# Analysis and Implementation of MSC-Specifications

Carsten Kern

Lehrstuhl Informatik 2
**Software Modeling and Verification**
RWTH Aachen

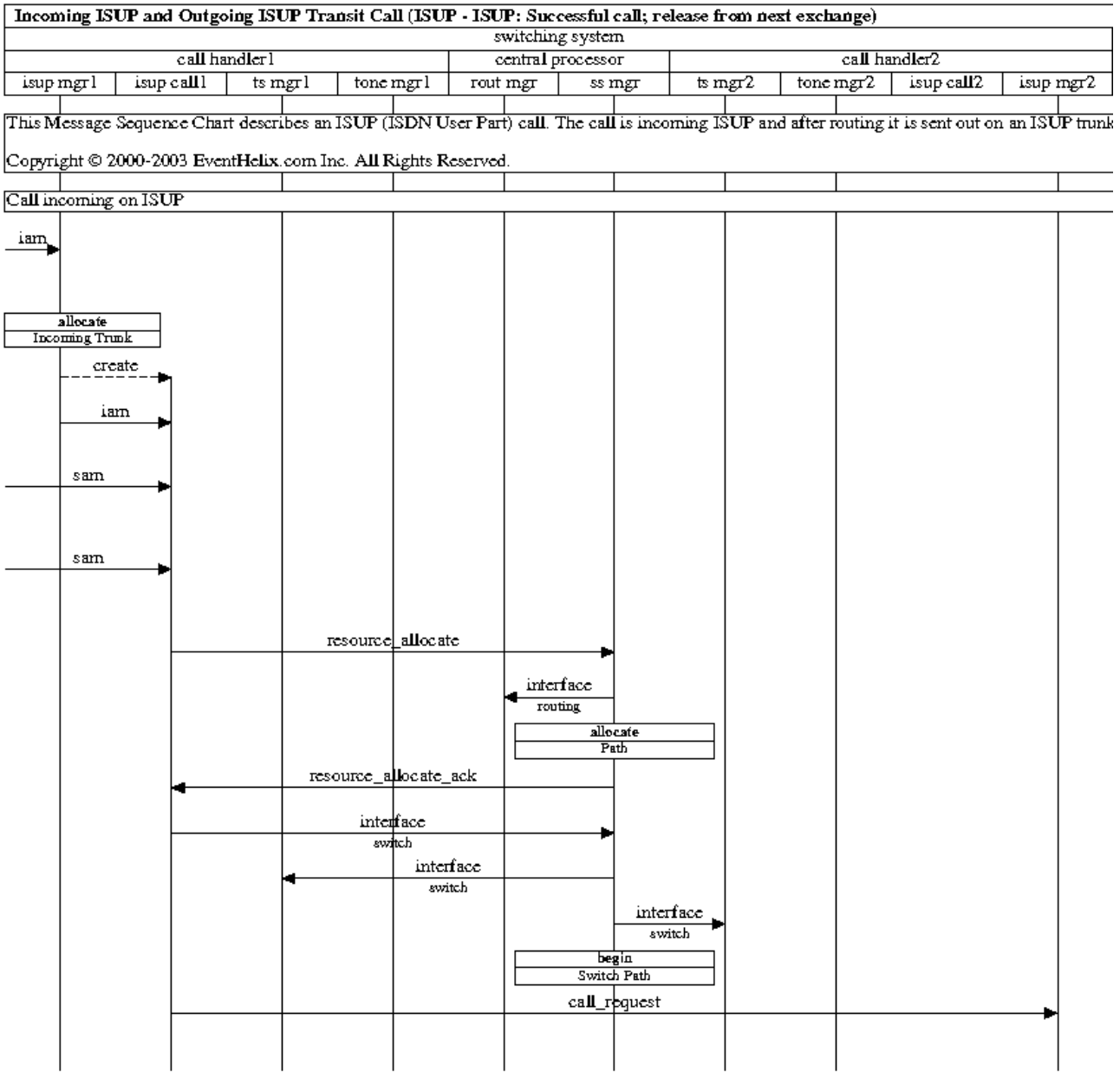
November 29, 2005

# Outline

# Presentation outline

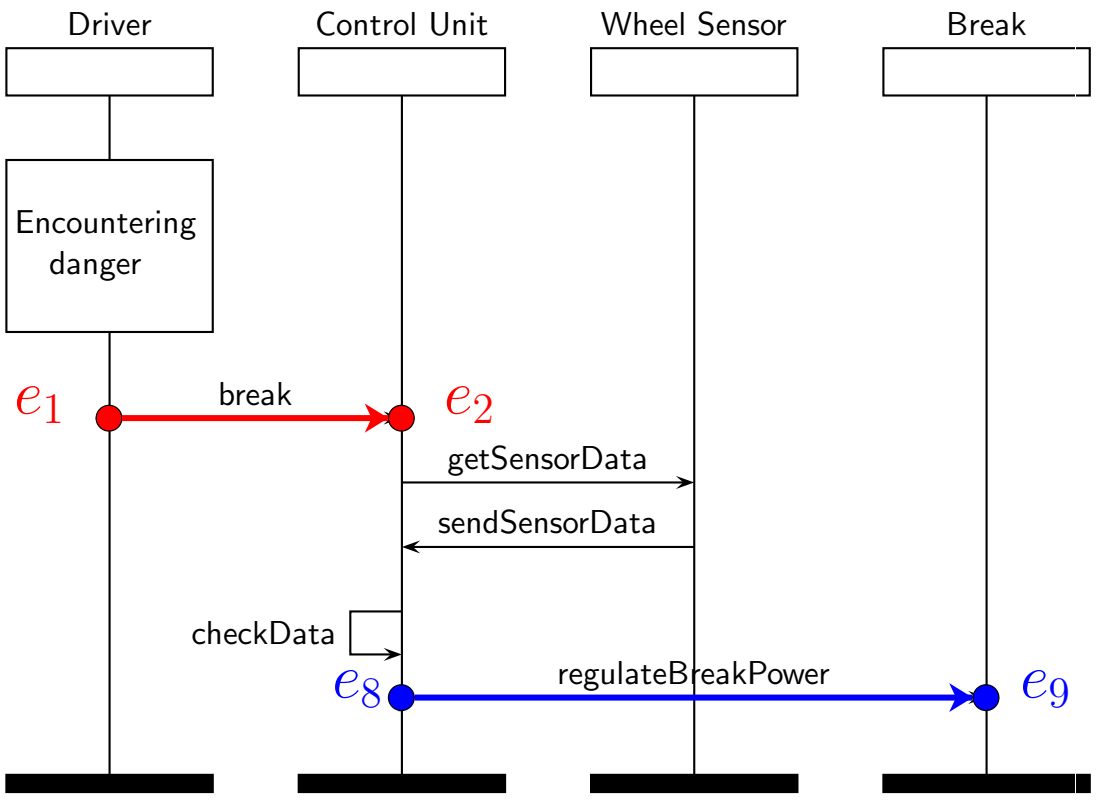
**Objects of interest: Message Sequence Charts**

- standardized modeling language at high level of abstraction
- used for specification of communication protocols
- similar to UML sequence diagrams


Incoming ISUP and Outgoing ISUP Transit Call (ISUP - ISUP: Successful call; release from next exchange)
switching system
call handler1
central processor
call handler2
isup mgr1
isup call1
ts mgr1
tone mgr1
rout mgr
ss mgr
ts mgr2
tone mgr2
isup call2
isup mgr2
This Message Sequence Chart describes an ISUP (ISDN User Part) call. The call is incoming ISUP and after routing it is sent out on an ISUP trunk
Copyright © 2000-2003 EventHelix.com Inc. All Rights Reserved.
Call incoming on ISUP
iam
allocate
Incoming Trunk
create
iam
sam
sam
resource_allocate
interface
routing
allocate
Path
resource_allocate_ack
interface
switch
interface
switch
interface
switch
begin
Switch Path
call_request

