Model Checking I
alias
Reactive Systems Verification

Luca Tesei

MSc in Computer Science, University of Camerino

Topics

- Channel Systems
- Synchronous and Asynchronous Communication
- SOS Semantics of Channel Systems as Transition Systems
- Examples

Material

Reading:

Chapter 2 of the book, pages 53–63.

More:

The slides in the following pages are taken from the material of the course “Introduction to Model Checking” held by Prof. Dr. Ir. Joost-Pieter Katoen at Aachen University.

Overview

Introduction

Modelling parallel systems

- Transition systems
- Modeling hard- and software systems
- Parallelism and communication ←

Linear Time Properties

Regular Properties

Linear Temporal Logic

Computation-Tree Logic

Equivalences and Abstraction

Channel systems

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

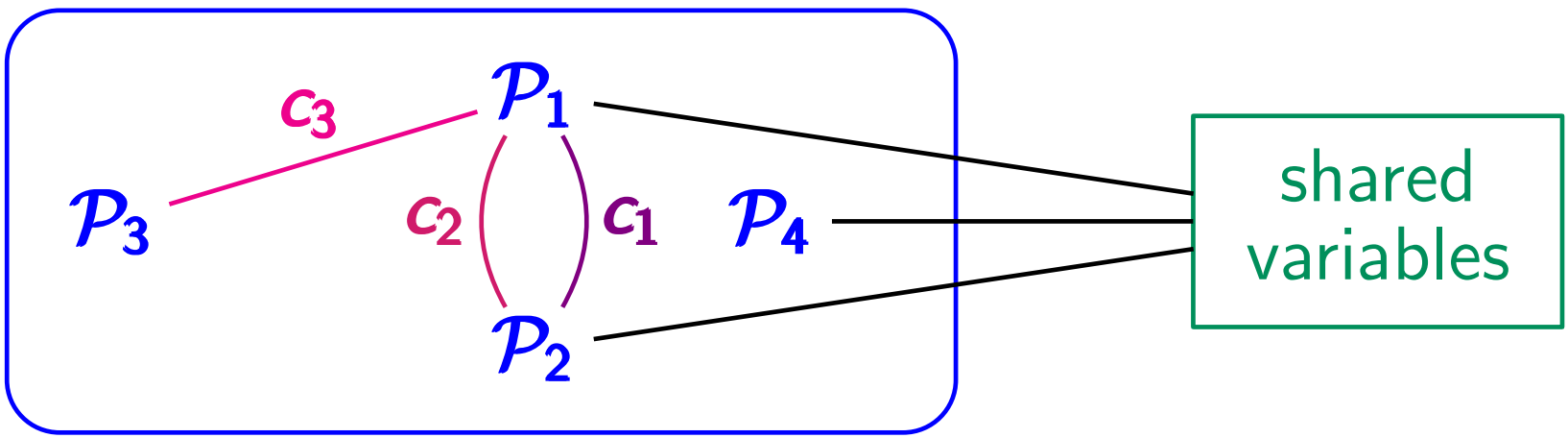
} communication over channels

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels

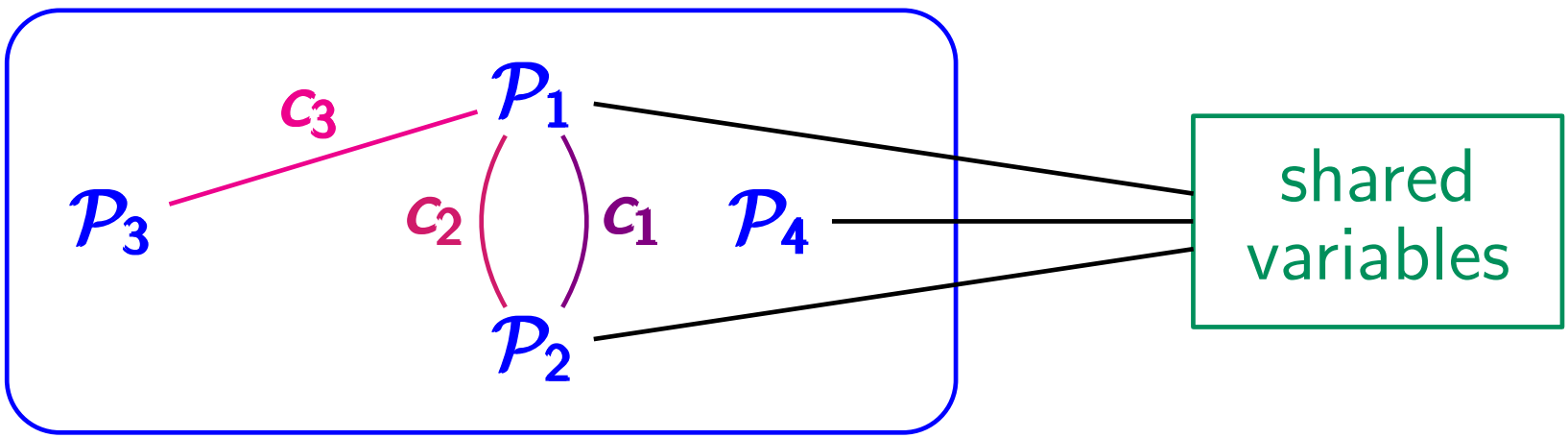


Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels



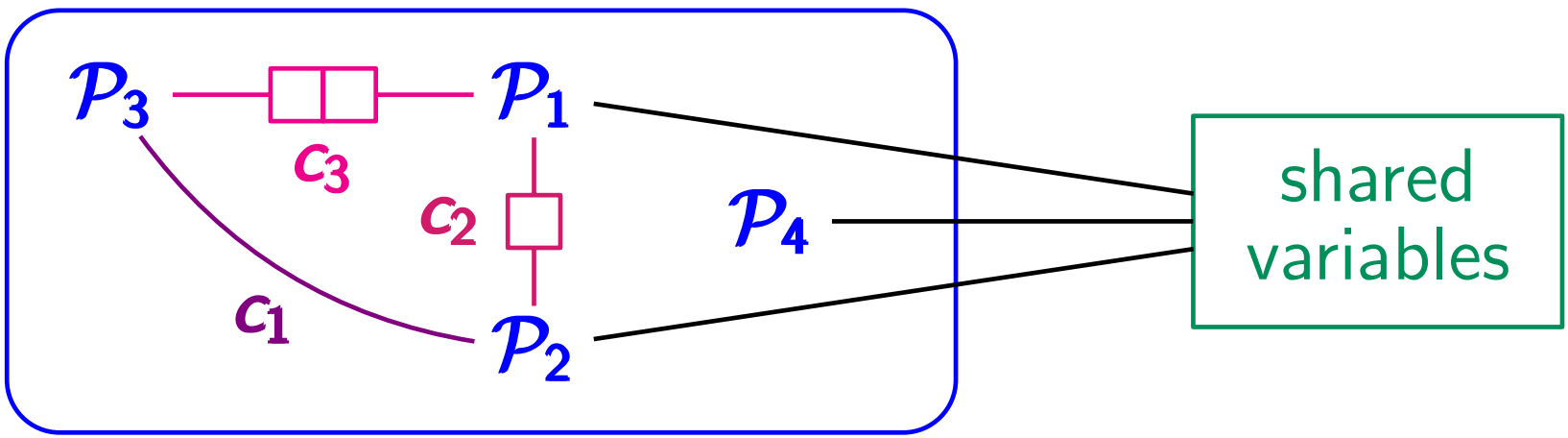
channel types: synchronous or FIFO

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels



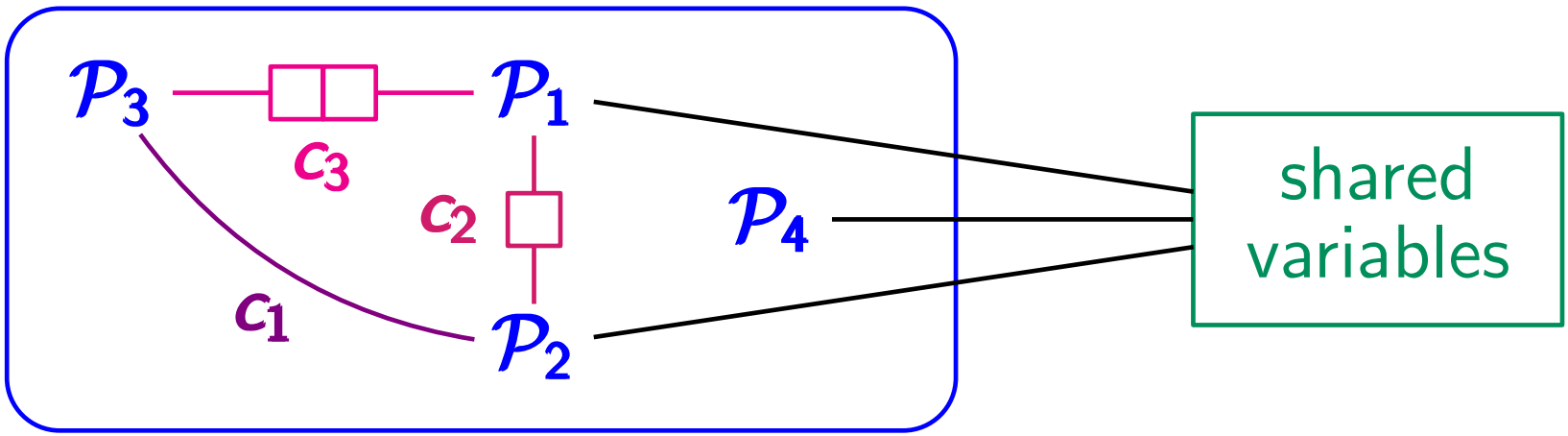
channel types: synchronous or FIFO

capacity $\widehat{=}$ number of buffer cells

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing ⟵ capacity **0**
- asynchronous message passing ⟵ capacity $\geq$ **1**



channel types: synchronous or FIFO

capacity **0** capacity $\widehat{=}$ number of buffer cells

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels

formalization through program graphs for $\mathcal{P}_1, \ldots, \mathcal{P}_n$

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels

formalization through program graphs for $\mathcal{P}_1, \ldots, \mathcal{P}_n$

- with conditional transitions $\ell_i \xrightarrow{g:\alpha} \ell_i'$ (as before)

Channel systems

representation of data-dependent parallel systems with

- communication over shared variables
- synchronous message passing
- asynchronous message passing

} communication over channels

formalization through program graphs for $\mathcal{P}_1, \ldots, \mathcal{P}_n$

- with conditional transitions $\ell_i \xrightarrow{g:\alpha} \ell_i'$ (as before)
- and communication actions

$\ell_i \xrightarrow{c!v} \ell_i'$	sending value v via channel c
$\ell_i \xrightarrow{c?x} \ell_i'$	receiving a value for variable x via channel c

Typed variables and channels

typed variable: variable x with data domain $Dom(x)$

Typed variables and channels

typed variable: variable x with data domain $Dom(x)$

evaluation for a set Var of typed variables:

type-consistent function $\eta : Var \to Values$

$\uparrow$

i.e., $\eta(x) \in Dom(x)$

Typed variables and channels

typed variable: variable x with data domain $Dom(x)$

evaluation for a set Var of typed variables:

type-consistent function $\eta : Var \to Values$

$\uparrow$

i.e., $\eta(x) \in Dom(x)$

typed channel: channel c with

capacity $cap(c) \in \mathbb{N} \cup \{\infty\}$ and domain $Dom(c)$

Typed variables and channels

typed variable: variable x with data domain $Dom(x)$

evaluation for a set Var of typed variables:

type-consistent function $\eta : Var \to Values$

$\uparrow$

i.e., $\eta(x) \in Dom(x)$

typed channel: channel c with

capacity $cap(c) \in \mathbb{N} \cup \{\infty\}$ and domain $Dom(c)$

evaluation for a set $Chan$ of typed channels:

type-consistent function $\xi : Chan \to Values^*$

Typed variables and channels

typed variable: variable x with data domain $Dom(x)$

evaluation for a set Var of typed variables:

type-consistent function $\eta : Var \to Values$

$\uparrow$

i.e., $\eta(x) \in Dom(x)$

typed channel: channel c with

capacity $cap(c) \in \mathbb{N} \cup \{\infty\}$ and domain $Dom(c)$

evaluation for a set $Chan$ of typed channels:

type-consistent function $\xi : Chan \to Values^*$

s.t. $\xi(c)$ is a word over $Dom(c)$ of length $\leq cap(c)$

Channel system (CS)

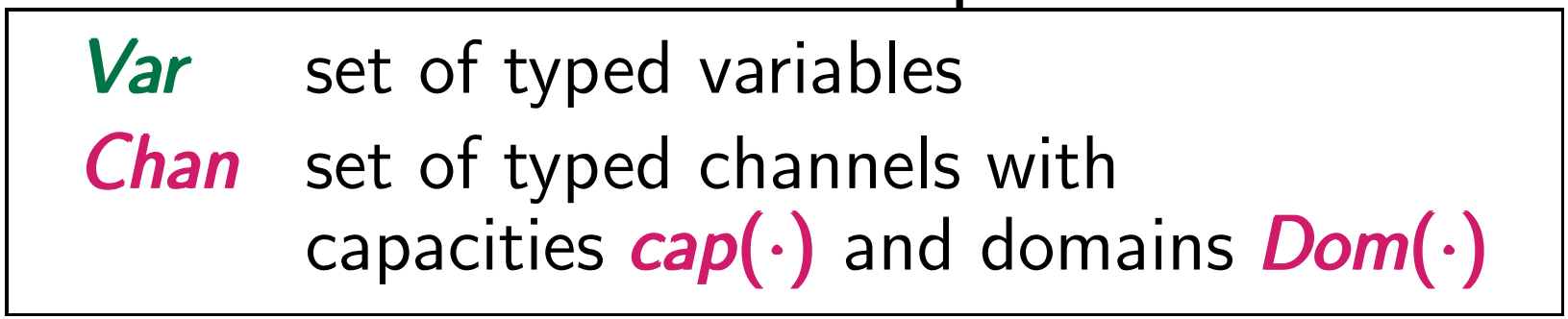
$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs

Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(Var, Chan)$

Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(\mathit{Var}, \mathit{Chan})$



Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(\mathit{Var}, \mathit{Chan})$

Var	set of typed variables
Chan	set of typed channels with capacities $\mathit{cap}(\cdot)$ and domains $\mathit{Dom}(\cdot)$

program graphs $\mathcal{P}_i = (\mathit{Loc}_i, \mathit{Act}_i, \mathit{Effect}_i, \hookrightarrow_i, \mathit{Loc}_{0,i}, g_0)$ with conditional transitions

$\ell \xrightarrow{g:\alpha}_i \ell'$ guarded command

Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(Var, Chan)$

Var	set of typed variables
$Chan$	set of typed channels with capacities $cap(\cdot)$ and domains $Dom(\cdot)$

program graphs $\mathcal{P}_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_0)$ with conditional transitions

$\ell \xrightarrow{g:\alpha}_i \ell'$ where $g \in Cond(Var)$, $\alpha \in Act_i$

Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(Var, Chan)$

Var	set of typed variables
$Chan$	set of typed channels with capacities $cap(\cdot)$ and domains $Dom(\cdot)$

program graphs $\mathcal{P}_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_0)$ with conditional transitions

$\ell \xrightarrow{g:\alpha}_i \ell'$ guarded command

$\ell \xrightarrow{c!v}_i \ell'$ sending value v via channel c

Channel system (CS)

$[\mathcal{P}_1 | \mathcal{P}_2 | \ldots | \mathcal{P}_n]$ where $\mathcal{P}_i$ are program graphs over a pair $(Var, Chan)$

Var	set of typed variables
$Chan$	set of typed channels with capacities $cap(\cdot)$ and domains $Dom(\cdot)$

program graphs $\mathcal{P}_i = (Loc_i, Act_i, Effect_i, \hookrightarrow_i, Loc_{0,i}, g_0)$ with conditional transitions

$\ell \xhookrightarrow{g:\alpha}_i \ell'$ guarded command

$\ell \xhookrightarrow{c!v}_i \ell'$ sending value v via channel c

$\ell \xhookrightarrow{c?x}_i \ell'$ receiving a value for variable x via channel c

Effect of communication actions in CS

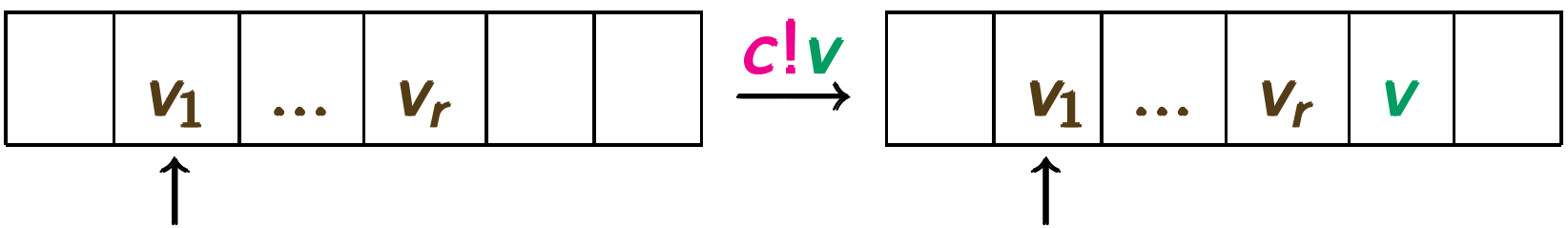
asynchronous message passing via channels of capacity ≥ 1

	enabled if ...	effect
sending $c!v$		
receiving $c?x$		

Effect of communication actions in CS

asynchronous message passing via channels of capacity ≥ 1

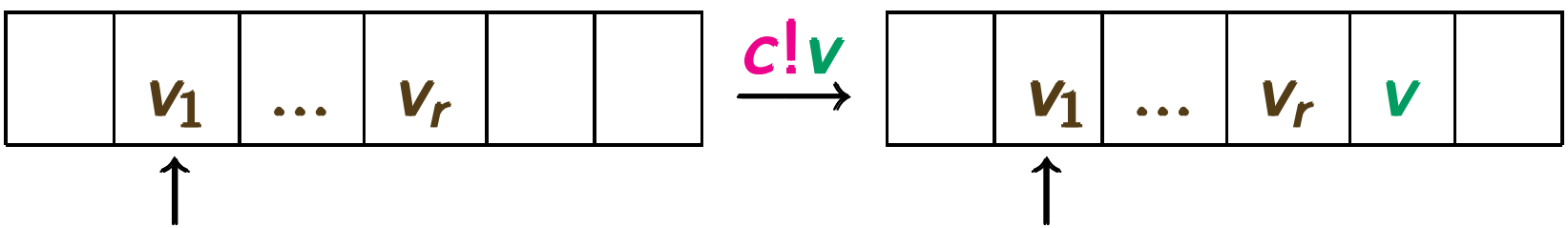
	enabled if ...	effect
sending $c!v$	channel c not full	$add(c, v)$
receiving $c?x$		



Effect of communication actions in CS

asynchronous message passing via channels of capacity ≥ 1

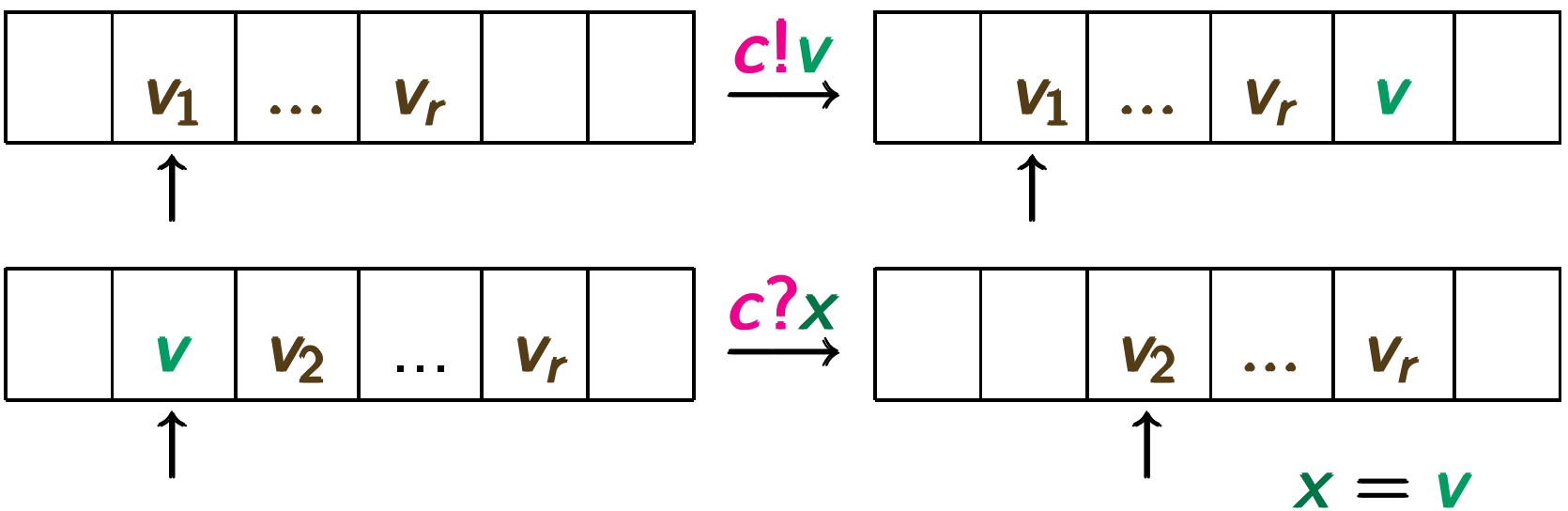
	enabled if ...	effect
sending $c!v$	channel c not full	$add(c, v)$
receiving $c?x$	channel c not empty $v = front(c)$	$x := v$ $remove(c)$



