Team Automata for Spatial Access Control

Maurice H. ter Beek[0]

Tuesday 23 April 2002

•

Seminar on Automata Theory
University of Szeged

Thursday 30 May 2002

•

MTA-SZTAKI, Budapest

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

[0]ERCIM fellow at MTA-SZTAKI, Budapest

[1]Dept. of Computer Science, Univ. of Colorado

[2]Leiden Institute of Advanced Computer Science, Univ. Leiden

Motivation

Computer Supported Cooperative Work is:

How do groups of people work together?

How can technology (groupware) assist?

CSCW needs:

Conceptual level:
define concepts and terminology consistently

Architectural level:
describe and compare groupware systems

Ellis introduced concept of *team automata* for modelling both levels informally in

"Team Automata for Groupware Systems"
(GROUP'97 - International ACM Conference on Supporting Group Work)

Motivation - continued

Ter Beek, Ellis, Kleijn, and Rozenberg formalized and elaborated this concept in

"Synchronizations in Team Automata for Groupware Systems"

(CSCW - Journal of Collaborative Computing, 2002)

"Team Automata for Spatial Access Control"

(ECSCW 2001 - 7th European CSCW Conference, 2001)

"Team Automata for CSCW"

(2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, 2001)

Team Automata

•

A formal approach towards the modelling
of collaboration between system components
(ter Beek, 2002)

- model logical architecture of a design
- abstract from concrete data and actions
- describe behaviour in terms of

  - state-action diagram (structure)
  - role of actions (input, output, internal)
  - shared actions (synchronizations)

- force user to make explicit design choices
- provide mathematically precise analytic tools
  for proving crucial design properties,
  before implementation!

Access Control

- security and information access control vital components of any system

- CSCW / groupware systems require:

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

  - more fine-grained access granularity
    e.g. in software development team

  - 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

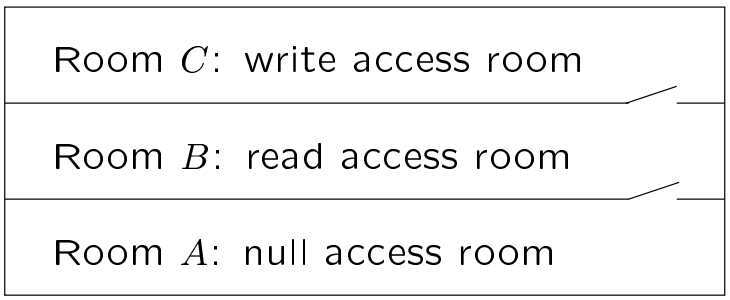
SPACE metaphor (Bullock et al., 1998)

$\Downarrow$

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

and

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



Virtual Reality

- user traverses from room to room
  by using keyboard keys or mouse

- *authorization* check at boundaries (doors)
  between spaces (rooms)

- if access is OK, then user can enter and
  use resources of newly entered room

- here abstract from *authentication*

SPACE Access Model

- basic tenet: a fundamental component of any collaborative environment is the environment itself (i.e. the space)

- hide explicit technical security mechanisms from end users through natural spatial makeup of the environment

- users can use their knowledge of environment to understand the implicit security policies

- avoids users tackling technical concepts such as so-called access matrices, which helps avoiding misunderstandings

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* $\quad q \xrightarrow{a} q'$

