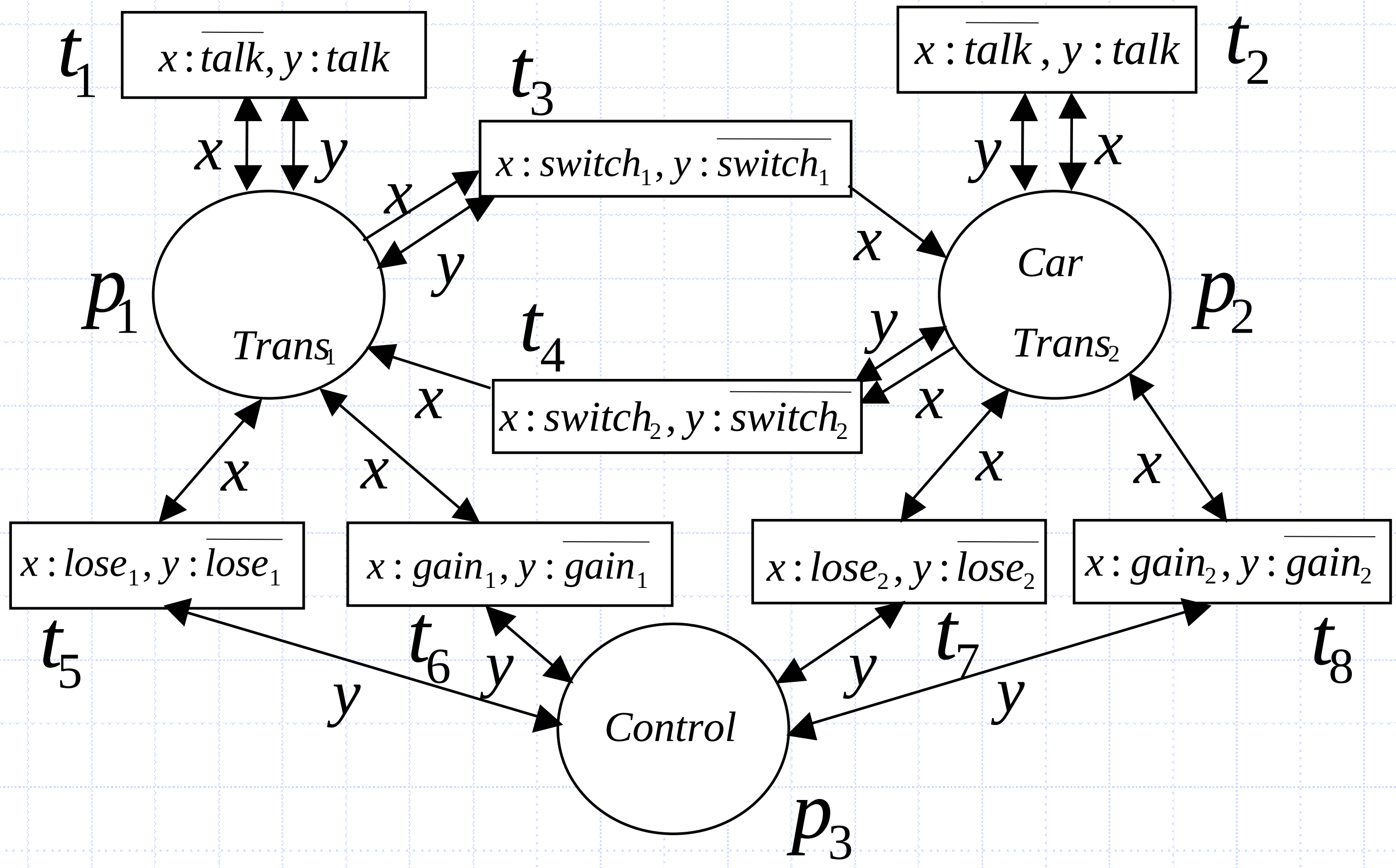


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
Car
$Trans_1$
p_2
$Trans_2$
p_3
Control
$x:lose_1, y:\overline{lose_1}$
t_5
$x:gain_1, y:\overline{gain_1}$
t_6
$x:lose_2, y:\overline{lose_2}$
t_7
$x:gain_2, y:\overline{gain_2}$
t_8
x
y