Effect of communication actions in CS

asynchronous message passing via channels of capacity ≥ 1

	enabled if ...	effect
sending $c!v$	channel c not full	$add(c, v)$
receiving $c?x$	channel c not empty $v = front(c)$	$x := v$ $remove(c)$



Effect of communication actions in CS

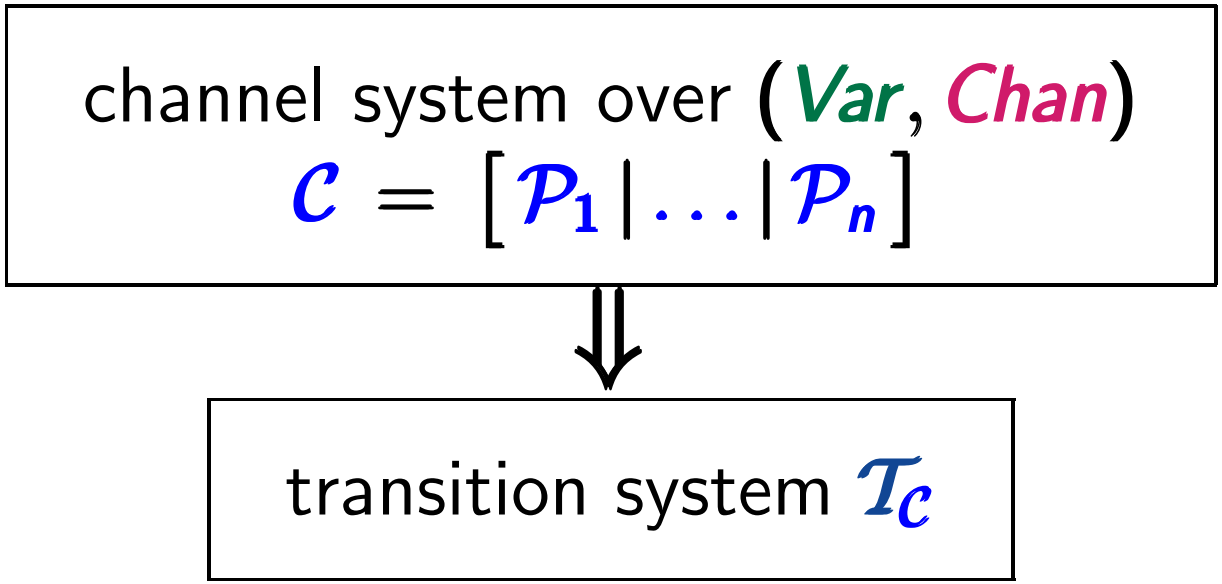
asynchronous message passing via channels of capacity ≥ 1

	enabled if ...	effect
sending $c!v$	channel c not full	$add(c, v)$
receiving $c?x$	channel c not empty $v = front(c)$	$x := v$ $remove(c)$

synchronous message passing via channels of capacity 0

- $c!v$ and $c?x$ are executed at the same time
- effect $x := v$

TS-semantics of channel systems



TS-semantics of channel systems

channel system over $(Var, Chan)$

$$\mathcal{C} = [\mathcal{P}_1 | \ldots | \mathcal{P}_n]$$

$\Downarrow$

transition system $\mathcal{T}_{\mathcal{C}}$

states of $\mathcal{T}_{\mathcal{C}}$ have the form

$$\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$$

- $\ell_1, \ldots, \ell_n$: locations of $\mathcal{P}_1, \ldots, \mathcal{P}_n$
- η: variable valuation
- ξ: channel evaluation

TS-semantics of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where

- ℓ_i location of program graph $\mathcal{P}_i$,
- $\eta \in \mathit{Eval}(\mathit{Var})$ variable evaluation
- $\xi \in \mathit{Eval}(\mathit{Chan})$ channel evaluation

TS-semantics of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where

- ℓ_i location of program graph $\mathcal{P}_i$,
- $\eta \in \mathit{Eval}(\mathit{Var})$ variable evaluation
- $\xi \in \mathit{Eval}(\mathit{Chan})$ channel evaluation

variable evaluation:

$$\eta : \mathit{Var} \longrightarrow \bigcup_{x \in \mathit{Var}} \mathit{Dom}(x) \quad \text{with } \eta(x) \in \mathit{Dom}(x)$$

channel evaluation:

$$\xi : \mathit{Chan} \rightarrow \bigcup_{c \in \mathit{Chan}} \mathit{Dom}(c)^* \quad \text{with } \xi(c) \in \mathit{Dom}(c)^* \text{ and } |\xi(c)| \leq \mathit{cap}(c)$$

TS-semantics of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where

- ℓ_i location of program graph $\mathcal{P}_i$,
- $\eta \in \mathit{Eval}(\mathit{Var})$ variable evaluation
- $\xi \in \mathit{Eval}(\mathit{Chan})$ channel evaluation

variable evaluation:

$$\eta : \mathit{Var} \longrightarrow \bigcup_{x \in \mathit{Var}} \mathit{Dom}(x) \quad \text{with } \eta(x) \in \mathit{Dom}(x)$$

channel evaluation:

$$\xi : \mathit{Chan} \rightarrow \bigcup_{c \in \mathit{Chan}} \mathit{Dom}(c)^* \quad \text{with } \xi(c) \in \mathit{Dom}(c)^* \text{ and } |\xi(c)| \leq \mathit{cap}(c)$$

only channels c with $\mathit{cap}(c) \geq 1$ are relevant

Transition relation of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where $\ell_i \in \mathit{Loc}_i$, $\eta \in \mathit{Eval}(\mathit{Var})$, $\xi \in \mathit{Eval}(\mathit{Chan})$

Transition relation of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where $\ell_i \in \mathit{Loc}_i$, $\eta \in \mathit{Eval}(\mathit{Var})$, $\xi \in \mathit{Eval}(\mathit{Chan})$

transition relation $\longrightarrow$ is given by SOS-rules:

- interleaving rules for $\alpha \in \mathit{Act}_i$
- rules for message passing along channels

Transition relation of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where $\ell_i \in \mathit{Loc}_i$, $\eta \in \mathit{Eval}(\mathit{Var})$, $\xi \in \mathit{Eval}(\mathit{Chan})$

transition relation $\longrightarrow$ is given by SOS-rules:

- interleaving rules for $\alpha \in \mathit{Act}_i$
- rules for message passing along channels

interleaving rule for actions $\alpha \in \mathit{Act}_i$:

$$\frac{\ell_i \xhookrightarrow{g:\alpha}_i \ell_i' \ \wedge \ \eta \models g}{\langle \ell_1, .., \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\alpha} \langle \ell_1, .., \ell_i', \ldots, \ell_n, \mathit{Effect}_i(\alpha, \eta), \xi \rangle}$$

Transition relation of channel systems

states $\langle \ell_1, \ldots, \ell_n, \eta, \xi \rangle$ where $\ell_i \in \mathit{Loc}_i$, $\eta \in \mathit{Eval}(\mathit{Var})$, $\xi \in \mathit{Eval}(\mathit{Chan})$

transition relation $\longrightarrow$ is given by SOS-rules:

- interleaving rules for $\alpha \in \mathit{Act}_i$
- rules for message passing along channels

interleaving rule for actions $\alpha \in \mathit{Act}_i$:

$$\frac{\ell_i \xrightarrow{g:\alpha}_i \ell_i' \ \wedge \ \eta \models g}{\langle \ell_1, .., \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\alpha} \langle \ell_1, .., \ell_i', \ldots, \ell_n, \mathit{Effect}_i(\alpha, \eta), \xi \rangle}$$

$\uparrow$

does not affect the channel evaluation ξ

SOS-rules for asynchronous message passing

for channel c with $cap(c) \geq 1$

SOS-rules for asynchronous message passing

for channel c with $cap(c) \geq 1$

receiving a message:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \ \wedge \ \xi(c) = v_1 v_2 \ldots v_k \ \wedge \ k \geq 1}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \xi' \rangle}$$

SOS-rules for asynchronous message passing

for channel c with $cap(c) \geq 1$

receiving a message:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \ \wedge \ \xi(c) = v_1 v_2 \ldots v_k \ \wedge \ k \geq 1}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \xi' \rangle}$$

where $\eta' = \eta[x{:=}v_1]$

$$\eta[x{:=}v_1](y) = \begin{cases} \eta(y) & \text{if } y \neq x \\ v_1 & \text{if } y = x \end{cases}$$

SOS-rules for asynchronous message passing

for channel c with $cap(c) \geq 1$

receiving a message:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \ \wedge \ \xi(c) = v_1 v_2 \ldots v_k \ \wedge \ k \geq 1}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \xi' \rangle}$$

where $\eta' = \eta[x{:=}v_1]$ and $\xi' = \xi[c{:=}v_2 \ldots v_k]$

$$\eta[x{:=}v_1](y) = \begin{cases} \eta(y) & \text{if } y \neq x \\ v_1 & \text{if } y = x \end{cases}$$

$$\xi[c{:=}v_2 \ldots v_k](d) = \begin{cases} \xi(d) & \text{if } d \neq c \\ v_2 \ldots v_k & \text{if } d = c \end{cases}$$

SOS-rules for asynchronous message passing

for channel c with $cap(c) \geq 1$

receiving a message:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \ \wedge \ \xi(c) = v_1 v_2 \ldots v_k \ \wedge \ k \geq 1}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta', \xi' \rangle}$$

where $\eta' = \eta[x := v_1]$ and $\xi' = \xi[c := v_2 \ldots v_k]$

sending a message:

$$\frac{\ell_i \xrightarrow{c!v}_i \ell_i' \ \wedge \ \xi(c) = v_1 \ldots v_k \ \wedge \ k < cap(c)}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_n, \eta, \xi[c := v_1 \ldots v_k v] \rangle}$$

SOS-rules for synchronous message passing

SOS-rules for synchronous message passing

for synchronous channel c:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \;\wedge\; \ell_j \xrightarrow{c!v}_j \ell_j' \;\wedge\; i \neq j}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_j', \ldots, \ell_n, \eta', \xi' \rangle}$$

SOS-rules for synchronous message passing

for synchronous channel c:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \;\wedge\; \ell_j \xrightarrow{c!v}_j \ell_j' \;\wedge\; i \neq j}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_j', \ldots, \ell_n, \eta', \xi' \rangle}$$

where $\eta' = \eta[x := v]$

SOS-rules for synchronous message passing

for synchronous channel c:

$$\frac{\ell_i \xrightarrow{c?x}_i \ell_i' \;\wedge\; \ell_j \xrightarrow{c!v}_j \ell_j' \;\wedge\; i \neq j}{\langle \ell_1, \ldots, \ell_i, \ldots, \ell_j, \ldots, \ell_n, \eta, \xi \rangle \xrightarrow{\tau} \langle \ell_1, \ldots, \ell_i', \ldots, \ell_j', \ldots, \ell_n, \eta', \xi' \rangle}$$

where $\eta' = \eta[x{:=}v]$ and $\xi' = \xi$

How many states...

has a transition system for a channel system with ... ?

- 2 processes with 2 locations each
- 2 Boolean variables
- 2 channels of capacity 10 and Boolean values

How many states...

has a transition system for a channel system with ... ?

- 2 processes with 2 locations each
- 2 Boolean variables
- 2 channels of capacity 10 and Boolean values

answer:

$$2*2*2*2*(2^{11}-1)*(2^{11}-1)$$

note: $2^{11}-1 = 1+2+2^2+...+2^{10}$

How many states...

has a transition system for a channel system with ... ?

- 2 processes with 2 locations each
- 2 Boolean variables
- 2 channels of capacity 10 and Boolean values

answer:

$2*2*2*2*(2^{11}-1)*(2^{11}-1) > 2^{24} > 25$ mio

note: $2^{11}-1 = 1+2+2^2+...+2^{10}$

How many states...

has a transition system for a channel system with ... ?

- 2 processes with 2 locations each
- 2 Boolean variables
- 2 channels of capacity 10 and Boolean values

answer:

$2*2*2*2*(2^{11}-1)*(2^{11}-1) > 2^{24} > 25$ mio

note: $2^{11}-1 = 1+2+2^2+...+2^{10}$

... with an unbounded channel ?

How many states...

has a transition system for a channel system with ... ?

- 2 processes with 2 locations each
- 2 Boolean variables
- 2 channels of capacity 10 and Boolean values

answer:

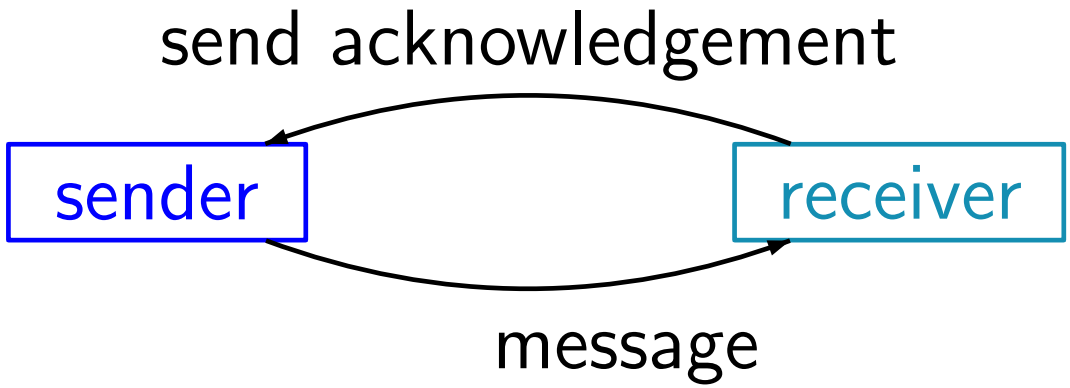
$2*2*2*2*(2^{11}-1)*(2^{11}-1) > 2^{24} > 25$ mio

note: $2^{11}-1 = 1+2+2^2+...+2^{10}$

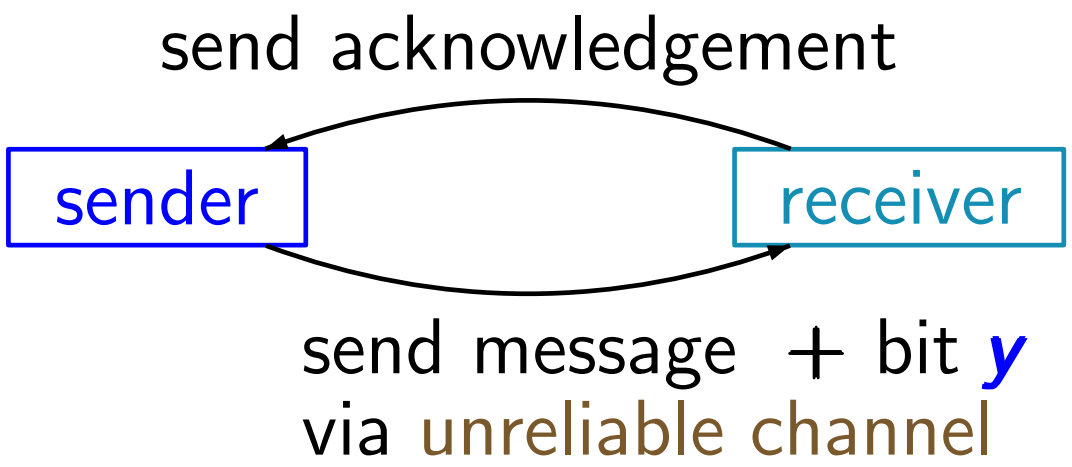
... with an unbounded channel ?

answer: ∞

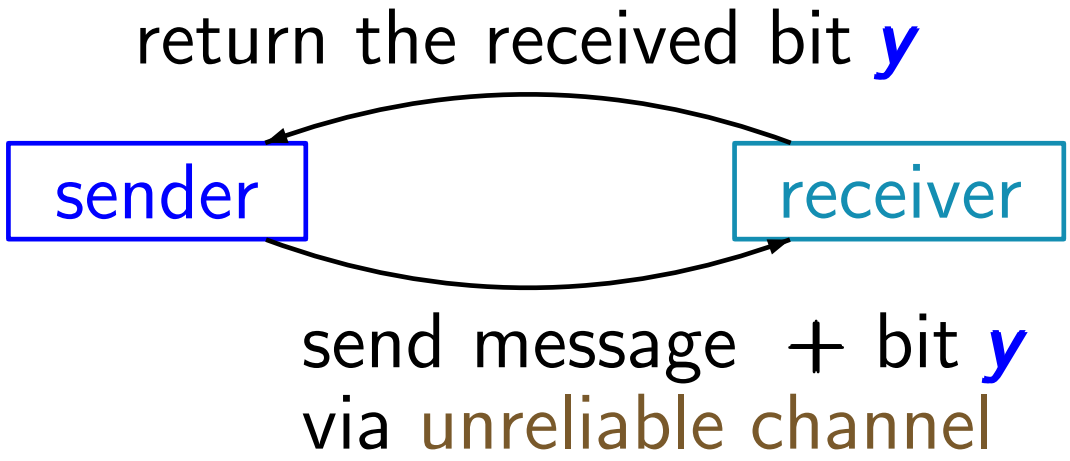
Alternating bit protocol (ABP)



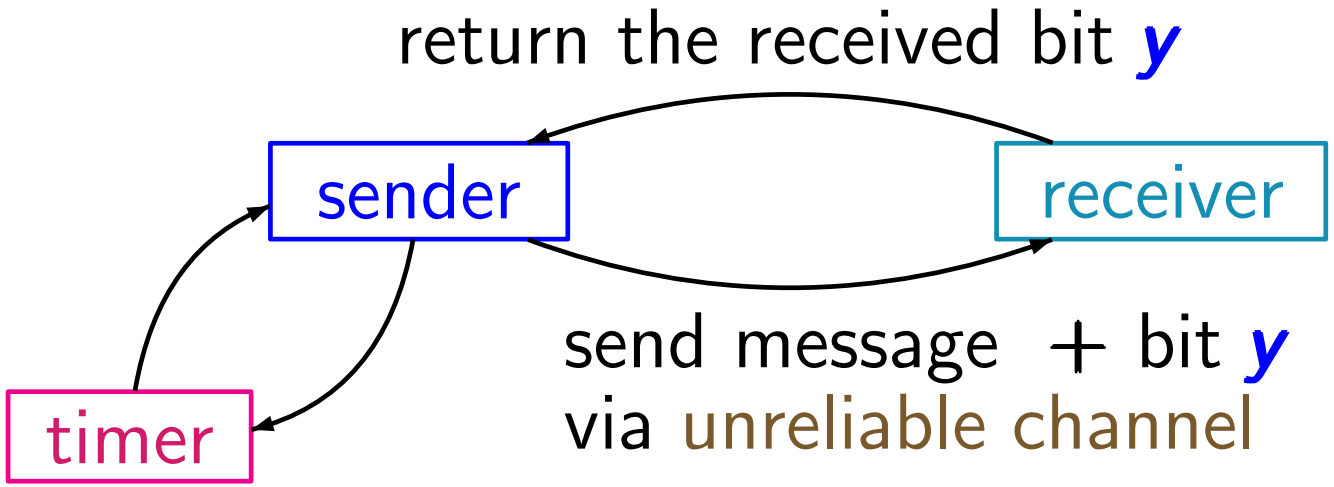
Alternating bit protocol (ABP)