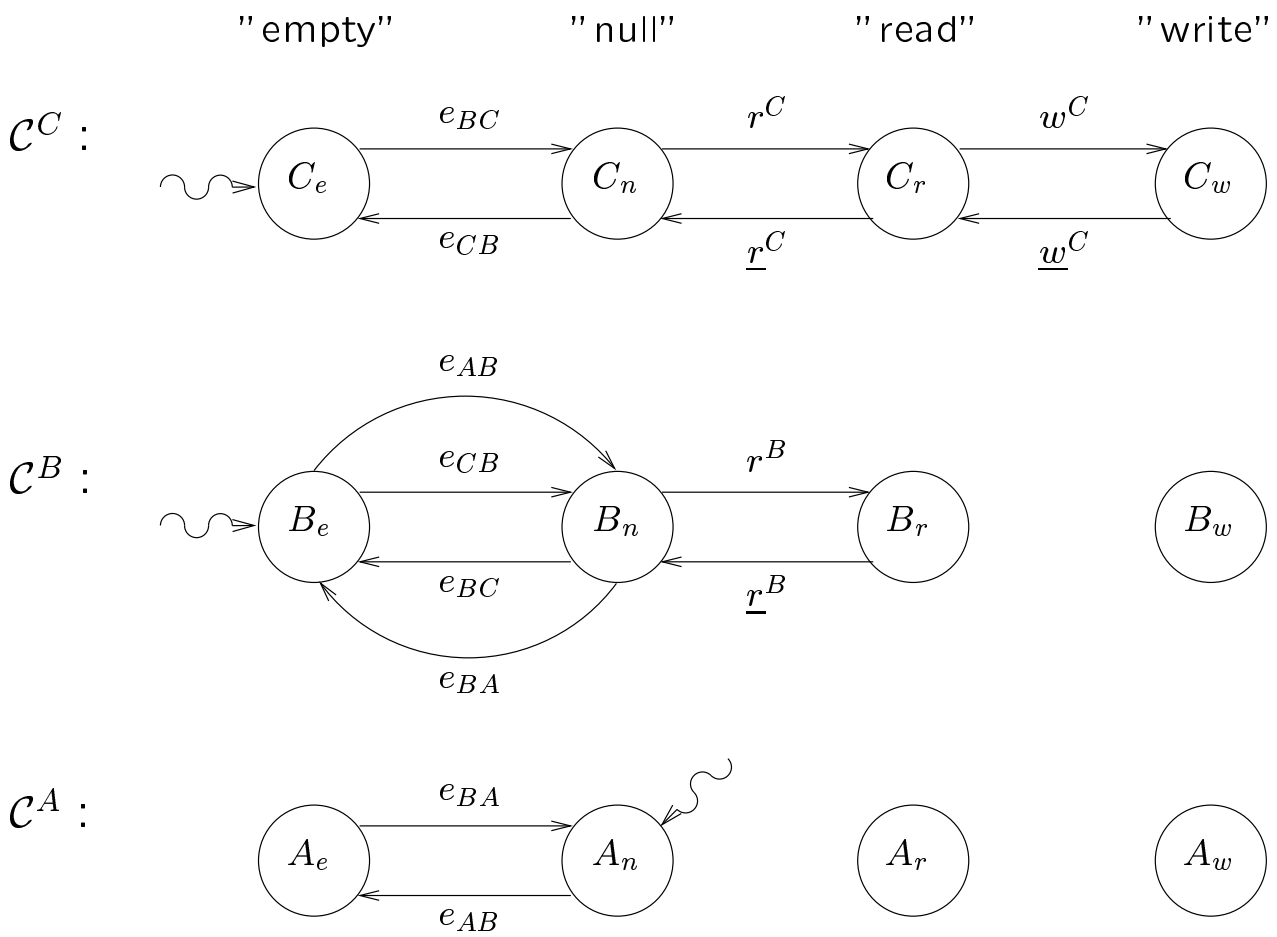
$I \subseteq Q$ set of *initial states*

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

Σ_{int} *internal actions* cannot be observed

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

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, and $\underline{r}^B$ **internal** actions

e_{BC}, e_{CB}, e_{AB}, and e_{BA} **external** actions

Composable System

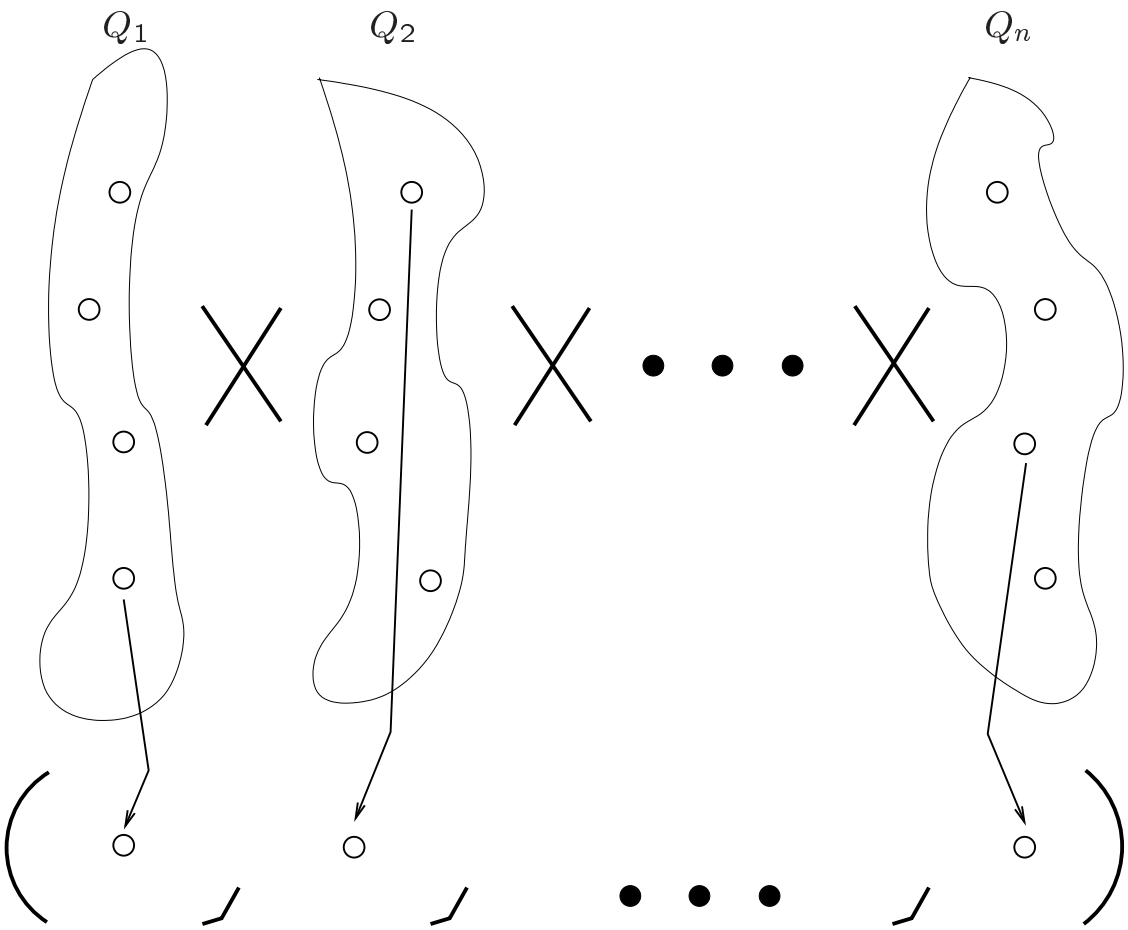
a set $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ of component automata is a *composable system* if for all $i \in \{1, \ldots, n\}$,

