The Security of Mobile Agent Systems

Wiesław Pawłowski

Modeling and Reasoning about Mobile Agent Systems

– with some applications to security –

Joint work with

- Marek A. Bednarczyk – Institute of Computer Science PAS, PL
- Luca Bernardinello – University of Milano Bicocca, IT
- Tomasz Borzyszkowski - Gdańsk University, PL
- Wojciech Jamroga – University of Clausthal, DE
- Lucia Pomello – University of Milano Bicocca, IT

Systems, views, models

- System
  - dynamic (in a non-technical sense)
  - composed of (mobile) agents possessing "identity"
- View
  - description of a certain aspect of system behavior
- System model
  - "synchronous composition" of several views

Systems, views, specifications

- System
  - dynamic (in a non-technical sense)
  - composed of (mobile) agents possessing "identity"
- View
  - description of a certain aspect of system behavior
- System specification
  - "synchronous composition" of several views

Simple example

"Flip-flop" message exchange

- agents:
  - communicators – senders/receivers
  - messages
- assumptions
  - there is a message flow infrastructure between communicators
  - messages are addressed – contain info about sender and addressee
  - only addressee can receive a message
  - addressee can send the message back to the sender

Different aspects of system behavior

- physical view – message flow
  - agents: communicators, messages, communication infrastructure
  - describes: *topology* of the communication infrastructure
- logical view – addressing of the messages
  - agents: communicators and messages
  - describes:
    - *sender* and *addressee* for each message
    - message *delivery control*

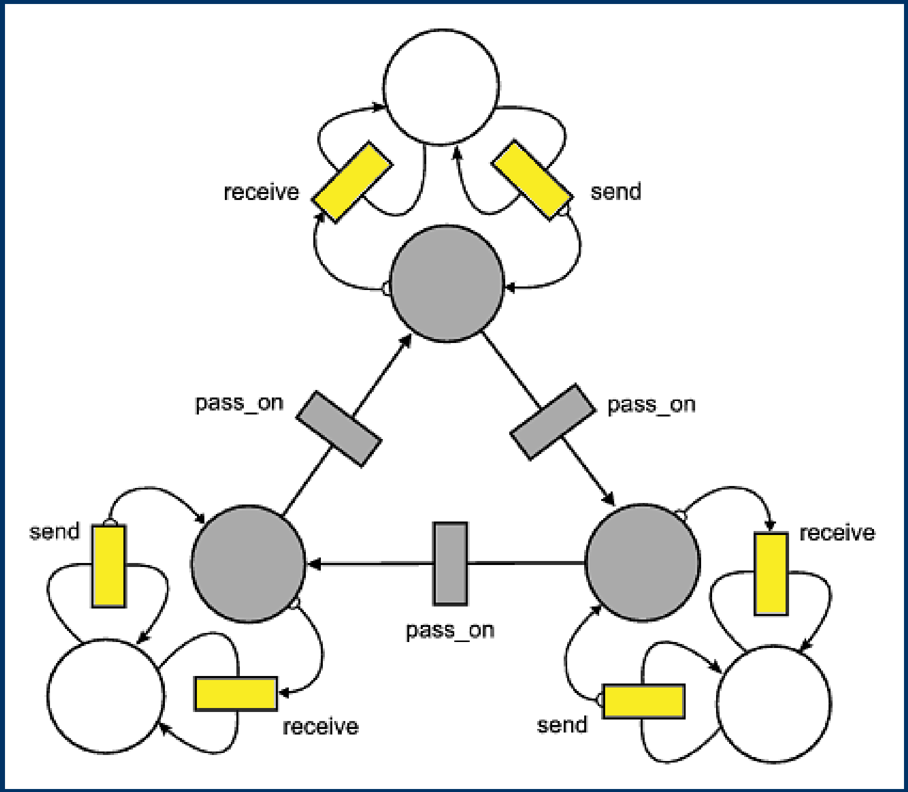
Modeling the physical view

- We shall use Petri hypernets as a modeling framework
- agents:
  - communicators, messages and communication infrastructure
- actions (transitions):
  - send – a communicator sends a message
  - receive – a message is delivered to a communicator
  - pass_on – a message changes its location within the communication infrastructure

Petri hypernets in a nutshell

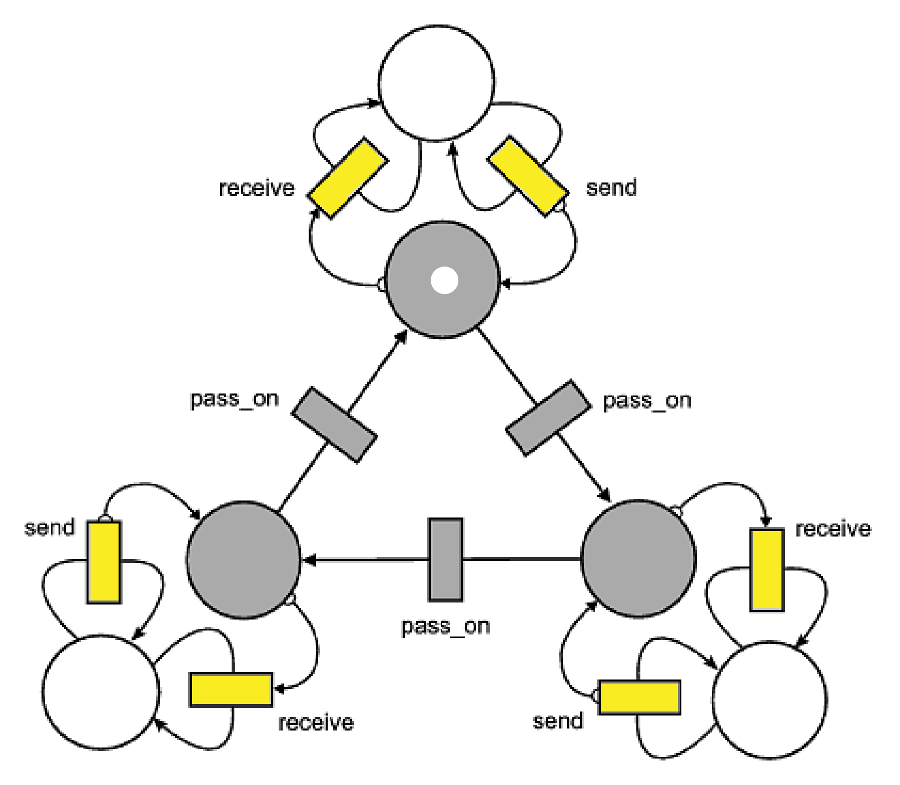
- **Agents** $\approx$ Petri nets with "communication ports"
  - based on the "*nets-within-nets*" idea (nets as tokens)
  - **locations** $=$ places in nets
- Hypernet $\approx$ set of "typed" nets
- Dynamics
  - hypernet state (hypermarking) $\approx$ type-preserving placement of agents in locations (agent has ***at most one*** location at a given time)
  - agents form consortia to perform common actions (transitions)
  - hierarchy of agents evolves as a result of firing consortia
  - number and "identity" of agents ***does not change***