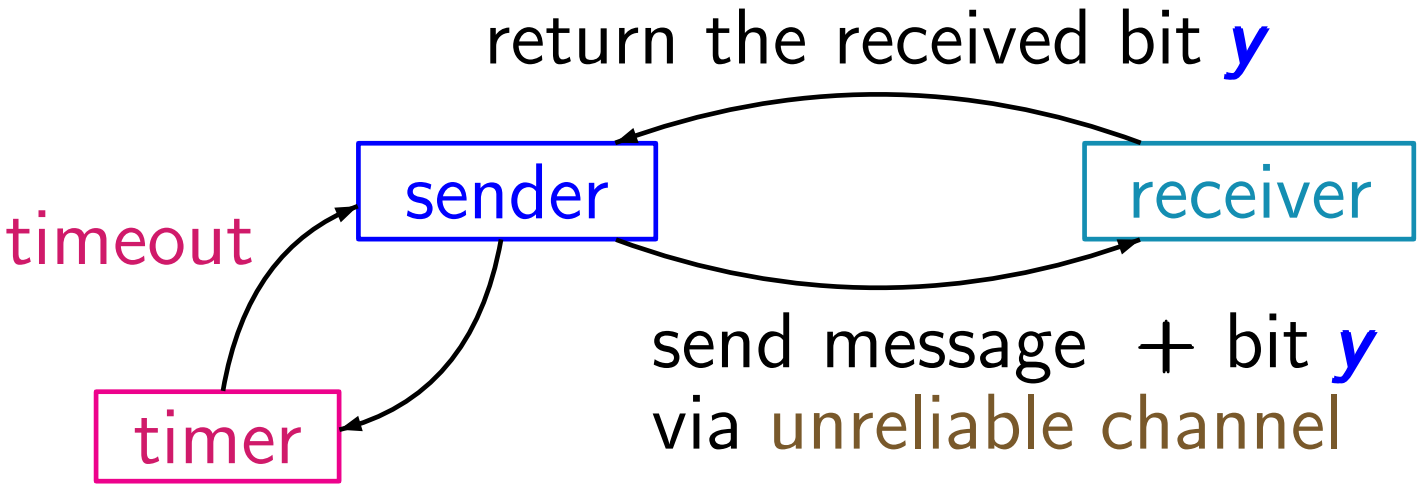
Alternating bit protocol (ABP)



Alternating bit protocol (ABP)

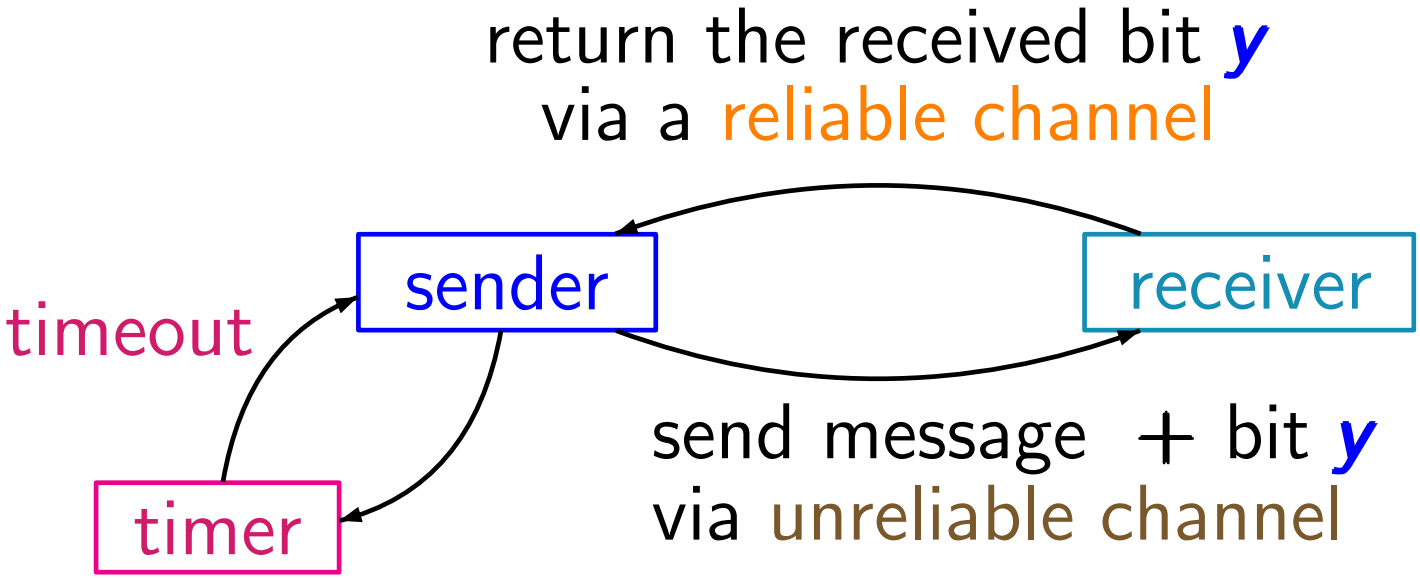


Protocol for the sender



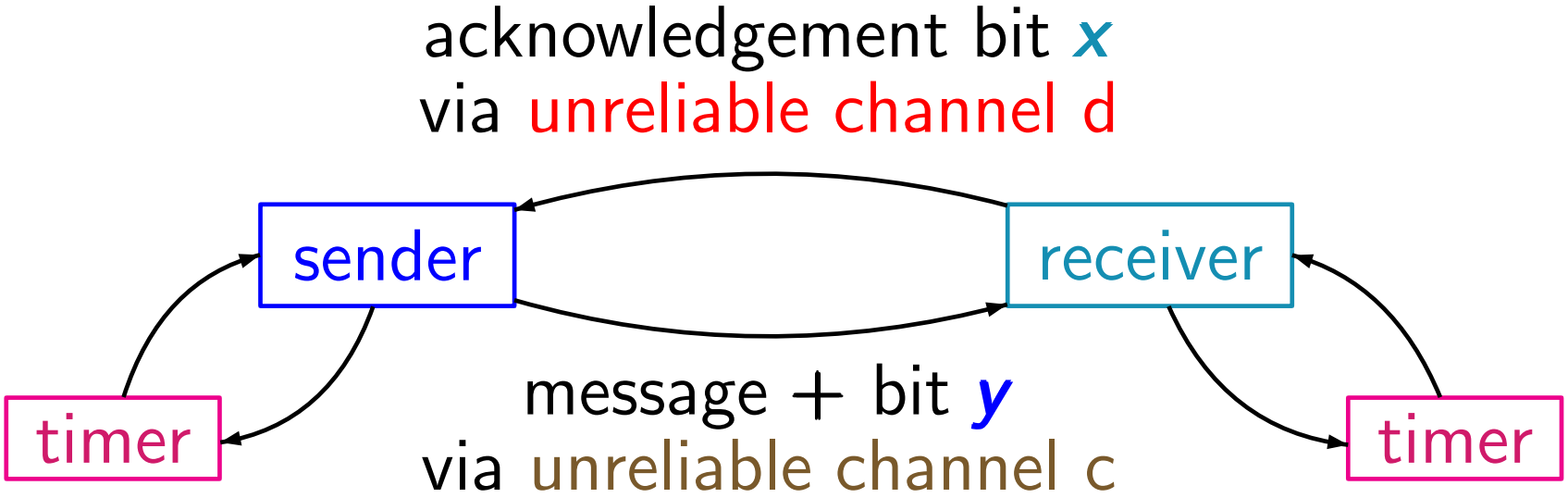

```
LOOP FOREVER
(1)  send message + bit y and activate timer
(2)  AWAIT timeout or acknowledgement DO
          IF timeout THEN goto (1)
                     ELSE turn off timer; y:=¬y
          FI
     OD
```

Protocol for the sender

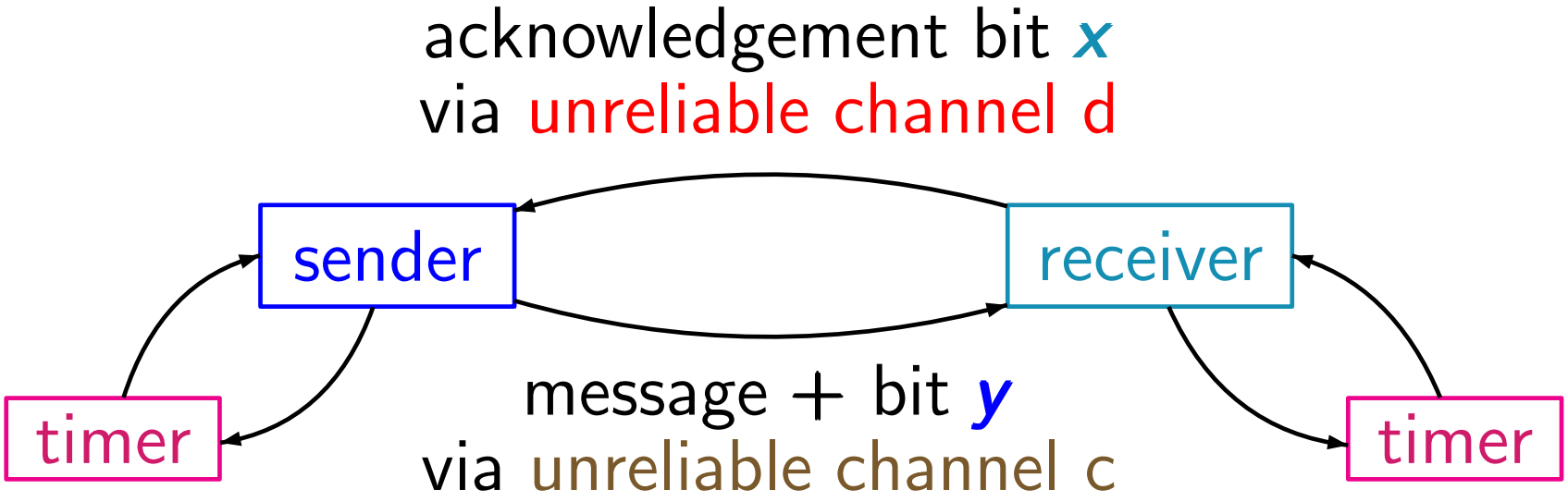



```
LOOP FOREVER
(1)  send message + bit y and activate timer
(2)  AWAIT timeout or acknowledgement DO
          IF timeout THEN goto (1)
                     ELSE turn off timer; y:=¬y
          FI
     OD
```

If both channels are unreliable ...

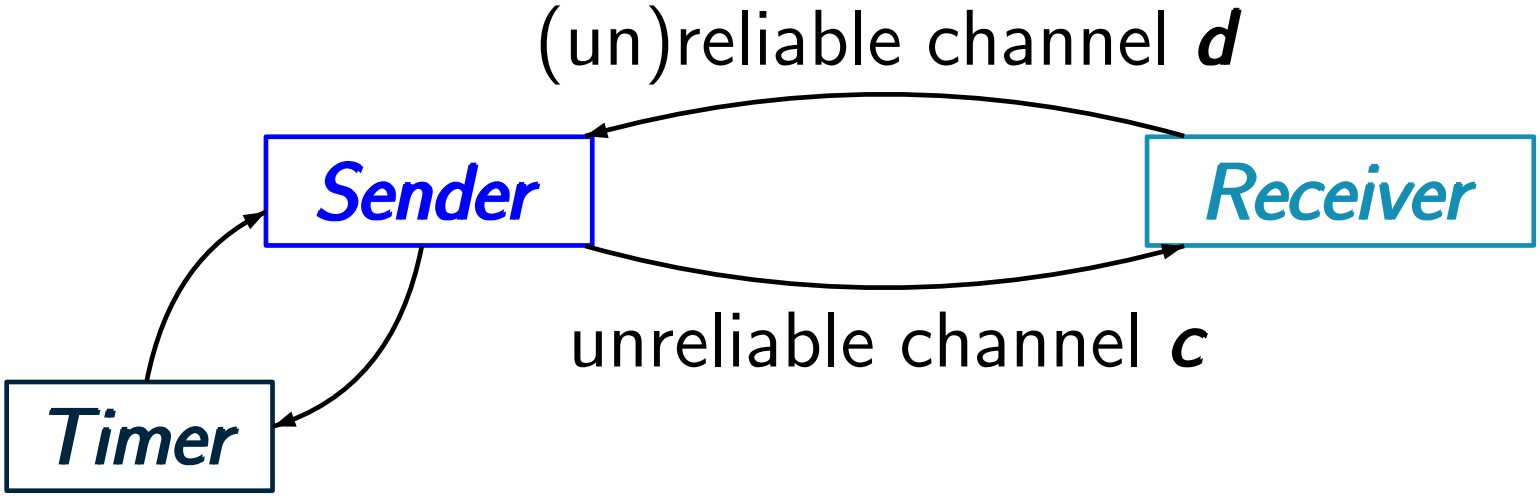


If both channels are unreliable ...



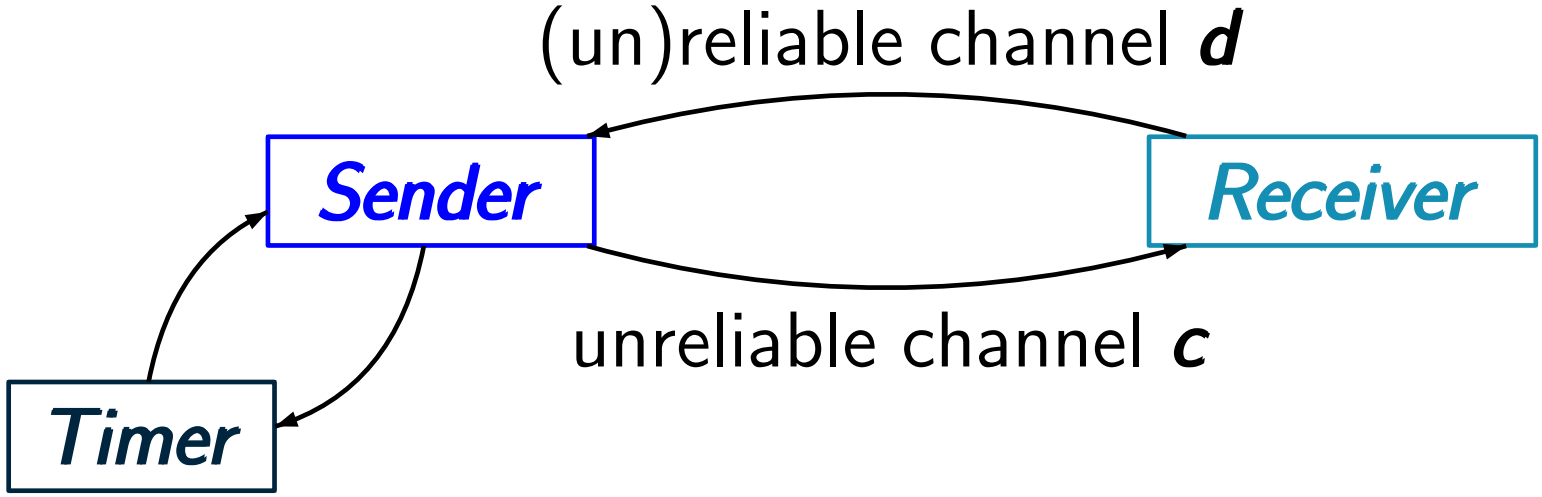

```
LOOP FOREVER
(1)  send message + bit y and activate timer
(2)  AWAIT  timeout or acknowledgement x DO
          IF  timeout THEN  goto (1)
           ELSE  IF  x=y THEN  turn off timer; y:=¬y
                             ELSE  ignore x
                 FI
          FI
     OD
```

Alternating bit protocol (ABP)



channel system: $[\,\textit{Sender} \mid \textit{Timer} \mid \textit{Receiver}\,]$

Alternating bit protocol (ABP)

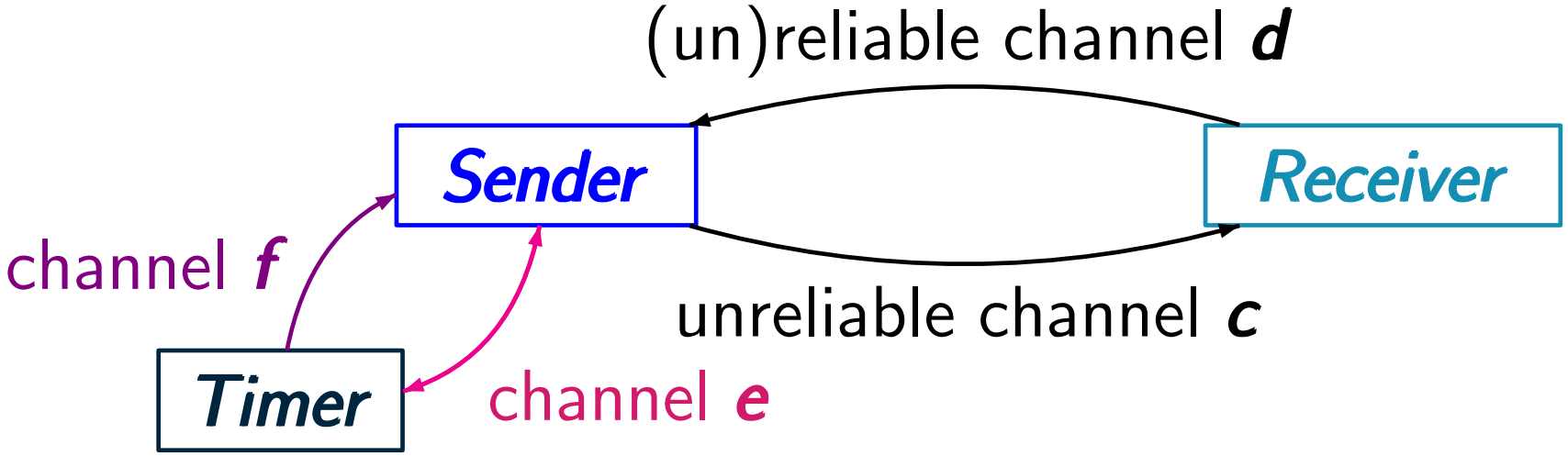


channel system: $[\ \textit{Sender} \mid \textit{Timer} \mid \textit{Receiver}\ ]$

synchronous message passing between
Timer and *Sender*

asynchronous message passing between
Receiver and *Sender*

Alternating bit protocol (ABP)

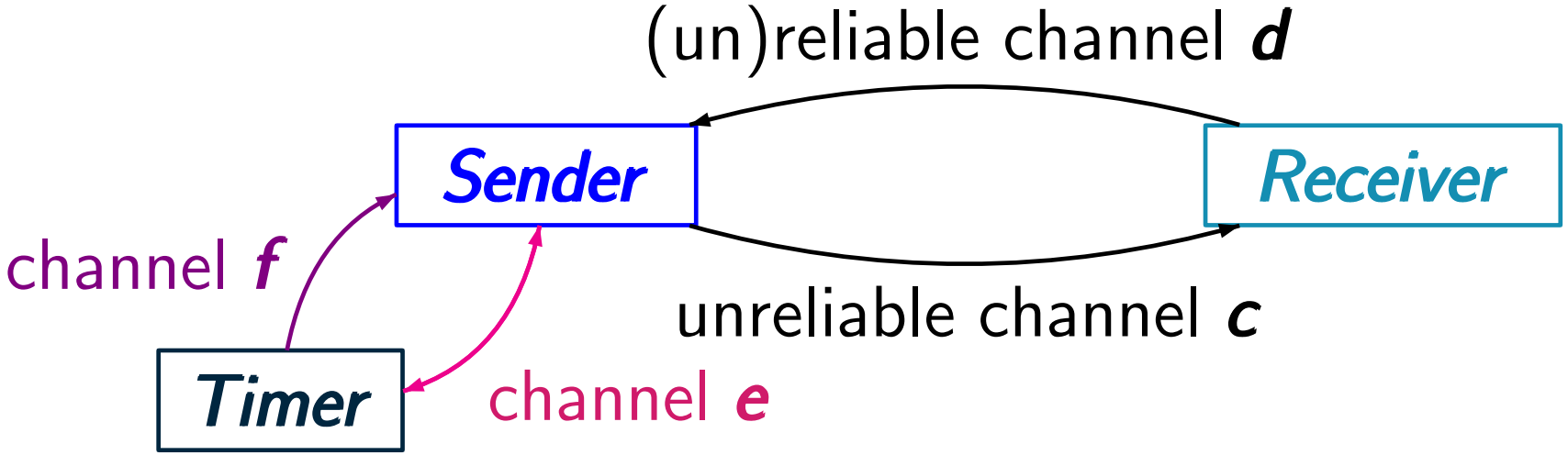


channel system: $[\ \textit{Sender} \mid \textit{Timer} \mid \textit{Receiver}\ ]$