# Presentation outline

# A CMSC Example (Antiblock System)



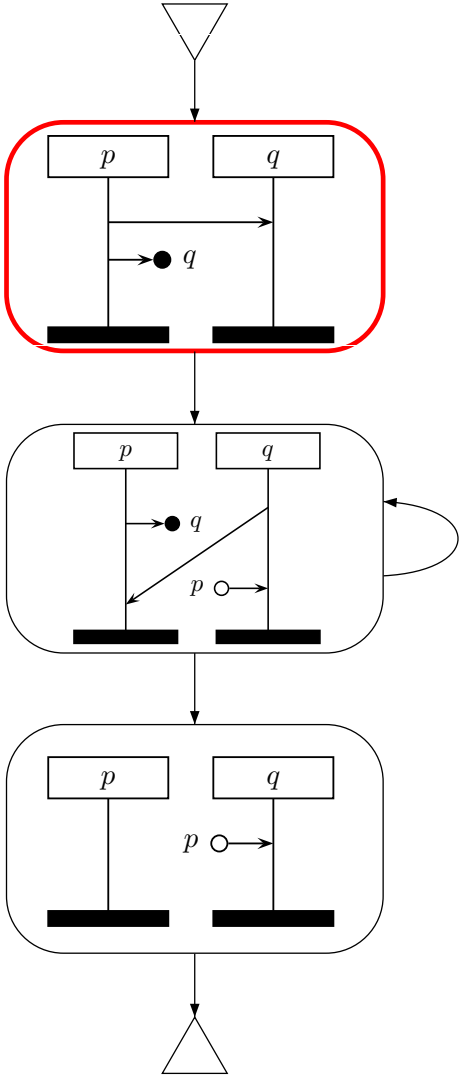
# Definition: compositional MSCs (CMSCs)

## A CMSC is a tuple $M = \langle \mathcal{P}, E, t, m, < \rangle$

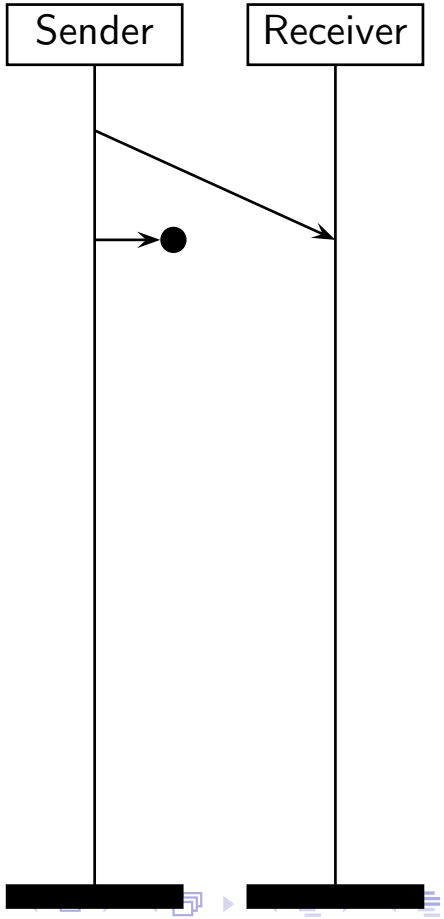
- $\mathcal{P}$: finite, non-empty set of processes
- $E = \biguplus_{p \in \mathcal{P}} E_p = S \uplus R$: set of events
  ($S$: sending and $R$: receiving events)
- $t$: event labeling function $(t : E \to Act)$
  (e.g.: $t(e) = p!q$ or $t(e') = p?q$)
- $m$: injective and partial matching function $(m : S \not\to R)$
  (not every sending event needs to have a corresponding receive event)
- $< \subseteq E \times E$: partial order on events

If $m$ is total and bijective we call $M$ an MSC

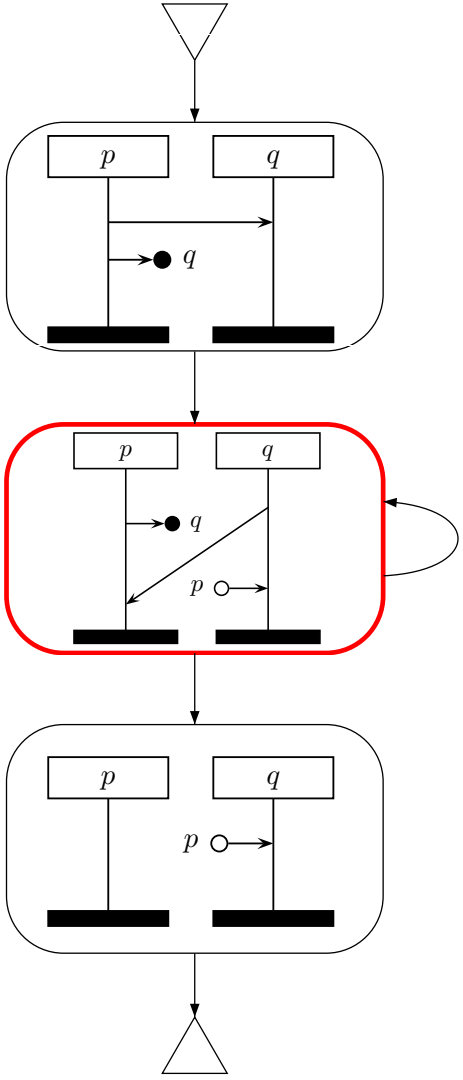
# A CMSG Example (Alternating-Bit Protocol)



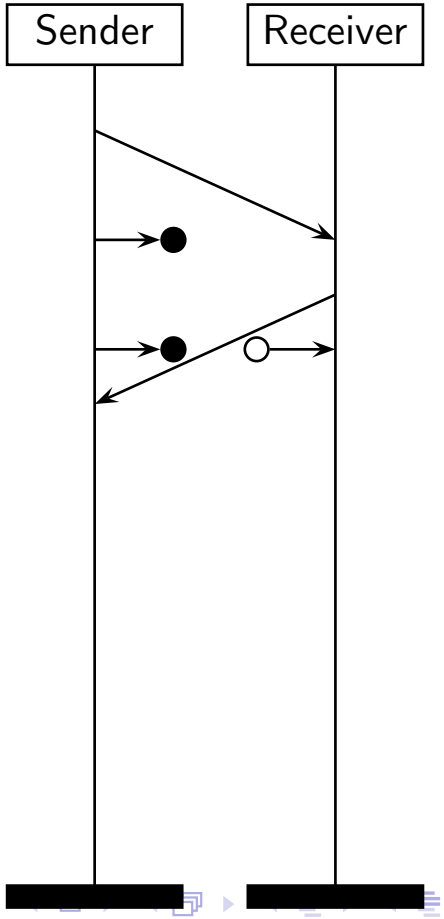

An execution



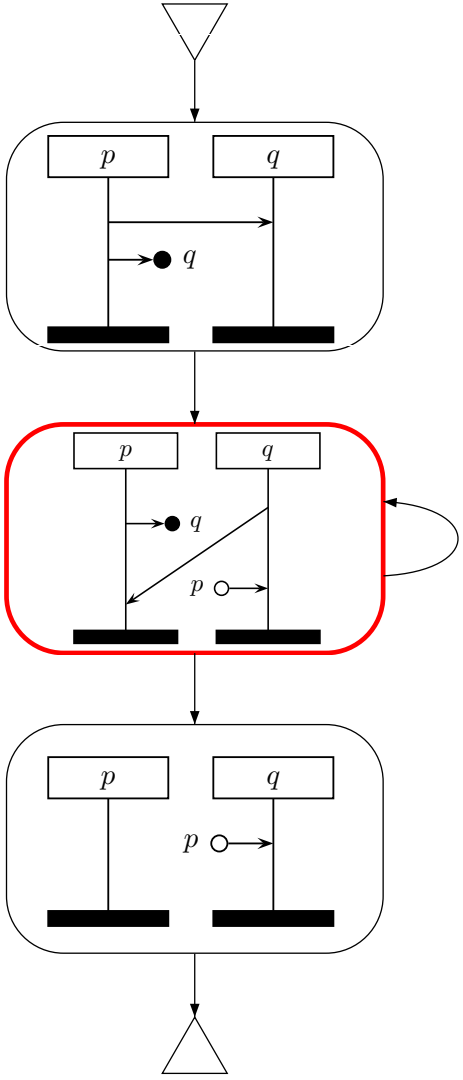
# A CMSG Example (Alternating-Bit Protocol)



