Team Automata for Groupware Systems

Maurice H. ter Beek[1]

Friday 13 November 1998

⇒ joint work with Clarence A. Ellis[2],
Jetty Kleijn[1] and Grzegorz Rozenberg[1,2]

[1]Leiden Institute of Advanced Computer Science, Leiden University

[2]Department of Computer Science, University of Colorado

Computer Supported Cooperative Work

How do groups *communicate*, *collaborate* and *cooperate* when working together?

How can technology assist?

Groupware $\approx$ group $+$ technology

Need for:

Conceptual level:
define concepts and terminology consistently

Architectural level:
describe and compare groupware systems

Ellis introduced concept of *team automata* for modelling both levels in semi-formal way in

"Team Automata for Groupware Systems"

“Synchronizations in Team Automata for Groupware Systems”

Aim: formalize and elaborate this concept

Model:

Conceptual level:
provides unambiguous formal framework for defining basic groupware notions such as *communication*, *collaboration* and *cooperation*

Architectural level:
allows use of team automata as building blocks in design of groupware systems

Component Automaton

$$\mathcal{C} = (Q, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, I)$$

Q set of *states* (possibly infinite)

Σ_{inp} *input* alphabet (of input actions)

Σ_{out} *output* alphabet (of output actions)

Σ_{int} *internal* alphabet (of internal actions)

$\delta \subseteq Q \times \Sigma \times Q$ set of *labelled transitions* (transition relation)

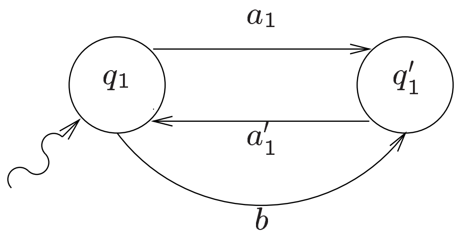
$I \subseteq Q$ set of initial states

$\delta_a = \{(q, q') \mid (q, a, q') \in \delta\}$ set of a-*transitions*

a $\mathbf{en}_{\mathcal{C}}$ q if $\exists q' \in Q : (q, q') \in \delta_a$

Example Component Automaton

$\mathcal{C}_1$:



$a_1, a_1' \in \Sigma_{1,int}$

$b \in \Sigma_{1,out}$

$b \ \mathbf{en}_{\mathcal{C}_1} \ q_1$

$w = q_1 b q_1' a_1' q_1 a_1 q_1' a_1' q_1 b q_1' \in \mathsf{C}_{\mathcal{C}}$
(w *computation* of $\mathcal{C}$)

input record of w is empty

output record of w is bb

internal record of w is $a_1' a_1 a_1'$

$bb \in \mathcal{B}_{\mathcal{C}}^{\Sigma_{out}}$ (bb *output behaviour* of $\mathcal{C}$)

$\{\lambda\} = \mathcal{B}_{\mathcal{C}}^{\Sigma_{inp}}$ (no *input behaviour* of $\mathcal{C}$)

Composable System

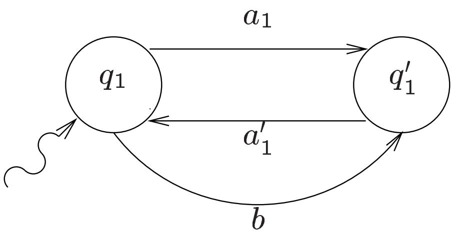
$\mathcal{I} \subseteq \mathbb{N}$ index set

A set of component automata $\mathcal{S} = \{\mathcal{C}_i \mid i \in \mathcal{I}\}$ is a *composable system* if, for all $i \in \mathcal{I}$,

$$\Sigma_{i,int} \cap \bigcup_{j \in \mathcal{I} \setminus \{i\}} \Sigma_j = \varnothing$$

Example Component Automata

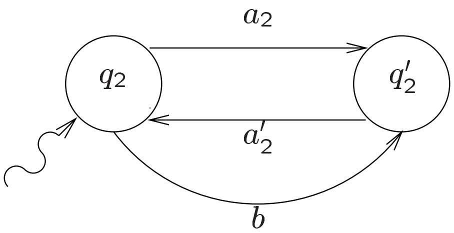
$\mathcal{C}_1$:



$$a_1, a_1' \in \Sigma_{1,int}$$

$$b \in \Sigma_{1,out}$$

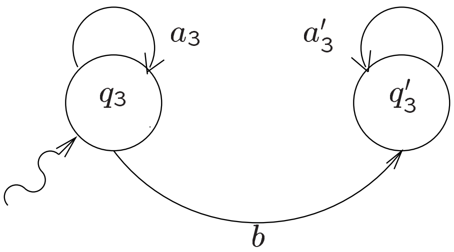
$\mathcal{C}_2$:



$$a_2, a_2' \in \Sigma_{2,int}$$

$$b \in \Sigma_{2,out}$$

$\mathcal{C}_3$:

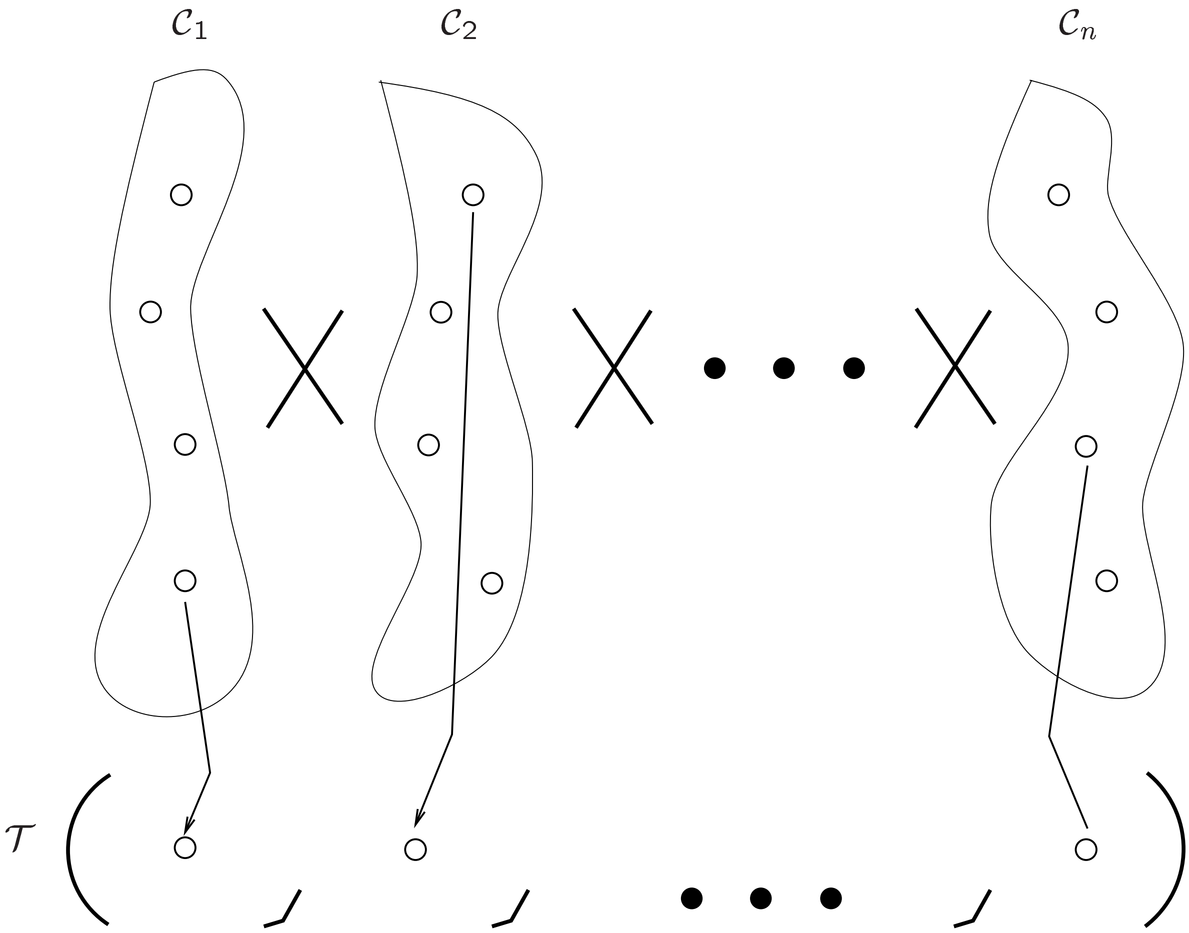


$$a_3, a_3' \in \Sigma_{3,int}$$

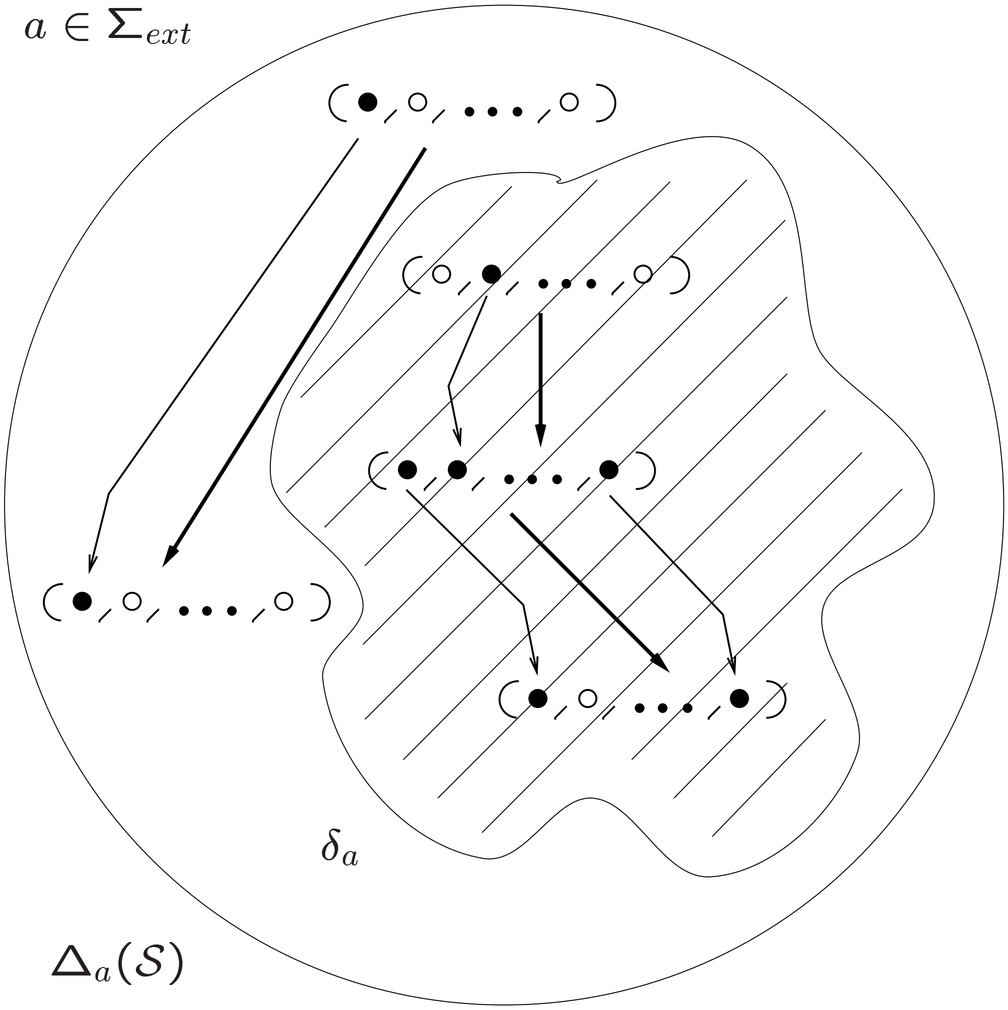
$$b \in \Sigma_{3,\boldsymbol{inp}}$$

$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ is a composable system

State Space



Transition Space



Complete Transition Space

The *complete transition space* of $a \in \bigcup_{i\in\mathcal{I}} \Sigma_i$ in $\mathcal{S} = \{\mathcal{C}_i \mid i \in \mathcal{I}\}$ is

$$\begin{aligned}\Delta_a(\mathcal{S}) = \{(q,q') \in \prod_{i\in\mathcal{I}} Q_i \times \prod_{i\in\mathcal{I}} Q_i \mid \\ \exists j \in \mathcal{I} :\ & (\mathrm{proj}_j(q), a, \mathrm{proj}_j(q')) \in \delta_j \wedge \\ (\forall i \in \mathcal{I} :\ & [(\mathrm{proj}_i(q), a, \mathrm{proj}_i(q')) \in \delta_i] \vee \\ & [\mathrm{proj}_i(q) = \mathrm{proj}_i(q')])\}\end{aligned}$$

In every team transition at least one component acts according to its δ

All other components are either idle or join in this team transition

Each time one chooses a specific δ, one defines a team automaton

Team Automaton

$$\mathcal{T} = (\prod_{i \in \mathcal{I}} Q_i, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, \prod_{i \in \mathcal{I}} I_i)$$

is a *team automaton over* $\mathcal{S} = \{\mathcal{C}_i \mid i \in \mathcal{I}\}$ if

$\Sigma_{int} = \bigcup_{i \in \mathcal{I}} \Sigma_{i,int}$