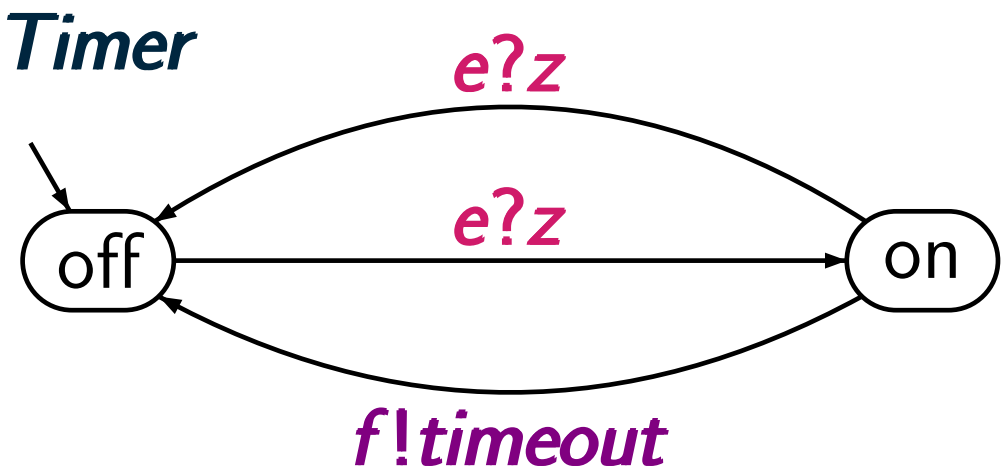
synchronous message passing between
Timer and *Sender* ← channels ***e*** and ***f***

asynchronous message passing between
Receiver and *Sender* ← channels ***c***, ***d***

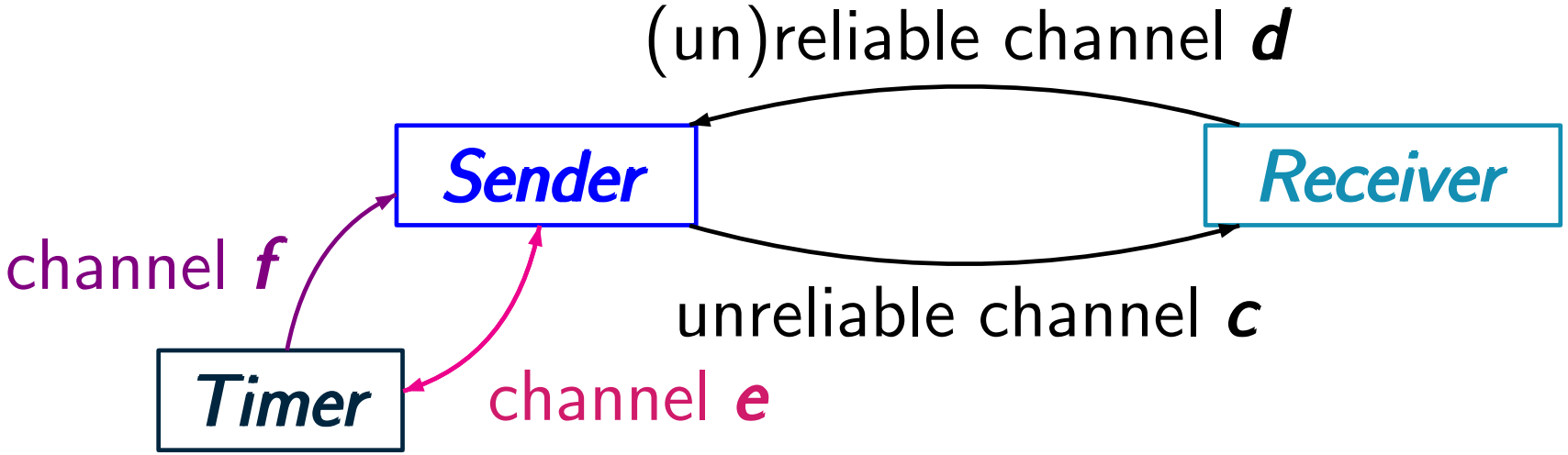
Alternating bit protocol (ABP)



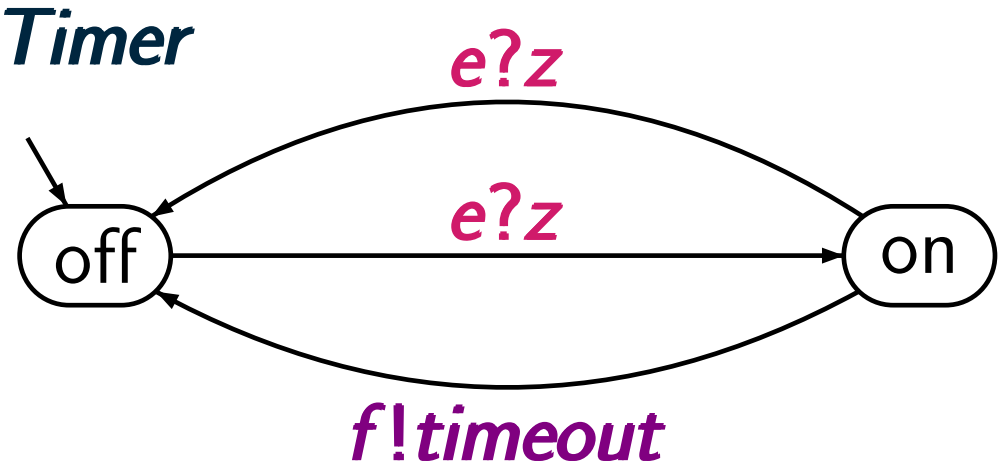
channel system: [*Sender* | *Timer* | *Receiver*]



Alternating bit protocol (ABP)



channel system: $[\ Sender \mid Timer \mid Receiver\ ]$



actions of *Sender*:

$\vdots$

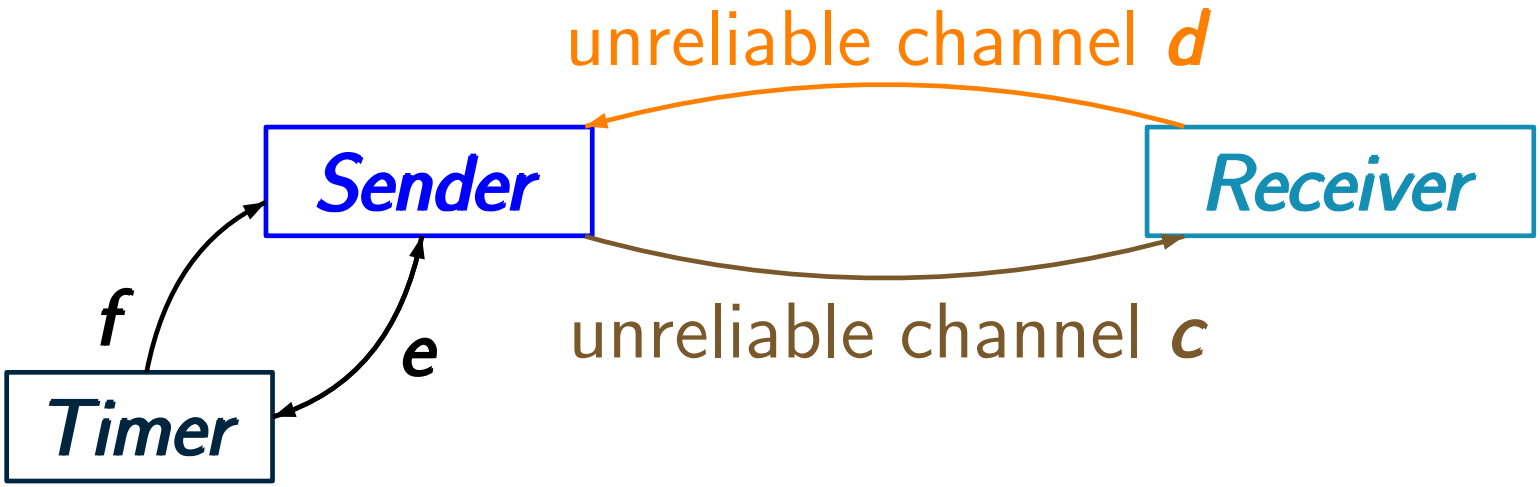
$e!timer\_on$

$e!timer\_off$

$f?z'$

$\vdots$

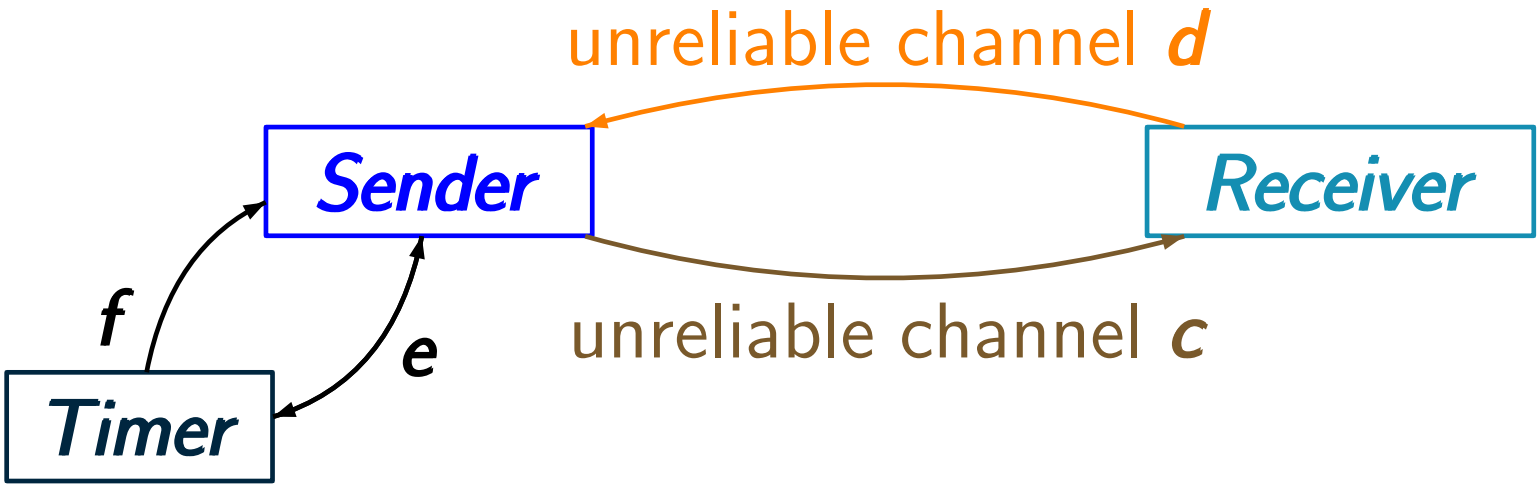
Program graph for the sender



specify the sender by a program graph using

- asynchronous channels ***c*** and ***d***
- synchronous channels ***e*** and ***f***

Program graph for the sender

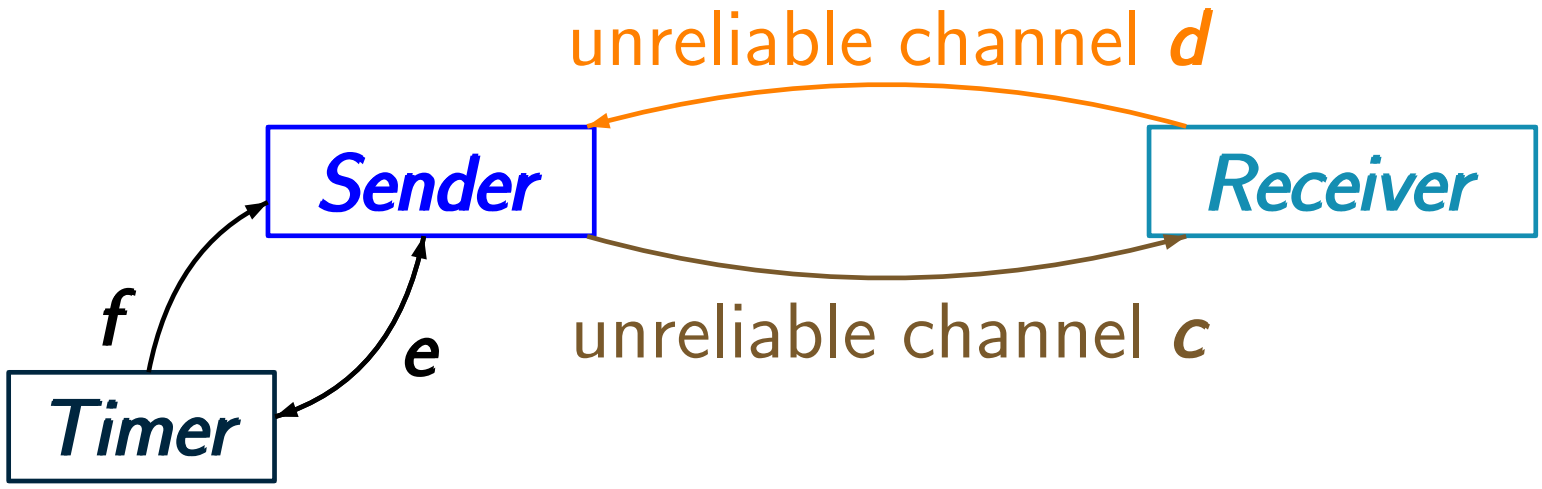


specify the sender by a program graph using

- asynchronous channels *c* and *d*
- synchronous channels *e* and *f*

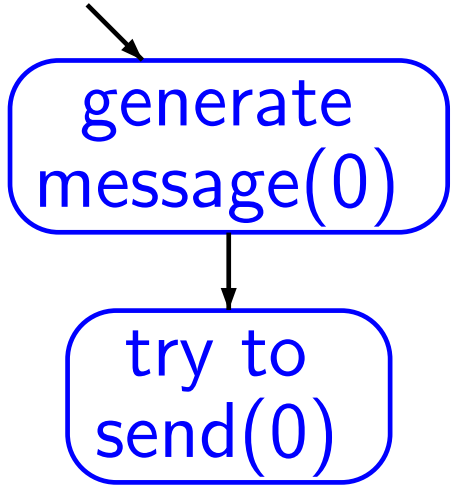
simply write *!timeout*
?timer_on
?timer_off

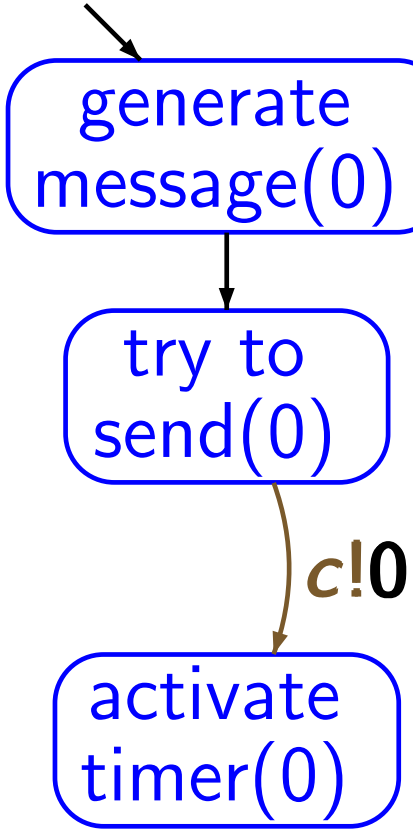
Program graph for the sender

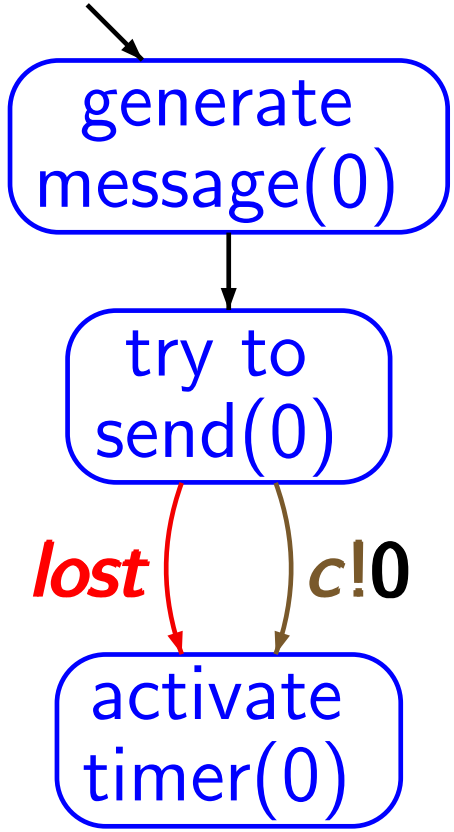


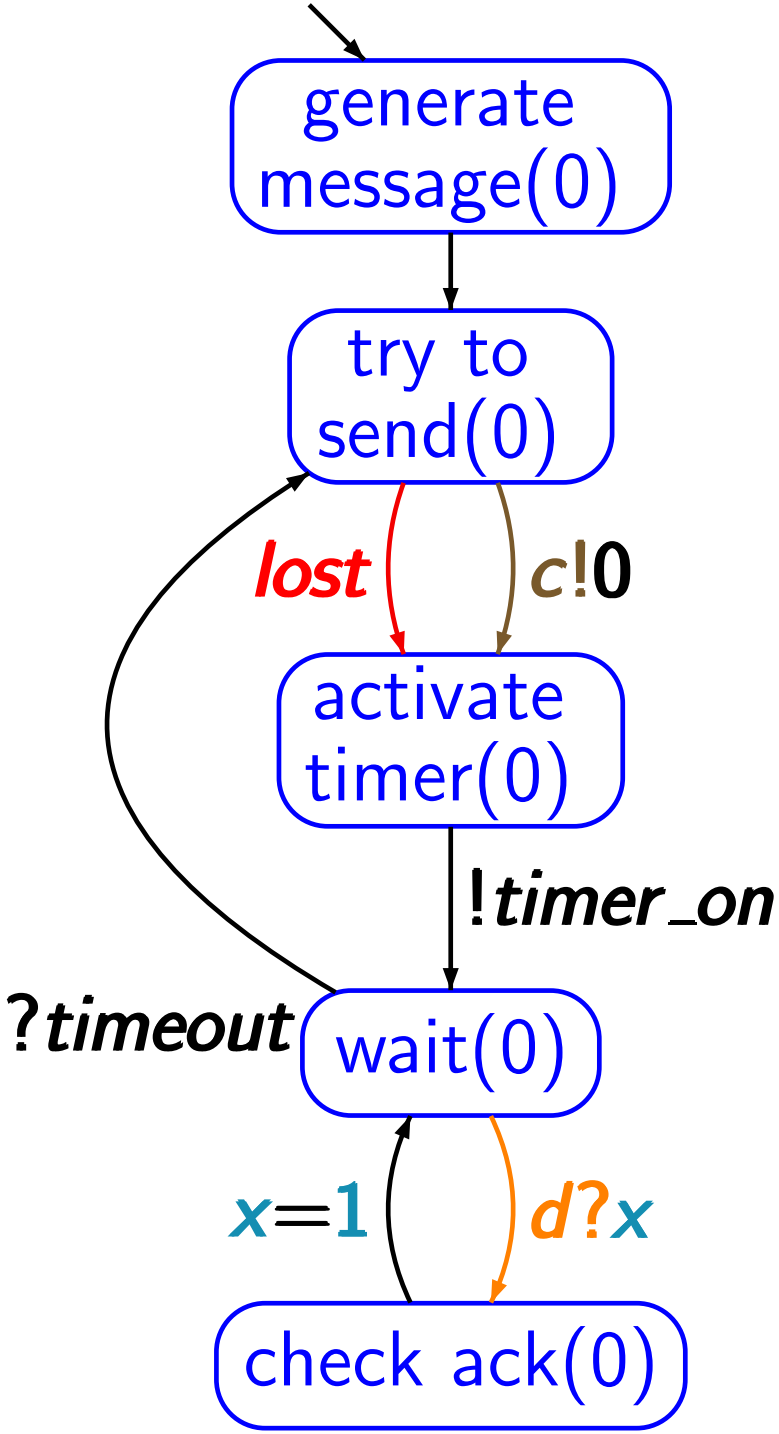
specify the sender by a program graph using

