Team Automata

A Formal Approach to the Modeling of Collaboration Between System Components

Maurice H. ter Beek

ISTI-CNR, Pisa

Monday 8 December 2003

LIACS, Leiden

Outline

Background & Origins of Team Automata

Thesis Goals:

(a) precisely defining team automata

(b) studying their characteristics

(c) comparing them with other models

(d) begin applying them in practice

Some Current & Future Work

Background of Team Automata

Computer Supported Cooperative Work is:

? how do groups of people work together

? how can technology (groupware) assist

CSCW needs:

Conceptual level:
define concepts & terminology consistently

Architectural level:
describe & compare groupware systems

$\Rightarrow$ Ellis introduced idea of *team automata* for modeling both levels informally in

Team Automata for Groupware Systems

(GROUP'97 - International ACM Conference on Supporting Group Work, 1997)

Origins of Team Automata

Theoretical source of inspiration:

Input/Output automata – of Lynch & Tuttle

Conceptual source of inspiration:

Collective Intelligence in Computer Based Collaboration – by Smith

(Computers, Cognition and Work series, 1994)

"well-structured groups [i.e. teams] outperform individuals in certain tasks" – but: also calls for "models capturing concepts of group behavior"

Thesis shows:

usefulness of team automata not limited to modeling groupware notions & systems – but:

extends to modeling collaboration in reactive, distributed systems in general !

(a) Precisely Defining Team Automata

[BEKR03] ter Beek, Ellis, Kleijn & Rozenberg

Synchronizations in Team Automata for Groupware Systems

(Computer Supported Cooperative Work—the Journal of Collaborative Computing, 2003)

Team automata:

- model logical architecture of a design
- abstract from concrete data & actions
- describe behavior in terms of
  - state-action diagram (automaton)
  - role of actions (input, output, internal)
  - synchronizations (simultaneous execution of common actions)

Component Automaton

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

Q set of *states*

$\Sigma = \Sigma_{inp} \cup \Sigma_{out} \cup \Sigma_{int}$ *alphabet* (partition)

$\delta \subseteq Q \times \Sigma \times Q$ *transition relation* $\qquad q \xrightarrow{a} q'$

$I \subseteq Q$ set of *initial states*

$$\left.\begin{array}{l}\Sigma_{inp} \text{ input actions} \\ \Sigma_{out} \text{ output actions}\end{array}\right\} \Sigma_{ext} \text{ externally observable}$$

Σ_{int} *internal actions* $\qquad$ cannot be observed

$\delta_a = \{(q, q') \mid (q, a, q') \in \delta\}$ set of a-*transitions*
(i.e. all arcs labeled with a)

a **en**$_{\mathcal{C}}$ q if $\exists q' \in Q : (q, q') \in \delta_a$
(i.e. a *enabled* at q)

Composable System

a set $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ of component automata is a *composable system* if $\forall\, i \in \{1, \ldots, n\}$:

$$\Sigma_{i,int} \cap \bigcup_{j \in \{1,\ldots,n\} \setminus \{i\}} \Sigma_j = \varnothing$$

$\Rightarrow$ internal actions are private (i.e. uniquely associated to one component automaton)

Fix: $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ is a composable system

? how to form a team automaton over a composable system

Complete Transition Space

The *complete transition space* of $a \in \Sigma = \bigcup_{i\in\{1,\ldots,n\}}(\Sigma_{i,inp} \cup \Sigma_{i,out} \cup \Sigma_{i,int})$ in $\mathcal{S}$ is

$$\Delta_a(\mathcal{S}) = \{(q, q') \in \prod_{i\in\{1,\ldots,n\}} Q_i \times \prod_{i\in\{1,\ldots,n\}} Q_i \mid$$

$$\exists j \in \{1, \ldots, n\} : \; (\mathbf{proj}_j(q), a, \mathbf{proj}_j(q')) \in \delta_j \wedge$$

$$\forall i \in \{1, \ldots, n\} : \; (\mathbf{proj}_i(q), a, \mathbf{proj}_i(q')) \in \delta_i \;\vee \\ \mathbf{proj}_i(q) = \mathbf{proj}_i(q')\}$$

in every team transition <u>at least one</u> component acts <u>according</u> to its transition relation

all other components either <u>join</u> or are <u>idle</u>

Team Automaton

$$\mathcal{T} = (\prod_{i \in \{1,\ldots,n\}} Q_i, (\Sigma_{inp}, \Sigma_{out}, \Sigma_{int}), \delta, \prod_{i \in \{1,\ldots,n\}} I_i)$$

is a *team automaton composed over* $\mathcal{S}$ if

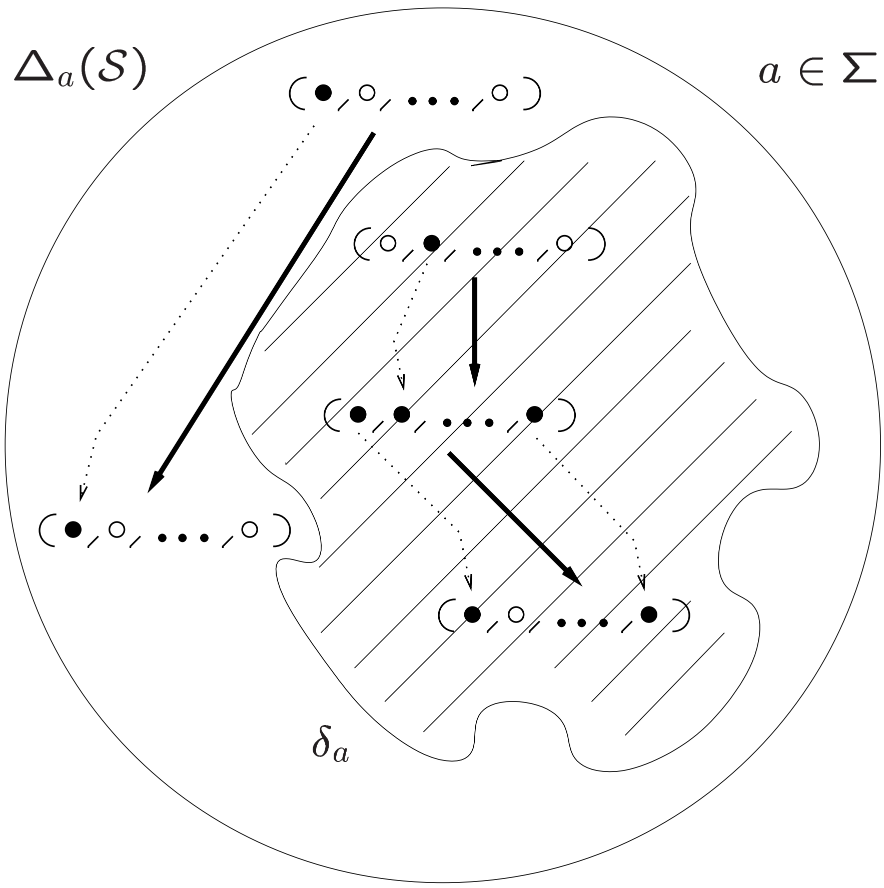
$$\left.\begin{array}{l} \Sigma_{int} = \bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,int} \\[1em] \Sigma_{out} = \bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,out} \\[1em] \Sigma_{inp} = (\bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,inp}) \setminus \Sigma_{out} \end{array}\right\} = \Sigma$$

$\delta \subseteq \prod_{i\in\{1,\ldots,n\}} Q_i \times \Sigma \times \prod_{i\in\{1,\ldots,n\}} Q_i$ such that

$\forall a \in \Sigma$: $\quad \delta_a \subseteq \Delta_a(\mathcal{S})$
$\quad\quad \& \ \ \delta_a = \Delta_a(\mathcal{S})$ if $a \in \Sigma_{int}$

$\Rightarrow$ every team is a component & thus can be used to iteratively compose team automata !

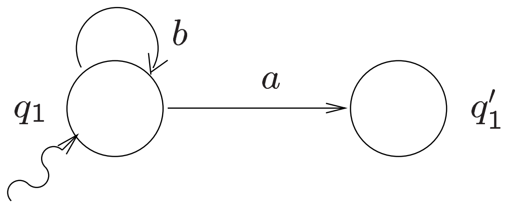
Transition Space of Team Automaton

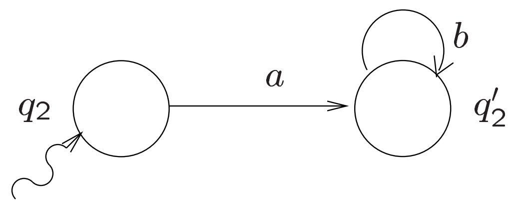


$\Rightarrow$ each <u>choice</u> of a team transition relation δ_a, $\forall a \in \Sigma$, defines a specific team automaton !