$\Sigma_{out} = \bigcup_{i \in \mathcal{I}} \Sigma_{i,out}$

$\Sigma_{inp} = (\bigcup_{i \in \mathcal{I}} \Sigma_{i,inp}) \setminus \Sigma_{out}$

$\delta \subseteq \prod_{i \in \mathcal{I}} Q_i \times \Sigma \times \prod_{i \in \mathcal{I}} Q_i$ such that $\forall a \in \Sigma$

$$\delta_a \subseteq \Delta_a(\mathcal{S})$$

and moreover $\delta_a = \Delta_a(\mathcal{S})$ if $a \in \Sigma_{int}$

<u>Lemma</u>: Every team automaton is a component automaton

Example Team Automaton over $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$

$$\mathcal{T} = \left(\prod_{i \in \{1,2,3\}} Q_i, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, \{(q_1, q_2, q_3)\} \right)$$

$\Sigma_{inp} = \varnothing$

$\Sigma_{out} = \{b\}$

$\Sigma_{int} = \{a_1, a_1', a_2, a_2', a_3, a_3'\}$

δ is defined by

$\delta_a = \Delta_a(\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\})$ for all $a \in \Sigma_{int}$

$\delta_b = \{((q_1, q_2, q_3), (q_1', q_2', q_3'))\}$

$$\begin{aligned} \mathcal{T}' : \quad \delta_b' = \{&((q_1, q_2, q_3), (q_1', q_2', q_3')), \\ &((q_1, q_2, q_3'), (q_1', q_2', q_3'))\} \end{aligned}$$

Subteam

The subteam of $\mathcal{T}$ *determined by* $J \subseteq \mathcal{I}$ is

$$SUB_J(\mathcal{T}) = (\prod_{j \in J} Q_j, (\Sigma_{J,inp}, \Sigma_{J,out}, \Sigma_{J,int}), \delta_J, \prod_{j \in J} I_j)$$