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
t_5
$x:lose_1, y:\overline{lose_1}$
t_6
$x:gain_1, y:\overline{gain_1}$
t_7
$x:lose_2, y:\overline{lose_2}$
t_8
$x:gain_2, y:\overline{gain_2}$
p_1
Car
$Trans_1$
p_2
$Trans_2$
p_3
Control
x
y


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
$Trans_1$
p_2
Car
$Trans_2$
t_5
$x:lose_1, y:\overline{lose_1}$
t_6
$x:gain_1, y:\overline{gain_1}$
t_7
$x:lose_2, y:\overline{lose_2}$
t_8
$x:gain_2, y:\overline{gain_2}$
p_3
$Control$
x
y


t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
$Trans_1$
p_2
Car
$Trans_2$
t_5
$x:lose_1, y:\overline{lose_1}$
t_6
$x:gain_1, y:\overline{gain_1}$
t_7
$x:lose_2, y:\overline{lose_2}$
t_8
$x:gain_2, y:\overline{gain_2}$
p_3
$Control$
x
y



t_1
$x:\overline{talk}, y:talk$
t_2
$x:\overline{talk}, y:talk$
t_3
$x:switch_1, y:\overline{switch_1}$
t_4
$x:switch_2, y:\overline{switch_2}$
p_1
$Trans_1$
p_2
Car
$Trans_2$
p_3
Control
t_5
$x:lose_1, y:\overline{lose_1}$
t_6
$x:gain_1, y:\overline{gain_1}$
t_7
$x:lose_2, y:\overline{lose_2}$
t_8
$x:gain_2, y:\overline{gain_2}$
x
y

Example: π-calculus and *PN* ^{2}

- Differences
  - π-calculus: Process *Car* keeps the communication link to one of transmitters.
  - *PN* ^{2}: Which one of transmitters to talk with is determined by the location of Agent net *Car*.

Mathematical analysis

- Modeling power in the theoretical sense
- Equivalent P/T nets
- Matrix representation
- State equation
- Invariant analysis

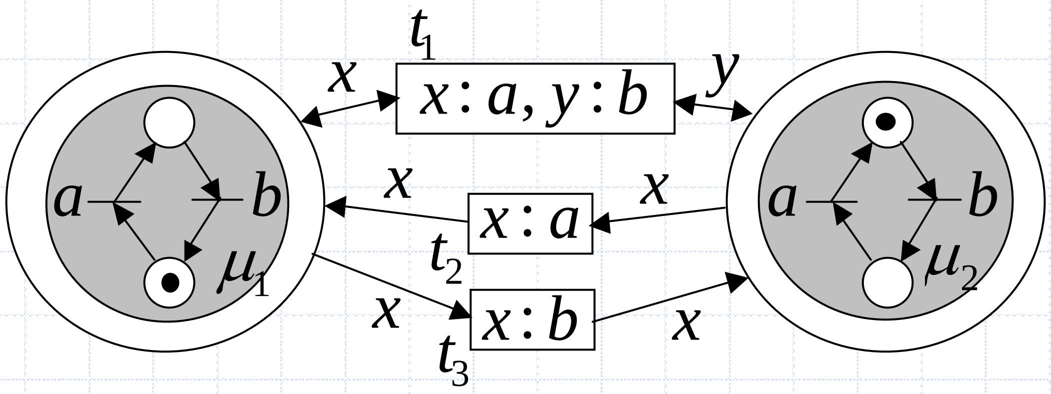
Modeling power (in the theoretical sense)