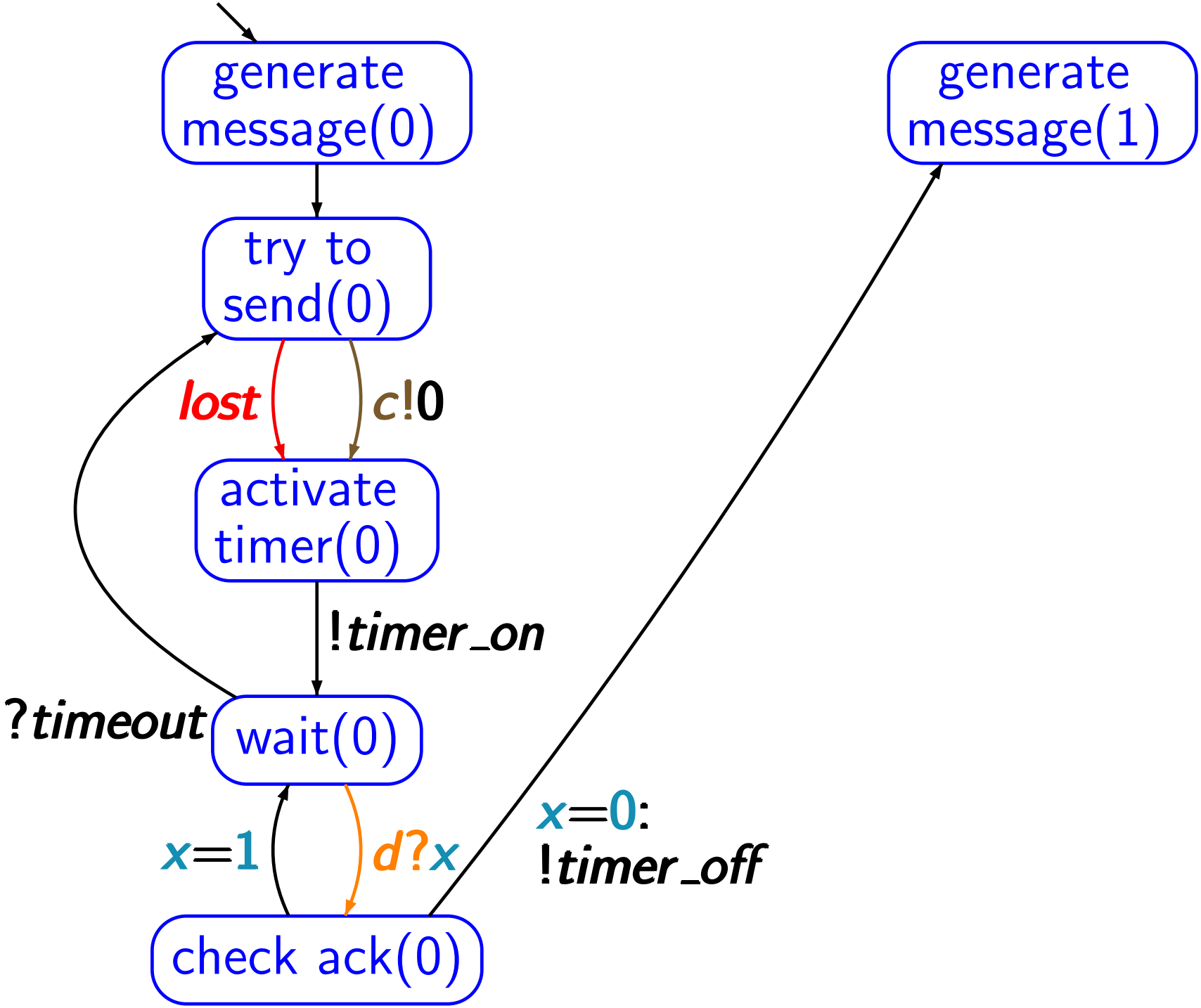
- asynchronous channels ***c*** and ***d***
- synchronous channels ***e*** and ***f***
- Boolean variable ***x*** for the acknowledgement bit sent by the receiver

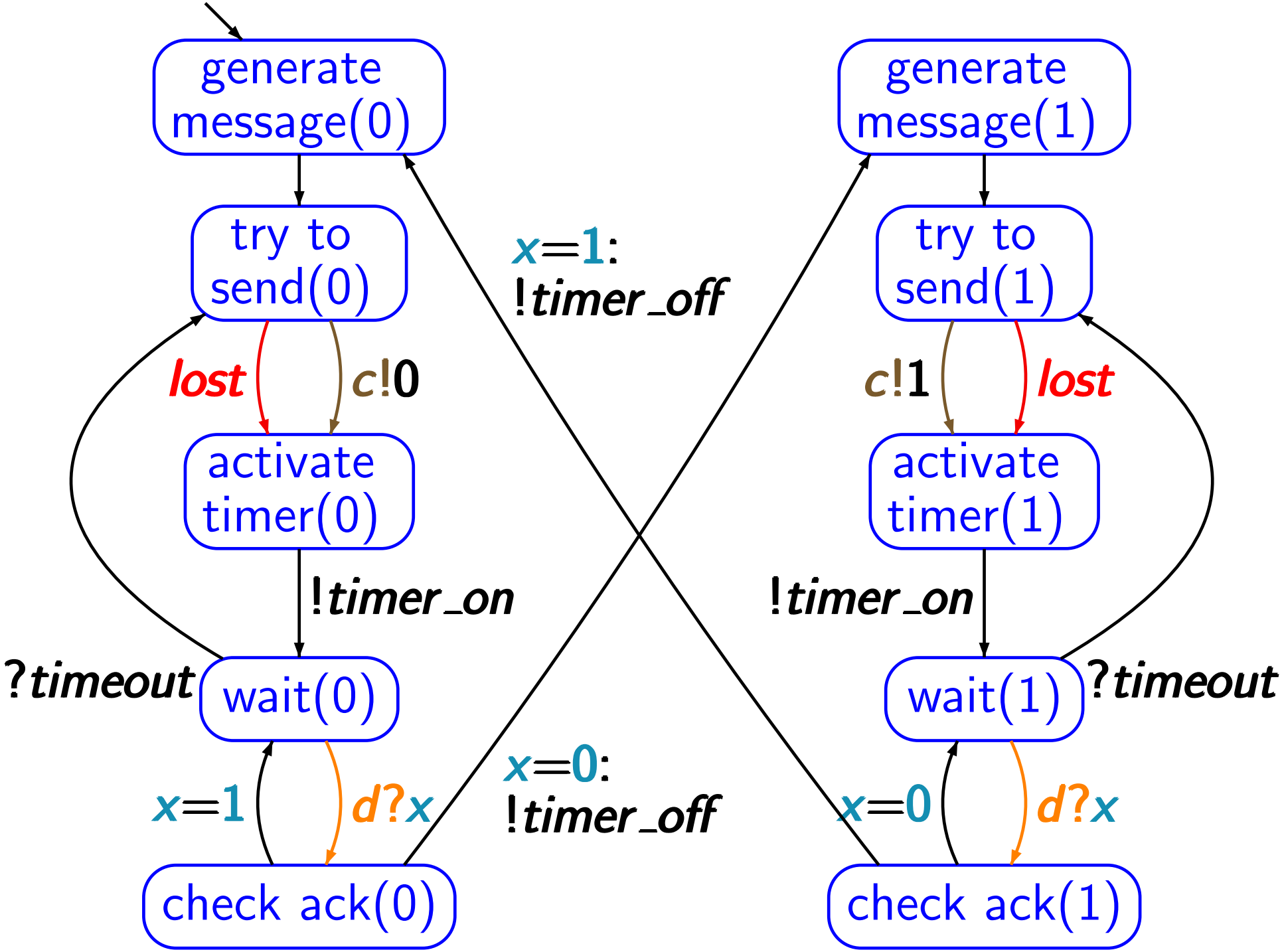

generate
message(0)
try to
send(0)


generate
message(0)
try to
send(0)
c!0
activate
timer(0)

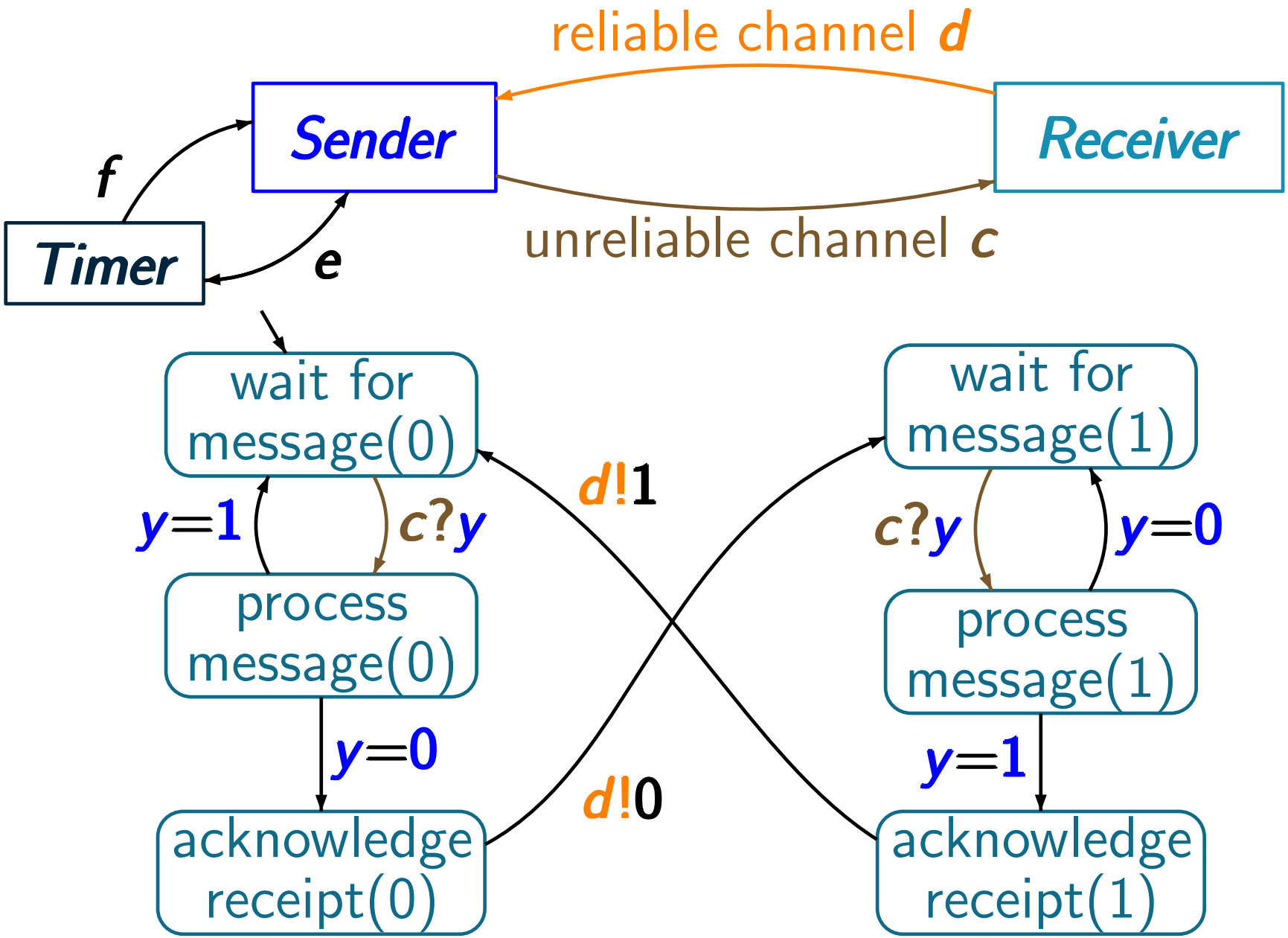

generate
message(0)
try to
send(0)
lost
c!0
activate
timer(0)

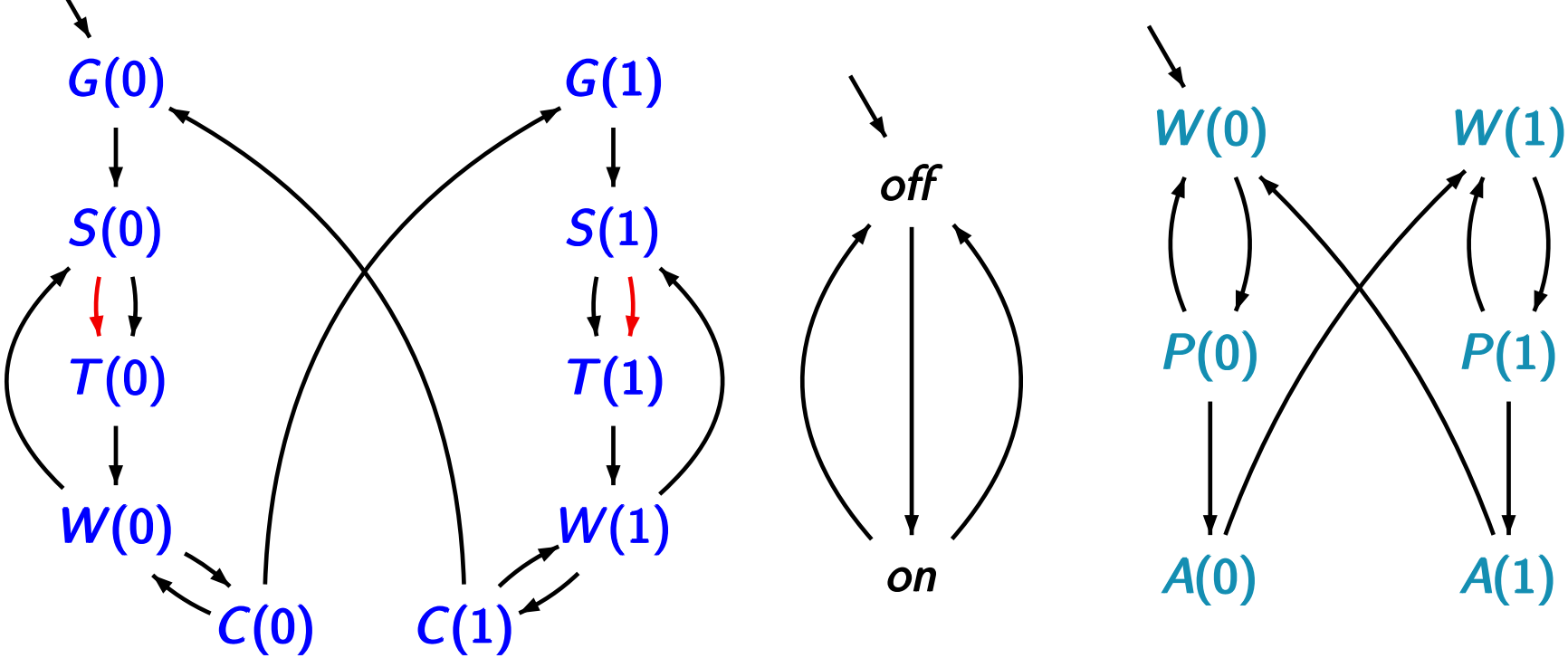

generate message(0)
try to send(0)
lost
c!0
activate timer(0)
!timer_on
?timeout
wait(0)
x=1
d?x
check ack(0)

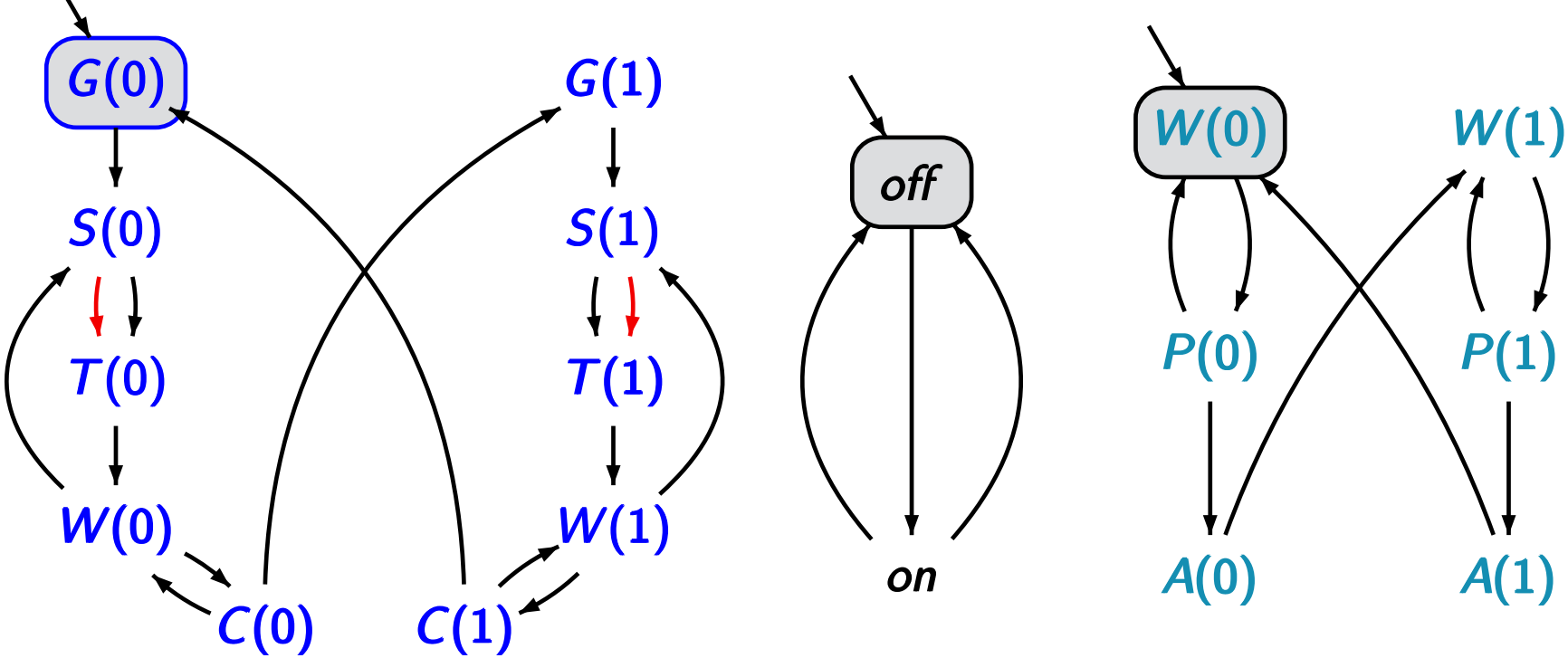

generate message(0)
try to send(0)
lost
c!0
activate timer(0)
!timer_on
?timeout
wait(0)
x=1
d?x
check ack(0)
x=0:
!timer_off
generate message(1)


generate message(0)
try to send(0)
lost
c!0
activate timer(0)
!timer_on
?timeout
wait(0)
x=1
d?x
check ack(0)
x=1:
!timer_off
x=0:
!timer_off
generate message(1)
try to send(1)
c!1
lost
activate timer(1)
!timer_on
wait(1)
?timeout
x=0
d?x
check ack(1)

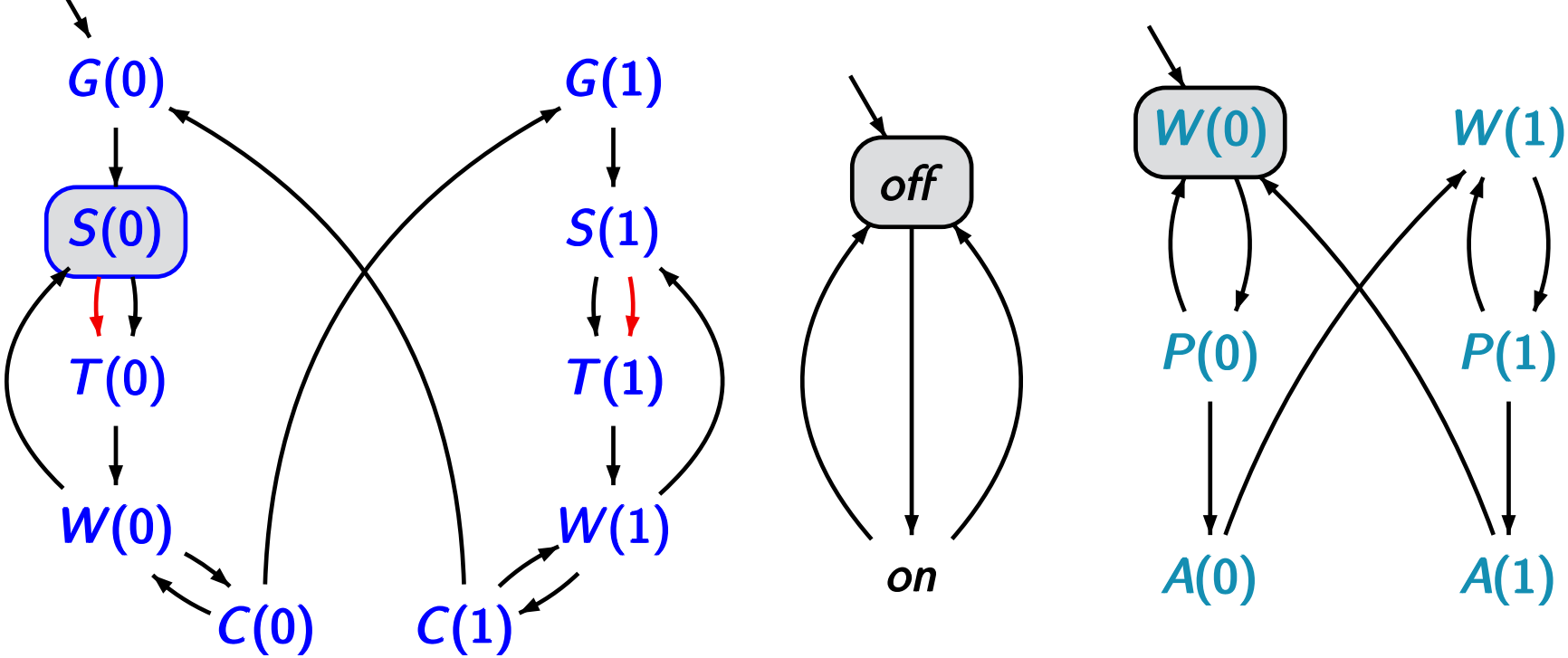
Program graph for the receiver




G(0)
S(0)
T(0)
W(0)
C(0)
G(1)
S(1)
T(1)
W(1)
C(1)
off
on
W(0)
W(1)
P(0)
P(1)
A(0)
A(1)