$\Sigma_{J,int} = \bigcup_{j \in J} \Sigma_{j,int}$

$\Sigma_{J,out} = \bigcup_{j \in J} \Sigma_{j,out}$

$\Sigma_{J,inp} = (\bigcup_{j \in J} \Sigma_{j,inp}) \setminus \Sigma_{J,out}$

δ_J is defined by, for all $a \in \Sigma_J = \bigcup_{j \in J} \Sigma_j$,

$$(\delta_J)_a = \{(\mathrm{proj}_J(q), \mathrm{proj}_J(q')) \mid (q, q') \in \delta_a)\} \cap \Delta_a(\{\mathcal{C}_j \mid j \in J\})$$

<u>Lemma</u>: $SUB_J(\mathcal{T})$ is a team automaton over $\{\mathcal{C}_j \mid j \in J\}$

Teams versus Subteams

Lemma: Let $J \subseteq \mathcal{I}$, let $a \in \Sigma_J = \bigcup_{j \in J} \Sigma_j$, and let $X \in \{\mathit{Free}, \mathit{AI}, \mathit{SI}\}$

$$\text{If } a \in X(\mathcal{T}), \text{ then } a \in X(\mathit{SUB}_J(\mathcal{T}))$$

Team to subteam: transition relation of subteam must still satisfy requirement that at least one component (now of the subteam) participates in every subteam transition

Modulo this observation the properties of the team are inherited by the subteam

The other way around is less obvious (extra conditions needed)

Key notions so far

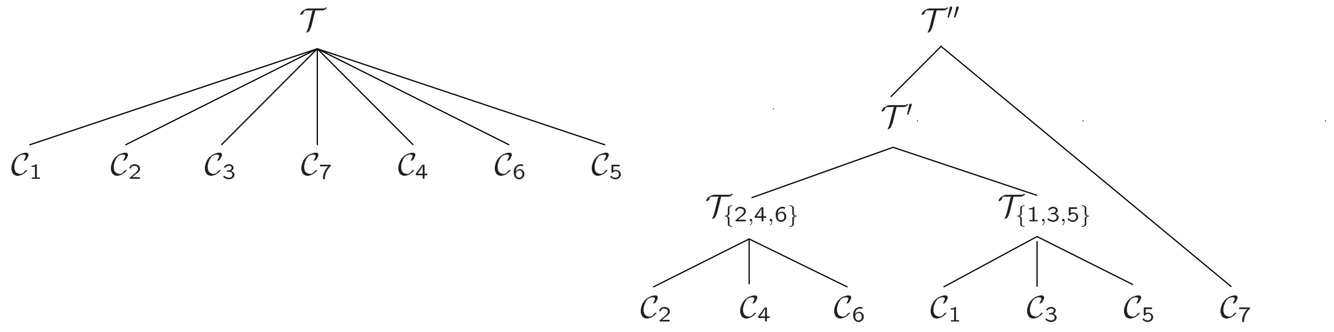
Component automaton

Composable system (set of component automata)

Team automaton (over a composable system)

Subteam (of a team automaton)

Teams over teams



$\mathcal{T}$ team over $\{\mathcal{C}_i \mid i \in \{1, 2, \ldots, 7\}\}$
$\mathcal{T}''$ iterated team over $\{\mathcal{C}_i \mid i \in \{1, 2, \ldots, 7\}\}$

Theorem: after "reordering" elements of $\mathcal{T}''$:

state space $\mathcal{T}''$ = state space $\mathcal{T}$

alphabets $\mathcal{T}''$ = alphabets $\mathcal{T}$ (incl. partition)

initial state space $\mathcal{T}''$ = initial state space $\mathcal{T}$

(potentially the same transition relation)

In fact, also $\mathcal{T}''$ can be considered as a team over $\{\mathcal{C}_i \mid i \in \{1, 2, \ldots, 7\}\}$

Synchronizations

In $\mathcal{T}$ action a is

free if no a-transition of $\mathcal{T}$ contains a simultaneous execution of a by two or more components

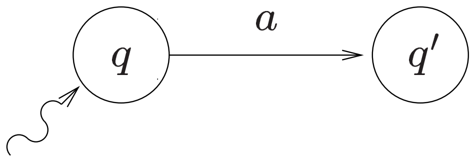
ai (*action-indispensable*) if all components which have a as an action participate in every a-transition of $\mathcal{T}$ (blocking can occur!)

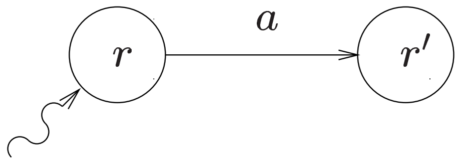
si (*state-indispensable*) if all a-transitions of $\mathcal{T}$ involve all components in which a is enabled in the current state (no blocking can occur!)

Lemma: (1) $\mathit{AI}(\mathcal{T}) \subseteq \mathit{SI}(\mathcal{T})$
(2) $\Sigma_{int} \subseteq \mathit{Free}(\mathcal{T}) \cap \mathit{AI}(\mathcal{T})$

Example Synchronizations

$\mathcal{C}_1$: (a external) $\mathcal{C}_2$:





$\mathcal{T}$	a *free*	*ai*		*si*
$\delta = \{((q,r),a,(q',r'))\}$	$-$	$+$	$\Rightarrow$	$+$
$\delta = \{((q,r),a,(q',r))\}$	$+$	$-$	$\Leftarrow$	$-$
$\delta = \{((q,r'),a,(q',r'))\}$	$+$	$-$		$+$
$\delta = \{((q,r),a,(q',r)),$ $((q,r),a,(q',r'))\}$	$-$	$-$	$\Leftarrow$	$-$

Till now synchronizations of one type of action

Now more detail: what type of action synchronizes with what type of action?