Example Component & Team Automata

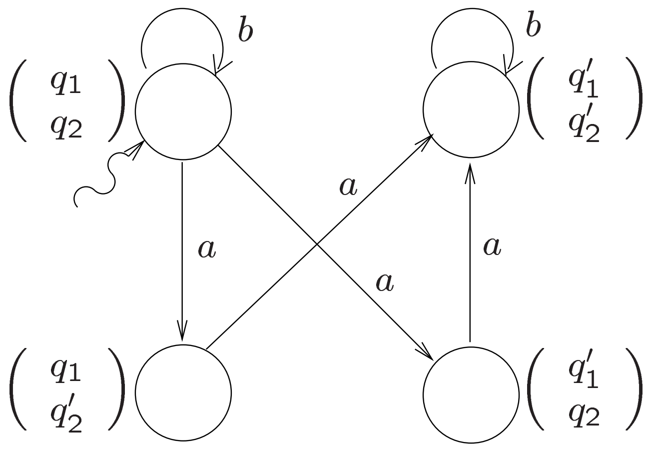
$\mathcal{C}_1$: $\qquad\qquad$ $\mathcal{C}_2$:

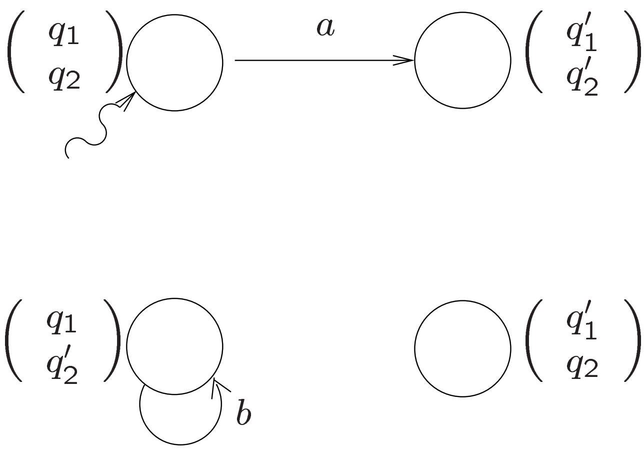




$\Rightarrow$ team automata $\mathcal{T}^{free}$ & $\mathcal{T}^{ai}$ over $\{\mathcal{C}_1, \mathcal{C}_2\}$ are defined by choosing their transitions!

$\mathcal{T}^{free}$: $\qquad\qquad$ $\mathcal{T}^{ai}$:





? who participates in $((q_1, q_2'), b, (q_1, q_2'))$ of $\mathcal{T}^{ai}$: only $\mathcal{C}_1$ or only $\mathcal{C}_2$ or both (loop problem)

$\Rightarrow$ adopt maximal interpretation: both execute b!

Computations...

$$\alpha = \binom{q_1}{q_2} a \binom{q_1}{q_2'} a \binom{q_1'}{q_2'} \in \mathbf{C}_{\mathcal{T}free}$$

$\pi_{\mathcal{C}_1}(\alpha) = q_1 a q_1' \in \mathbf{C}_{\mathcal{C}_1}$ $\qquad$ $\pi_{\mathcal{C}_2}(\alpha) = q_2 a q_2' \in \mathbf{C}_{\mathcal{C}_2}$

<u>Theorem</u>: if $\mathcal{T}$ is a team automaton over $\mathcal{S}$, then

$$\forall i \in \{1, \ldots, n\} : \ \pi_{\mathcal{C}_i}(\mathbf{C}_{\mathcal{T}}) \subseteq \mathbf{C}_{\mathcal{C}_i}$$

...and Behavior

pres$_{\{a,b\}}(\alpha) = aa \in \mathbf{B}_{\mathcal{T}free}$ $\qquad$ $aa \notin \mathbf{B}_{\mathcal{C}_1} \cup \mathbf{B}_{\mathcal{C}_2}$

$ab \notin \mathbf{B}_{\mathcal{T}free} \cup \mathbf{B}_{\mathcal{T}ai}$ $\qquad$ $ab \in \mathbf{B}_{\mathcal{C}_2}$

Subteam

The subteam of $\mathcal{T}$ determined by $J \subseteq \{1,\ldots,n\}$ is

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

$$\left.\begin{aligned} \Sigma_{J,int} &= \textstyle\bigcup_{j\in J} \Sigma_{j,int} \\ \Sigma_{J,out} &= \textstyle\bigcup_{j\in J} \Sigma_{j,out} \\ \Sigma_{J,inp} &= (\textstyle\bigcup_{j\in J} \Sigma_{j,inp}) \setminus \Sigma_{J,out} \end{aligned}\right\} = \Sigma_J$$

δ_J is defined by, $\forall\, a \in \Sigma_J$,

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

$\Rightarrow$ $SUB_J(\mathcal{T})$ is a team automaton over $\{\mathcal{C}_j \mid j \in J\}$!

Key Notions so far

Component automaton

Composable system (of component automata)

Team automaton (over a composable system)

Subteam (of a team automaton)

So far Published only in Thesis

Properties of Team Automata:

- action, transition & state reduced versions
- enabling
- determinism

(b) Studying Team Automata Characteristics

[BEKR03]

&

[BK03] ter Beek & Kleijn

Team Automata Satisfying Compositionality

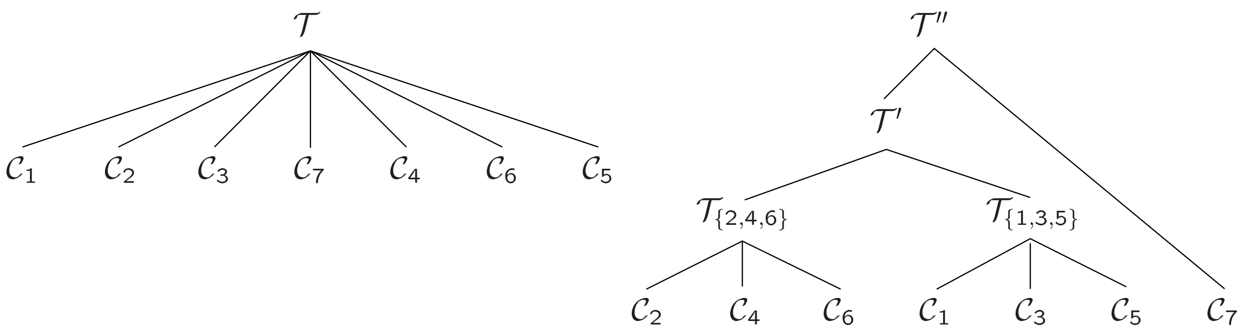
(FM 2003 - International Symposium of Formal Methods, 2003)

<u>Characteristics of:</u>

- iterated team automata
- synchronizations in team automata
- fixed team automata
- behavior of team automata

Teams over Teams

Recall: each team automaton can itself be used as a component automaton



$\mathcal{T}$ team automaton over $\{\mathcal{C}_1, \ldots, \mathcal{C}_7\}$
$\mathcal{T}''$ *iterated* team automaton over $\{\mathcal{C}_1, \ldots, \mathcal{C}_7\}$

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

$\mathcal{T}''$ team automaton over $\{\mathcal{C}_1, \ldots, \mathcal{C}_7\}$

Synchronizations: free, ai & si

In team automaton $\mathcal{T}$ over $\mathcal{S}$ an action a is:

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

ai (*action-indispensable*) if all components with 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) $AI(\mathcal{T}) \subseteq SI(\mathcal{T})$
(2) $\Sigma_{int} \subseteq Free(\mathcal{T}) \cap AI(\mathcal{T})$

Fixed Team Automata: free, ai & si

$\mathcal{T}$ is <u>the</u> *maximal-free team automaton* over $\mathcal{S}$ if

$$\delta_a = \{(q,q') \in \Delta_a(\mathcal{S}) \mid \#\{i \in \{1,\ldots,n\} \mid a \in \Sigma_i \wedge (\mathbf{proj}_i(q), \mathbf{proj}_i(q')) \in \delta_{i,a}\} = 1\} \ \forall a \in \Sigma$$

$\mathcal{T}$ is <u>the</u> *maximal-ai team automaton* over $\mathcal{S}$ if

$$\delta_a = \{(q,q') \in \Delta_a(\mathcal{S}) \mid \forall i \in \{1,\ldots,n\} : a \in \Sigma_i \Rightarrow (\mathbf{proj}_i(q), \mathbf{proj}_i(q')) \in \delta_{i,a}\} \ \forall a \in \Sigma$$

$\mathcal{T}$ is <u>the</u> *maximal-si team automaton* over $\mathcal{S}$ if

$$\delta_a = \{(q,q') \in \Delta_a(\mathcal{S}) \mid \forall i \in \{1,\ldots,n\} : (a \in \Sigma_i \wedge a \ \mathbf{en}_{\mathcal{C}_i} \ \mathbf{proj}_i(q)) \Rightarrow (\mathbf{proj}_i(q), \mathbf{proj}_i(q')) \in \delta_{i,a}\} \ \forall a \in \Sigma$$