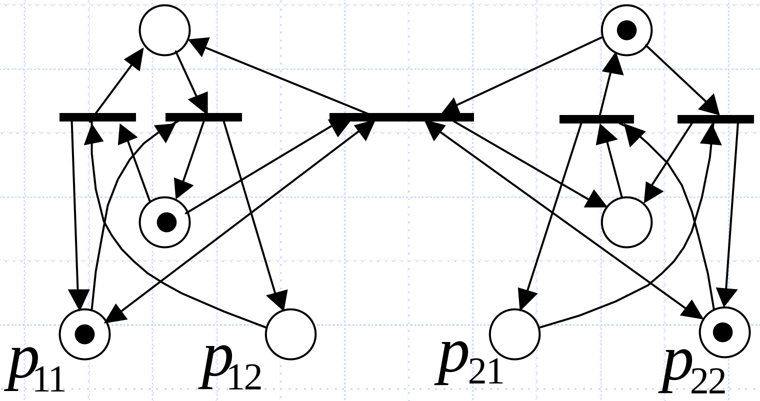
- There are two cases in which PN^2 has infinitely many states:
  - Some agent net are duplicated infinitely many times (unbounded PN^2).
  - Some agent nets are unbounded P/T nets (infinite-sort PN^2).
- When a PN^2 is finite-sort, there exists a marked P/T net that simulates the behavior of the PN^2.
- The class of infinite-sort PN^2s has the same level of modeling power as Turing machines.

Equivalent P/T nets

- One of ways to analyze PN^2s is to construct equivalent P/T nets, because there exists various analysis techniques developed for P/T nets.
- Current results:
  - For any finite-sort PN^2, we can construct an equivalent marked P/T net, where the equivalence is defined by isomorphism of state transition diagrams.
  - If every transition component has exactly one input and one output, then we can also construct an equivalent marked P/T net (even if PN^2 is infinite-sort).