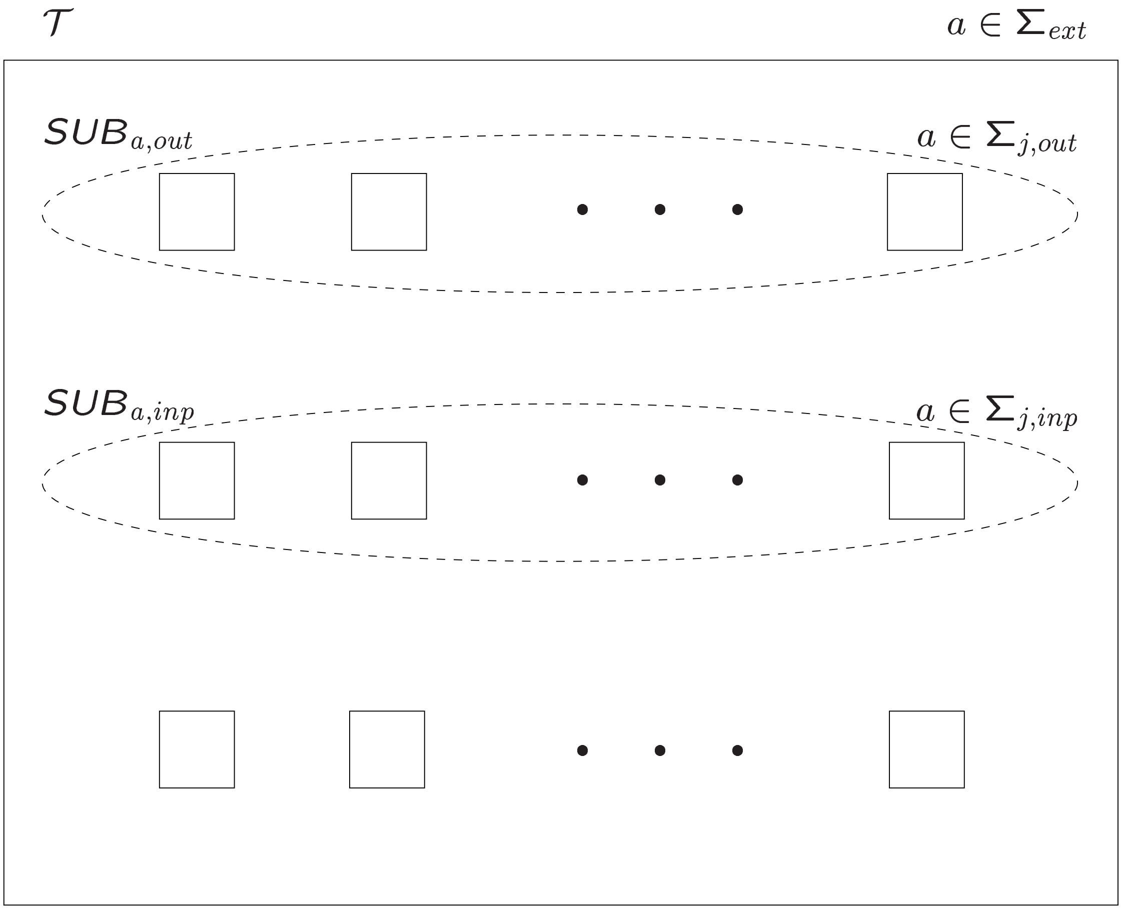
Internal actions never synchronize
Synchronizations thus combinations of external actions