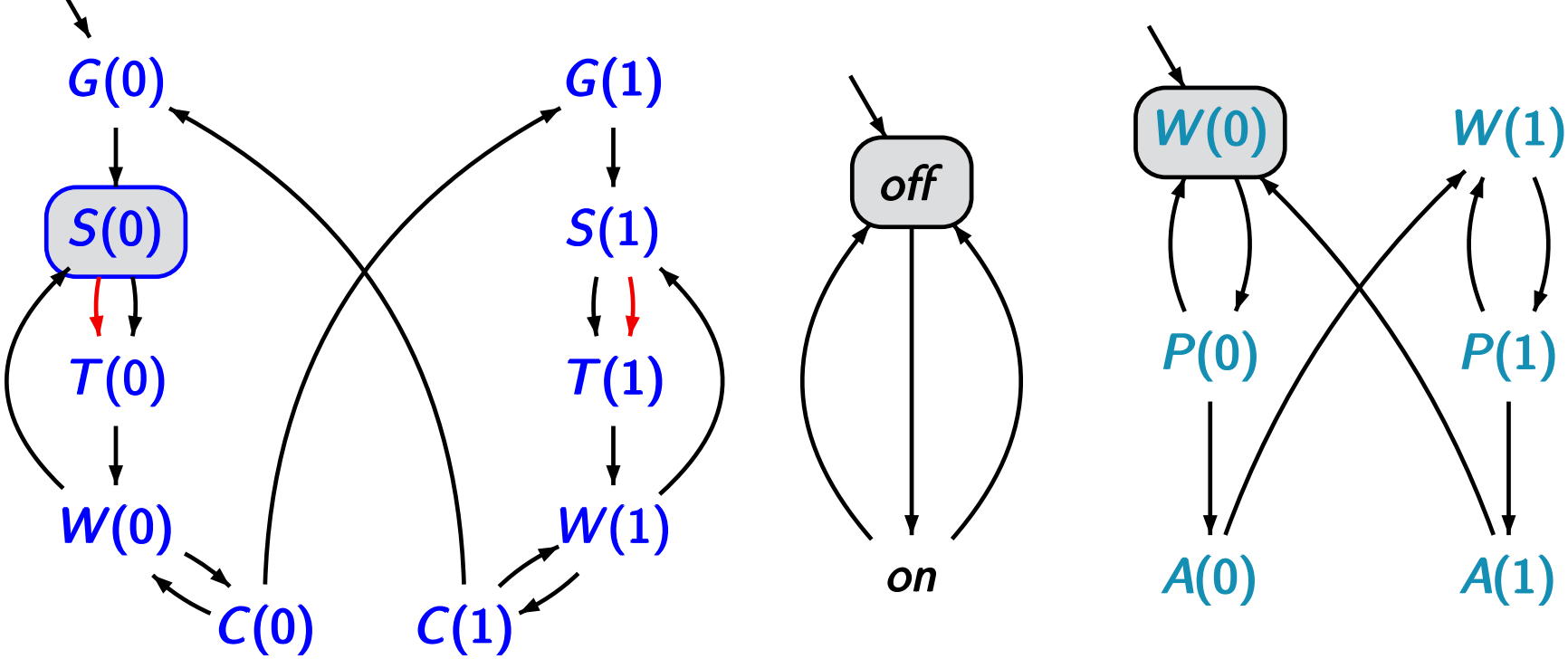


Generate(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$



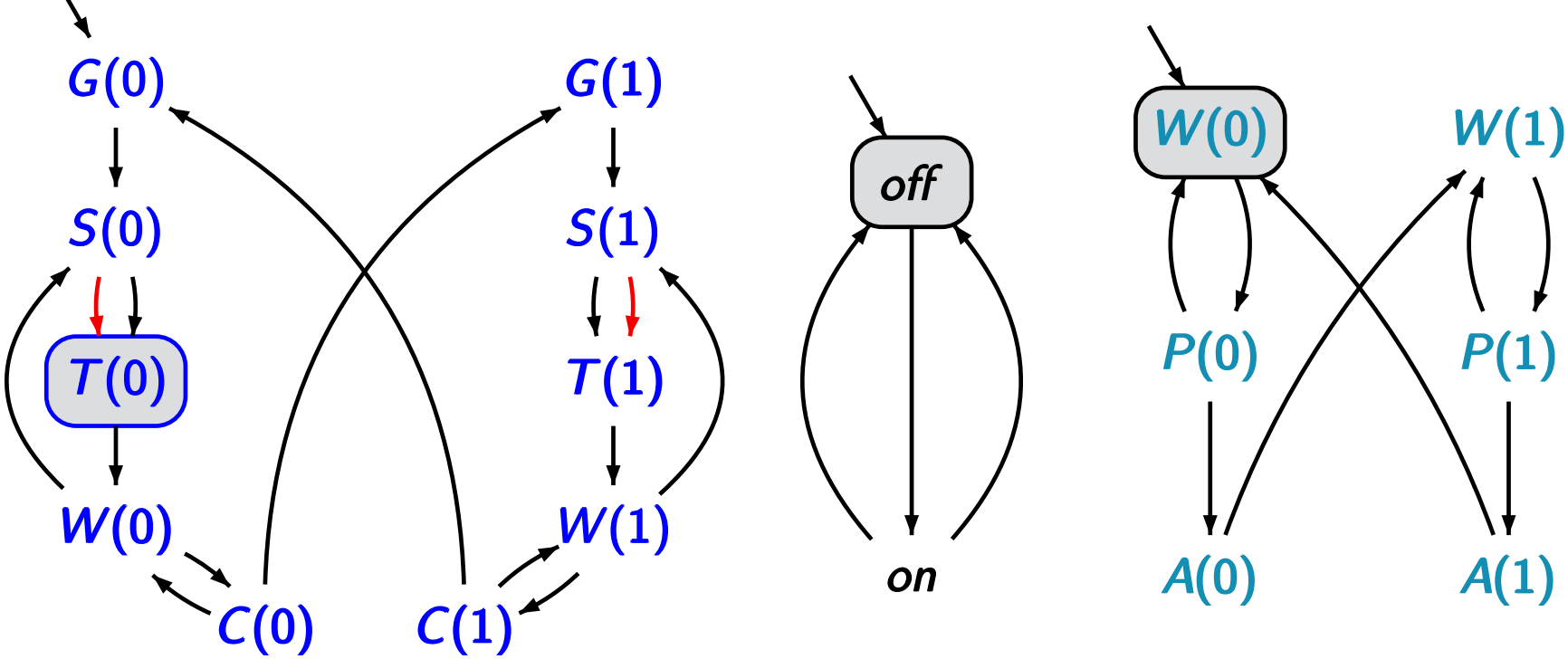
Generate(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$

Send(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$

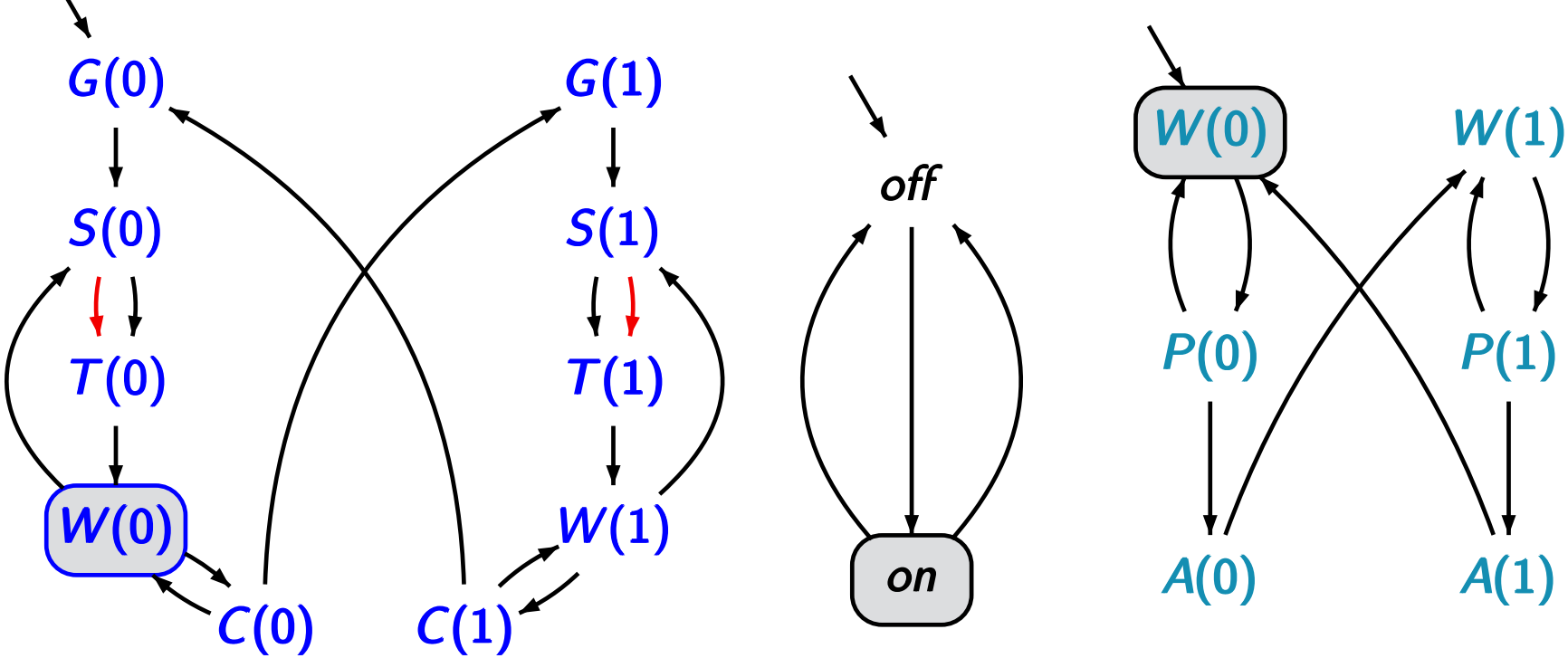


Generate(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$

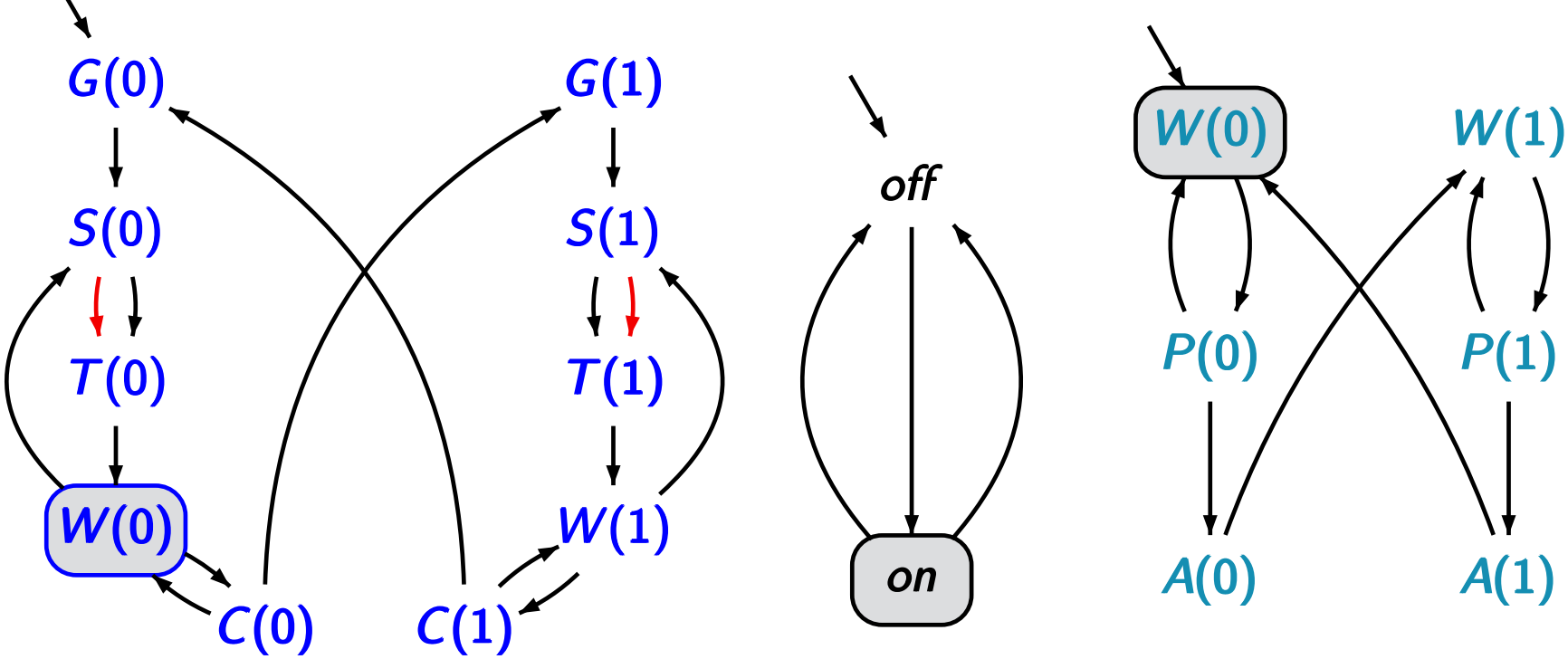
Send(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$ message lost



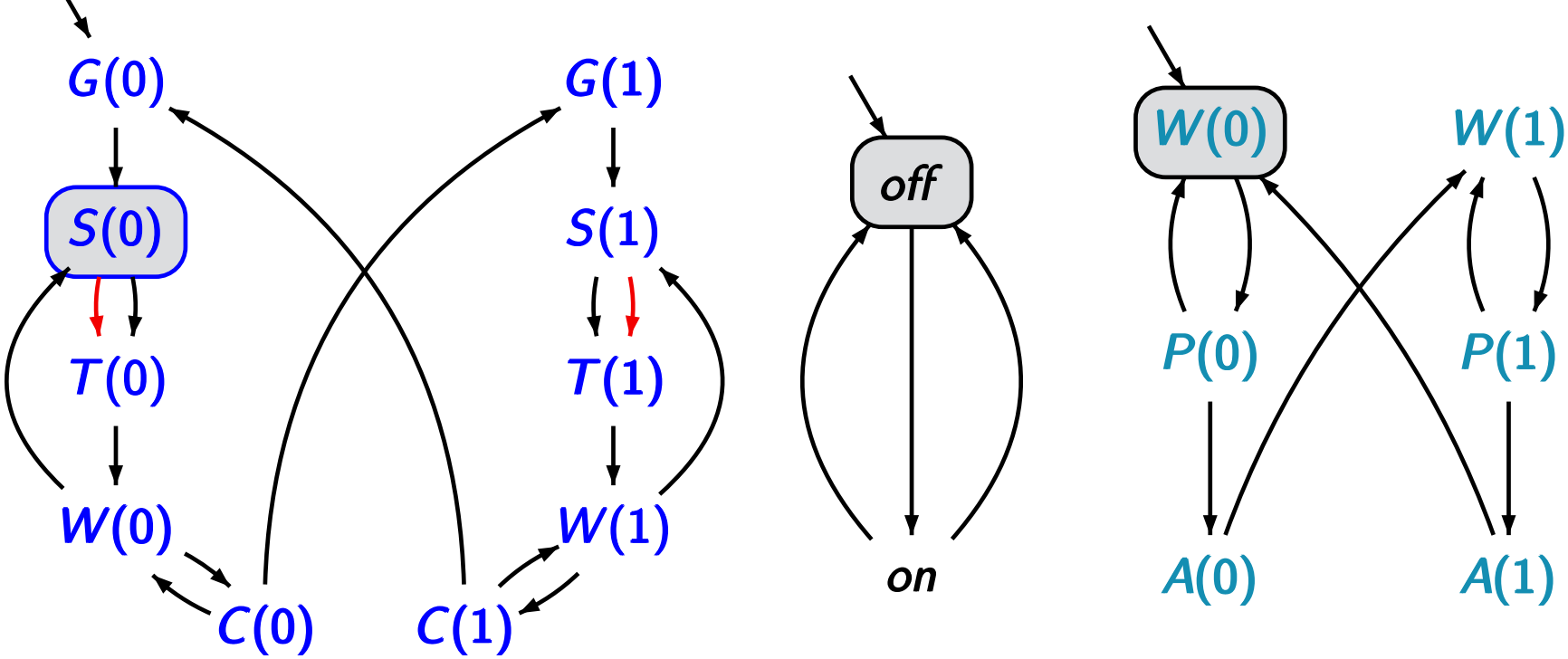
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message lost
Timer_on(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	



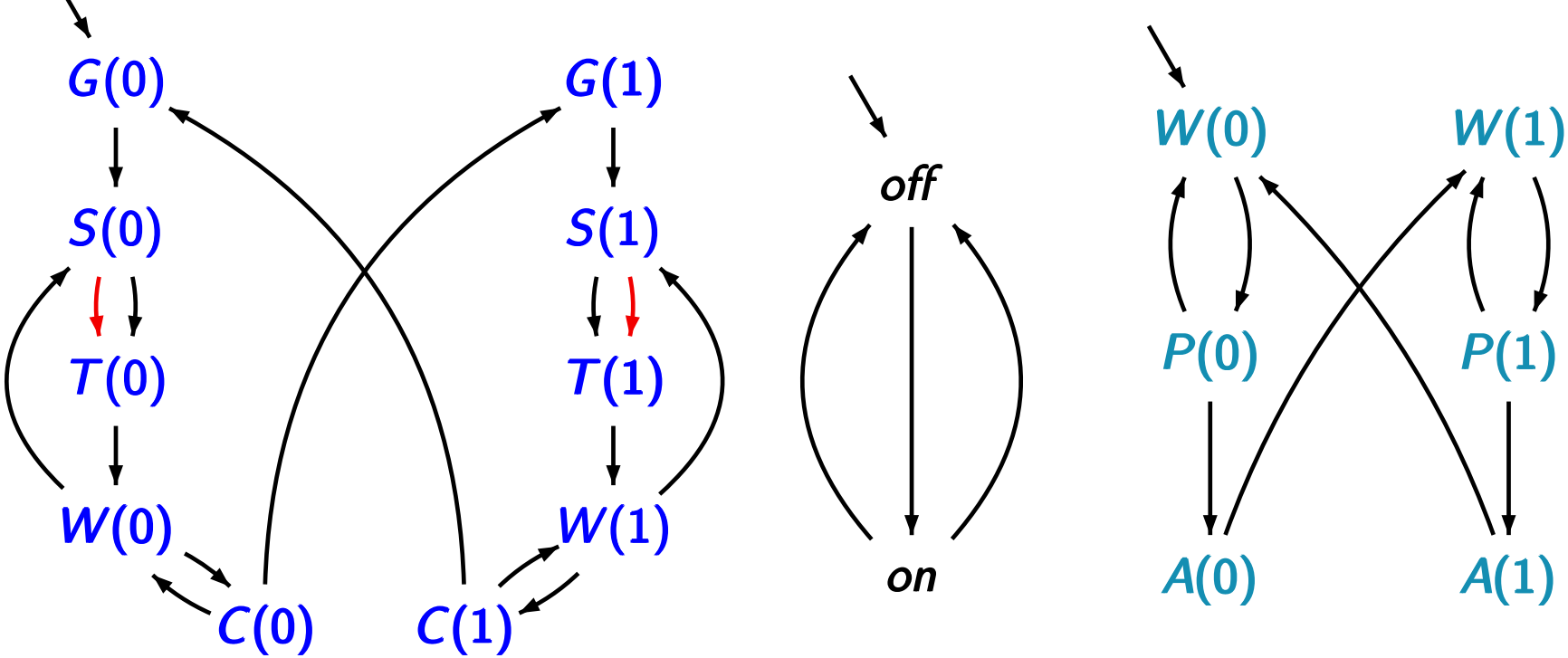
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message lost
Timer_on(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	