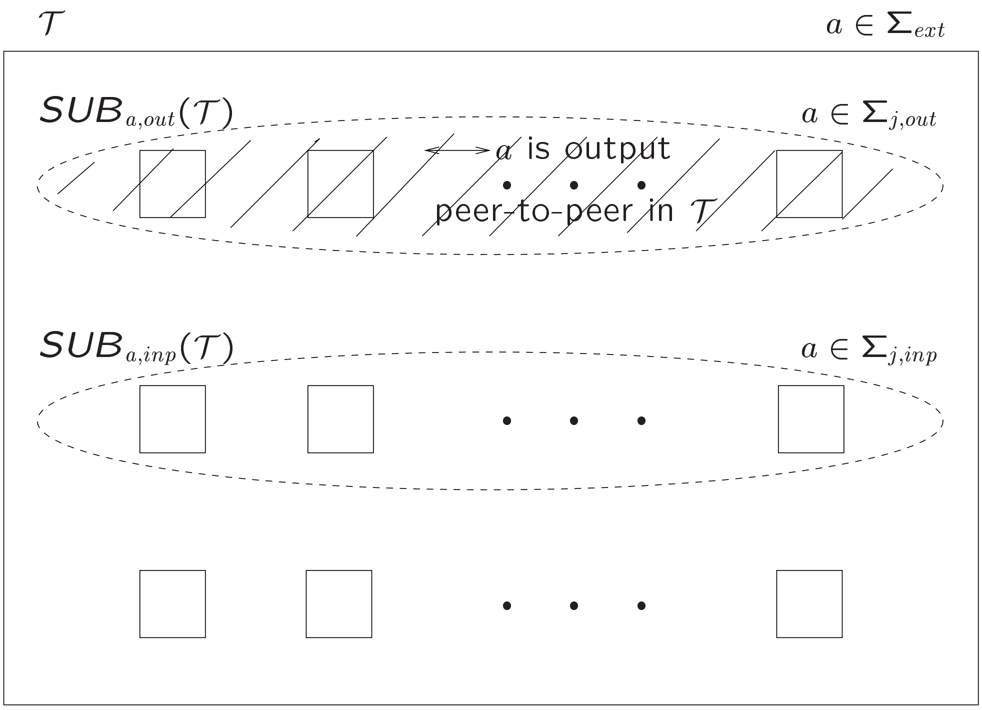
$\Rightarrow$ δ_a is the largest & hence unique transition relation in $\Delta_a(\mathcal{S})$ by which a is *free* / *ai* / *si* !

$\Rightarrow$ maximal interpretation: $((q_1,q_2'),b,(q_1,q_2'))$ is contained in $\mathcal{T}^{ai}$, but not in $\mathcal{T}^{free}$!

Synchronizations: peer-to-peer

collaboration of peers (input or output)

Action a is *strong (weak) output peer-to-peer* in $\mathcal{T}$ if a is *ai* (*si*) in $SUB_{a,out}(\mathcal{T})$



Analogous:

strong (weak) input peer-to-peer

maximal-peer-to-peer team automaton

Synchronizations: master-slave

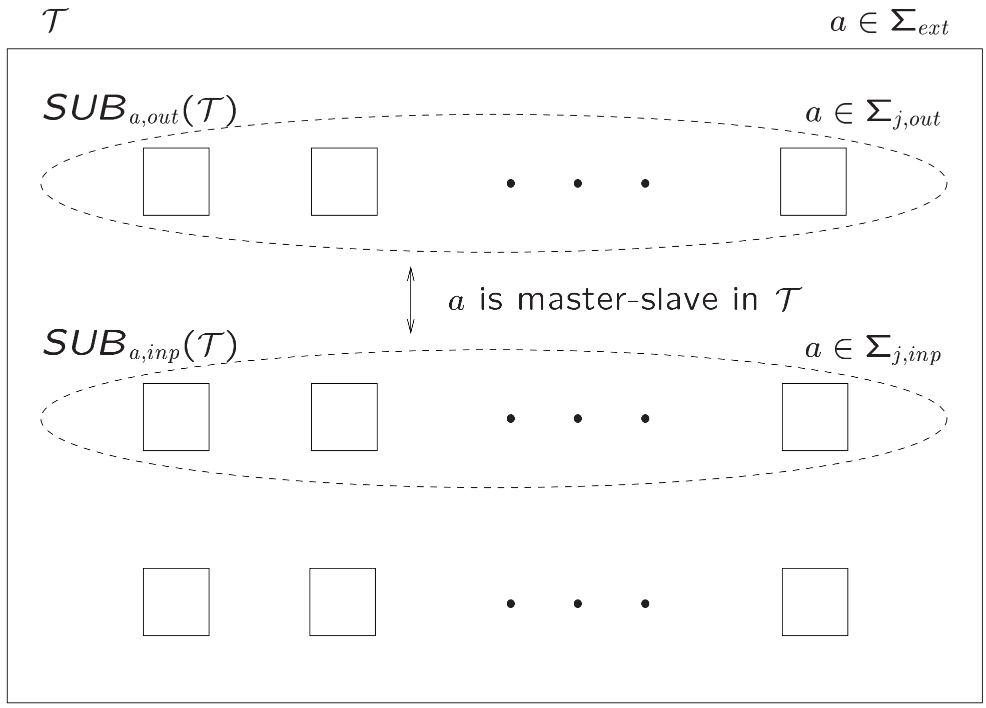
cooperation of masters (output) & slaves (input)

Action a is *strong (weak) master-slave* in $\mathcal{T}$ if

$SUB_{a,out}$ participates in every a-transition of $\mathcal{T}$
i.e. slaves may never proceed on their own

&

$SUB_{a,inp}$ participates in every a-transition of $\mathcal{T}$
i.e. slaves must obey the master (if enabled)



Analogous *maximal-master-slave team automaton*

Behavior: from Team to Component

We have seen that

Theorem: if $\mathcal{T}$ is a team automaton over $\mathcal{S}$, then

$$\forall\, i \in \{1, \ldots, n\} : \; \pi_{\mathcal{C}_i}(\mathbf{C}_{\mathcal{T}}) \subseteq \mathbf{C}_{\mathcal{C}_i}$$

but also that $aa \in \mathbf{B}_{\mathcal{T}free}$ while $aa \notin \mathbf{B}_{\mathcal{C}_1} \cup \mathbf{B}_{\mathcal{C}_2}$

We do have that

Theorem: if $\mathcal{T}$ is the maximal-ai team automaton over $\mathcal{S}$, then

$$\forall\, i \in \{1, \ldots, n\} : \; \mathbf{pres}_{\Sigma_i}(\mathbf{B}_{\mathcal{T}}) \subseteq \mathbf{B}_{\mathcal{C}_i}$$

$\Rightarrow$ no such result for other fixed team automata !

From Component to Maximal-free Team

First intuition: independent, non-synchronizing components, thus $\forall\, i \in \{1, \ldots, n\} : \mathbf{C}_{\mathcal{C}_i} = \pi_{\mathcal{C}_i}(\mathbf{C}_{\mathcal{T}})$
However: recall loop problem !

Fix: Γ is an arbitrary alphabet

$\mathcal{S}$ is Γ-*loop limited* if $\forall\, i \in \{1, \ldots, n\},\ \forall\, a \in \Gamma \cap \Sigma_i$:

whenever $(q, a, q) \in \delta_i$, for some $q \in Q_i$,

then $\forall\, j \in \{1, \ldots, n\} \setminus \{i\}$: $\delta_{j,a} = \varnothing$

Theorem: if $\mathcal{T}$ is the maximal-free team automaton over $\mathcal{S}$ and $\mathcal{S}$ is Σ-loop limited, then

$$\forall\, i \in \{1, \ldots, n\} : \ \mathbf{C}_{\mathcal{C}_i} = \pi_{\mathcal{C}_i}(\mathbf{C}_{\mathcal{T}})$$

Theorem: if $\mathcal{T}$ is the maximal-free team automaton over $\mathcal{S}$ and $\mathcal{S}$ is Σ-loop limited, then

$$\forall\, i \in \{1, \ldots, n\} : \ \mathbf{B}_{\mathcal{C}_i} \subseteq \mathbf{B}_{\mathcal{T}}$$

From Component to Maximal-ai Team

$\alpha \in \prod_{i\in\{1,\ldots,n\}} \mathrm{C}_{\mathcal{C}_i}$ is *ai-consistent* if $\exists\, w \in \Sigma^*$:

$\forall\, i \in \{1,\ldots,n\}$: $\textbf{pres}_{\Sigma_i}(w) = \textbf{pres}_{\Sigma_i}(\textbf{proj}_i(\alpha))$

$\mathcal{S}$ is *ai-consistent* if $\forall\, i \in \{1,\ldots,n\},\ \forall\, \beta \in \mathrm{C}_{\mathcal{C}_i}$:

$\exists$ ai-consistent $\alpha \in \prod_{i\in\{1,\ldots,n\}} \mathrm{C}_{\mathcal{C}_i}$: $\textbf{proj}_i(\alpha) = \beta$