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

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

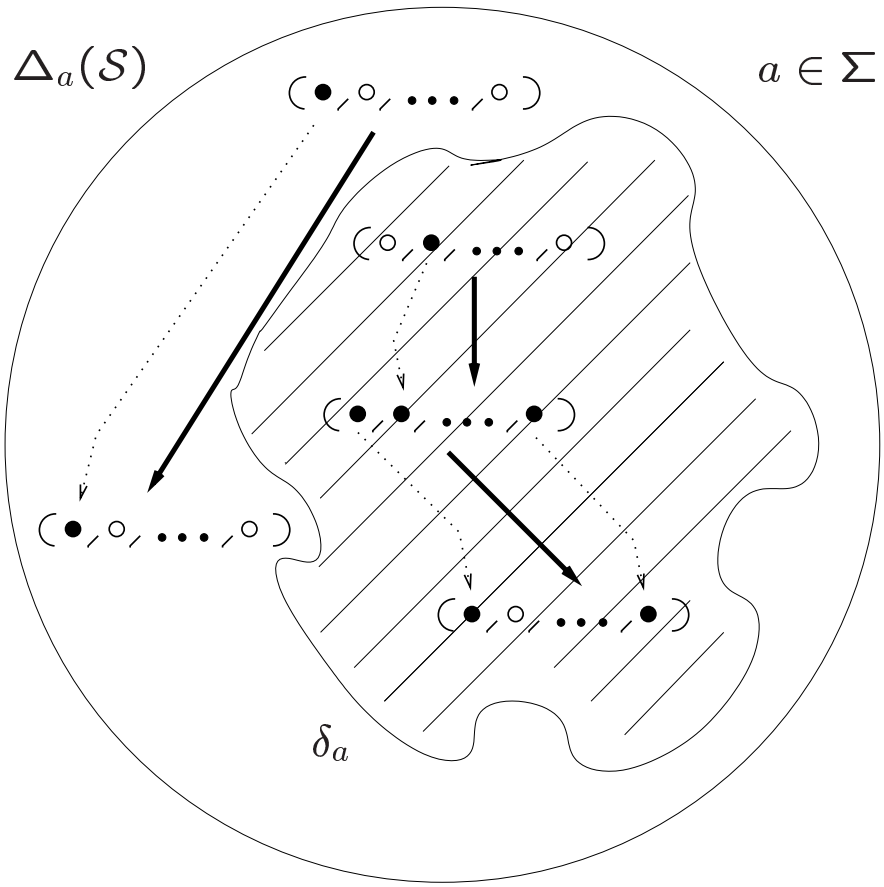
? how to form a team automaton over such a composable system $\mathcal{S}$

State Space of Team Automata



$\in \prod_{i \in \{1,\ldots,n\}} Q_i$

Transition Space of Team Automata



in every team transition at least one component acts according to its transition relation

all other components either join or are idle

$\Rightarrow$ the choices of team transition relations δ_a, $\forall a \in \Sigma$, define a specific team automaton!

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 over* $\mathcal{S} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ if