Communication infrastructure agent



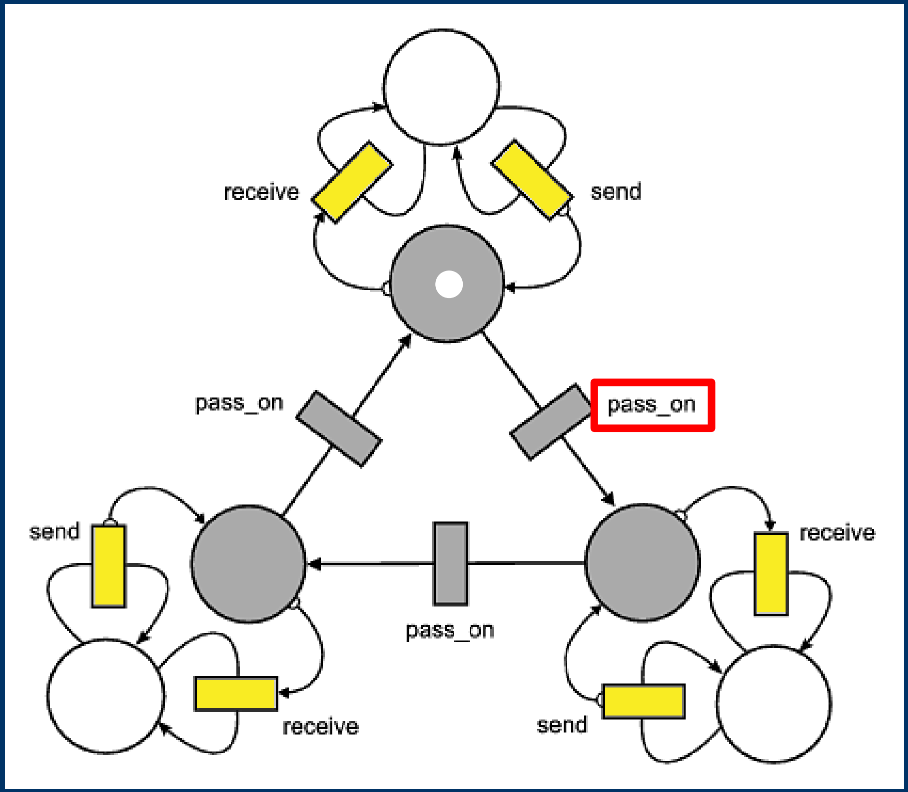
- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

Infrastructure & messages



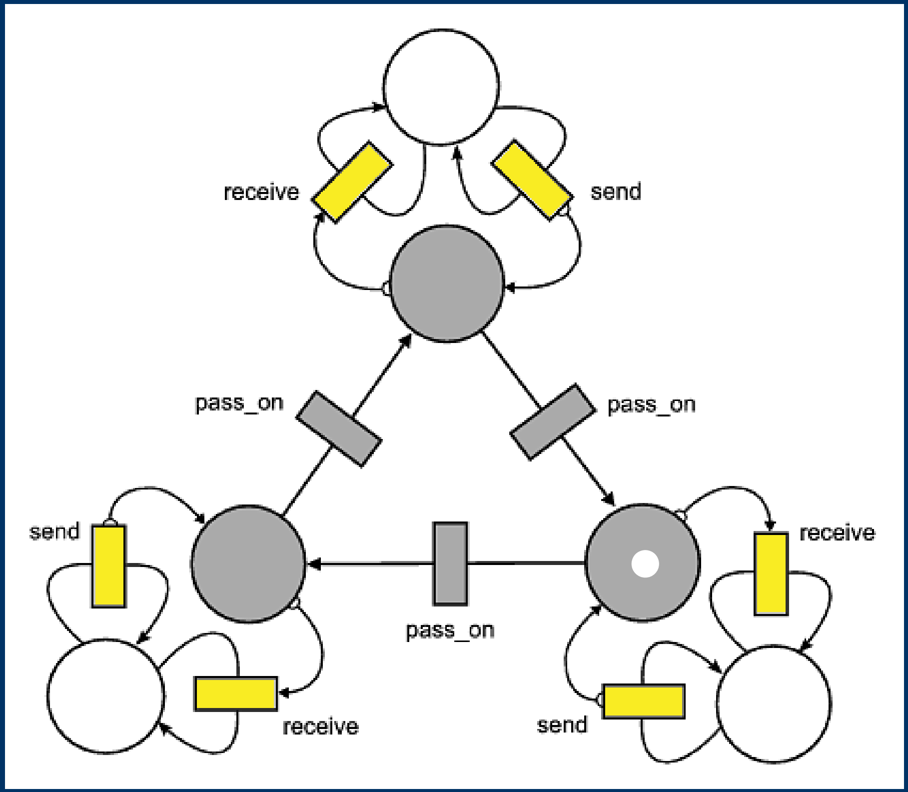
- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

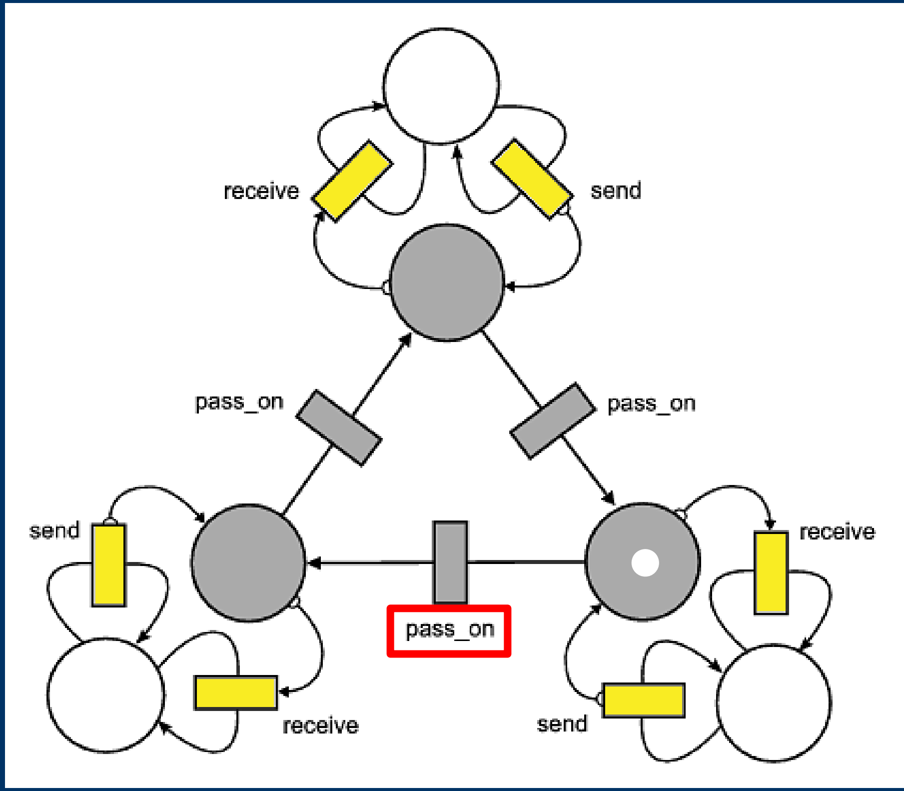
Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