Theorem: if $\mathcal{T}$ is the maximal-ai team automaton over $\mathcal{S}$, then

$\forall\, i \in \{1,\ldots,n\}$: $\mathrm{C}_{\mathcal{C}_i} = \pi_{\mathcal{C}_i}(\mathrm{C}_{\mathcal{T}})$ iff $\mathcal{S}$ is ai-consistent

Theorem: if $\mathcal{T}$ is the maximal-ai team automaton over $\mathcal{S}$ and $\mathcal{S}$ is ai-consistent, then

$$\forall\, i \in \{1,\ldots,n\} :\ \mathrm{B}_{\mathcal{C}_i} = \textbf{pres}_{\Sigma_i}(\mathrm{B}_{\mathcal{T}})$$

? what about other fixed team automata

(Synchronized) Shuffles

Well-known language-theoretic operations
e.g. formal languages, concurrency

- $u \mid\mid v$ denotes *shuffle* of $u, v \in \Delta^*$

Fix: $\Delta = \{a, b, c, d\}$, $u = abc \in \Delta^*$, $v = cd \in \Delta^*$

Then $u \mid\mid v = \{abccd, acbcd, cabcd, abcdc, acbdc, cabdc, acdbc, cadbc, cdabc\}$

- $u \mid\mid^{\Gamma} v$ denotes *S-shuffle on* Γ of $u, v \in \Delta^*$

Now $u \mid\mid^{\{c\}} v = \{abcd\}$, but $u \mid\mid^{\{b,c\}} v = \varnothing$

Full & Relaxed Synchronized Shuffles

Both obtained from S-shuffle by varying the symbols to be synchronized

- $u \;{}_{\Delta_1}\!\sqcup\!\sqcup_{\Delta_2}\, v$ denotes *fS-shuffle* of $u \in \Delta_1^*$, $v \in \Delta_2^*$

<u>Fix</u>: $\Delta_1 = \{a, b, c\}$, $\Delta_2 = \{c, d\}$

Now $u \;{}_{\Delta}\!\sqcup\!\sqcup_{\Delta}\; v = \varnothing$, but $u \;{}_{\Delta_1}\!\sqcup\!\sqcup_{\Delta_2}\; v = \{abcd\}$

- $u \;{}_{\Delta_1}\!\sqcup\!\sqcup^{\Gamma}_{\Delta_2}\; v$ denotes *rS-shuffle on* Γ of $u \in \Delta_1^*$, $v \in \Delta_2^*$

Now $u \;{}_{\Delta_1}\!\sqcup\!\sqcup^{\{c\}}_{\Delta_2}\; v = u \;{}_{\Delta_1}\!\sqcup\!\sqcup^{\{b,c\}}_{\Delta_2}\; v = \{abcd\}$

$\Rightarrow$ all extended to languages in the usual way

Compositionality

"A specification of a composite system can be obtained from specifications of its components"

$\Rightarrow$ automata-based specification model: behavior !

Theorem: if $\mathcal{T}$ is the *maximal-free* team automaton over $\mathcal{S}$ and $\mathcal{S}$ is Σ-loop limited, then

$$\mathbf{B}_{\mathcal{T}} = ||_{i \in \{1,\ldots,n\}} \mathbf{B}_{\mathcal{C}_i}$$

Theorem: if $\mathcal{T}$ is the *maximal-ai* team automaton over $\mathcal{S}$, then

$$\mathbf{B}_{\mathcal{T}} = \underline{||}_{\{\Sigma_i \,|\, i \in \{1,\ldots,n\}\}} \mathbf{B}_{\mathcal{C}_i}$$

Indeed

$$\mathbf{B}_{\mathcal{T}ai} = \{\lambda, a\} = \{b^n, b^n a \mid n \geq 0\} \;{}_{\Sigma_1}\underline{||}_{\Sigma_2}\; \{\lambda, ab^n \mid n \geq 0\} = \mathbf{B}_{\mathcal{C}_1} \;{}_{\Sigma_1}\underline{||}_{\Sigma_2}\; \mathbf{B}_{\mathcal{C}_2}$$

Heterogeneous Fixed Team Automata

Let $\Gamma' = \Sigma \setminus \Gamma$

&

let $\mathcal{T}'$ be the team automaton over $\mathcal{S}$ that has the largest and hence unique transition relation by which

all its actions in $\Sigma \cap \Gamma$ are *ai*

&

all its actions in Γ' are *free*

Theorem: if $\mathcal{S}$ is Γ'-loop limited, then

$$\mathbf{B}_{\mathcal{T}'} = \underline{||}^{\Gamma}_{\{\Sigma_i \,|\, i \in \{1,\ldots,n\}\}} \mathbf{B}_{\mathcal{C}_i}$$

? what about other fixed team automata

So far Published only in Thesis

Effect of synchronizations:

- inheritence & preservation of properties
  i.e. reducedness, enabling, determinism

Inheritence & preservation of synchronizations

Homogeneous vs. heterogeneous fixed teams

Theory of synchronized shuffles:

- infinitary words & languages
- fair (synchronized) shuffles
- commutativity & associativity

(c) Comparing Team Automata with Other Models

[BEKR03]

&

[BEKR01a] ter Beek, Ellis, Kleijn & Rozenberg

Team Automata for CSCW

(International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, 2001)

Sketchy comparisons with:

- safe Input/Output (I/O) automata
- a model of vector labeled Petri nets

$\Rightarrow$ so far details published only in thesis !

(Safe) I/O Automata

[Tut87] Tuttle (& supervisor Lynch)

Hierarchical Correctness Proofs for Distributed Algorithms

(Master's thesis, 1987)

A *(safe) I/O automaton* is a component automaton $\mathcal{C}$ such that $\mathcal{C}$ is *input enabling*

i.e. $\forall\, a \in \Sigma_{inp},\ \forall\, q \in Q$: $(q, q') \in \delta_a$ for some $q' \in Q$

$\mathcal{S}$ is a *compatible system* if $\forall\, i \in \{1, \ldots, n\}$:

$$\Sigma_{i,out} \cap \bigcup_{j \in \{1,\ldots,n\} \setminus \{i\}} \Sigma_{j,out} = \varnothing$$

$\Rightarrow$ output actions of I/O automata are disjoint

Team I/O Automata

$\mathcal{T}$ is <u>the</u> *team I/O automaton* over $\mathcal{S}$ if:

(1) $\mathcal{S}$ is a compatible system

(2) $\forall\, i \in \{1, \ldots, n\}$: $\mathcal{C}_i$ is a (safe) I/O automaton

(3) $\mathcal{T}$ is the *maximal-ai* team automaton over $\mathcal{S}$

$\Rightarrow$ each team I/O automaton is an I/O automaton !

"It might be interesting to study other notions of composition"

(from the Conclusion of [Tut87])

$\Rightarrow$ team automata extend I/O automata !

Iterated Composition & Subteams

$\mathcal{S}$ is a compatible system
$\mathcal{T}$ is the team I/O automaton over $\mathcal{S}$
$\mathcal{T}'$ is the *iterated* team I/O automaton over $\mathcal{S}$

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

$\mathcal{T}'$ is the team I/O automaton over $\mathcal{S}$

$\Rightarrow$ $SUB_J(\mathcal{T})$ need not be an I/O automaton, but

Theorem: if $J \subseteq \{1,\ldots,n\}$ & $\forall\, a \in \Sigma_{out}$, $\forall\, j$ such that $a \in \Sigma_{j,out}$: $\delta_{j,a} \neq \varnothing$, then

$SUB_J(\mathcal{T})$ is the team I/O automaton over $\{\mathcal{C}_j \,|\, j \in J\}$

$\Rightarrow$ the peer-to-peer & master-slave types of synchronization cannot be distinguished in (team) I/O automata !

Recall: Loop Problem

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

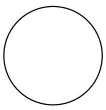
$\mathcal{C}_1$: $p \xrightarrow{a} p'$ — $\mathcal{C}_1$ executed a

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

$\mathcal{C}_3$: $r \circlearrowleft a$ — did $\mathcal{C}_3$ execute a
or was $\mathcal{C}_3$ idle?

Vector Team Automata

$$\begin{pmatrix} a \\ \lambda \\ \end{pmatrix}$$



$\underline{\mathcal{C}_3 \text{ executed } a}$!

From Vector Team Automata to Petri Nets

Fix: $\forall\, i \in \{1, \ldots, n\}$: Q_i & Σ_i are finite

Translation of vector team automaton $\mathcal{T}^v$ into *Individual Token Net Controller* (ITNC) *PN*($\mathcal{T}^v$)

ITNCs are a particular kind of state machine decomposable nets from the theory of VCCSs:

[Kee96] Keesmaat (& supervisors Kleijn & Rozenberg)

Vector Controlled Concurrent Systems

(Ph.D. thesis, 1996)

Theorem: if $\mathcal{T}^v$ is a *non-state-sharing* vector team automaton, then

$$\mathbf{B}_{PN(\mathcal{T}^v)} = \mathbf{B}_{\mathcal{T}^v} \quad \& \quad \mathbf{V}_{PN(\mathcal{T}^v)} = \mathbf{V}_{\mathcal{T}^v}$$

Subnets vs. Subteams

The *subnet* $SUB_J(\mathcal{K})$ *of* ITNC $\mathcal{K}$ *determined by* J is obtained by focusing on a subset J of the set $\{1,\ldots,n\}$ of individual tokens of $\mathcal{K}$

$\Rightarrow$ $SUB_J(\mathcal{K})$ is an ITNC !