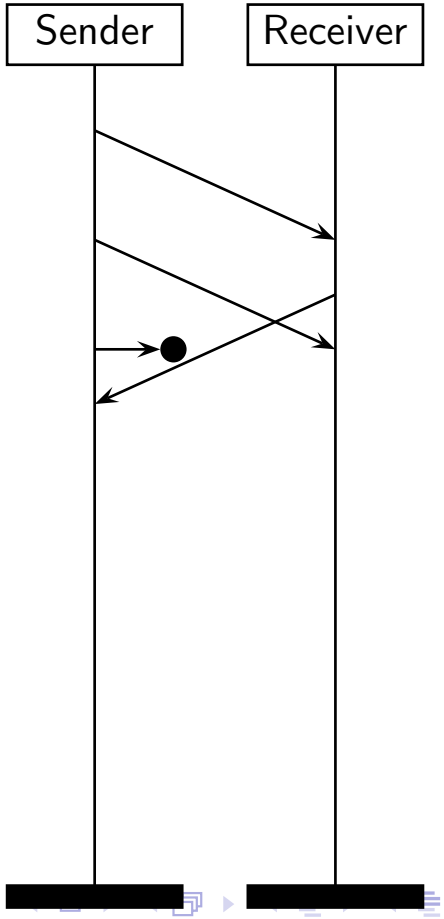

An execution



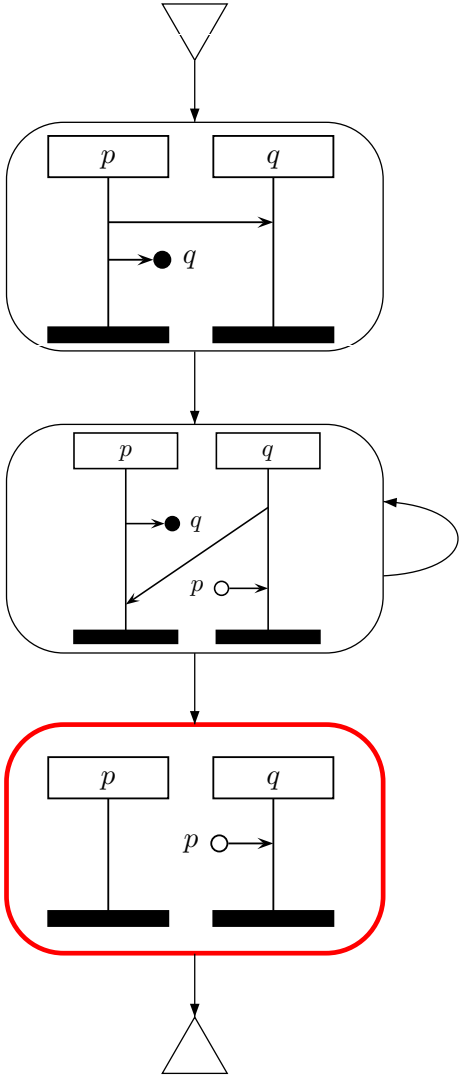
# A CMSG Example (Alternating-Bit Protocol)



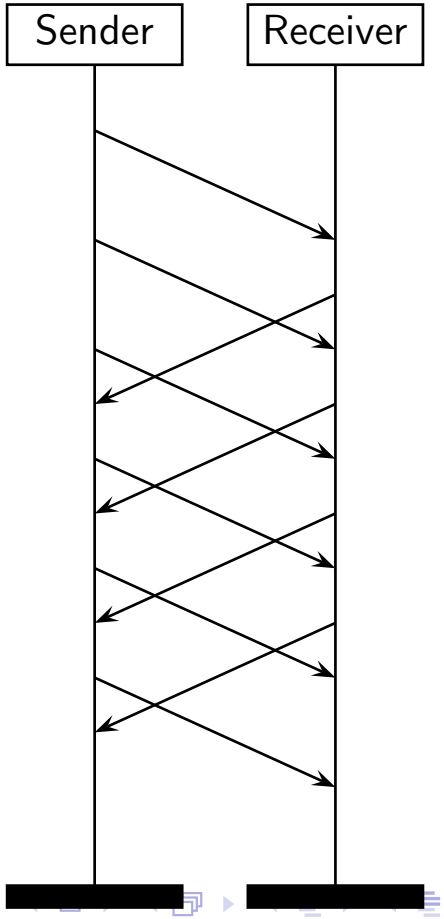

An execution



# A CMSG Example (Alternating-Bit Protocol)




An execution



# Definition: compositional MSGs (CMSGs)

**A CMSG is a tuple $G = \langle V, R, V^0, V^f, \lambda \rangle$**

- $\langle V, R \rangle$: graph ($V \neq \varnothing$, $R \subseteq V \times V$)
- $V^0$: non-empty set of start nodes ($V^0 \subseteq V$)
- $V^f$: set of end nodes ($V^f \subseteq V$)
- $\lambda$: node labeling function ($\lambda : V \to \mathbb{CMSC}$)

If $\lambda$ maps to $\mathbb{MSC}$ $G$ is called a Message Sequence Graph (MSG)

# Presentation outline

**What do we need CMSG properties for?**

$\Longrightarrow$ detecting classes of implementable (C)MSGs

# CMSG Property: "Regularity"

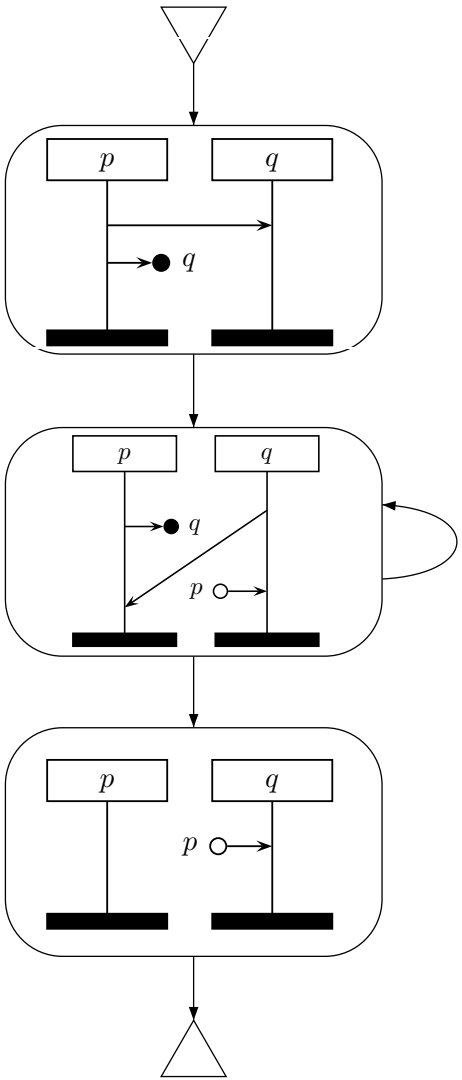
**Formal Definition:**

A CMSG $G$ is called *regular* iff the communication graph of every cycle is strongly connected.

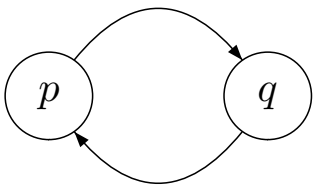
**Intuition of property: Regularity**

Regularity means that for each sent message an acknowledgement of the destination process can be received.

# Example: CMSG-Property "Regularity"




Communication graph of the loop:



# CMSG Properties: "Local-Choice" and "Locality"

**Intuition of property: Local Choice**

In every choice node only one process may decide on the progress of the system (*strong* local-choice).