More detailed look leads to classification of types of synchronization

Moreover, we find types of synchronization that actually occur in practice

Thus obtain formal definitions of notions from groupware

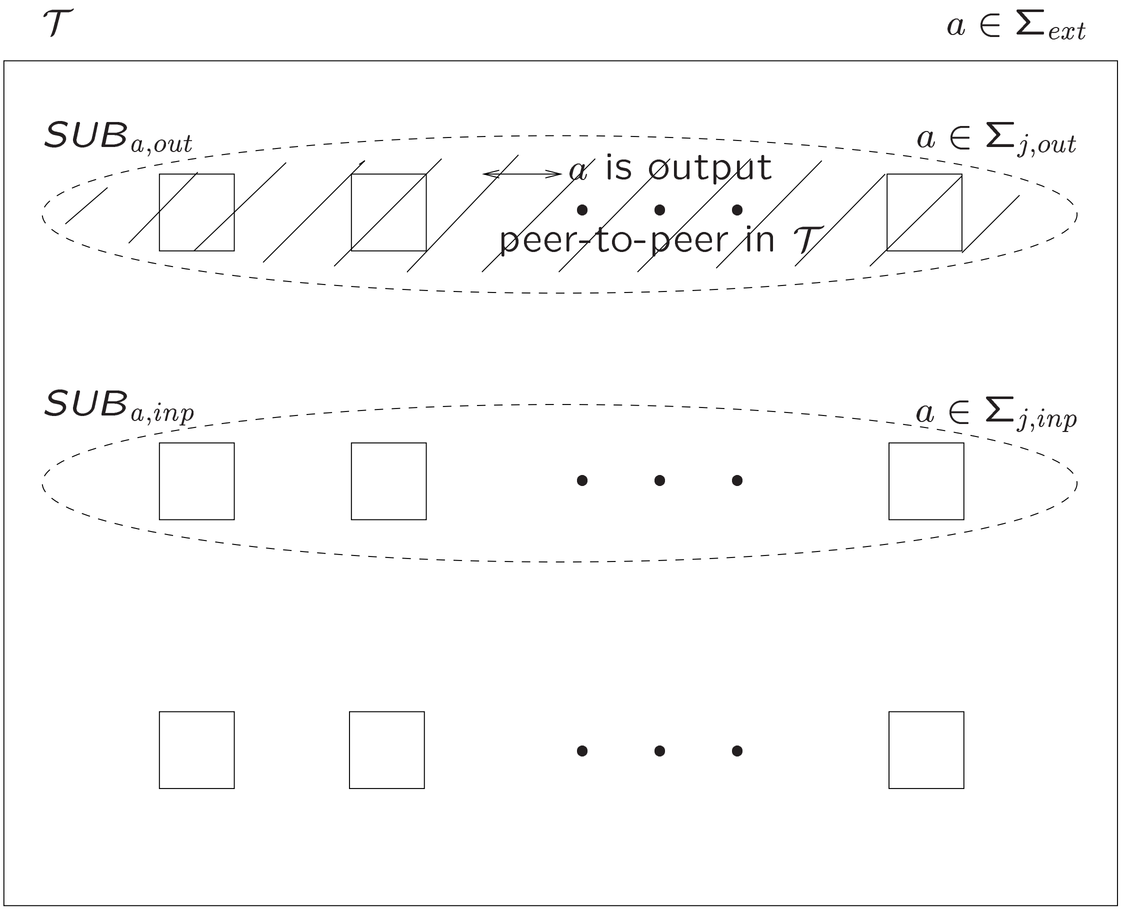
More Synchronizations



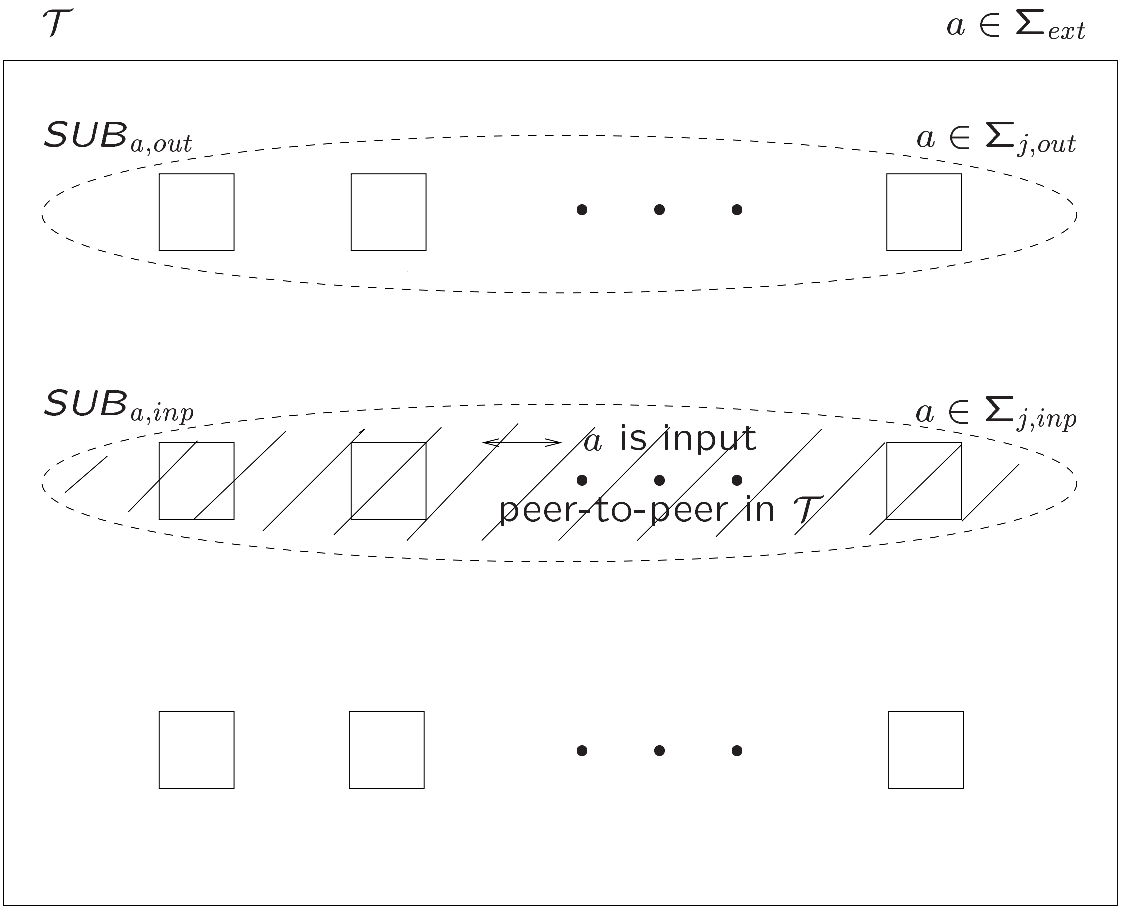
$\mathcal{I}_{a,out}(\mathcal{S}) = \{j \in \mathcal{I} \mid a \in \Sigma_{j,out}\}$ - *output domain*
$\mathcal{I}_{a,inp}(\mathcal{S})$ is the *input domain* of a in $\mathcal{S}$