Furthermore:

Theorem: if $J \subseteq \{1,\ldots,n\}$, then

$$SUB_J(PN(\mathcal{T}^v)) = PN(SUB_J(\mathcal{T}^v))$$

Further Related Models

Role of actions:

- input, output & internal: I/O automata, I/O systems, Interacting State Machines, Reactive Transition Systems & CCS

Synchronizations:

- *uniform*: I/O automata, mixed product of automata, path expressions & COSY
- *pluriform*: VCCS, free & synchronous product of automata
- other: CCS & (T)CSP

Composition:

- choice: synchronous product of automata
- *ai*-based: I/O automata, ISMs, RTSs, mixed product of automata, cooperating (pushdown) automata, timed cooperating automata, (T)CSP & statecharts

(d) Applying Team Automata

[BEKR03]

&

[HB00] 't Hoen & ter Beek

A Conflict-Free Strategy for Team-Based Model Development

(PDTSD'00 - International Workshop on Process support for Distributed Team-based Software Development, 2000)

&

[BEKR01b] ter Beek, Ellis, Kleijn & Rozenberg

Team Automata for Spatial Access Control

(ECSCW 2001 - European Conference on Computer-Supported Cooperative Work, 2001)

Team automata modeling:

- GROVE document editor architecture
- collaboration between software developers
- access control mechanisms

After the Thesis

[BB03] ter Beek & Bloem

Model Checking Team Automata for Access Control

(unpublished manuscript, 2003)

&

[BLP03] ter Beek, Lenzini & Petrocchi

Team Automata for Security Analysis of Multicast/Broadcast Communication

(WISP'03 - Workshop on Issues in Security and Petri Nets, 2003)

&

[BCM03] ter Beek, Csuhaj-Varjú & Mitrana

Teams of Pushdown Automata

(PSI'03 - International Conference on Perspectives of System Informatics, 2003)

(International Journal of Computer Mathematics, 2004)

&

. . .

Access Control

Information access control mechanisms are a vital component of any (computer) system

CSCW / groupware systems require:

⇒ more refined access rights
e.g. right to scroll a file synchronously edited by a group in real time

⇒ more control of meta access rights
e.g. grant/revoke a team member the right to grant/revoke file access to a non-team member (i.e. delegation)

Spatial Access Control

SPACE metaphor (Bullock et al., 1998)

$\Downarrow$

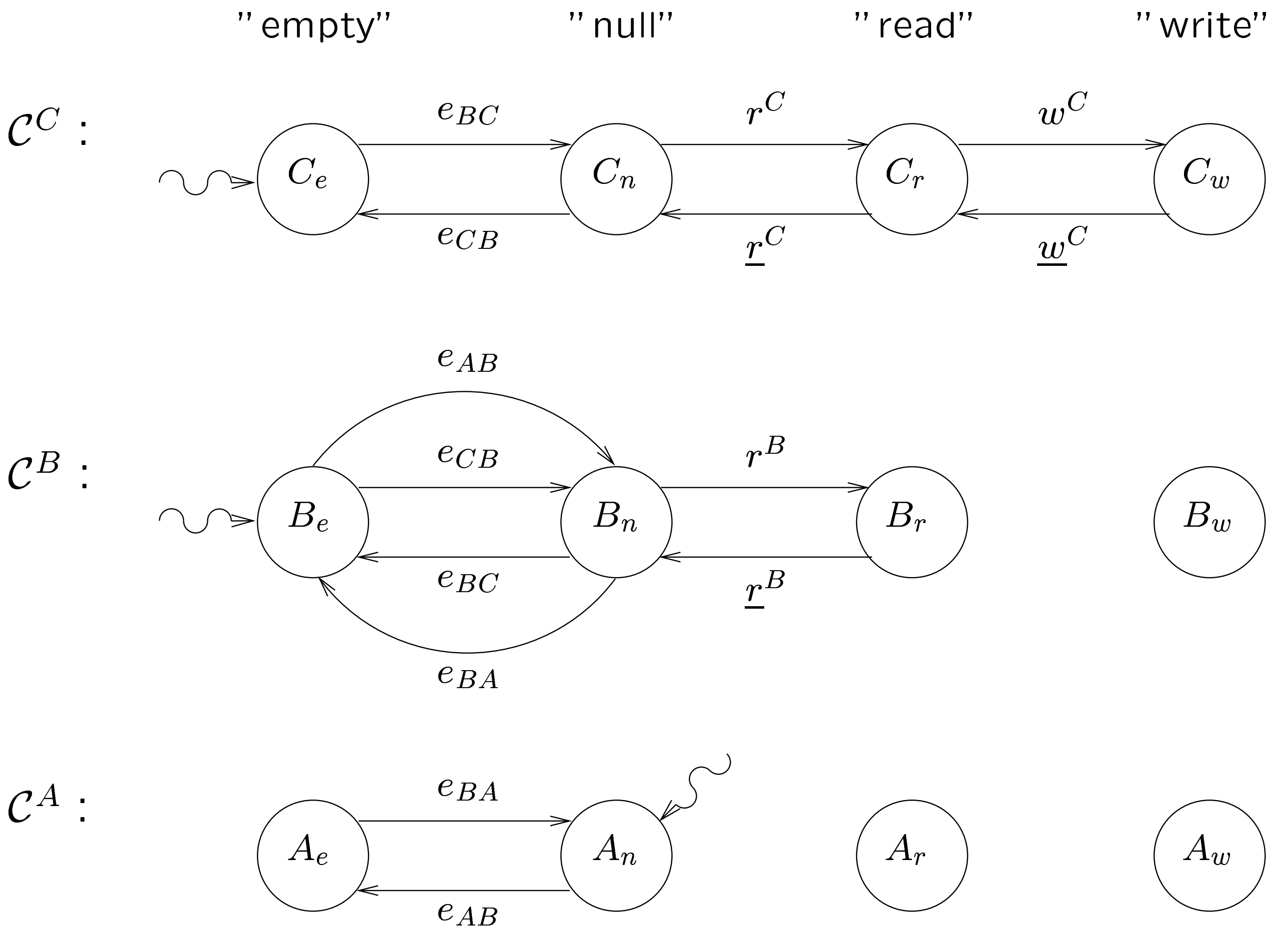
access control governed by spaces (rooms)
in which subjects and objects reside

&

the ability of a subject to traverse space
in order to get closer to an object

Room C: write access room
Room B: read access room
Room A: null access room

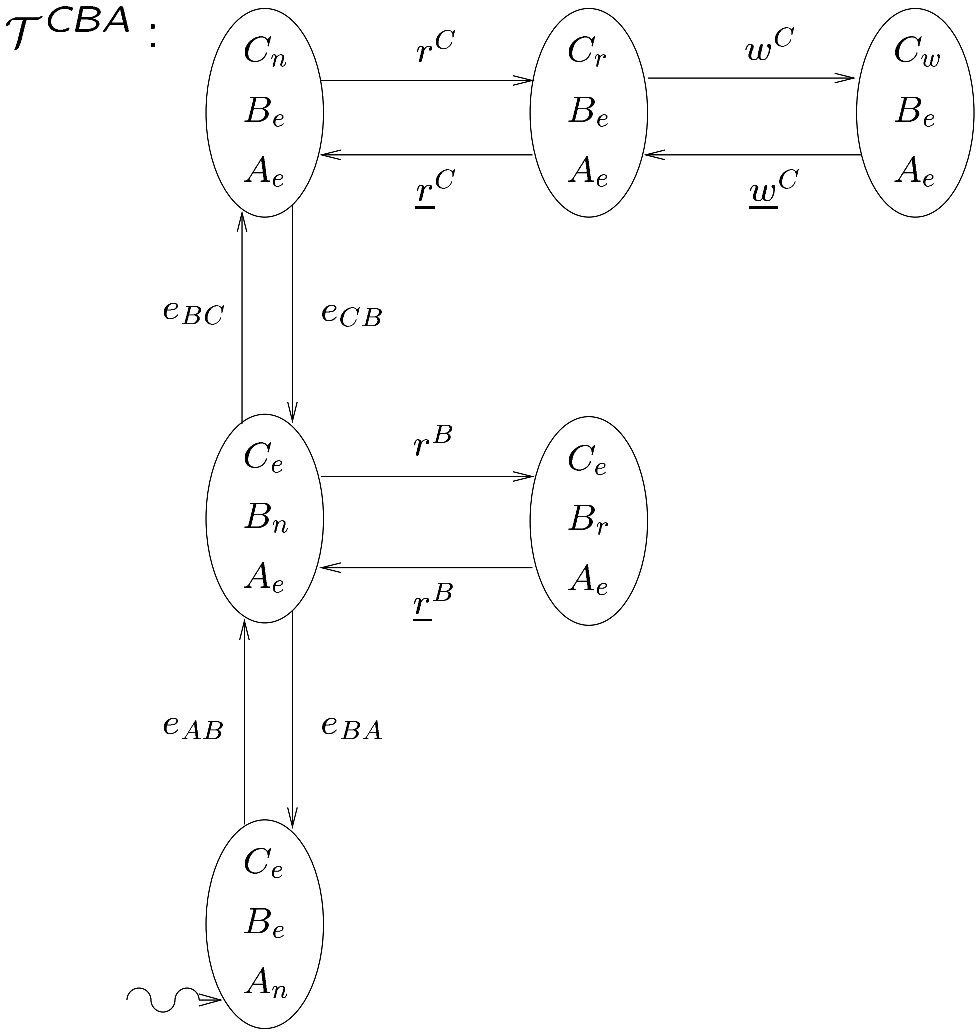
Component Automata $\mathcal{C}^C, \mathcal{C}^B, \mathcal{C}^A$: Rooms C, B and A