Equivalent P/T nets - Example

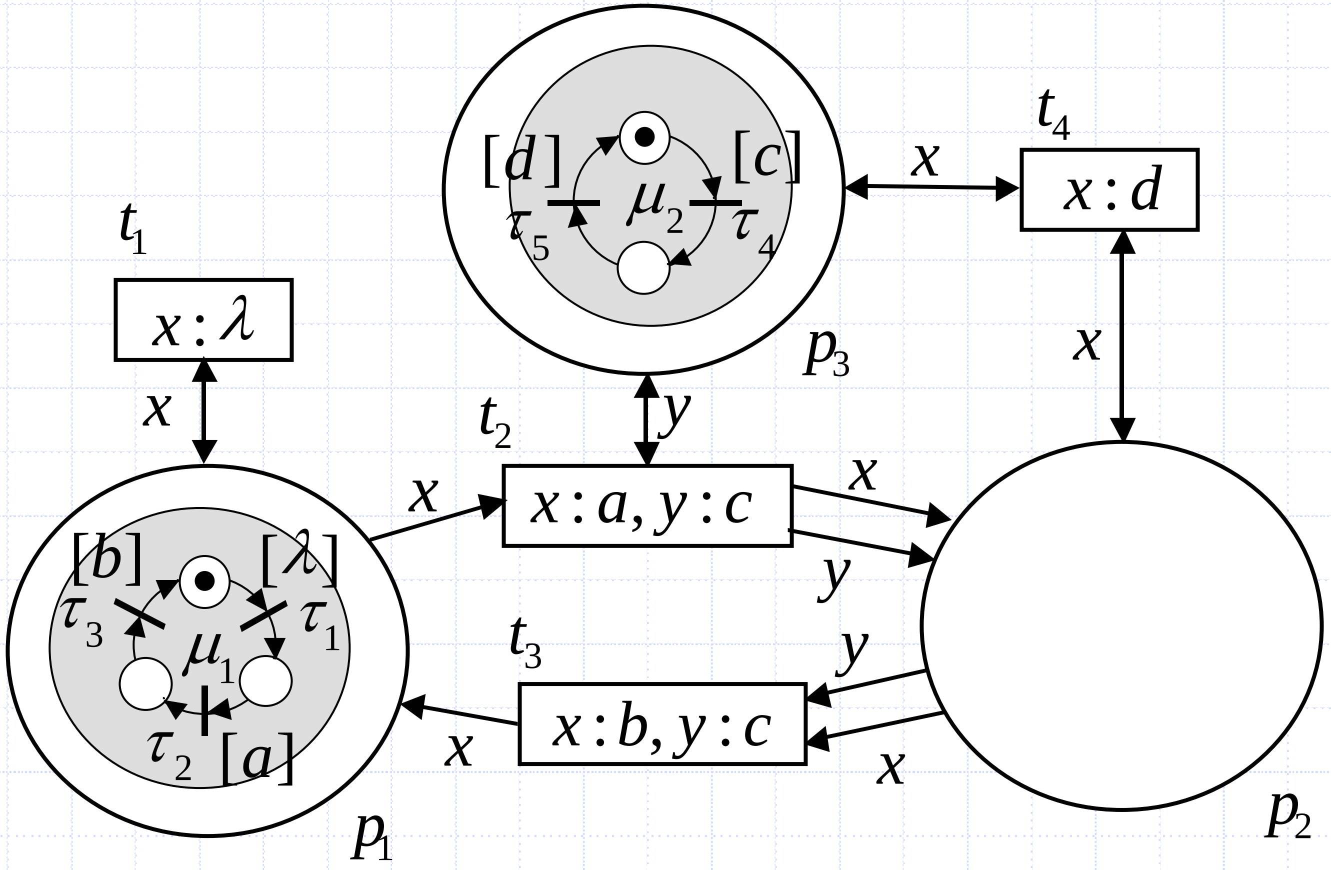




Matrix representation

- Graph structure of a P/T net is represented by an integer matrix, called incidence matrix.
- Linear algebraic analysis such as state equation and invariants are based on this representation.
- Using special structure of PN^2, we can obtain a compact representation of the incidence matrix.

Matrix representation - Example



Matrix representation - Example

Transitions

Transition components

Agent nets

$$
I = \begin{array}{cc|c|cc|cc|c}
 & & t_1 & t_2 & & t_3 & & t_4 \\
 & & x & x & y & x & y & x \\
 & & \mu_1 & \mu_1' & \mu_2 & \mu_1'' & \mu_2 & \mu_2' \\
\hline
p_1 & \mu_1 & -1 & 0 & 0 & 1 & 0 & 0 \\
 & \mu_1' & 1 & -1 & 0 & 0 & 0 & 0 \\
\hline
p_2 & \mu_1'' & 0 & 1 & 0 & -1 & 0 & 0 \\
 & \mu_2 & 0 & 0 & 0 & 0 & -1 & 1 \\
 & \mu_2' & 0 & 0 & 1 & 0 & 0 & -1 \\
\hline
p_3 & \mu_2 & 0 & 0 & -1 & 0 & 0 & 1 \\
 & \mu_2' & 0 & 0 & 1 & 0 & 0 & -1
\end{array}
$$

Places

Incidence Matrix

State equation

- (A necessary condition for reachability)

  If state s is reachable from state s_0, then there exists a firing count vector x such that

$$s = s_0 + I\,x$$

where I is the incidence matrix.

State equation - Example

$$\begin{tikzcd}
{[\mu_1,\varnothing,\mu_2]} \arrow[d, "{t_1 : x \mapsto \tau_1 \text{ in } \mu_1}"] \arrow[d, shift right=2] & s_0 \\
{[\mu_1',\varnothing,\mu_2]} \arrow[d, "{t_2 : x \mapsto \tau_2 \text{ in } \mu_1'; \; y \mapsto \tau_4 \text{ in } \mu_2}"] \arrow[d, shift right=2] & s_1 \\
{[\varnothing,\mu_1''+\mu_2',\mu_2']} \arrow[d, "{t_4 : x \mapsto \tau_5 \text{ in } \mu_2'}"] \arrow[d, shift right=2] & s_2 \\
{[\varnothing,\mu_1''+\mu_2,\mu_2]} \arrow[uuu, bend left=60, "{t_3 : x \mapsto \tau_3 \text{ in } \mu_1''; \; y \mapsto \tau_4 \text{ in } \mu_2}"] & s_3
\end{tikzcd}$$