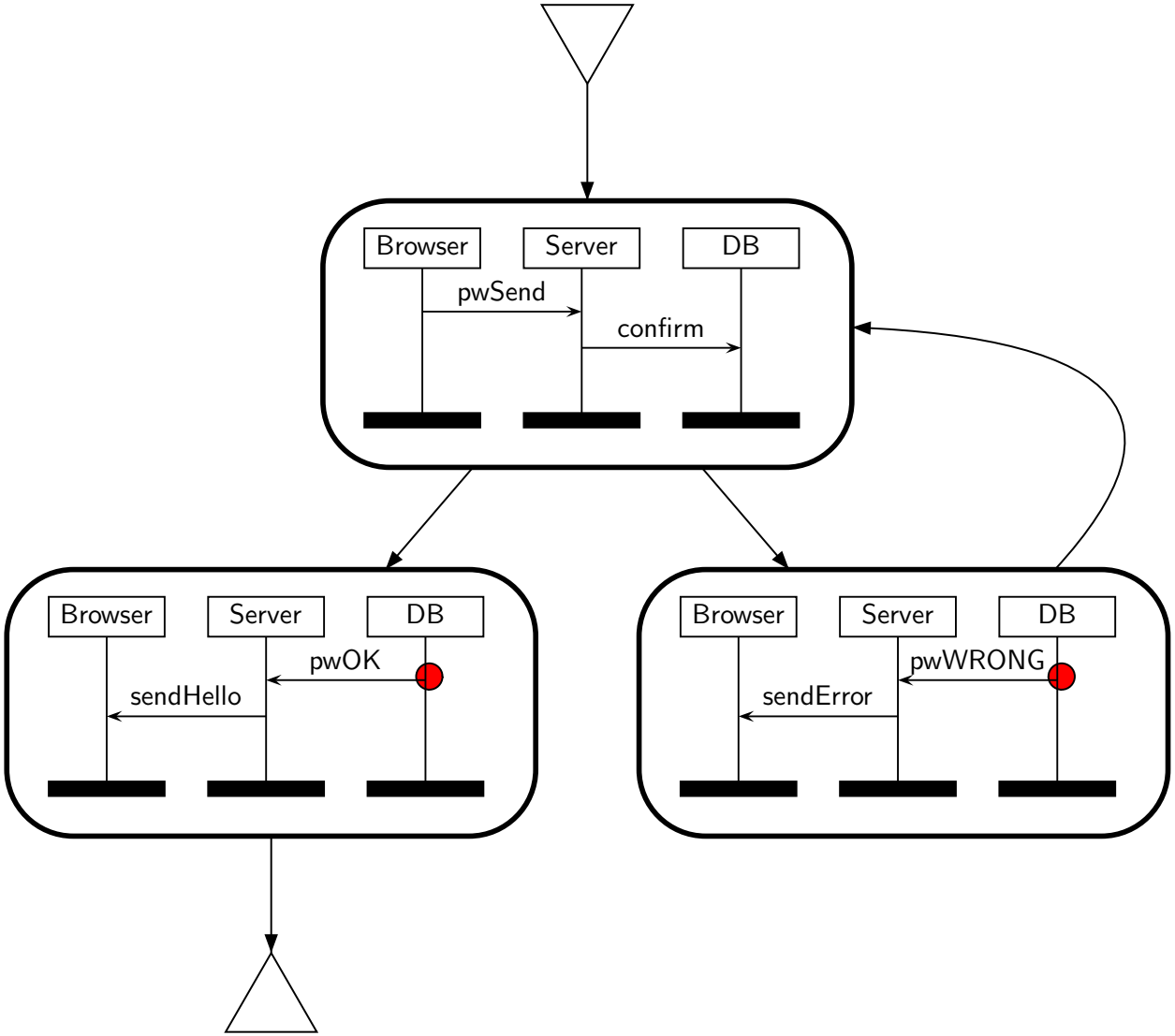
**Intuition of property: Locality**

The local property assures the local-choice property for every node in the graph.

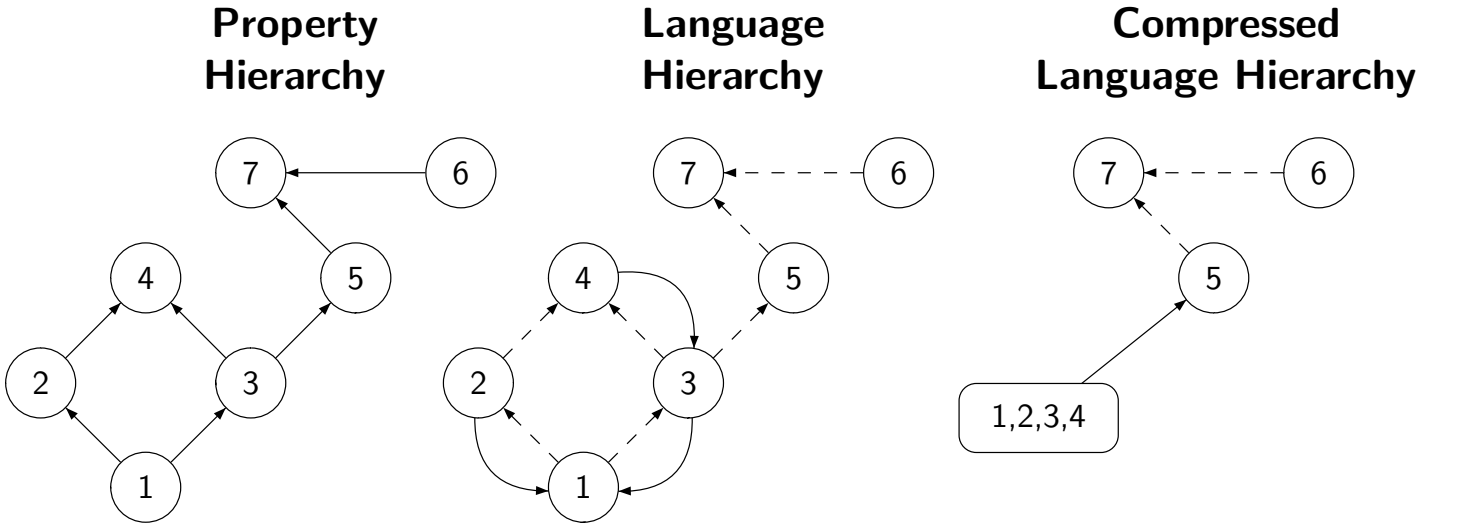
**Observation:**

Every local CMSG is also local-choice.

# Example: CMSG-Property "Local-Choice"



# Property and Language Hierarchy




1: strong local
2: strong local-choice
3: weak local
4: weak local-choice
5: locally cooperative
6: regular
7: globally cooperative

# Presentation outline

# Implementation of CMSGs

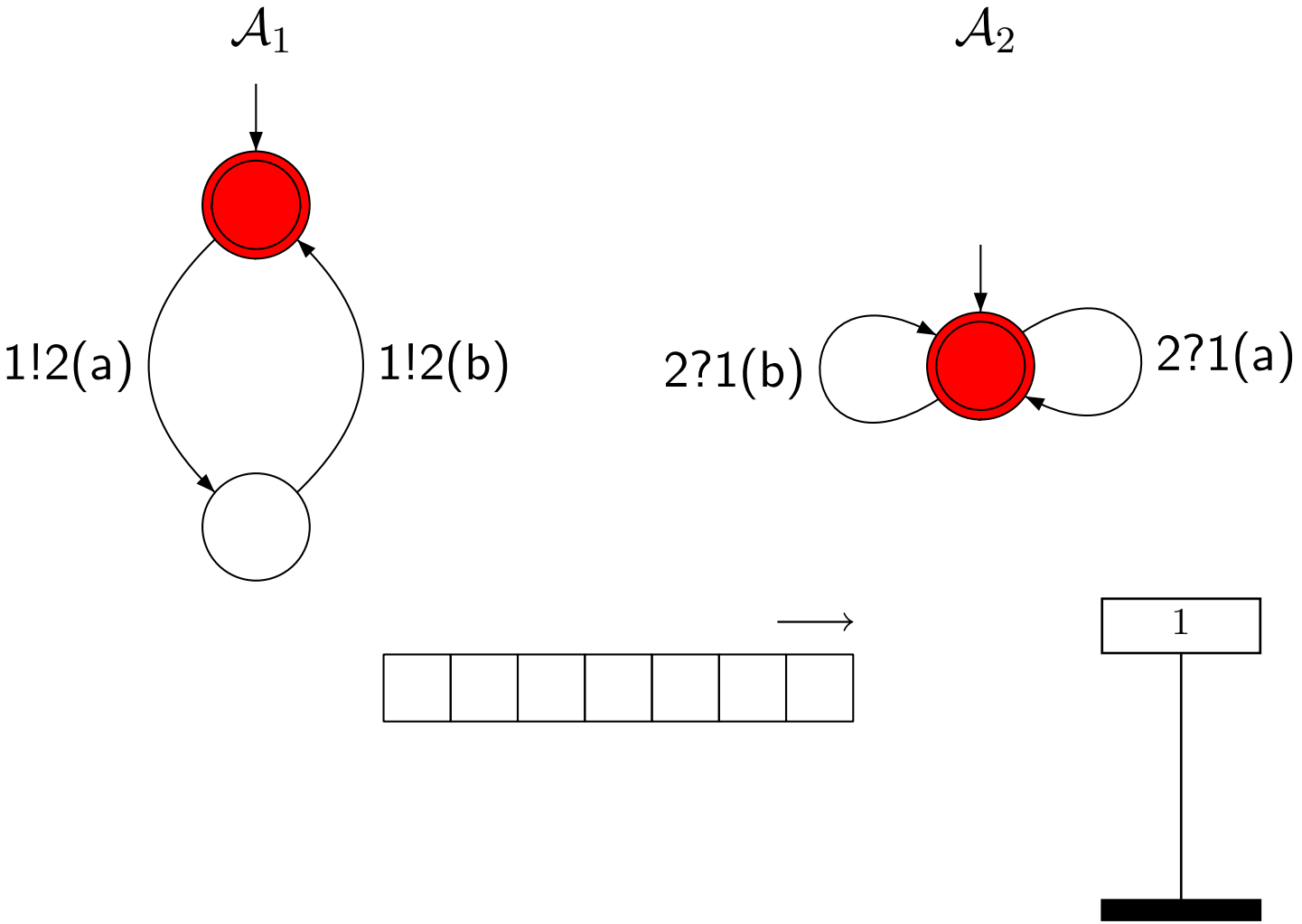
## Communicating Finite-State Machines $\mathcal{A} = \langle (\mathcal{A}_p)_{p \in \mathcal{P}}, F \rangle$