r^C, $\underline{r}^C$, w^C, $\underline{w}^C$, r^B & $\underline{r}^B$ <u>internal</u> actions
e_{BC}, e_{CB}, e_{AB} & e_{BA} <u>input</u>/<u>output</u> actions

$\Rightarrow$ choose a team automaton over $\{\mathcal{C}^C, \mathcal{C}^B, \mathcal{C}^A\}$

Team Automaton $\mathcal{T}^{CBA}$ over $\{\mathcal{C}^C, \mathcal{C}^B, \mathcal{C}^A\}$



$\Rightarrow$ team automaton $\mathcal{T}^{CBA}$ defined by <u>choice</u> of synchronizing transitions !

Meta Access Control

$\Rightarrow$ rights to grant & revoke access can themselves be granted & revoked

e.g. read access rights during refereeing:

(m^3) authors of paper grant the right to grant read access to editors-in-chief

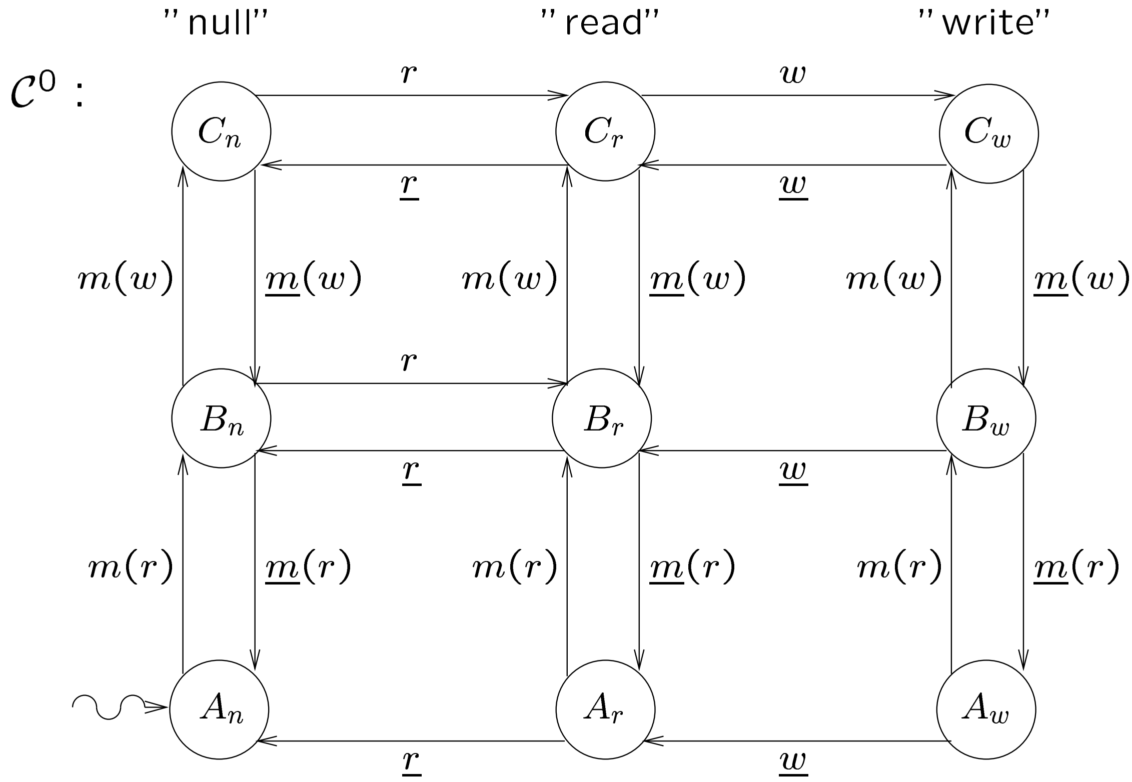
(m^2) editors-in-chief grant the right to grant read access to assistent editors

(m^1) assistent editors grant the right to read access to reviewers

(r) reviewers have read access

$$\underbrace{\text{grant right to}}_{m}\ \underbrace{\text{grant right to}}_{m} \ldots \underbrace{\text{read access}}_{r}$$

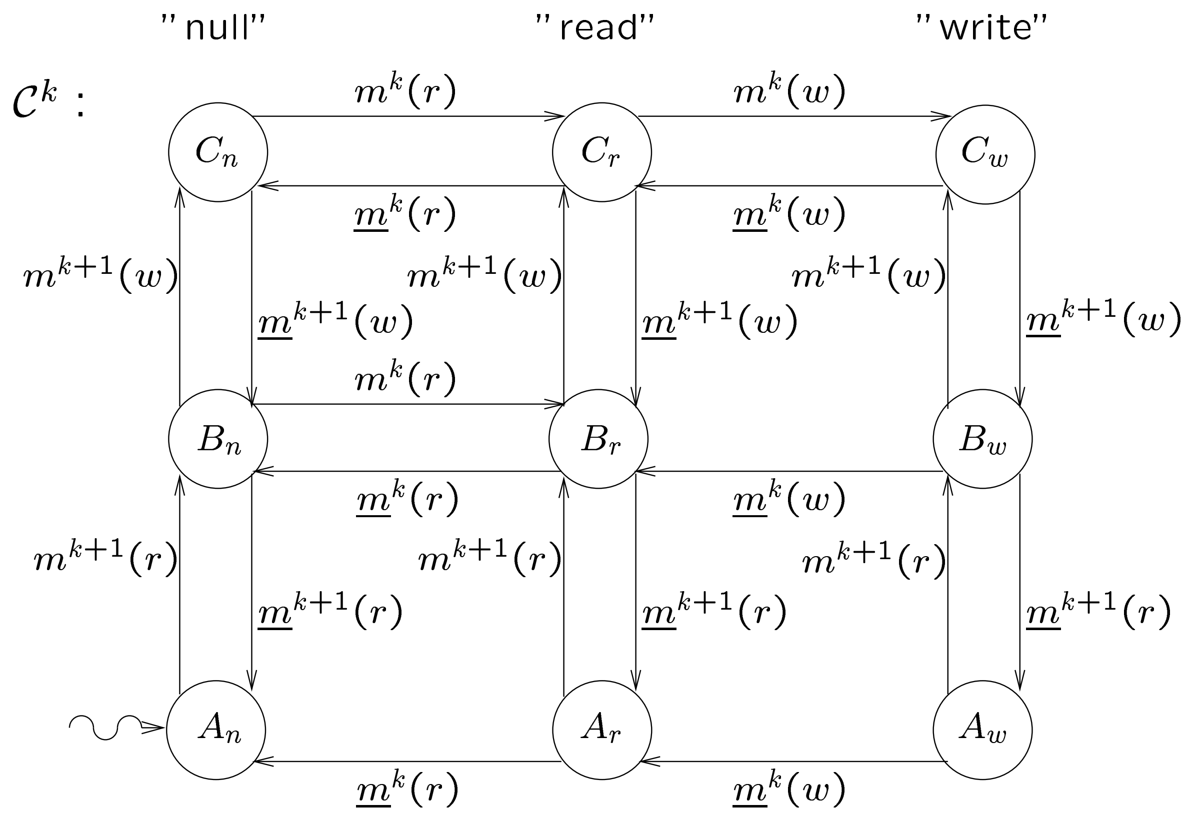
Component Automaton $\mathcal{C}^0$



$$
\left.
\begin{array}{ll}
m^0(r) = r & \text{read access} \\
\underline{m}^0(r) = \underline{r} & \text{no read access} \\
m^0(w) = w & \text{write access} \\
\underline{m}^0(w) = \underline{w} & \text{no write access}
\end{array}
\right\} \begin{array}{l} \underline{\text{internal}} \\ \text{actions} \end{array}
$$

$$
\underbrace{
\begin{array}{ll}
m^1(r) = m(r) & \text{grant right to read access} \\
\underline{m}^1(r) = \underline{m}(r) & \text{revoke right to read access} \\
m^1(w) = m(w) & \text{grant right to write access} \\
\underline{m}^1(w) = \underline{m}(w) & \text{revoke right to write access}
\end{array}
}_{\underline{\text{input}}\ \text{actions}}
$$