$SUB_{a,out}$ for $SUB_{\mathcal{I}_{a,out}}(\mathcal{T})$ - *output subteam*
$SUB_{a,inp}$ for the *input subteam* of a (in $\mathcal{T}$)

Output Peer-to-Peer



Input Peer-to-Peer



Strong and Weak Peer-to-Peer

peer-to-peer

strong	weak
ai	*si*
$a \in \mathit{AI}(\mathit{SUB}_{a,out})$	$a \in \mathit{SI}(\mathit{SUB}_{a,out})$

a strong output peer-to-peer if $a \in \mathit{AI}(\mathit{SUB}_{a,out})$

a weak output peer-to-peer if $a \in \mathit{SI}(\mathit{SUB}_{a,out})$

Likewise strong/weak input peer-to-peer

Example Peer-to-Peer

b *si* and *ai* in $\mathcal{T}$, but

b *si* and not *ai* in $\mathcal{T}'$

$\mathcal{I}_{b,inp} = \{3\}$ $\qquad$ $\mathcal{I}_{b,out} = \{1,2\}$

$$SUB_{\{1,2\}}(\mathcal{T}) = SUB_{\{1,2\}}(\mathcal{T}')$$

$$SUB_{\{3\}}(\mathcal{T}) = SUB_{\{3\}}(\mathcal{T}')$$

By Lemma: b also *si* and *ai* in $\mathit{SUB}_{\{1,2\}}(\mathcal{T})$
and $\mathit{SUB}_{\{3\}}(\mathcal{T})$

$\Rightarrow$ b strong output peer-to-peer in $\mathcal{T}$ and
b strong input peer-to-peer in $\mathcal{T}$

By equality of input/output subteam $\mathcal{T}$ and $\mathcal{T}'$:

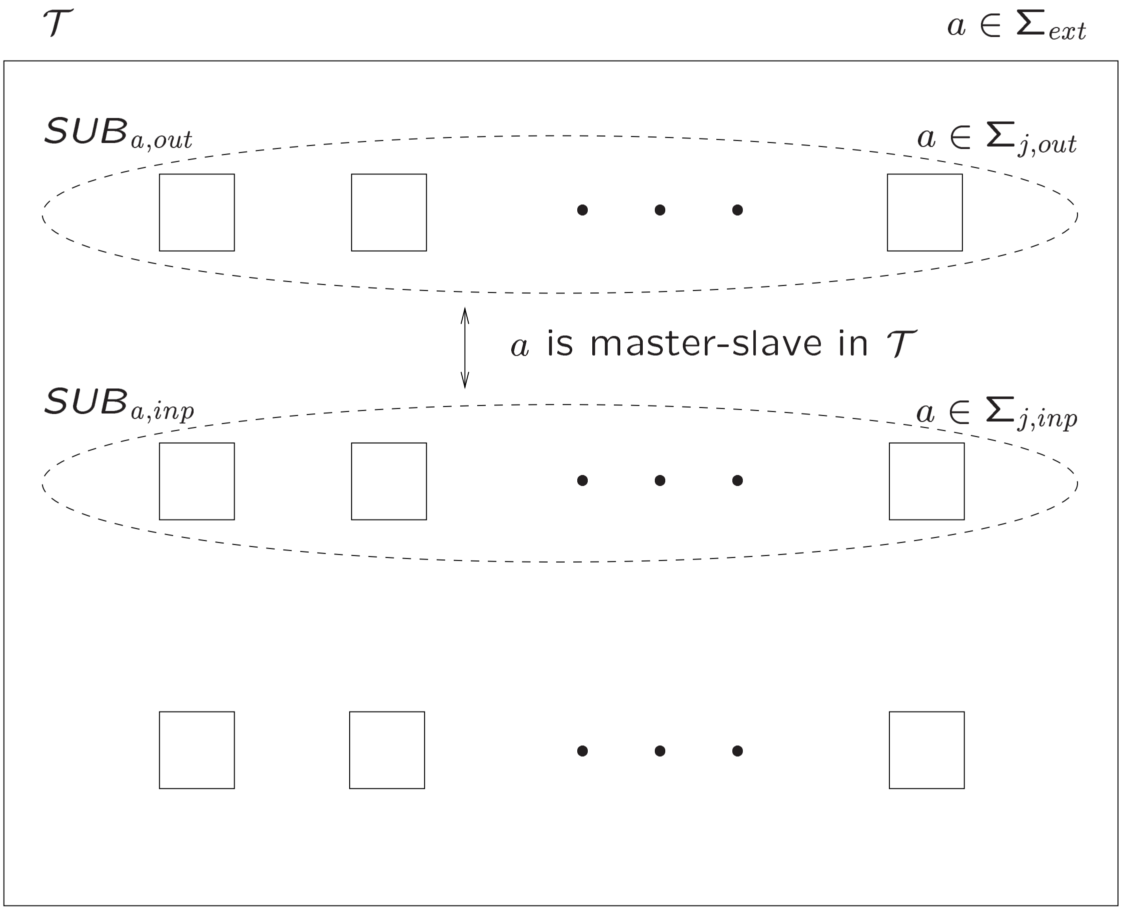
b *si* and *ai* in $\mathit{SUB}_{\{1,2\}}(\mathcal{T}')$ and $\mathit{SUB}_{\{3\}}(\mathcal{T}')$