- $(A_p)_{p \in \mathcal{P}}$: set of local automata
- $F \subseteq \prod_{p \in \mathcal{P}} S_p$: set of global final states

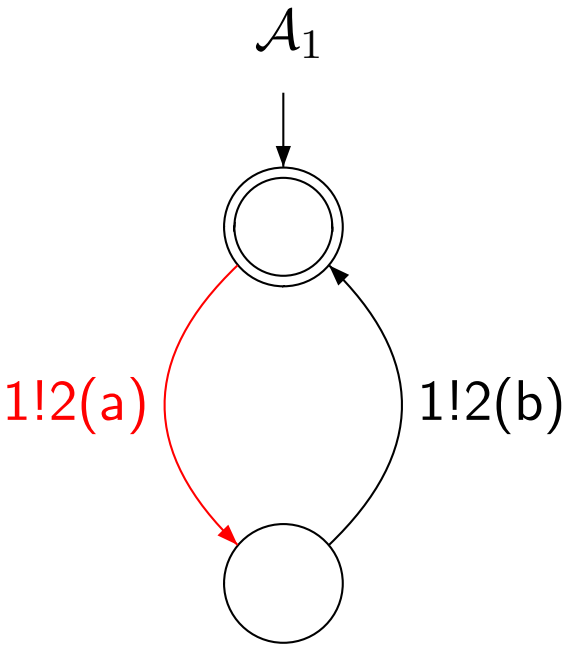
## local Automata $\mathcal{A}_p = \langle S_p, s_p, \rightarrow_p \rangle$

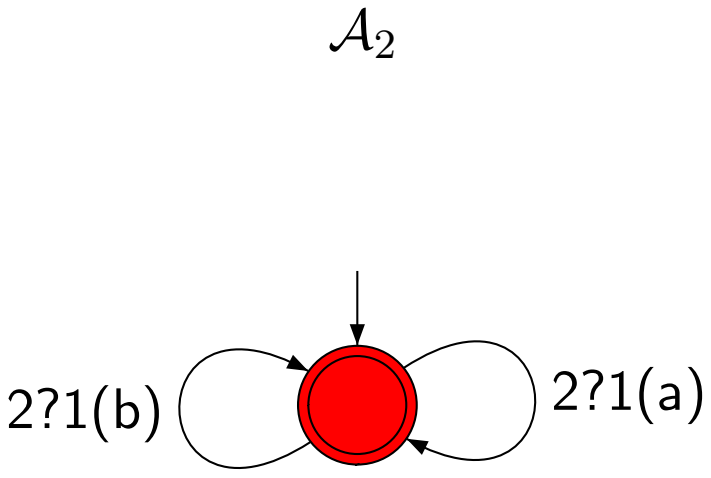
- $S_p$: final state set
- $s_p \in S_p$: starting state
- $\rightarrow \subseteq S_p \times Act \times S_p$: transition relation

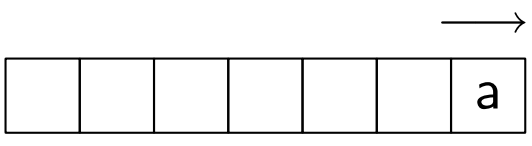
# A CFM Example (Producer-Consumer)