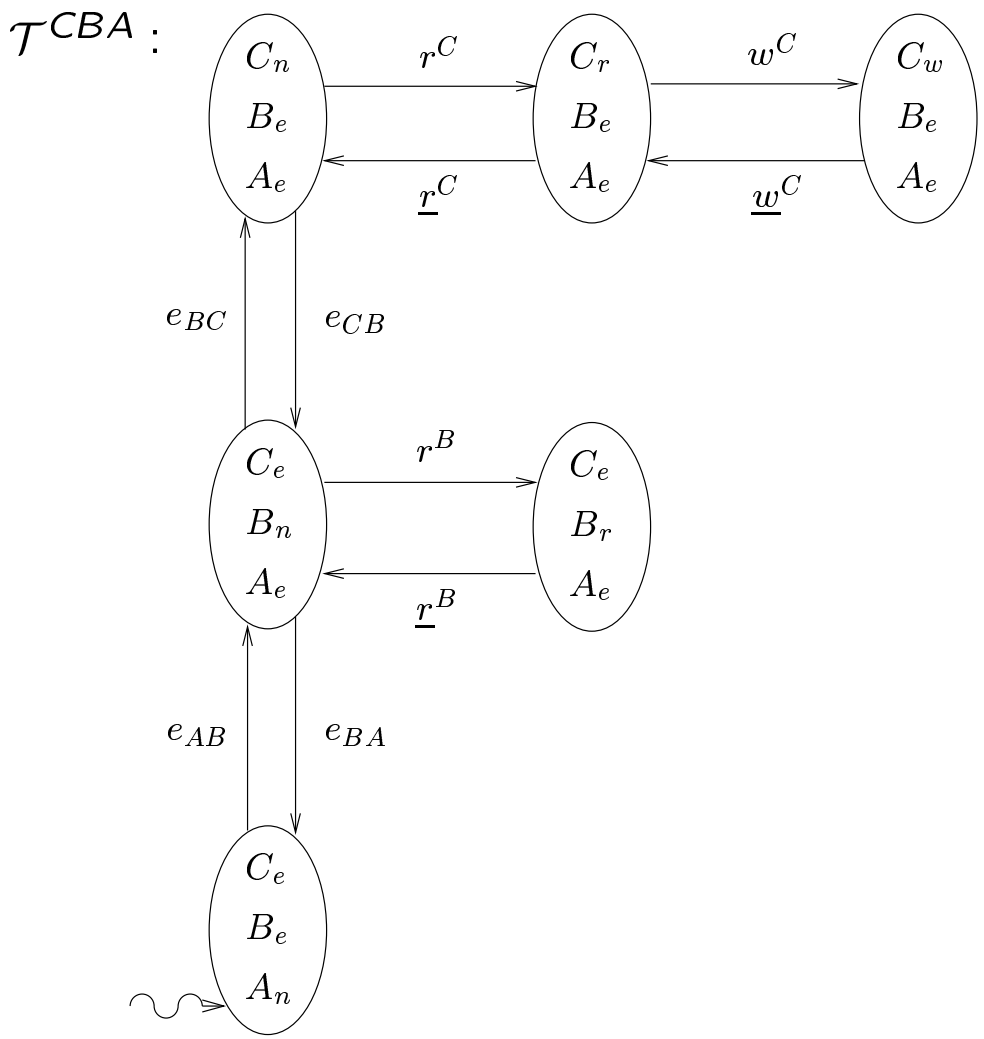
$$\left.\begin{array}{l} \Sigma_{int} = \bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,int} \\ \Sigma_{out} = \bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,out} \\ \Sigma_{inp} = (\bigcup_{i\in\{1,\ldots,n\}} \Sigma_{i,inp}) \setminus \Sigma_{out} \end{array}\right\} = \Sigma$$

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

$\forall a \in \Sigma \qquad \delta_a \subseteq \Delta_a(\mathcal{S})$
and $\delta_a = \Delta_a(\mathcal{S})$ if $a \in \Sigma_{int}$

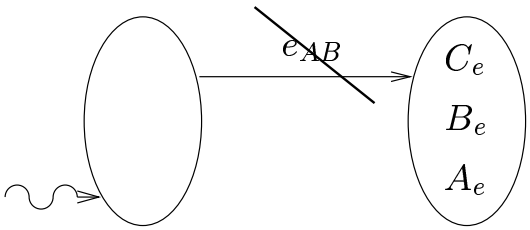
Lemma: every team automaton is
a component automaton

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



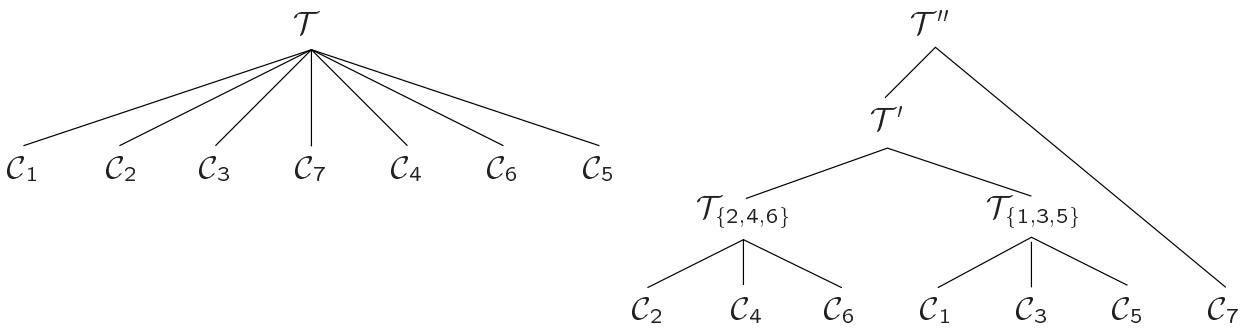
$\Rightarrow$ team automaton defined by
choice of synchronizing transitions!