$\Rightarrow$ b strong output peer-to-peer in $\mathcal{T}'$
b strong input peer-to-peer in $\mathcal{T}'$

Master-Slave

In $\mathcal{T}$ action a is

- *master-slave* if $SUB_{a,out}$ participates in every a-transition of $\mathcal{T}$
  i.e. a never participates only as input action in a synchronization, a slave never proceeds on its own



Master-Slave Continued

In $\mathcal{T}$ action a is

- *strong master-slave* if a master-slave in $\mathcal{T}$ <u>and</u>
  SUB$_{a,inp}$ has to participate in every a-transition of $\mathcal{T}$
  i.e. slave has to obey the master,
  has to in <u>*ai*</u> sense

- *weak master-slave* if a master-slave in $\mathcal{T}$ <u>and</u>
  SUB$_{a,inp}$ has to participate in every a-transition of $\mathcal{T}$ <u>if</u> it is enabled
  i.e. slave has to obey the master,
  has to in <u>*si*</u> sense

Example Master-Slave

(1) b master-slave in $\mathcal{T}$ and $\mathcal{T}'$ since

<u>only</u>
b-transition of $\mathcal{T}$:

$$\begin{tikzcd}
(\underline{q_1, q_2}, q_3) \arrow[r, "b"] \arrow[r, bend right=60, "b"'] & (\underline{q'_1, q'_2}, q'_3)
\end{tikzcd}$$

idem $\mathcal{T}'$

$SUB_{\{1,2\}}(\mathcal{T})$

(2) b strong master-slave in $\mathcal{T}$ since (1) and

<u>only</u>
b-transition of $\mathcal{T}$:

$$\begin{tikzcd}
(q_1, q_2, \underline{q_3}) \arrow[r, "b"] \arrow[r, bend right=60, "b"'] & (q'_1, q'_2, \underline{q'_3})
\end{tikzcd}$$

$b \in \Sigma_{3,inp}$

$SUB_{\{3\}}(\mathcal{T})$

$\Rightarrow$ b also weak master-slave in $\mathcal{T}$

Example Master-Slave Continued

(3) b weak master-slave in $\mathcal{T}'$ since (1) and

$\underline{\text{only}}$ in $(q_1, q_2, \underline{q_3}) \xrightarrow{b} (q'_1, q'_2, \underline{q'_3})$ $\quad b\ \mathbf{en}_{\mathcal{C}_3}\ q_3$

and indeed

$$q_3 \xrightarrow[SUB_{\{3\}}(\mathcal{T}')]{b} q'_3$$

(4) b not strong master-slave in $\mathcal{T}'$ since

in $(q_1, q_2, \underline{q'_3}) \xrightarrow{b} (q'_1, q'_2, \underline{q'_3})$

$$q'_3 \xrightarrow[SUB_{\{3\}}(\mathcal{T}')]{\cancel{b}} q'_3$$

$b \in \Sigma_{3,inp}$

$b\ \cancel{\mathbf{en}_{\mathcal{C}_3}}\ q_3$

Further Topics

- On architectural level our model forms a setup for comparing groupware systems

- Definition of the unique team automaton over a composable system

- "Tools" for really using team automata as application in groupware: hiding, renaming, closing, ...

- Rest of the paper presented next semester

Ellis about Example

b peer-to-peer between $\mathcal{C}_1$ and $\mathcal{C}_2$, because b output action of both

b not occur unless both $\mathcal{C}_1$ and $\mathcal{C}_2$ in initial state, and both voluntarily decide to execute b

$\mathcal{C}_3$ involved in b as slave automaton if in its initial state, because b input action to $\mathcal{C}_3$

Thus, $\mathcal{C}_3$ stuck in q_3 until its two masters agree to both execute b

At this time, $\mathcal{C}_3$ forced to transition into q'_3

In this state, $\mathcal{C}_3$ does not respond to b, although b *may occur many times*

Thus, $\mathcal{C}_3$ now free of its two masters; and also $\mathcal{C}_3$ can never return to its initial slave state

Loops

$\mathcal{T}$: $(p, q, r) \xrightarrow{a} (p', q, r)$

$\mathcal{C}_1$: $p \xrightarrow{a} p'$ — $\mathcal{C}_1$ executed a

$\mathcal{C}_2$: q — $\mathcal{C}_2$ cannot execute a

$\mathcal{C}_3$: $\begin{tikzcd} r \arrow[loop right, "a"] \end{tikzcd}$ — did $\mathcal{C}_3$ execute a?