pass_on

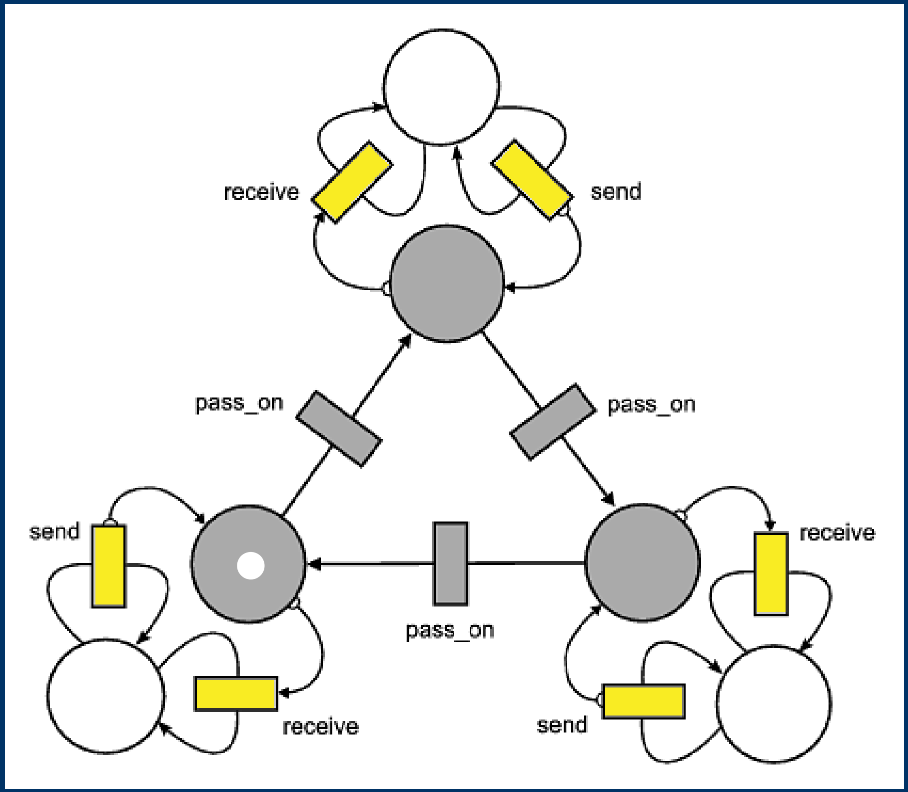
Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

pass_on

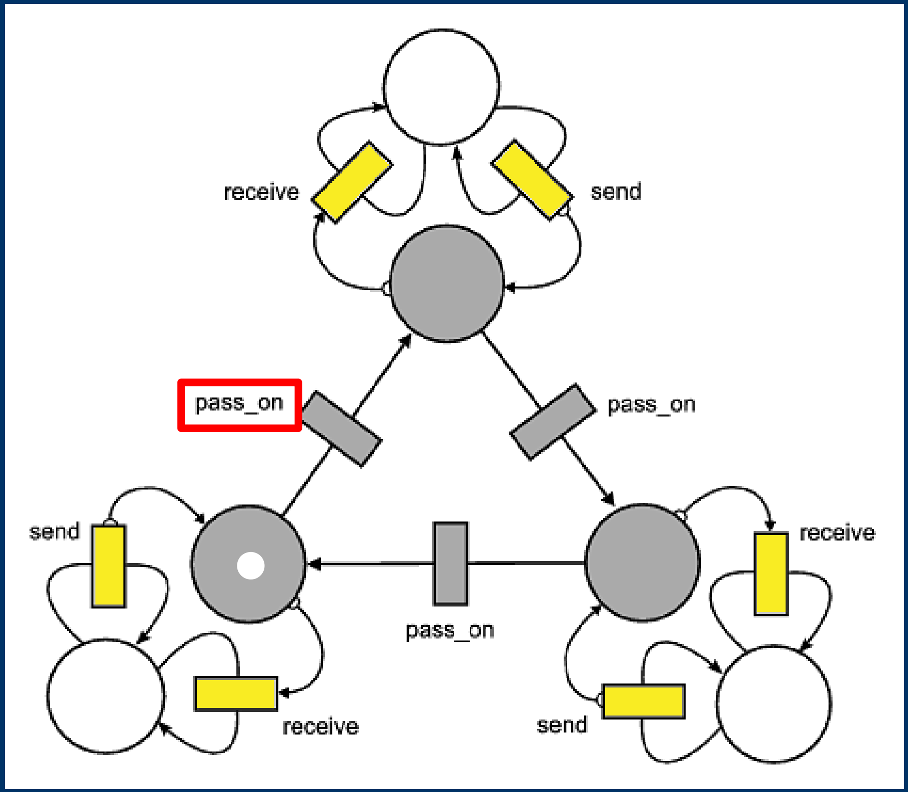
Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

pass_on, pass_on

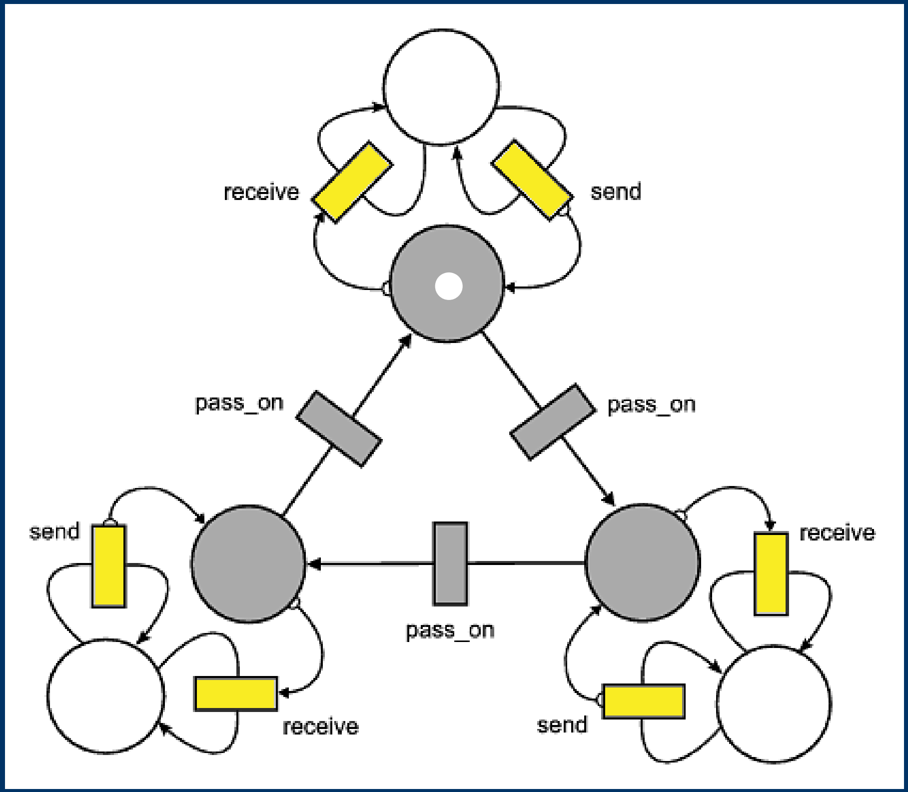
Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