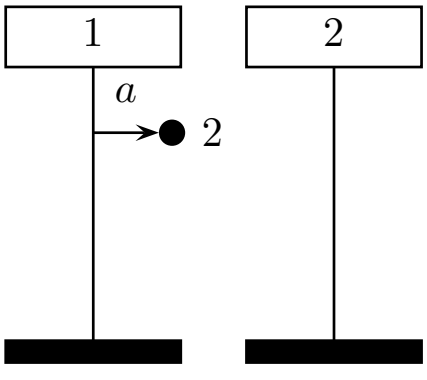


# A CFM Example (Producer-Consumer)

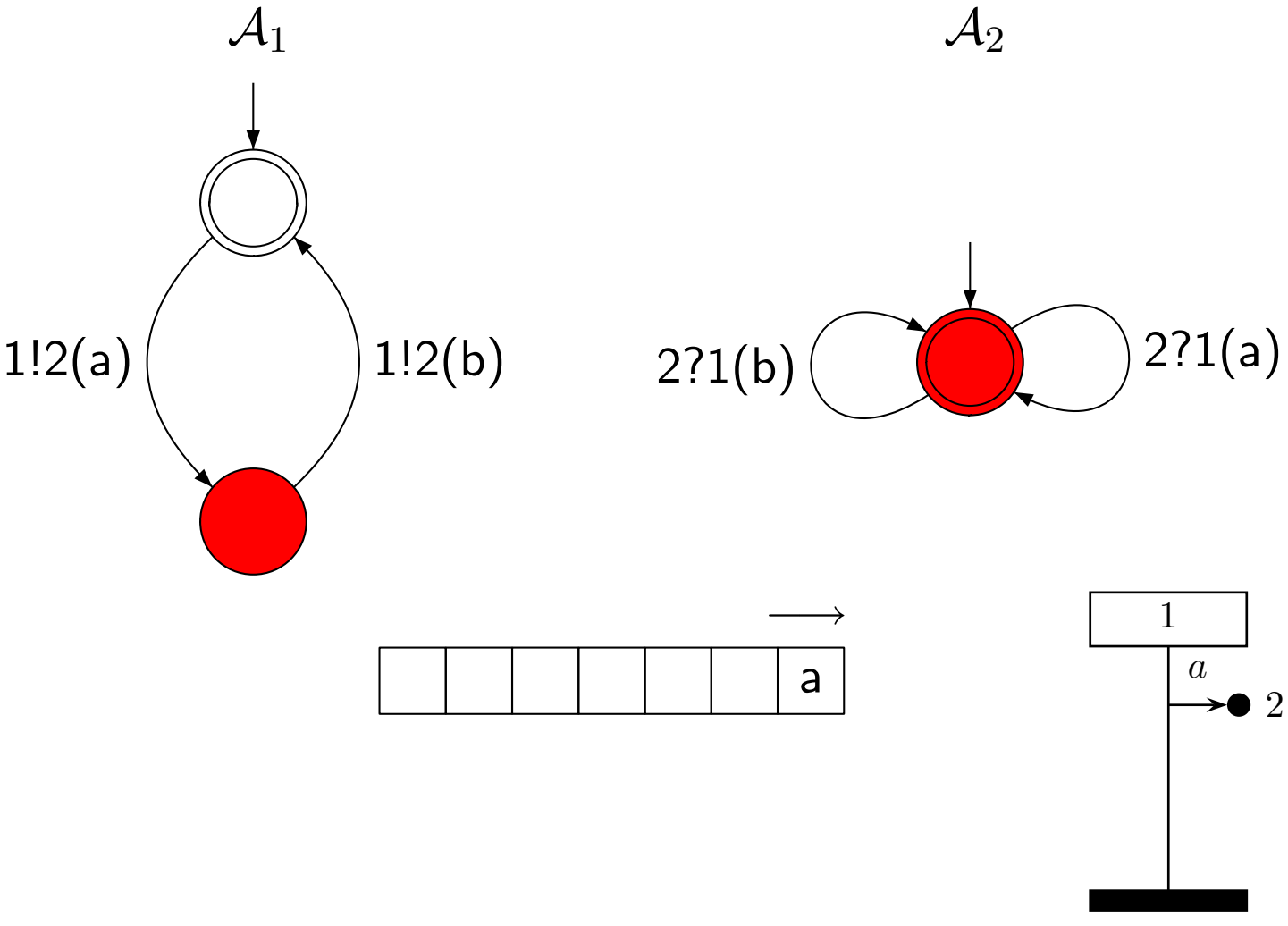


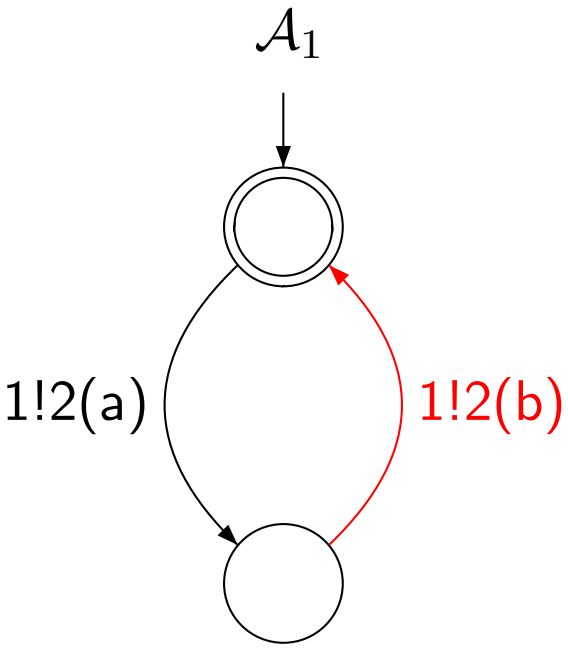



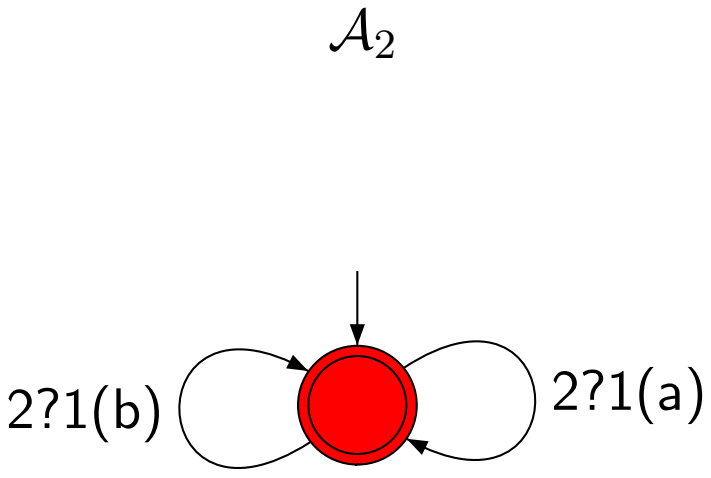



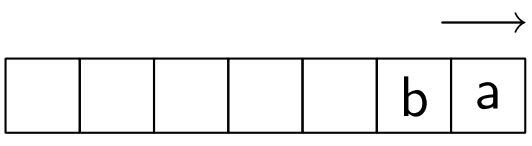
# A CFM Example (Producer-Consumer)