$\mathcal{T}^{CBA}$:



Teams over Teams

Note: every team automaton can be used as a component



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

Fact: after *reordering* elements of $\mathcal{T}''$:
$\mathcal{T}''$ team automaton over $\{\mathcal{C}_1, \ldots, \mathcal{C}_7\}$

Meta Access Control

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

e.g. read access rights during refereeing:

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

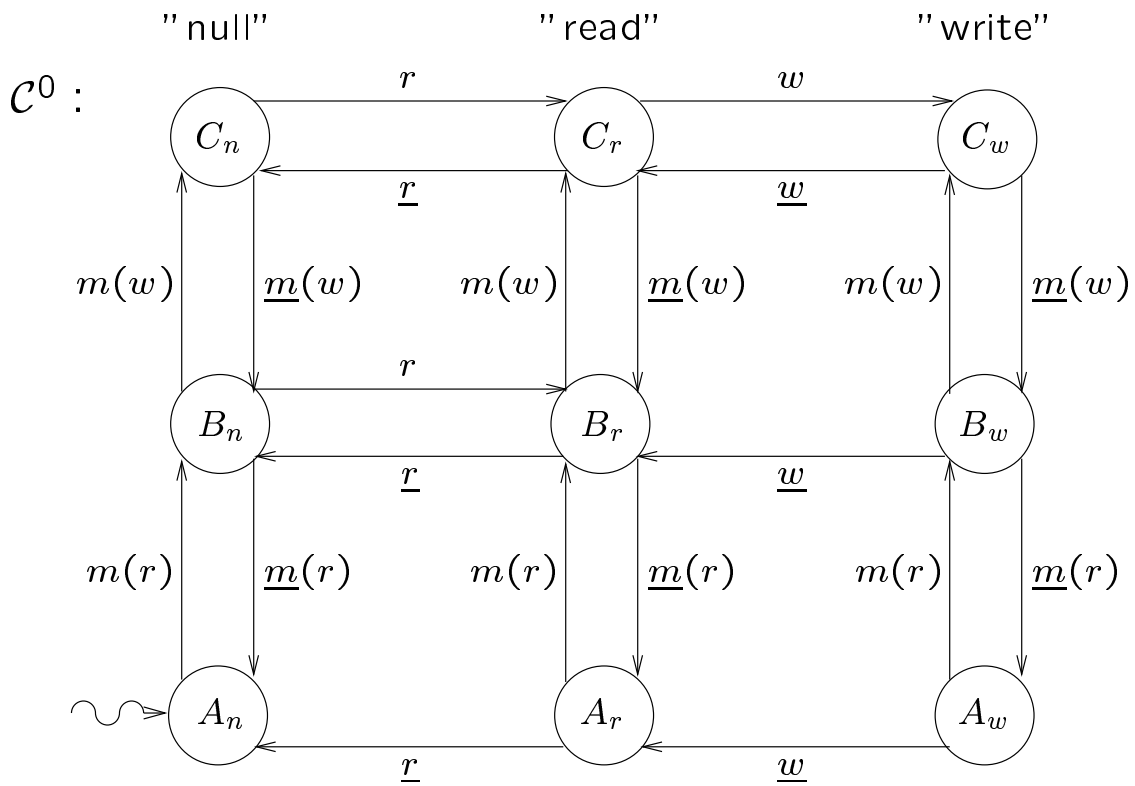
(m^2) editors-in-chief grant the right to grant read access to assistent editors