Component Automaton $\mathcal{C}^k$: Meta Access at Layer k

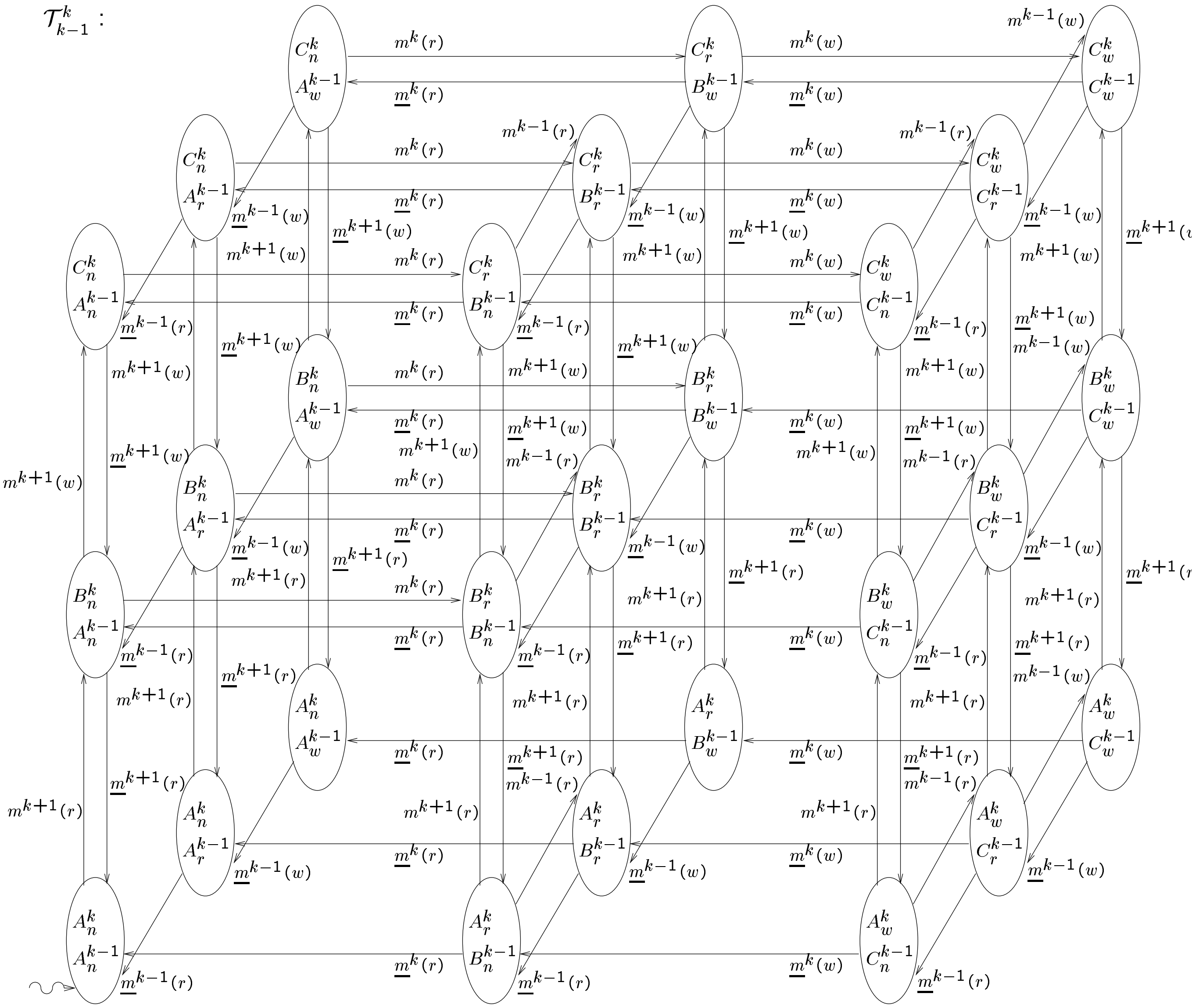


$m^k(r)$, $\underline{m}^k(r)$, $m^k(w)$ & $\underline{m}^k(w)$ <u>output</u> actions

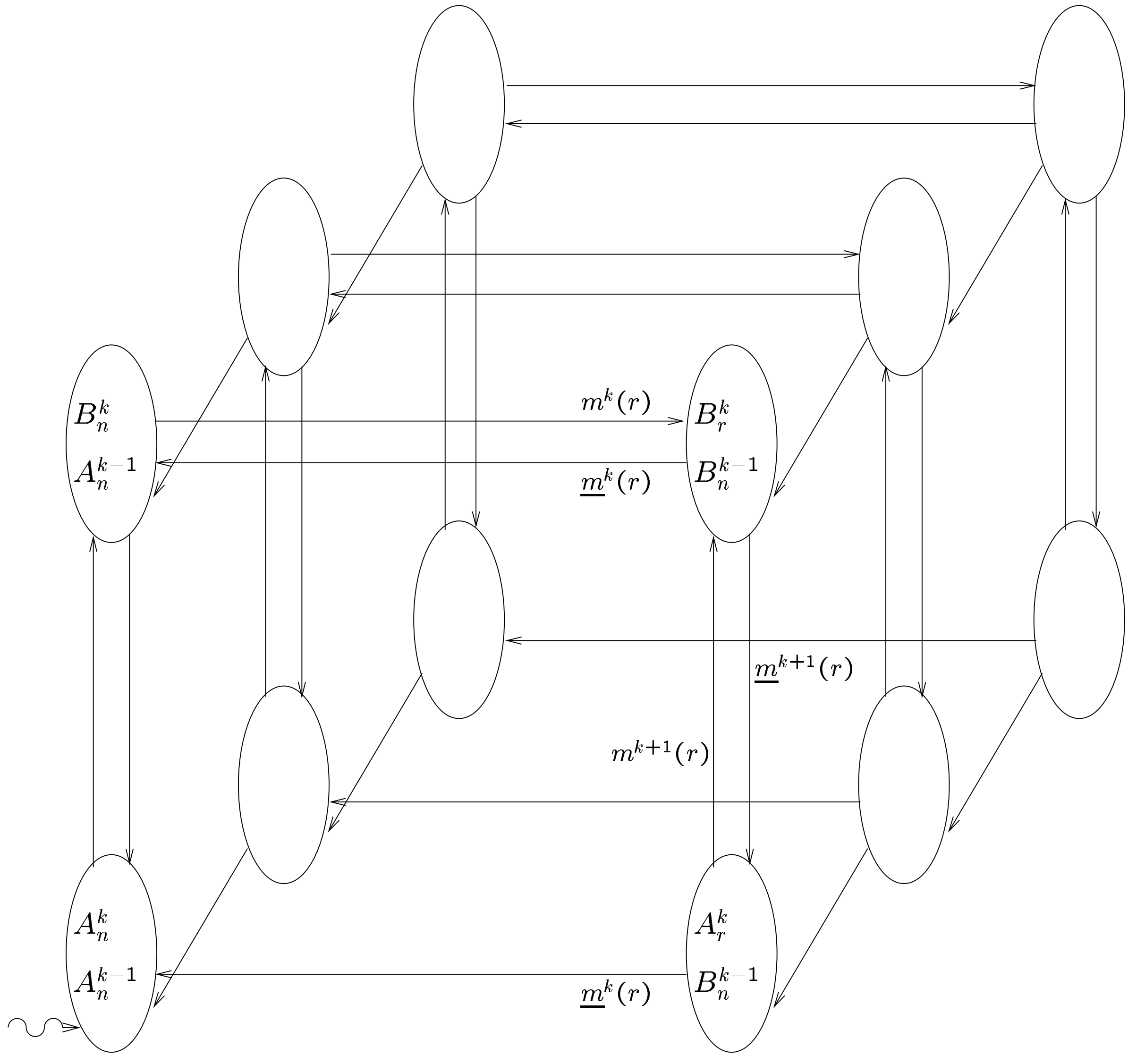
$m^{k+1}(r)$, $\underline{m}^{k+1}(r)$, $m^{k+1}(w)$ & $\underline{m}^{k+1}(w)$ <u>input</u>