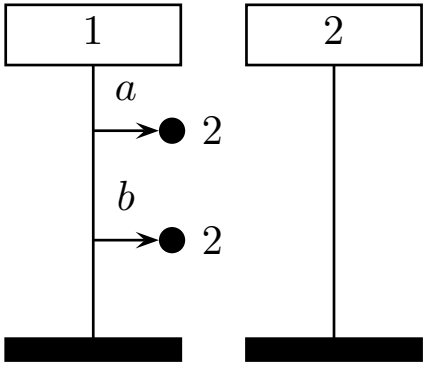


# A CFM Example (Producer-Consumer)

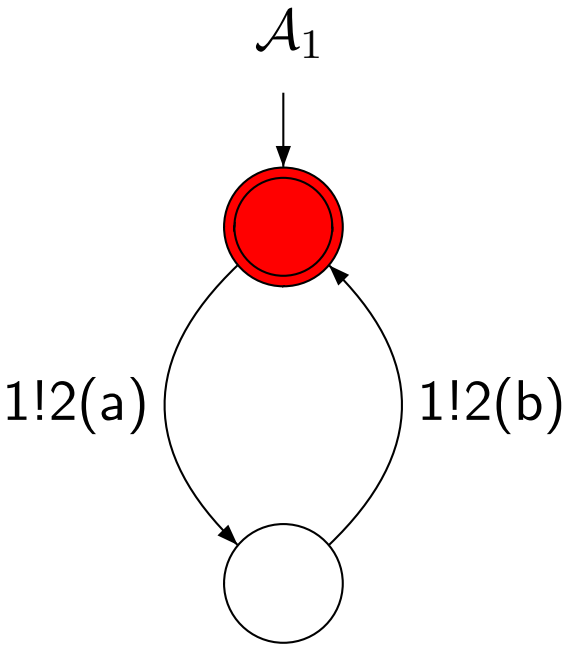


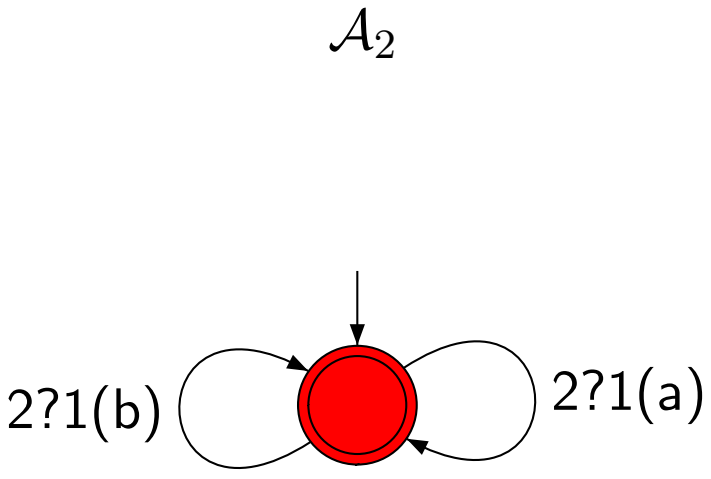



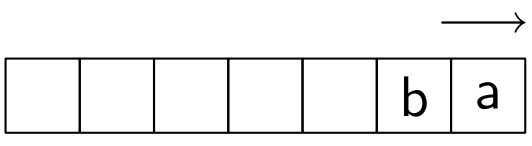



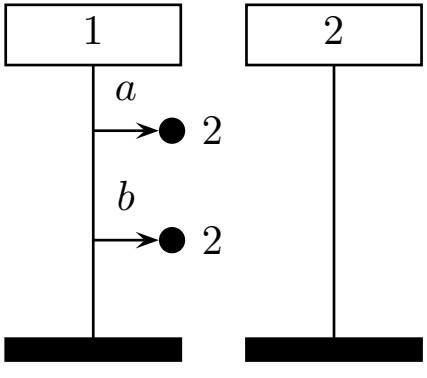



# A CFM Example (Producer-Consumer)

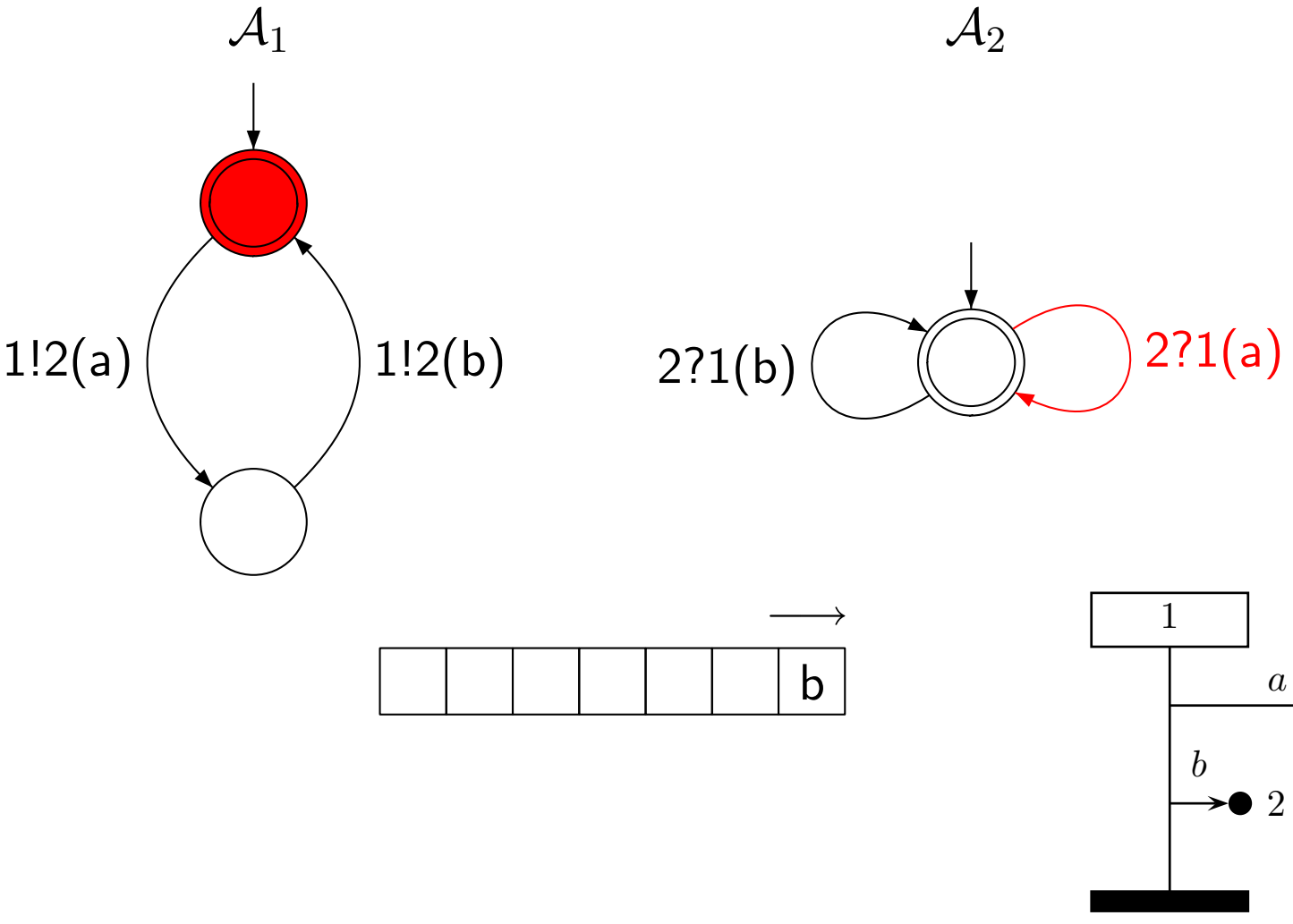


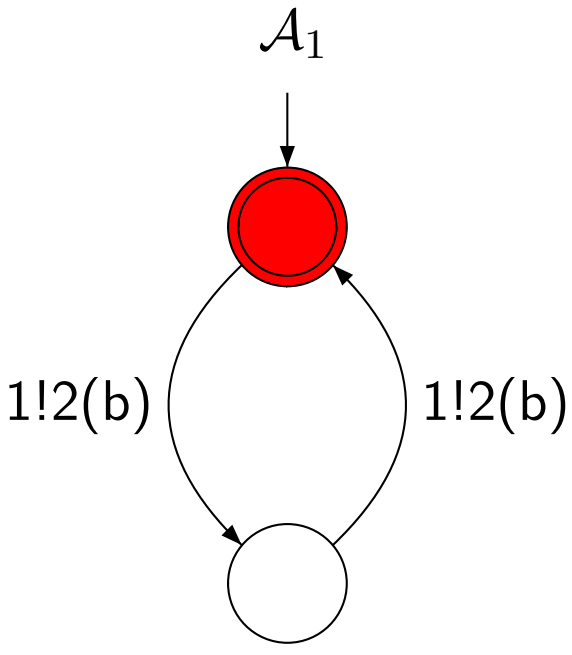



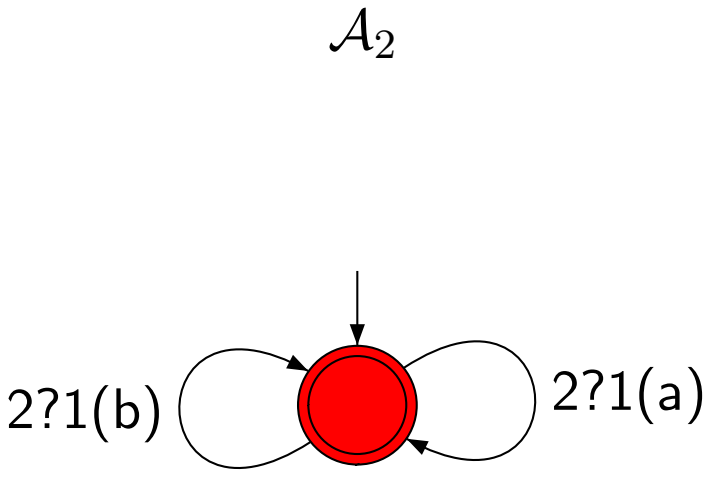



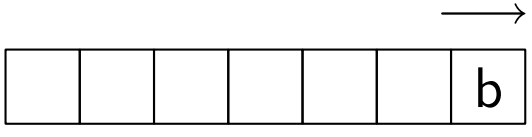
# A CFM Example (Producer-Consumer)