pass_on, pass_on

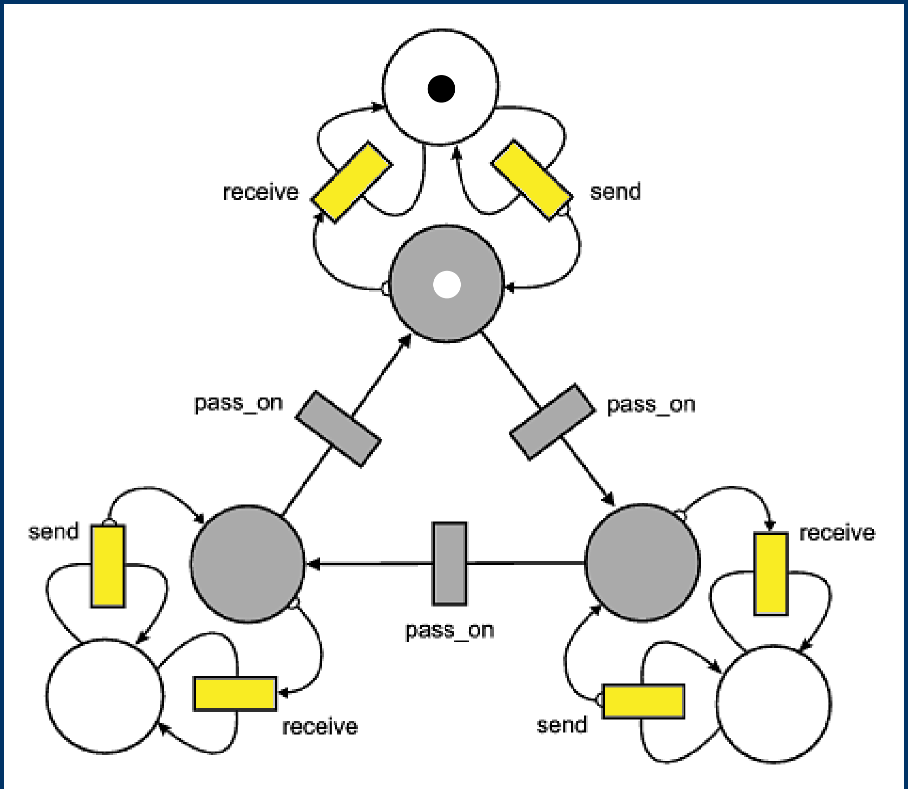
Infrastructure & messages



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

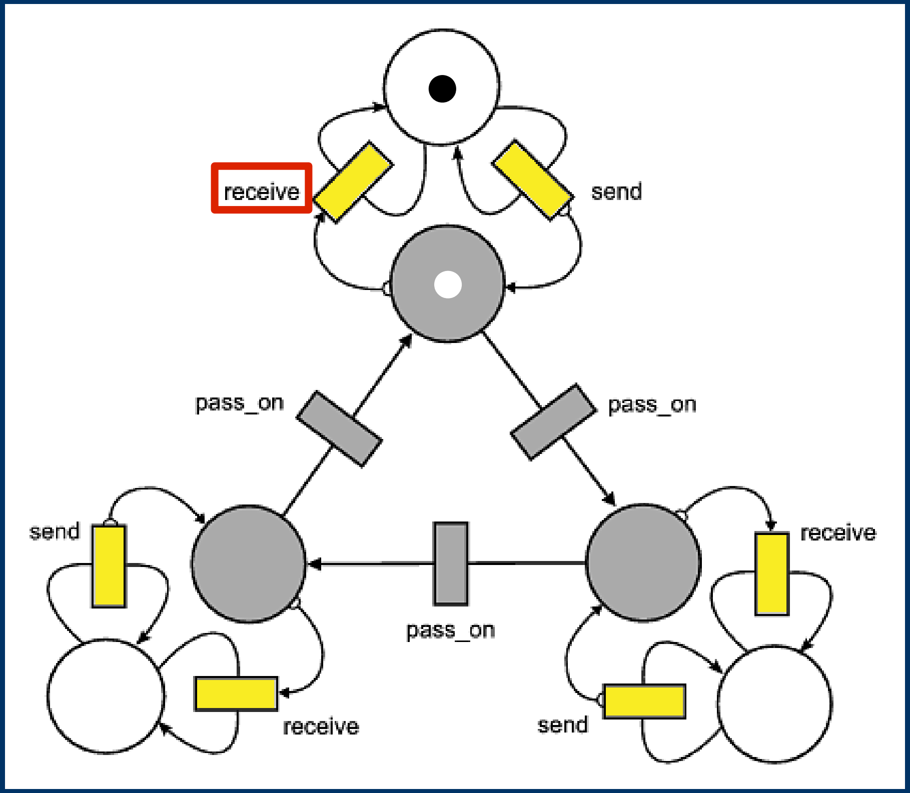
pass_on, pass_on, pass_on, ...

Infrastructure, messages & communicators



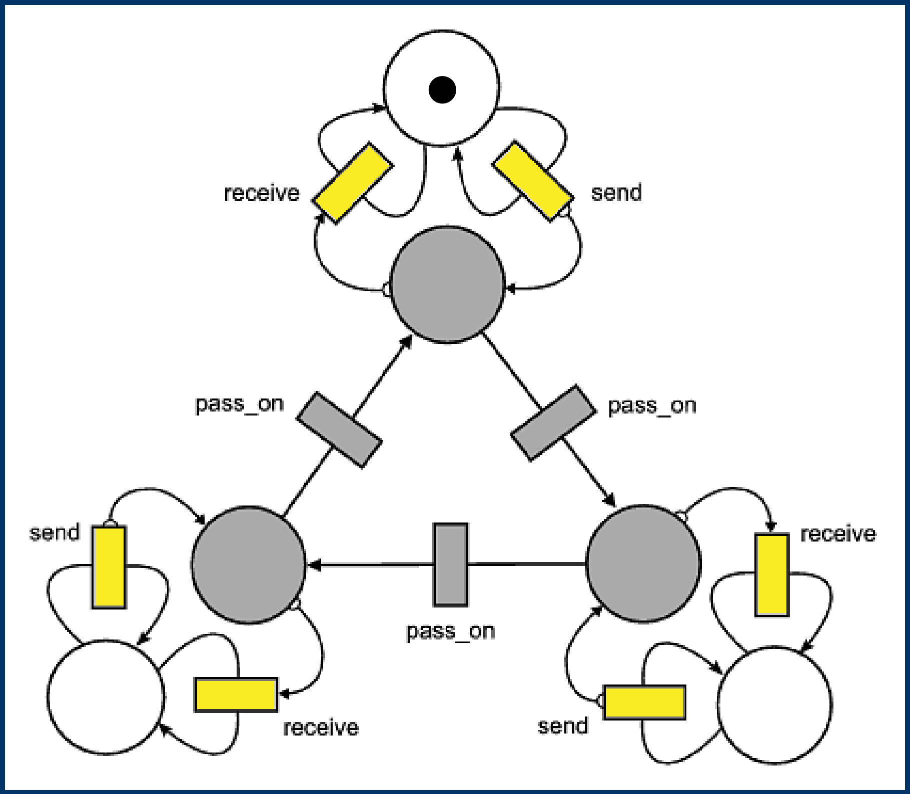
- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module
- Delivery/sending of messages is modeled by "inter-level" transitions (yellow)