State transition diagram

State equation - Example

$$
\begin{array}{ll}
p_1 & \mu_1 \\
 & \mu_1' \\
p_2 & \mu_1'' \\
 & \mu_2 \\
 & \mu_1' \\
p_3 & \mu_2 \\
 & \mu_2'
\end{array}
\left[\begin{array}{c}
0 \\ 0 \\ \hline 1 \\ 1 \\ 0 \\ \hline 1 \\ 0
\end{array}\right]
=
\left[\begin{array}{c}
1 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ \hline 1 \\ 0
\end{array}\right]
+ I \cdot
\left[\begin{array}{c}
1 \\ \hline 1 \\ 1 \\ \hline 0 \\ 0 \\ \hline 1
\end{array}\right]
\begin{array}{ll}
x & t_1 \\
x & t_2 \\
y & \\
x & t_3 \\
y & \\
x & t_4
\end{array}
$$

s_3 s_0 Firing count vector

Invariants

- Computing invariants is one of important and effective techniques in the analysis of Petri nets.
  - *A T-invariant* is a firing count vector x such that any firing sequence realizing x does not change the state.
  - A T-invariant corresponds to a cycle in the state transition diagram. If the set of reachable states is finite and the system never terminates, then there always exists a T-invariant.
  - *A P-invariant* is a vector of weights so that the weighted sum of the number of tokens in each place is constant by any occurrence of transitions.
  - By computing P-invariants, we can verify many dynamic properties.

Invariants - Example

$$\begin{tikzcd}
{[\mu_1,\emptyset,\mu_2]} \arrow[d, "t_1 : x \mapsto \tau_1 \text{ in } \mu_1"] & s_0 \\
{[\mu_1',\emptyset,\mu_2]} \arrow[d, "t_2 : x \mapsto \tau_2 \text{ in } \mu_1'{,}\ y \mapsto \tau_4 \text{ in } \mu_2"] & s_1 \\
{[\emptyset,\mu_1''+\mu_2',\mu_2']} \arrow[d, "t_4 : x \mapsto \tau_5 \text{ in } \mu_2'"] \arrow[uu, bend left=60, "\text{cycle}"] & s_2 \\
{[\emptyset,\mu_1''+\mu_2,\mu_2]} \arrow[uuu, bend left=80, "\begin{array}{c} t_3 : x \mapsto \tau_3 \text{ in } \mu_1''{,} \\ y \mapsto \tau_4 \text{ in } \mu_2 \end{array}"] & s_3
\end{tikzcd}$$

State transition diagram

Invariants - Example

$$0 = I \cdot \left[\begin{array}{c} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{array}\right] \begin{array}{l} x \quad t_1 \\ \left.\begin{array}{l} x \\ y \end{array}\right] t_2 \\ \left.\begin{array}{l} x \\ y \end{array}\right] t_3 \\ x \quad t_4 \end{array}$$

T-invariant

Invariants - Example

$$0 = \begin{array}{cc|ccc|cc} \multicolumn{2}{c}{\overbrace{\hspace{3em}}^{p_1}} & \multicolumn{3}{c}{\overbrace{\hspace{6em}}^{p_2}} & \multicolumn{2}{c}{\overbrace{\hspace{3em}}^{p_3}} \\ \mu_1 & \mu_1' & \mu_1'' & \mu_2 & \mu_2' & \mu_2 & \mu_2' \\ [1 & 1 & 1 & 0 & 0 & 1 & 1] \end{array} \cdot I^T$$

P-invariant

Further work

- Modeling real problems (including high-level version of PN^2)
- Efficient verification techniques using special structure of PN^2
- Several theoretical problems arising in the design of agents
- Developing computer tools