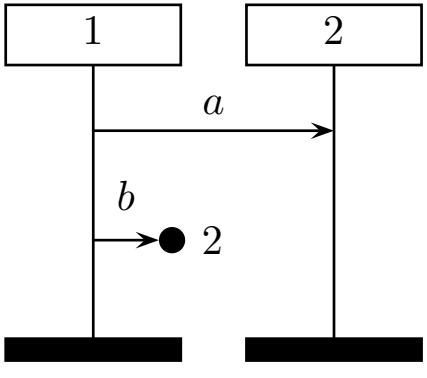


# A CFM Example (Producer-Consumer)

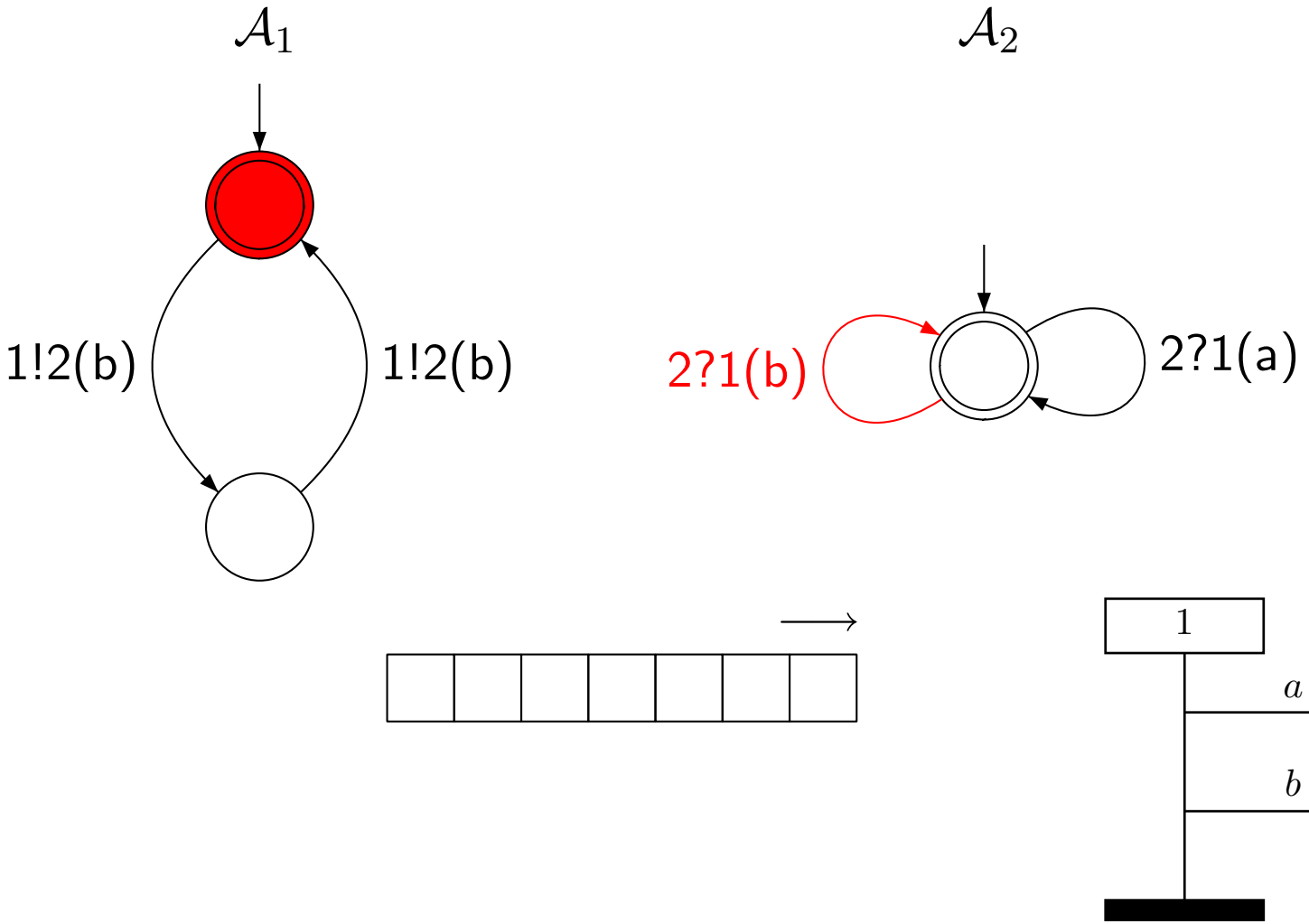


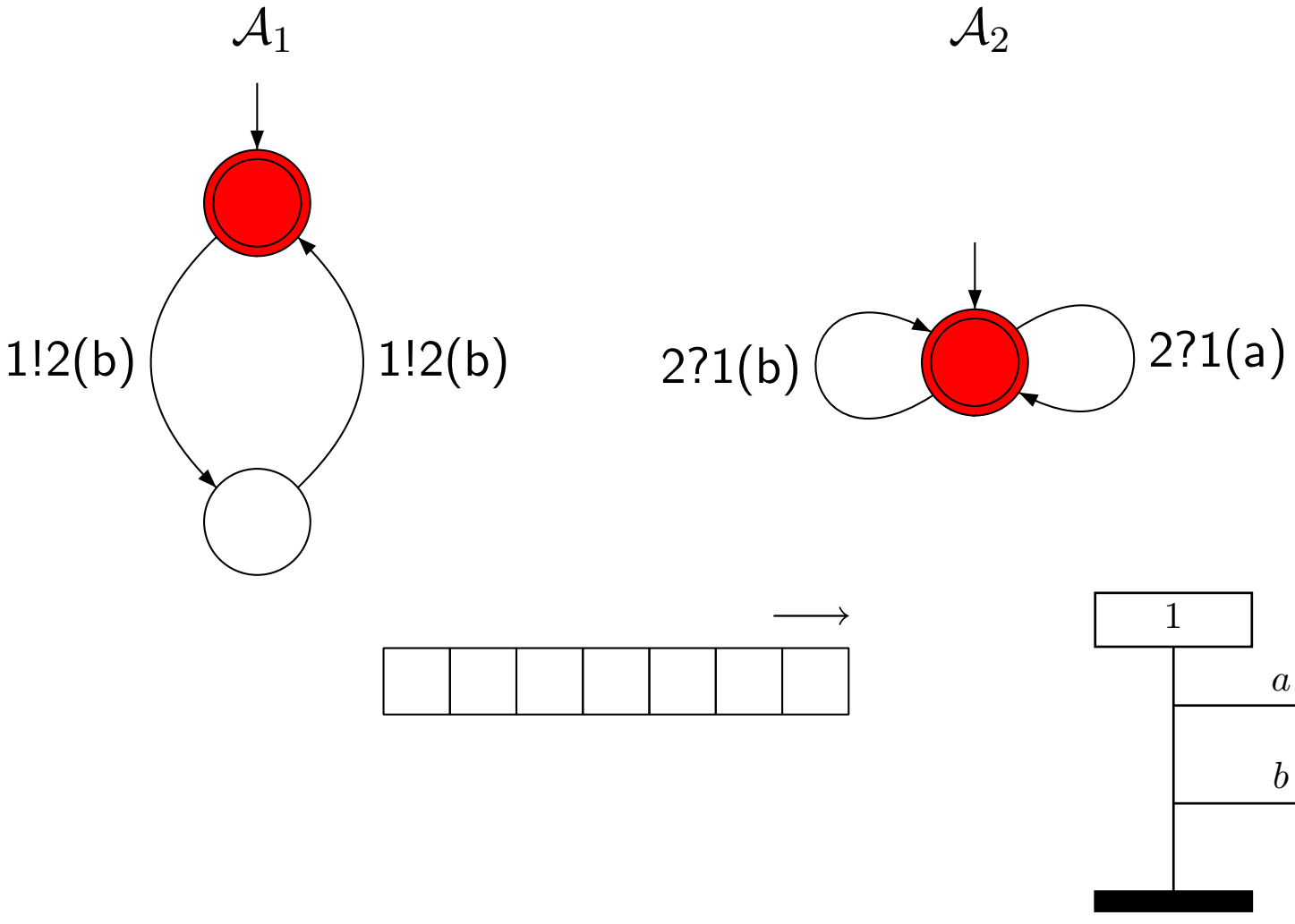



# A CFM Example (Producer-Consumer)



# A CFM Example (Producer-Consumer)



**Theorem**

Every local-choice CMSG is implementable (without deadlocks).

# Presentation outline

# Drawbacks of MSCs and MSGs

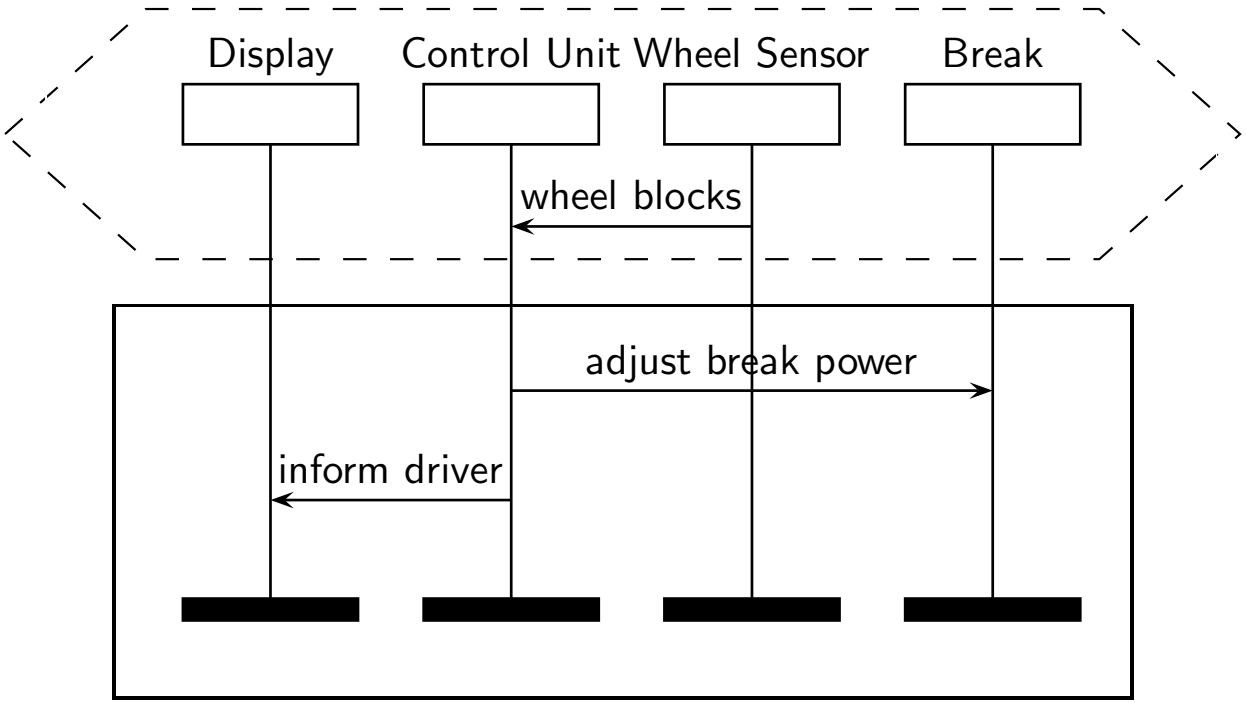
## Problems

- no distinction between what **must** and **may** happen

## Possible Approach

- using Life Sequence Charts (LSCs) (by Damm and Harel) to distinguish between
  - mandatory, optional and illegal behavior

# An LSC example (Antiblock System)



# Outlook

## Outlook

- Extension of MSCs and MSGs to simulate quantitative behavior

  Thus modeling:

  - uncertain events
  - unreliable channels
    (integrating probability into message transmission (MSCs))
  - probabilistic branching
    (integrating probability into node branching (MSGs))

- Approach: extending Life Sequence Charts to cope with quantitative behavior

## Goals:

- extensions need to be formally defined
- extensions need to be equipped with a reasonable semantics
- properties (like regularity etc.) will be defined and classified
- relation to existing probabilistic extensions of statecharts will be checked