Infrastructure, messages & communicators



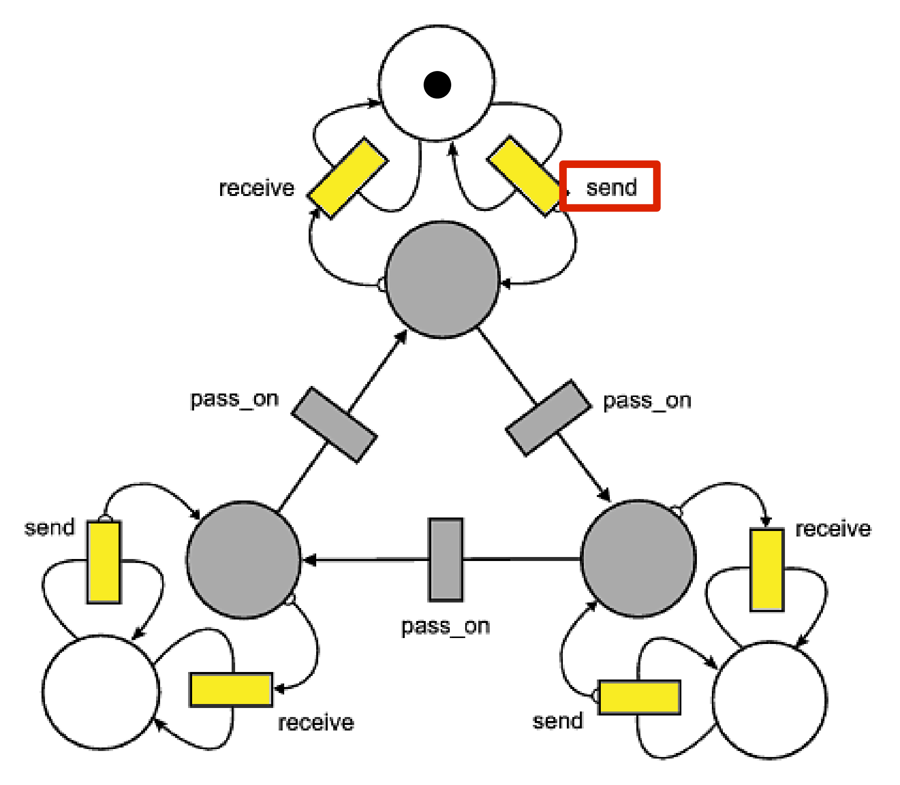
- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module
- Delivery/sending of messages is modeled by "inter-level" transitions (yellow)

Infrastructure, messages & communicators



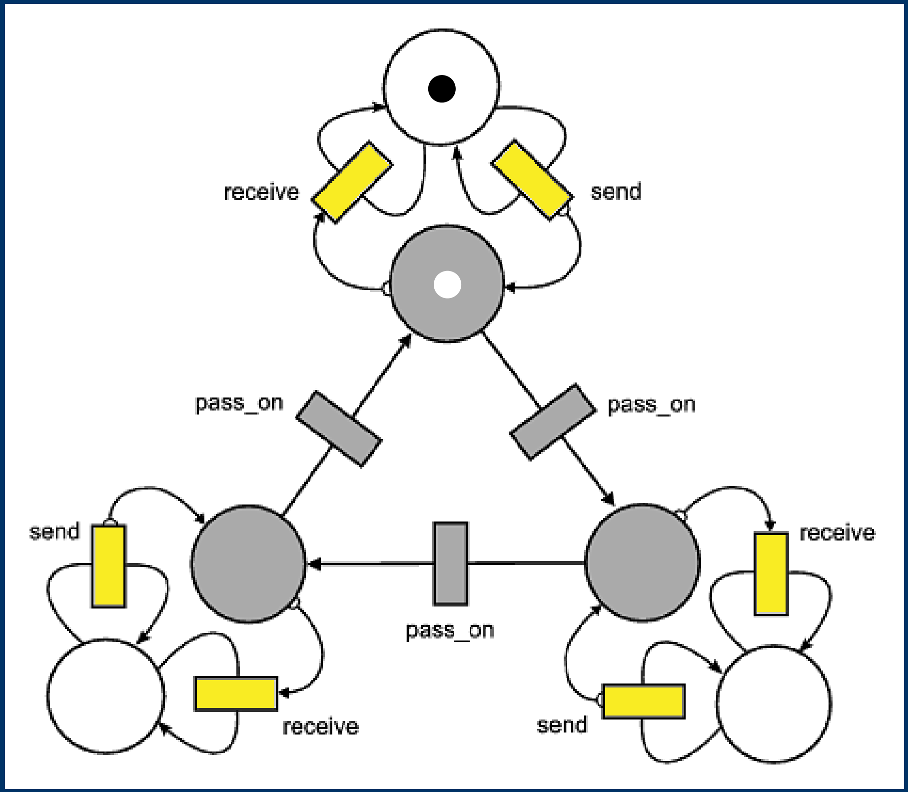
- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module
- Delivery/sending of messages is modeled by "inter-level" transitions (yellow)

Infrastructure, messages & communicators



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module
- Delivery/sending of messages is modeled by "inter-level" transitions (yellow)

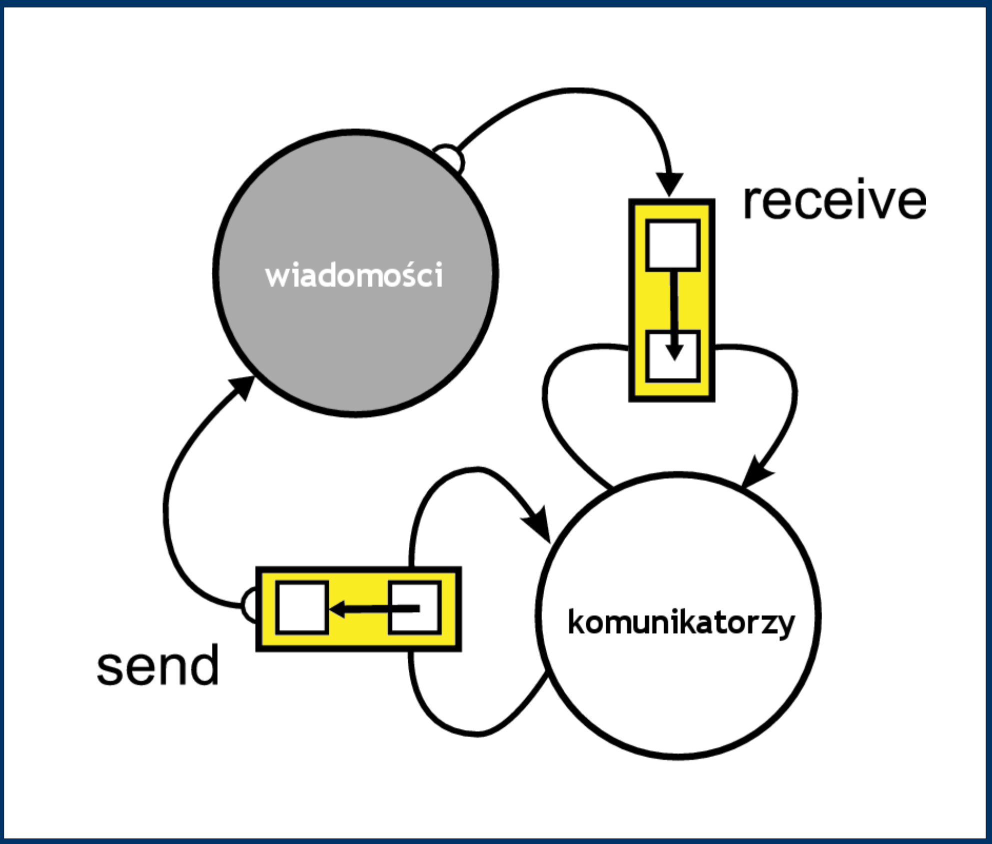
Infrastructure, messages & communicators