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

(r) reviewers have read access

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

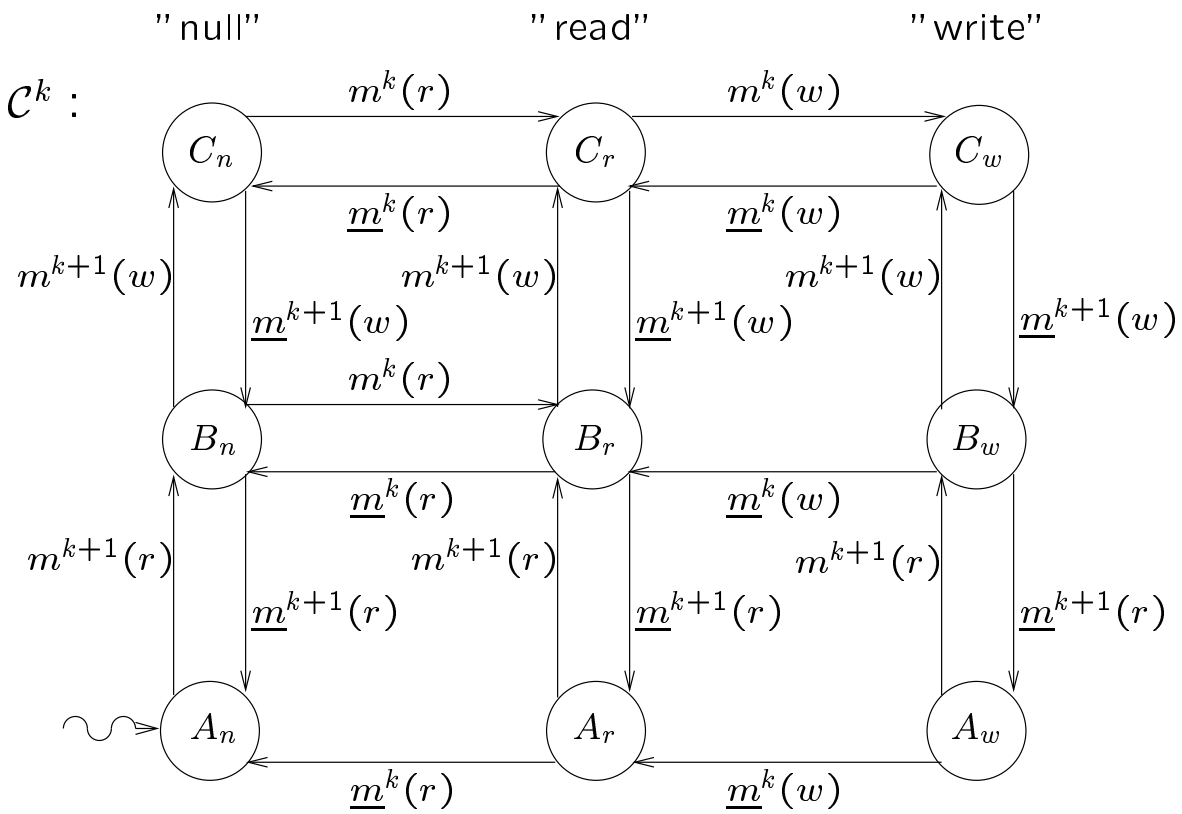
Component Automaton $\mathcal{C}^0$



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

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

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

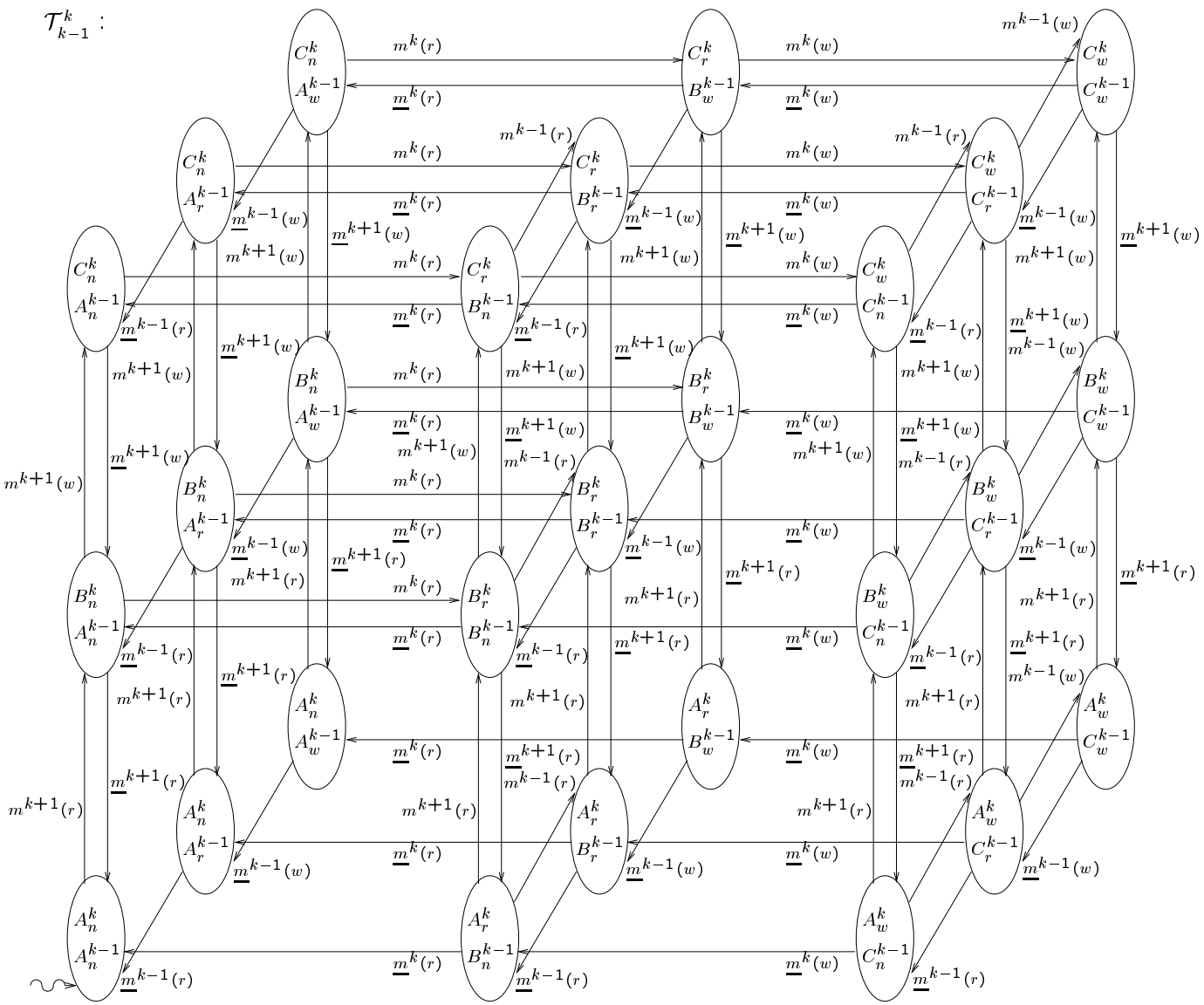


$m^k(r)$, $\underline{m}^k(r)$, $m^k(w)$, and $\underline{m}^k(w)$ **output** actions