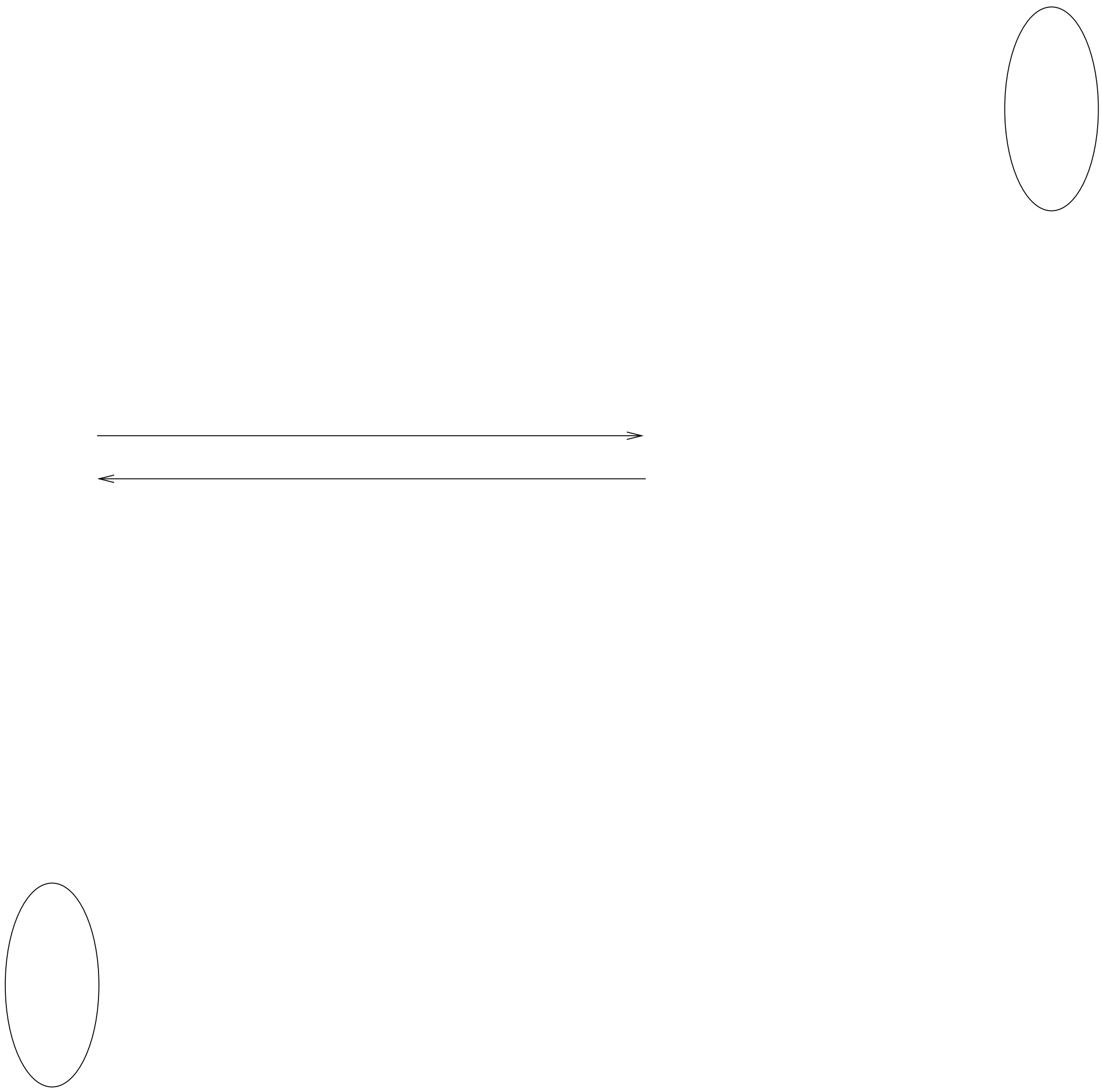
$\Rightarrow$ compose <u>iterated</u> team over $\{\mathcal{C}^0, \ldots, \mathcal{C}^n\}$ by choosing its synchronizing transitions !

Team Automaton $\mathcal{T}^{k}_{k-1}$ over $\{\mathcal{C}^{k-1}, \mathcal{C}^{k}\}$

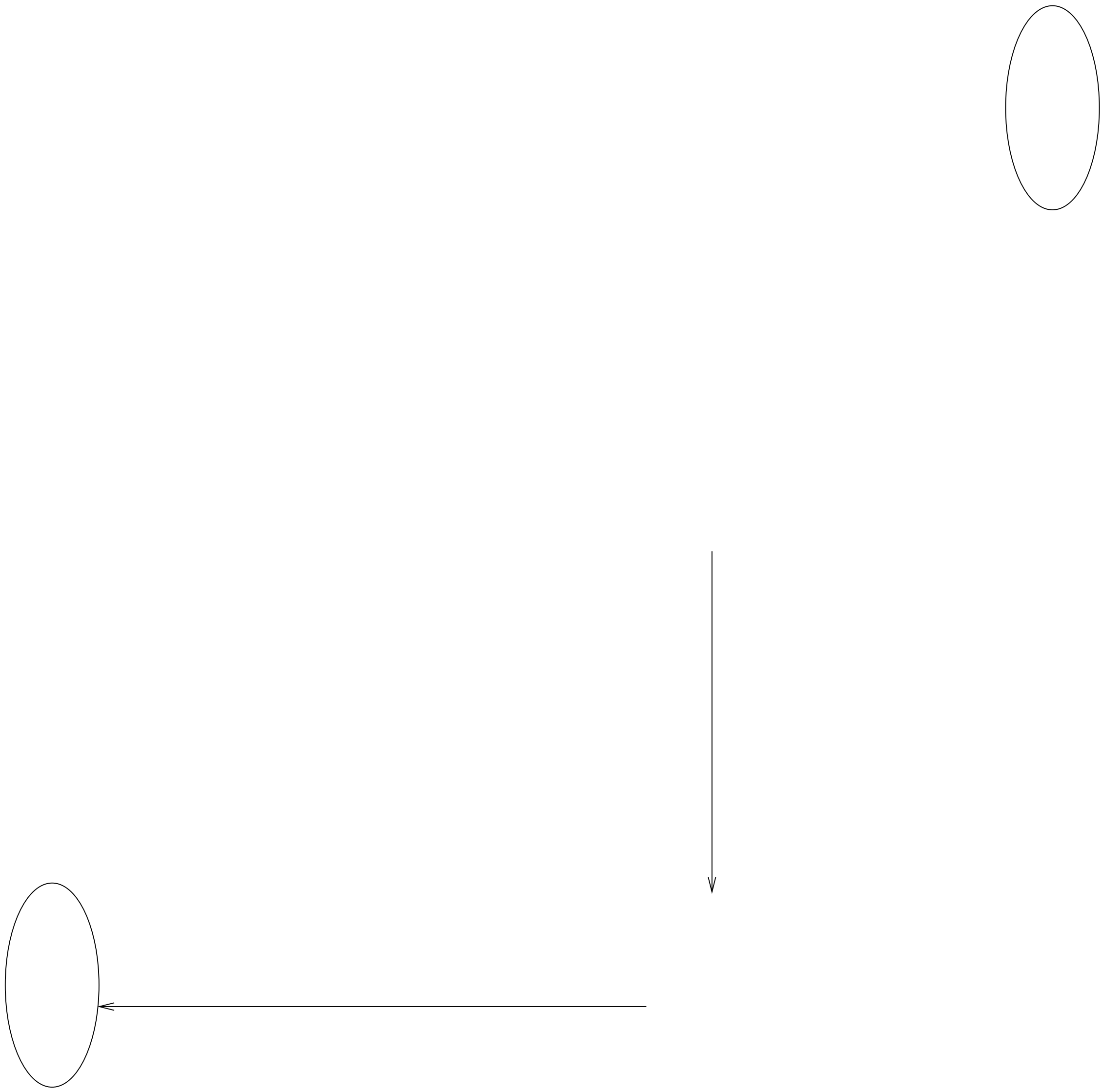


Team Automaton $\mathcal{T}^{k}_{k-1}$ over $\{\mathcal{C}^{k-1}, \mathcal{C}^{k}\}$ (zoomed in on detail)





$m^k(r)$, $\underline{m}^k(r)$, $m^k(w)$ & $\underline{m}^k(w)$ <u>master-slave</u> synchronizations with master $\mathcal{C}^k$ & slave $\mathcal{C}^{k-1}$



shallow & deep revocation: here deep, which is known as hard to model (Dewan et al., 1998)

Conclusions

Spatial access control metaphor end user friendly:

+ demystifies concepts of access control
+ understandable by non-computer people

Team automata framework precise and rigorous:

+ flexible (role of actions, choice of transitions)
+ scalable (modular construction, iterated teams)
+ extendible (time, priorities)
+ verifiable (automata-theoretic tools)

– no tool (yet)

⇒ metaphor & framework together form a powerful team !

Some Current & Future Work

[BB03] & [BLP03]

&

ter Beek, Massink, Latella & Gnesi

Model Checking Groupware Protocols

&

ter Beek & Mitrana

Synchronized Shuffles

&

ter Beek & Kleijn

Theory of Synchronized Shuffles

I/O Automata Versus Team Automata

Vector Team Automata and Petri Nets

but most importantly...

DEFEND Ph.D. THESIS !