- Infrastructure composed of two "modules":
  - message transport (gray)
  - communicator (white)
- Action (*transition*) pass_on is a part of the message transport module

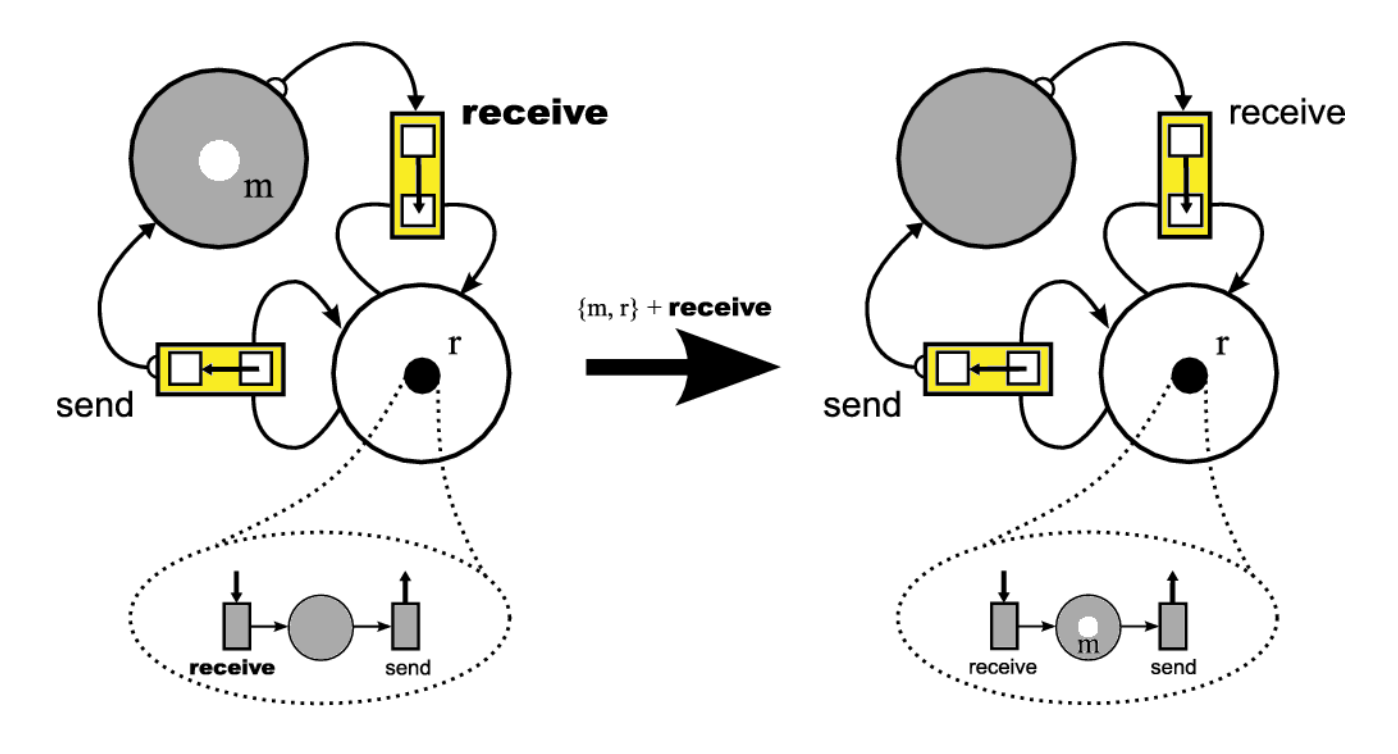
- Delivery/sending of messages is modeled by "inter-level" transitions (yellow)

Physical message flow



- firings of receive and send
  - manipulate pairs of agents message – receiver/sender
- receive – moves a message from a communication infrastructure location “inside” the receiver agent
- send – moves a message from “inside” the sender to a communication infrastructure location

Physical message flow – receive action



Physical message flow – an abstraction

- $A \stackrel{\text{def}}{=} \text{Mess} \cup \text{Comm} \cup \{\text{ infra }\}$ $\Leftarrow$ agents
- $S \stackrel{\text{def}}{=} (\text{Mess} \rightarrow \text{Loc}^{m}) \times (\text{Comm} \rightarrow \text{Loc}^{c})$ $\Leftarrow$ states

  where

  - Loc^{m} – set of message locations
  - Loc^{c} – set of communicator locations
- $T \stackrel{\text{def}}{=} \{\text{ pass\_on, send, receive }\}$ $\Leftarrow$ transitions
- $\delta : T \rightarrow \mathcal{P}(A) \rightarrow (S \rightharpoonup S)$ $\Leftarrow$ transition function

$\delta(t)(\alpha)(s) \stackrel{\text{def}}{=}$ **if** α is a t-consortium in s
then "result of firing the consortium s"
else "undefined"

Logical view – message addressing

- $A' \stackrel{\text{def}}{=} \text{Mess} \cup \text{Comm}$
- $S' \stackrel{\text{def}}{=} \text{Mess} \rightarrow \text{Comm} \times \text{Comm}$
- $T' \stackrel{\text{def}}{=} \{\ \text{send, receive}\ \}$

- $\delta'(\text{send})(\alpha)(s) \stackrel{\text{def}}{=}$ <u>**if**</u> $(\exists\ m,c,c')\ \alpha=\{m,c\} \wedge s(m)=(c,c')$ <u>**then**</u> s
  <u>**else**</u> "undefined"

- $\delta'(\text{receive})(\alpha)(s) \stackrel{\text{def}}{=}$ <u>**if**</u> $(\exists\ m,c,c')\ \alpha=\{m,c'\} \wedge s(m)=(c,c')$
  <u>**then**</u> $s\ [\ m \rightarrowtail (c',c)\ ]$ $\Leftarrow$ address transposition
  <u>**else**</u> "undefined"

Agent-aware transition systems

agent-aware transition system (A, T, S, δ):

- A set of agents
- T set of transitions
- S set of states
- $\delta : T \to \mathcal{P}(A) \to (S \rightharpoonup S)$ transition function

(simple modification of the *deterministic transition system* notion)