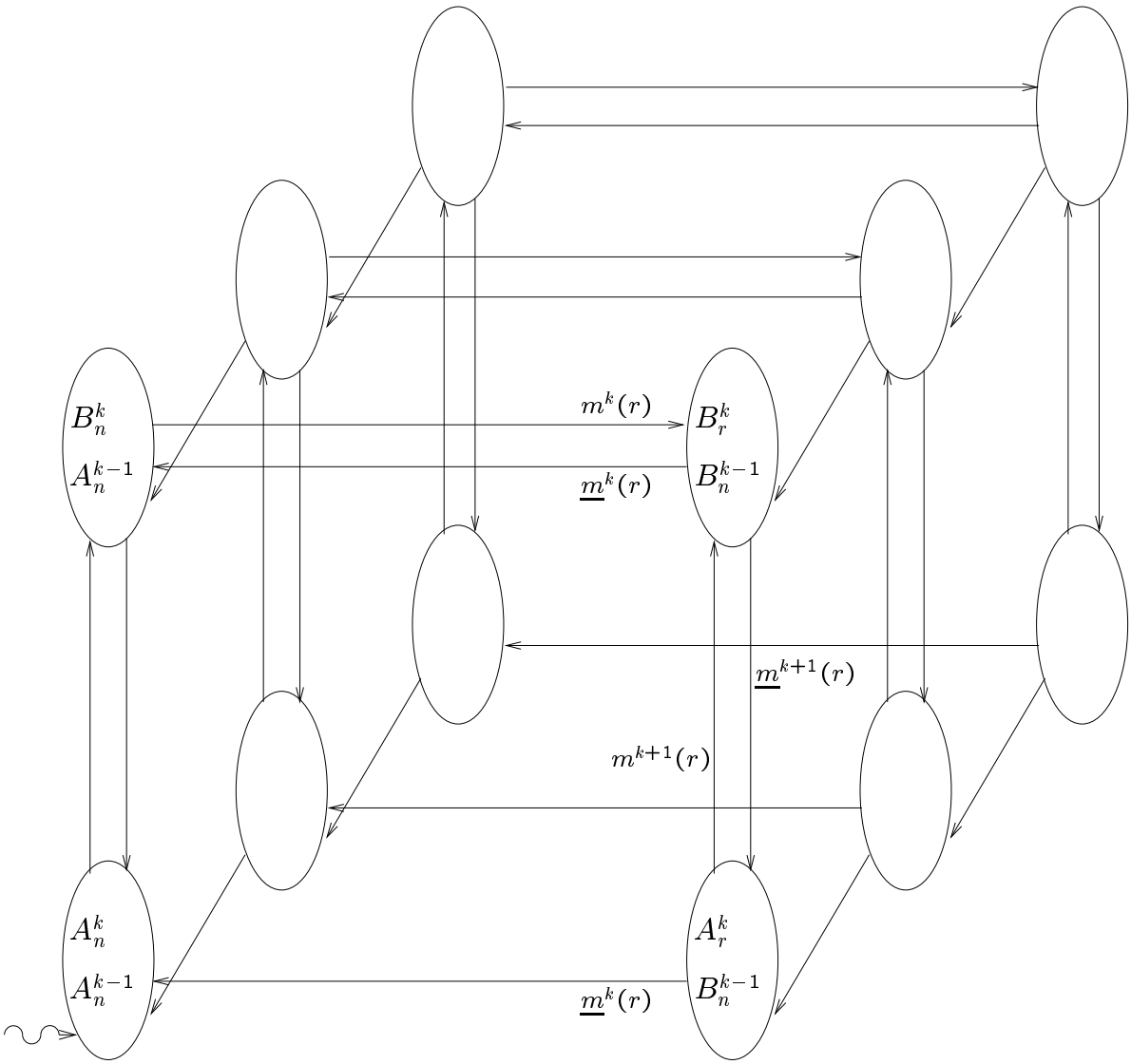
$m^{k+1}(r)$, $\underline{m}^{k+1}(r)$, $m^{k+1}(w)$, and $\underline{m}^{k+1}(w)$ **input**