View = agent-aware transition system

Synchronization of views

Let $\{ \mathcal{S}_i \mid i \in I \}$ – a family of views $\mathcal{S}_i = (A_i , T_i , S_i , \delta_i)$

$$\prod_{i \in I} \mathcal{S}_i \stackrel{\text{def}}{=} \langle \cup_{i \in I} A_i , \cup_{i \in I} T_i , \prod_{i \in I} S_i , \Delta \rangle$$

where

$$\begin{aligned}\Delta(t)(\alpha)(s) \approx\ & \underline{\mathbf{if}}\ \ (\exists\, k \in I)\ \ t \in T_k \wedge \ \delta_k(t)(\alpha \cap A_k)(s_k) \text{ is undefined} \\ & \underline{\mathbf{then}}\ \ \text{“undefined”} \\ & \underline{\mathbf{else}}\ \ \langle\ \underline{\mathbf{if}}\ t \in T_i\ \underline{\mathbf{then}}\ \delta_i(t)(\alpha \cap A_i)(s_i)\ \underline{\mathbf{else}}\ s_i \mid i \in I \rangle\end{aligned}$$

Synchronization result – system specification

Some “unwanted” features of views considered separately:

- physical: abstracts from addressing (send, receive)
- logical: disregards physical placement of agents (send, receive)

After synchronization:

- action pass_on (autonomous for the physical view) modeled exactly as before
- actions send and receive take into account both addressing and physical placement of message and sender/receiver

Synchronization as a product

morphism $\sigma : \mathcal{S}_1 \to \mathcal{S}_2$ of agent-aware t.s. $\mathcal{S}_i = (A_i , T_i , S_i , \delta_i)$ $i=1,2$

s.t. $A_1 \supseteq A_2$ and $T_1 \supseteq T_2$ is a function $\sigma : S_1 \to S_2$ such that:

- $\forall\ t \in T_1 , \alpha \subseteq A_1 , s \in S_1$ if $t \in T_2$ and $\delta_1(t)(\alpha)(s)$ is defined then $\delta_2(t)(\alpha \cap A_2)(\sigma(s)) = \sigma(\delta_1(t)(\alpha)(s))$
- $\forall\ t \in T_1 , \alpha \subseteq A_1 , s \in S_1$ if $t \notin T_2$ and $\delta_1(t)(\alpha)(s)$ is defined then $\sigma(s) = \sigma(\delta_1(t)(\alpha)(s))$

ATS – the category of agent-aware transition systems and their morphisms

Proposition. Synchronization = categorical product in ATS. $\square$

Synchronization as a product

morphism $\sigma : \mathcal{S}_1 \to \mathcal{S}_2$ of agent-aware t.s. $\mathcal{S}_i = (A_i , T_i , S_i , \delta_i)$ $i=1,2$

s.t. $A_1 \supseteq A_2$ and $T_1 \supseteq T_2$ is a function $\sigma : S_1 \to S_2$ such that:

$$
s \xrightarrow{t,\alpha} s' \text{ then } \begin{cases} \sigma(s)=\sigma(s') & t \in T_2 \\ \sigma(s) \xrightarrow{t,\alpha\cap A_2} \sigma(s') & \text{otherwise} \end{cases}
$$

ATS – the category of agent-aware transition systems and their morphisms

Proposition. Synchronization = categorical product in ATS.

□

Slightly more interesting example

Exchange of (possibly) encrypted messages

- agents:
  - communicators – senders/receivers
  - messages
  - keys – public and secret
- assumptions
  - there is a message flow infrastructure between communicators
  - message *might* be encrypted with receiver's public key
  - appropriate secret key is needed to decrypt an encrypted message

Five views

- physical
- logical 1 – encrypting and decrypting of messages
  - agents: messages and public keys
- logical 2 – public and secret keys relationship
  - agents: public and secret keys
- logical 3 and 4 – communicators' knowledge about public/secret keys
  - agents: communicators and public/secret keys

1. *Encrypting and decrypting messages*

- $A^1 \stackrel{\text{def}}{=} \text{Mess} \cup \text{PubKey}$
- $S^1 \stackrel{\text{def}}{=} \text{Mess} \longrightarrow \text{PubKey}^*$
- $T^1 \stackrel{\text{def}}{=} \{\ \text{send},\ \text{receive}\ \}$

- $\delta^1(\text{send})(\alpha)(s) \stackrel{\text{def}}{=}$ <u>**case**</u> α <u>**of**</u>

  $\{m, pk\} : \ s\,[\ m \rightarrowtail pk \cdot s(m)\ ]$ ⇐ encrypt with “pk”

  $\{m\} \quad : \ s$ ⇐ “plain-text”

  <u>**else**</u> “undefined”

1. *Encrypting and decrypting of messages*

- $A^1 \stackrel{\text{def}}{=} \text{Mess} \cup \text{PubKey}$
- $S^1 \stackrel{\text{def}}{=} \text{Mess} \rightarrow \text{PubKey}^*$
- $T^1 \stackrel{\text{def}}{=} \{\ \text{send}, \text{receive}\ \}$

- $\delta^1(\text{receive})(\alpha)(s) \stackrel{\text{def}}{=}$ $\underline{\textbf{if}}$ $\alpha = \{m,pk\} \wedge s(m) = pk \cdot pks$
  $\underline{\textbf{then}}$ $s\,[\ m \rightarrowtail pks\ ]$ $\Leftarrow$ message decrypting
  $\underline{\textbf{elseif}}$ $\alpha = \{m\} \wedge s(m) = \text{null}$
  $\underline{\textbf{then}}$ s $\Leftarrow$ „plain-text” message
  $\underline{\textbf{else}}$ „undefined”

2. *Public-Secret key relationship*

- $A^2 \stackrel{\text{def}}{=} \text{PubKey} \cup \text{SecKey}$
- $S^2 \stackrel{\text{def}}{=} \text{PubKey} \longrightarrow \text{SecKey}$
- $T^2 \stackrel{\text{def}}{=} \{ \text{ receive } \}$

- $\delta^2(\text{receive})(\alpha)(s) \stackrel{\text{def}}{=} \underline{\textbf{if}}\ \ \alpha = \emptyset \lor \ \alpha = \{\text{pk,sk}\} \land s(\text{pk}) = \text{sk}$
  $\underline{\textbf{then}}\ \ s$
  $\underline{\textbf{else}}$ "undefined"

3. *Communicators' knowledge – public keys*

- $A^3 \stackrel{\text{def}}{=} \text{Comm} \cup \text{PubKey}$
- $S^3 \stackrel{\text{def}}{=} \text{Comm} \to \mathcal{P}(\text{PubKey})$
- $T^3 \stackrel{\text{def}}{=} \{\ \text{send}\ \}$

- $\delta^3(\text{send})(\alpha)(s) \stackrel{\text{def}}{=} \underline{\textbf{if}}\ \ \alpha = \{c\} \lor \alpha = \{c,pk\} \land pk \in s(c)$

  $\underline{\textbf{then}}\ \ s$

  $\underline{\textbf{else}}$ "undefined"

4. *Communicators' knowledge - secret keys*

- $A^4 \stackrel{\text{def}}{=} \text{Comm} \cup \text{SecKey}$
- $S^4 \stackrel{\text{def}}{=} \text{Comm} \longrightarrow \mathcal{P}(\text{SecKey})$
- $T^4 \stackrel{\text{def}}{=} \{\ \text{receive}\ \}$

- $\delta^4(\text{receive})(\alpha)(s) \stackrel{\text{def}}{=} \underline{\textbf{if}}\ \ \alpha = \{c\} \vee \alpha = \{c, sk\} \wedge sk \in s(c)$

  $\underline{\textbf{then}}\ \ s$

  $\underline{\textbf{else}}$ "undefined"

Synchronous transition function – send

$\Delta(\text{send})(\alpha)(s)$

– physical:		{ inf, m, c }		
– logical 1:	{ m }	|	{ m, pk }	message encryption
– logical 2:		*not involved*		public/secret keys rel.
– logical 3:	{ c }	|	{ c, pk }	public keys knowledge
– logical 4:		*not involved*		secret keys knowledge

α α

Synchronous transition function – receive

$\Delta(\text{receive})(\alpha)(s)$

– physical:		$\{ \text{inf}, m, c \}$		
– logical 1:	$\{ m \}$	$\mid$	$\{ m, pk \}$	message encryption
– logical 2:	$\emptyset$	$\mid$	$\{ pk, sk \}$	public/secret keys rel.
– logical 3:		*not involved*		public keys knowledge
– logical 4:	$\{ c \}$	$\mid$	$\{ c, sk \}$	secret keys knowledge

α α

Synchronous transition function – pass_on

Δ(pass_on)(α)(s)

– physical:	{ inf, m }	
– logical 1:	*not involved*	message encryption
– logical 2:	*not involved*	public/secret keys rel.
– logical 3:	*not involved*	public keys knowledge
– logical 4:	*not involved*	secret keys knowledge

- We know:
  - how to model system views
  - how to obtain the description of the dynamics of the whole system from the composition of views
- Problem:
  - how to specify and verify (the desired) system properties

Interpreted agent-aware transition systems

interpreted agent-aware transition system $(A, T, S, \delta, \Sigma, \xi)$:

- (A, T, S, δ) agent-aware transition system
- Σ ranked alphabet (predicate names)
- $\xi : S \to \mathrm{RelStr}(\Sigma, A)$ interpretation function

where $\mathrm{RelStr}(\Sigma, A)$ is the set of Σ-relational structures with carrier A

Remarks:

- Interpreted agent-aware t.s. constitute a category IATS
- synchronization of *iaas* is a categorical product in IATS

Agent-terms and valuations

Agent-term: either agent name or agent variable

$$\mathrm{Term}(A) ::= \{ \lceil a \rceil \mid a \in A \} \cup X$$

Valuations and agent-term interpretations

$$v : X \longrightarrow A$$

$$v^{\#} : \mathrm{Term}(A) \longrightarrow A \quad \text{where } v^{\#}(\lceil a \rceil) \stackrel{\text{def}}{=} a$$

Formulas and satisfaction relation

- Formulas

$$\varphi ::= \text{false} \mid r(t_1,\ldots,t_n) \mid \neg\varphi \mid \varphi \vee \psi \mid E[\varphi U \psi] \mid EG\varphi$$

- Satisfaction relation for $M = (A, T, S, \delta, \Sigma, \xi)$

$s, v \vDash \text{false}$ never

$s, v \vDash \neg\varphi$ iff not $s, v \vDash \varphi$

$s, v \vDash \varphi \vee \psi$ iff $s, v \vDash \varphi$ or $s, v \vDash \psi$

$s, v \vDash r(t_1,\ldots,t_n)$ iff $(v^{\#}(t_1), \ldots, v^{\#}(t_n)) \in r_{\xi(s)}$

$s, a \vDash E[\varphi U \psi]$ iff there is a path π from s such that $\pi_k, v \vDash \psi$ for some k, and $\pi_j, v \vDash \varphi$, for $j=1,\ldots,k-1$

$s, a \vDash EG\varphi$ *iff there is a path* π *from* s *such that* $\pi_k, v \vDash \varphi$ for all k

Some useful derived formulas

- There exists a path such that φ eventually holds on it
  - $EF\varphi \equiv E[\text{true}U\varphi]$
- For all paths φ holds "globally" – for all states occurring in them
  - $AG\varphi \equiv \neg EF\neg\varphi$
- For all paths φ eventually holds on them
  - $AF\varphi \equiv \neg EG\neg\varphi$
- For all paths φ holds "until" ψ holds
  - $A[\varphi U\psi] \equiv \neg(E[\neg\psi U\neg(\varphi\vee\psi)] \vee EG\neg\psi)$

$\boxed{\text{Our logic} \approx \text{CTL – "next-step operator" + "n-ary predicates"}}$

Public key cryptography example revisited

Ranked alphabets and their interpretation for the views

- physical: a binary relational symbol SubAgent with the obvious agnet-containment interpretation
- logical 1: binary symbol LastEnc interpreted as relation between messages and public keys ("*last encrypted with*")
- logical 2: binary symbol MatchK interpreted as as a relation between public and secret keys ("matching keys")
- logical 4: binary symbol KnowsSecK interpreted as as the obvious relation between communicators and public keys

Public key cryptography example revisited

$$\mathrm{LastEnc}(m,pk) \wedge \mathrm{MatchK}(pk,sk) \wedge \mathrm{KnowsSecK}(c,sk)$$
$$\Downarrow$$
$$\mathrm{EF}\ \mathrm{SubAgent}(c,m)$$

if a communicator possesses "the right" secret key then there is a possible temporal evolution of the system leading to a state where the encrypted message has been delivered to him

$$\boxed{\text{Liveness property}}$$

Public key cryptography example revisited

$$\mathrm{MatchK}(pk,sk) \wedge \neg \mathrm{KnowsSecK}(c,sk)$$
$$\Downarrow$$
$$\neg \mathbf{E}[\mathrm{LastEnc}(m,pk)\,\mathbf{U}\,\mathrm{SubAgent}(c,m)]$$

if a communicator does not possess "the right" secret key he will not be able to receive the encrypted message

$$\boxed{\text{Safety property}}$$

Public key cryptography example revisited

$$\mathrm{KnowsSecK}(c,sk) \Rightarrow \mathrm{AG}\ \mathrm{KnowsSecK}(c,sk)$$
$$\wedge$$
$$\neg\mathrm{KnowsSecK}(c,sk) \Rightarrow \mathrm{AG}\ \neg\mathrm{KnowsSecK}(c,sk)$$

communicators neither forget nor acquire secret keys

$$\boxed{\text{Safety property}}$$

Public key cryptography example revisited

$$\mathsf{AG}\ (\ \mathsf{KnowsSecK}(c, \lceil k_a \rceil) \Rightarrow c = \lceil a \rceil\ )$$

k_a *is a "private key" of the communicator* a

Safety property

Conclusions

What has been presented

- agent-aware transition systems – a unified framework for modeling dynamic agent systems
- system model = categorical product of views
- simple (but expressive) logic for reasoning about properties of the system

Future work

- extend and refine the existing model checking tools for the presented logic using the Verics model-checker system
- apply the framework to a wide spectrum of case-studies