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

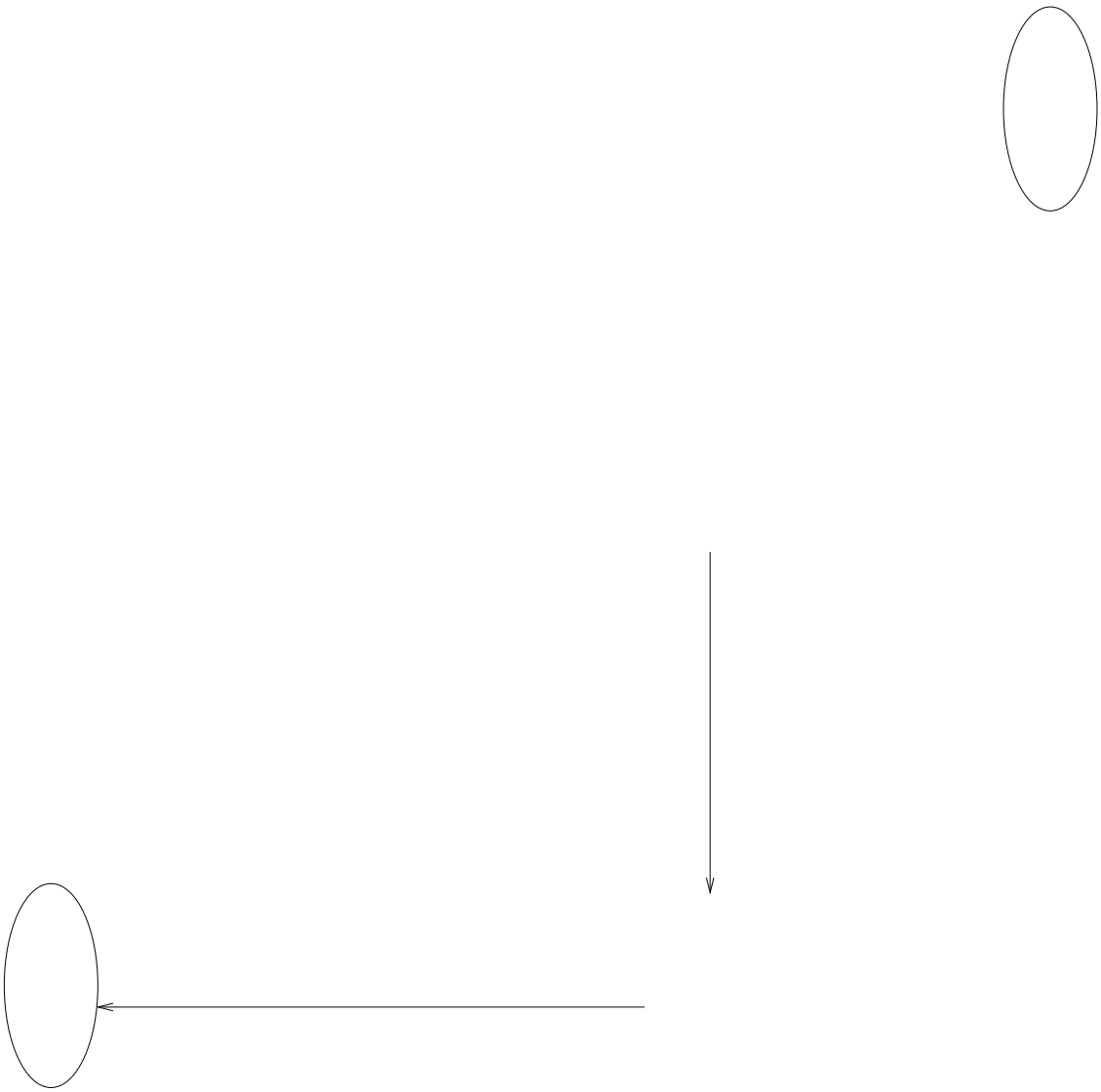
Example Team Automaton $\mathcal{T}^k_{k-1}$ over $\{\mathcal{C}^{k-1}, \mathcal{C}^k\}$



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



$m^k(r)$, $\underline{m}^k(r)$, $m^k(w)$, and $\underline{m}^k(w)$ **master-slave** synchronizations with master $\mathcal{C}^k$ and slave $\mathcal{C}^{k-1}$



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

Conclusion

- spatial access control metaphor end user friendly:

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

- team automata model precise and rigorous:

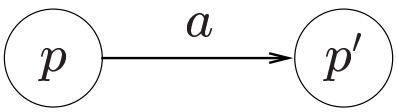
  + flexible (role of actions, choice of transitions)
  + scalable (modular construction, iterated teams)
  + extendible (time, priorities)
  + verifiable (automata theoretic tools)

  – no tool (yet)

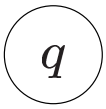
⇒ metaphor and model together form
a powerful team!

The Problem with Loops

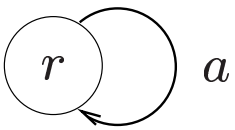
$\mathcal{T}$: $(p, q, r) \xrightarrow{a} (p', q, r)$

$\mathcal{C}_1$: 

$\mathcal{C}_1$ executed a

$\mathcal{C}_2$: 

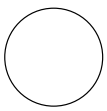
$\mathcal{C}_2$ cannot execute a

$\mathcal{C}_3$: 

did $\mathcal{C}_3$ execute a
or was $\mathcal{C}_3$ idle?

Vector Team Automata

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



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