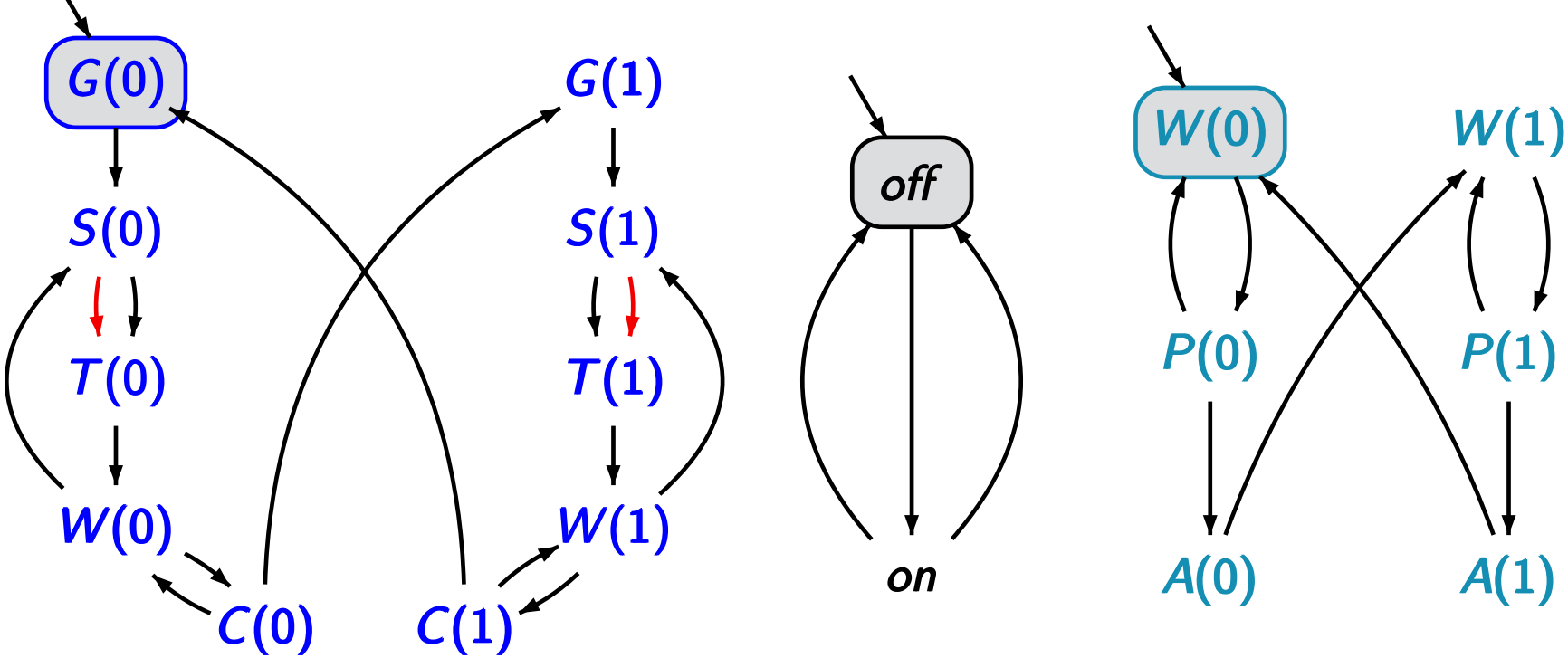


Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message lost
Timer_on(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	timeout

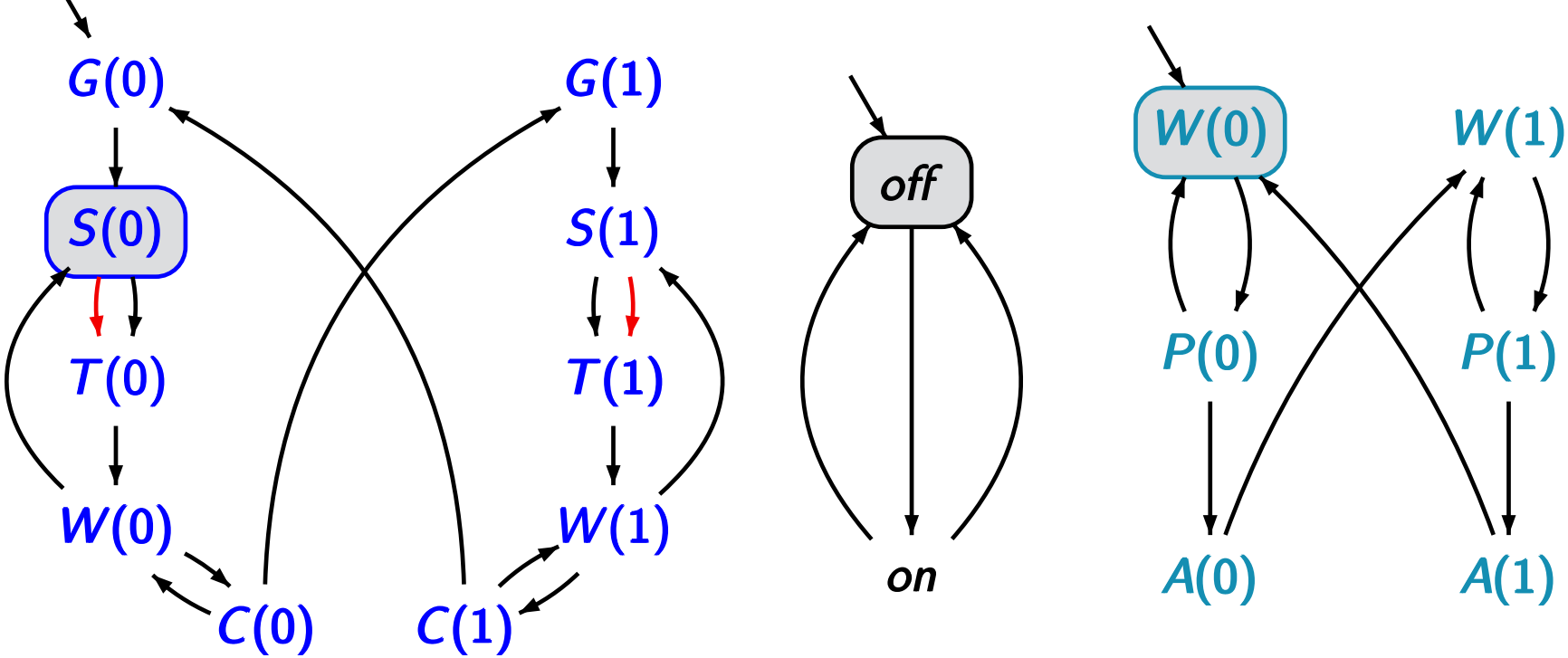


Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message lost
Timer_on(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
	⋮				try again

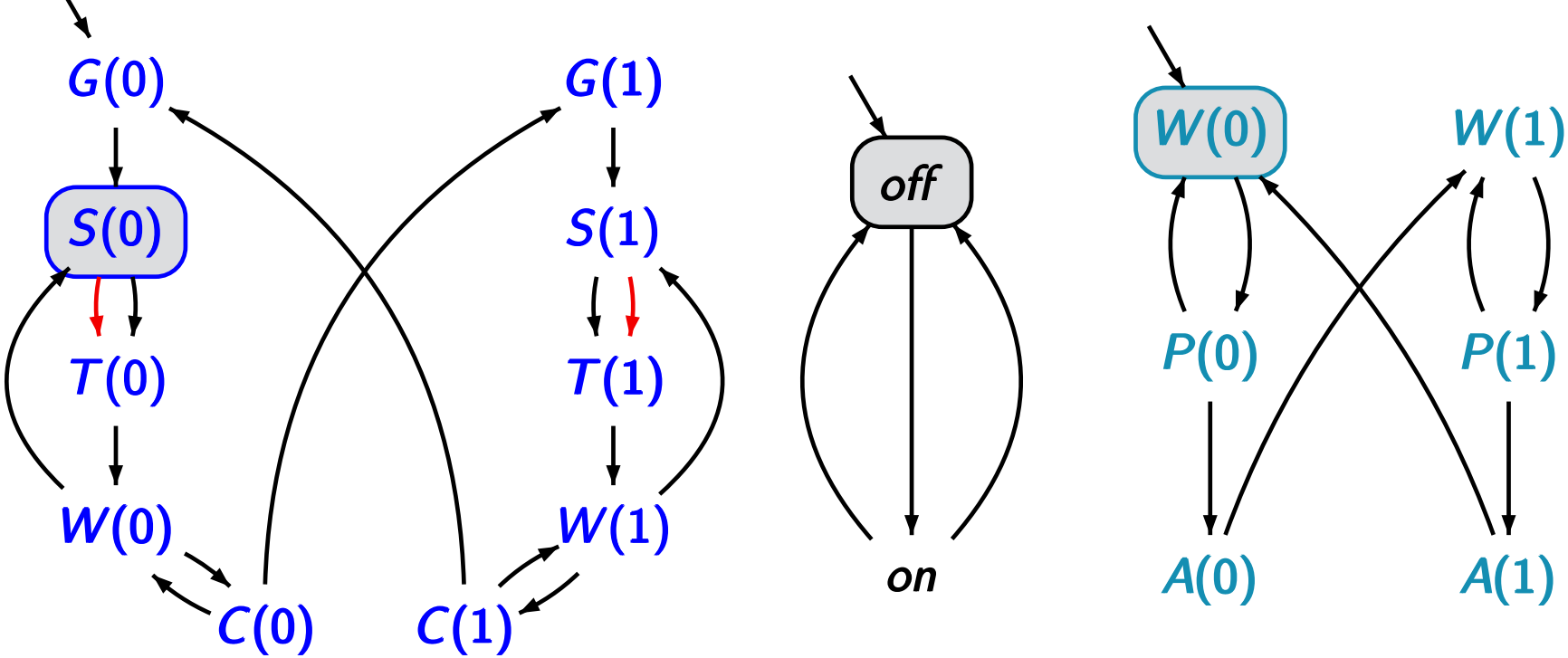

G(0)
S(0)
T(0)
W(0)
C(0)
G(1)
S(1)
T(1)
W(1)
C(1)
off
on
W(0)
W(1)
P(0)
P(1)
A(0)
A(1)



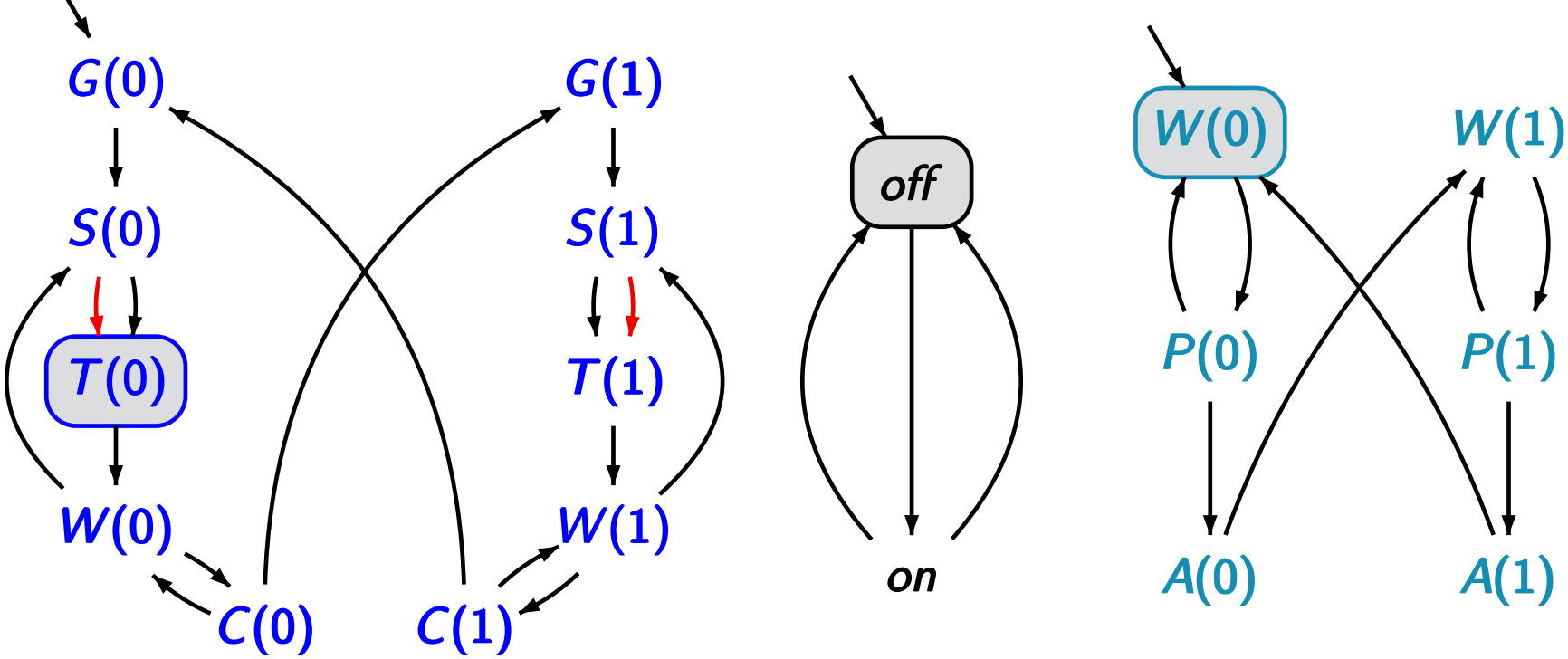
Generate(0) off Wait(0) $c{=}\varepsilon$ $d{=}\varepsilon$



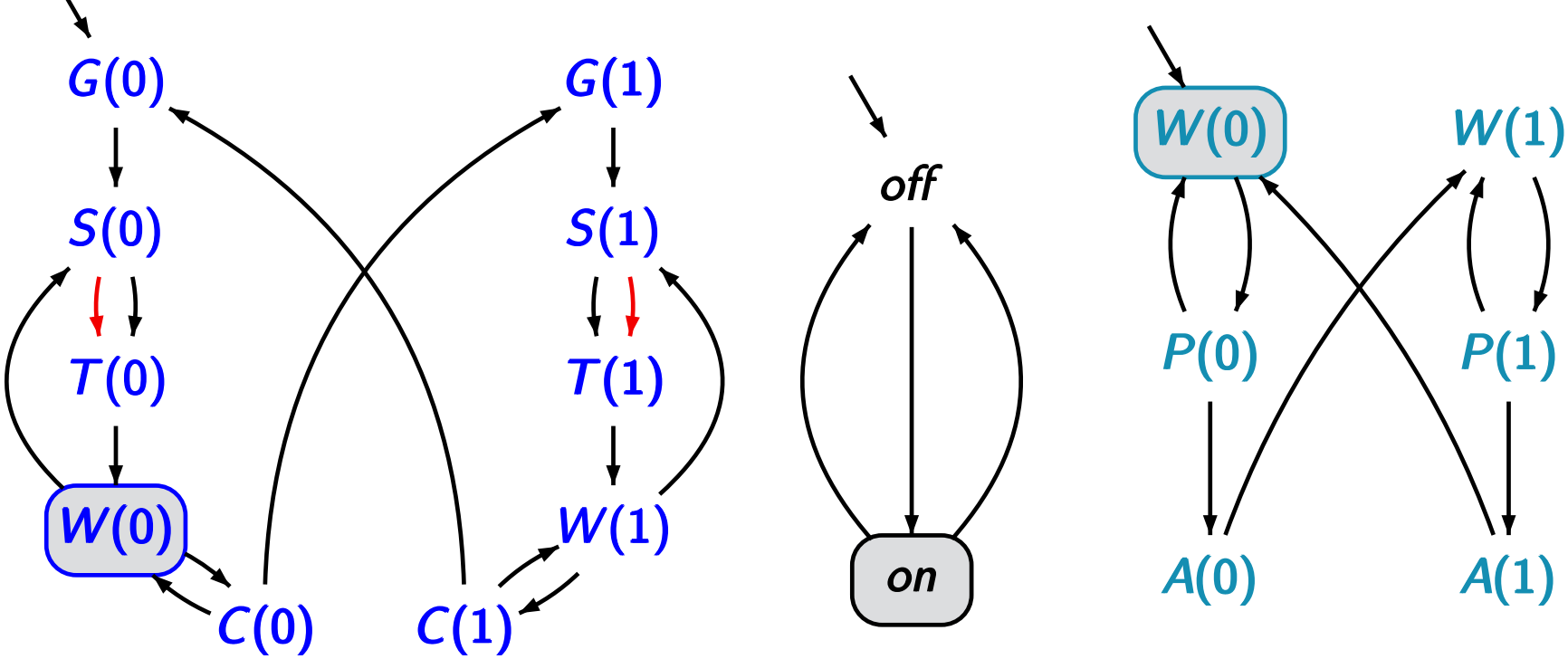
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$



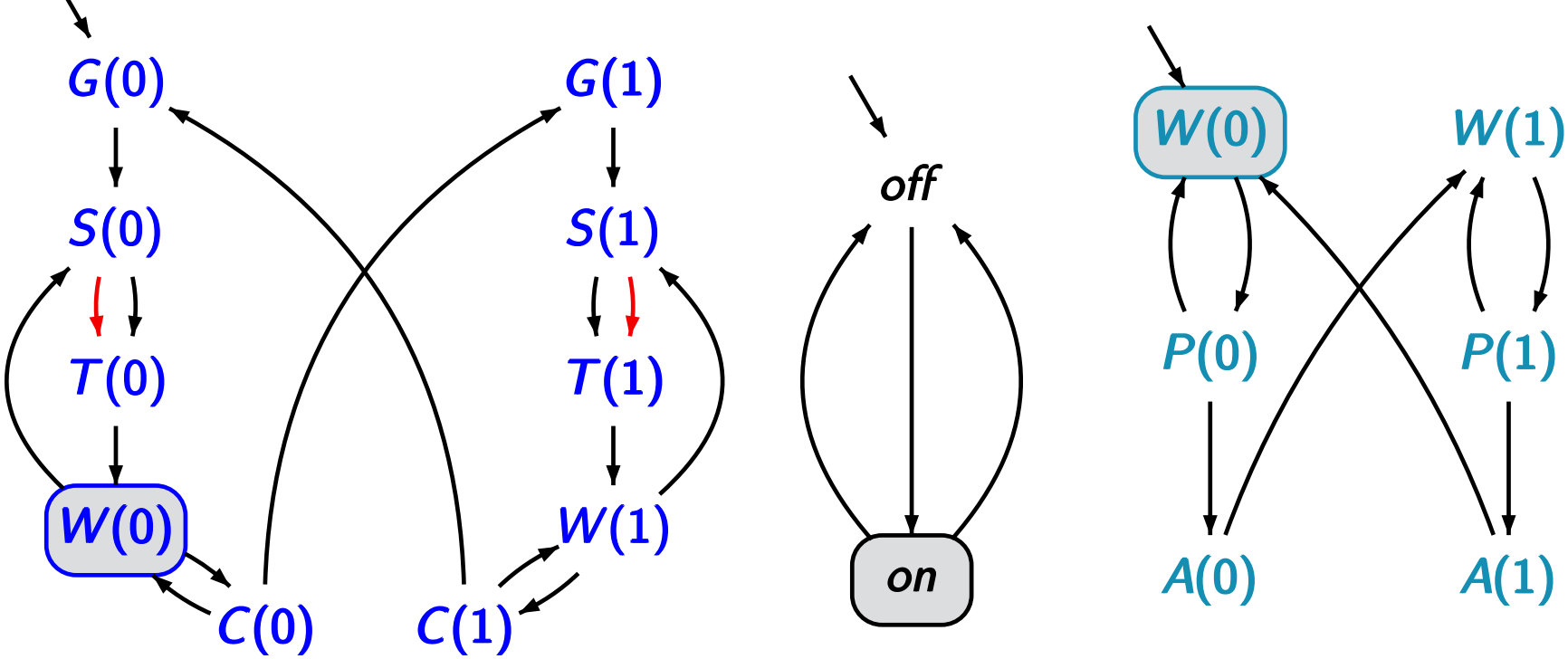
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent



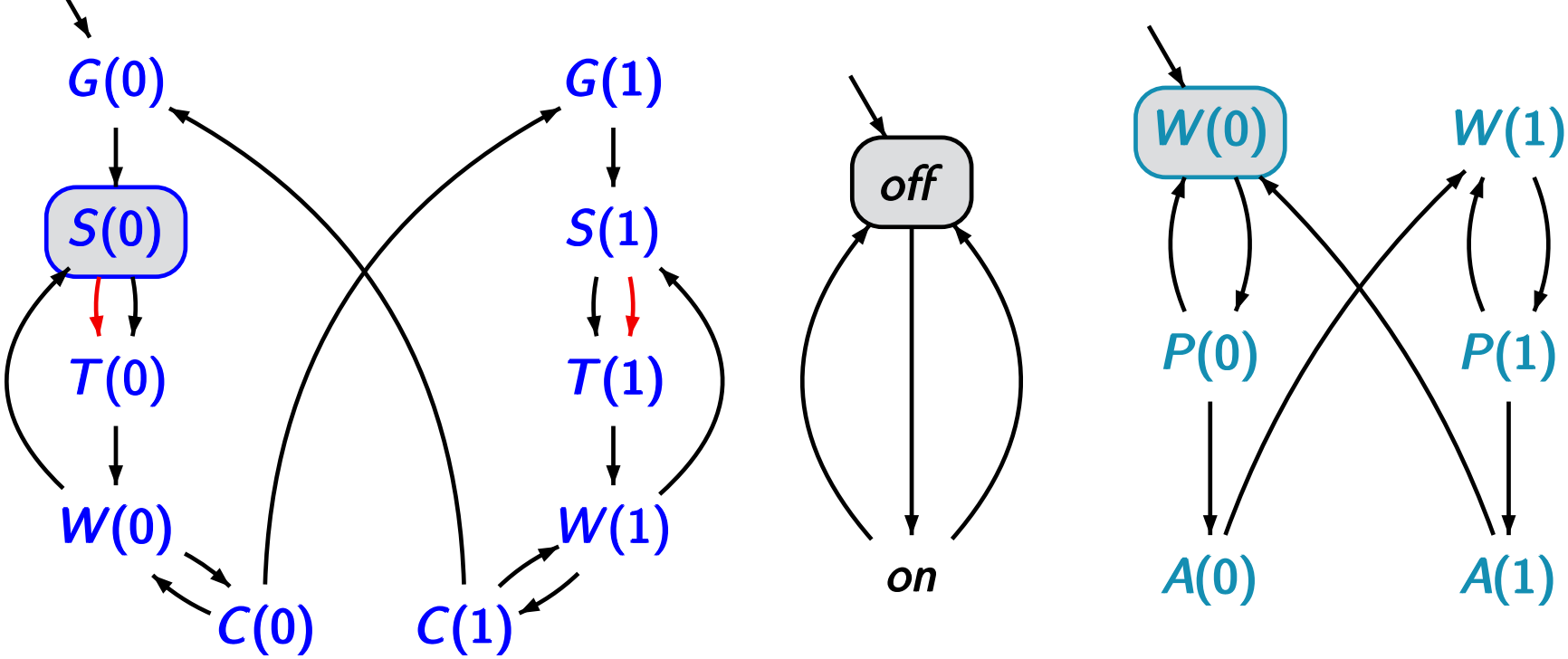
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	



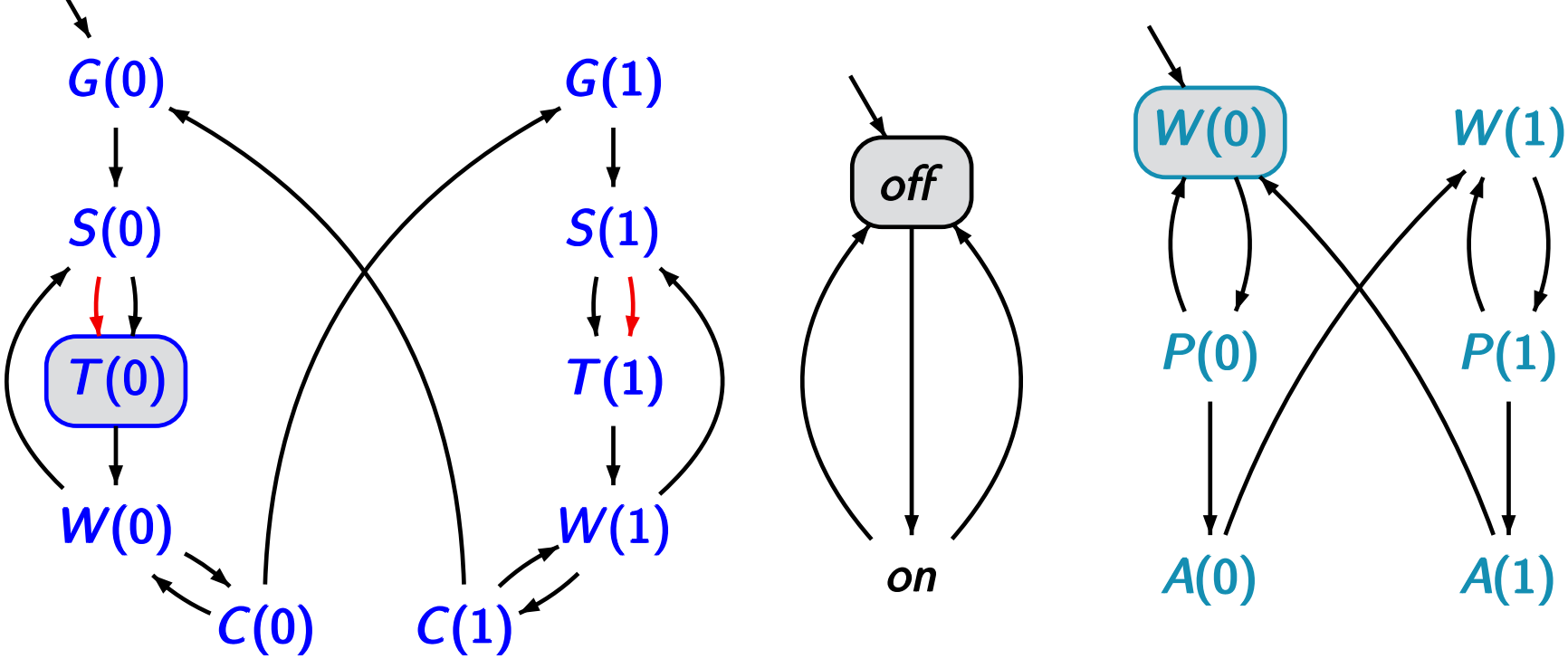
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	



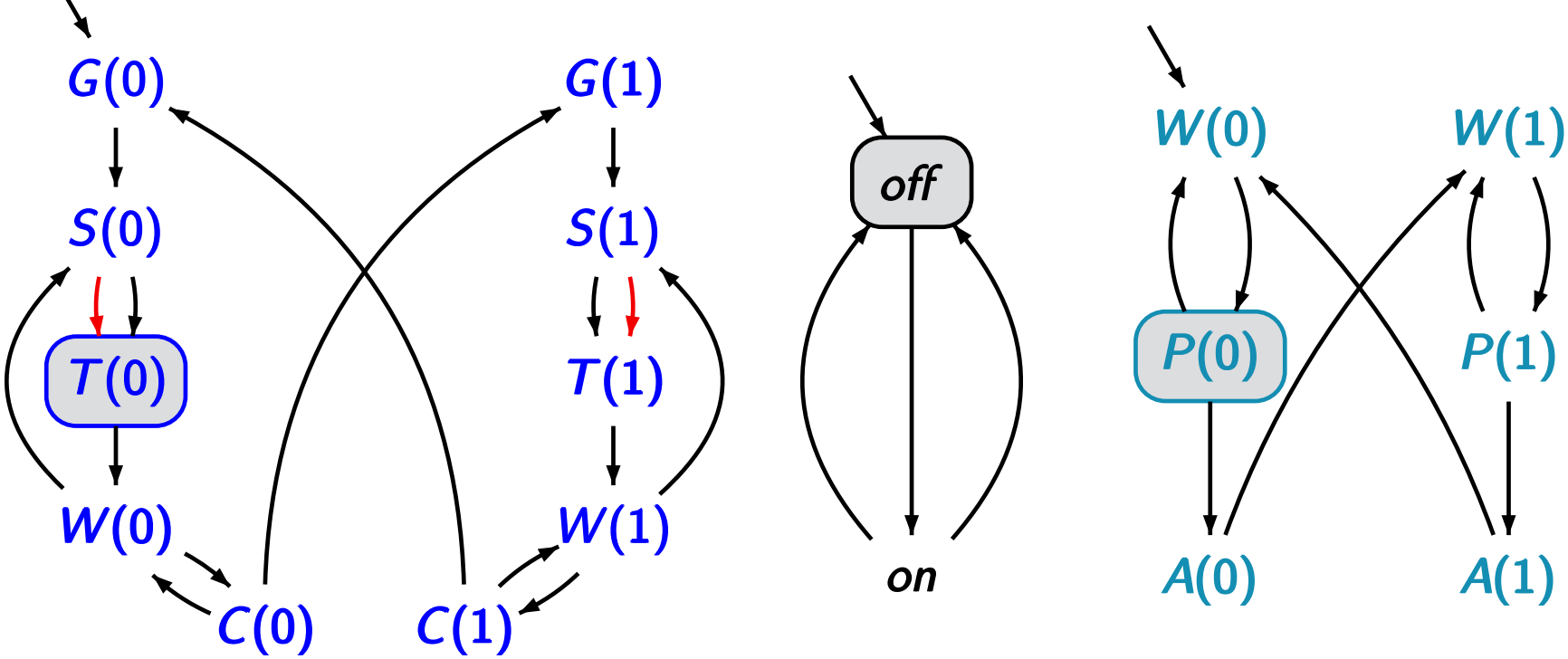
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout



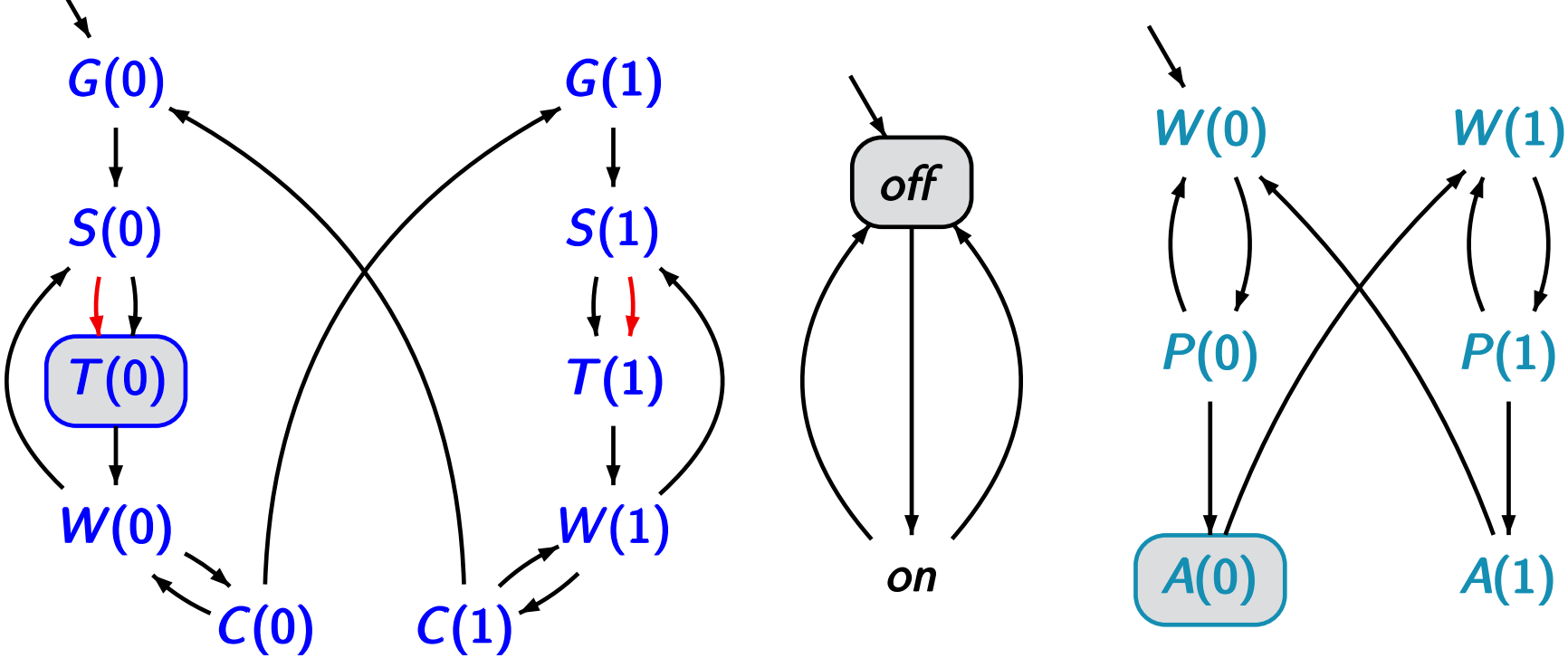
Generate(0)	off	Wait(0)	$c=\varepsilon$	$d=\varepsilon$	
Send(0)	off	Wait(0)	$c=\varepsilon$	$d=\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c=0$	$d=\varepsilon$	
Wait(0)	on	Wait(0)	$c=0$	$d=\varepsilon$	timeout
Send(0)	off	Wait(0)	$c=0$	$d=\varepsilon$	



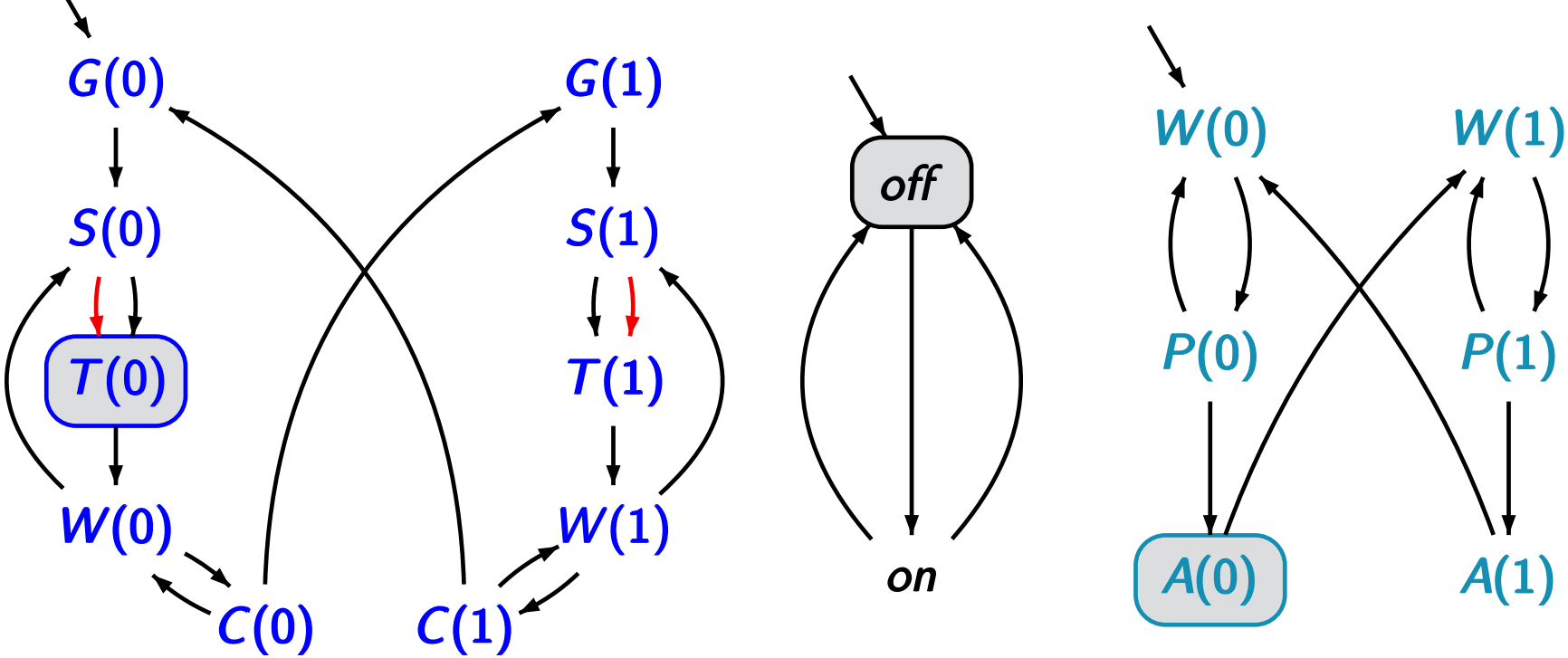
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	0 sent again
Timer_on(0)	off	Wait(0)	$c{=}00$	$d{=}\varepsilon$	



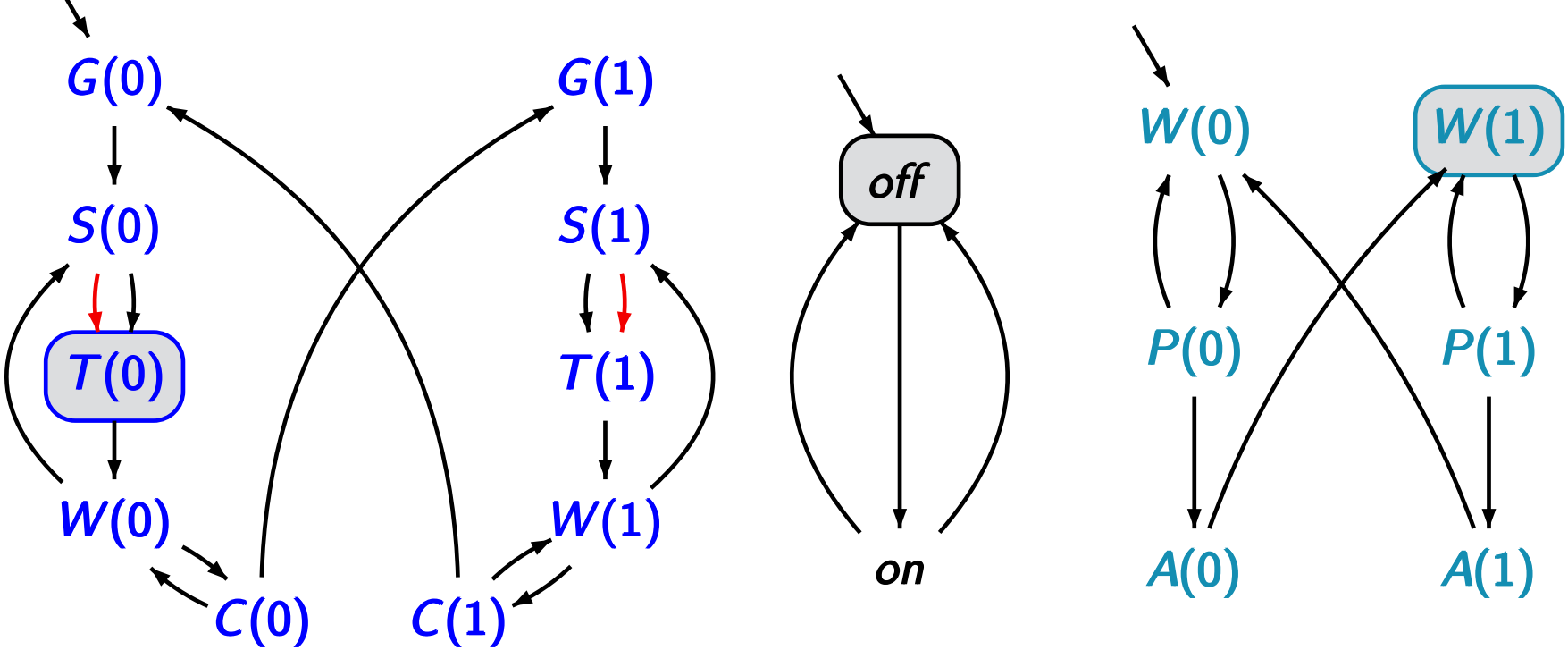
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	0 sent again
Timer_on(0)	off	Wait(0)	$c{=}00$	$d{=}\varepsilon$	
Timer_on(0)	off	Proc(0)	$c{=}0$	$d{=}\varepsilon$	message received



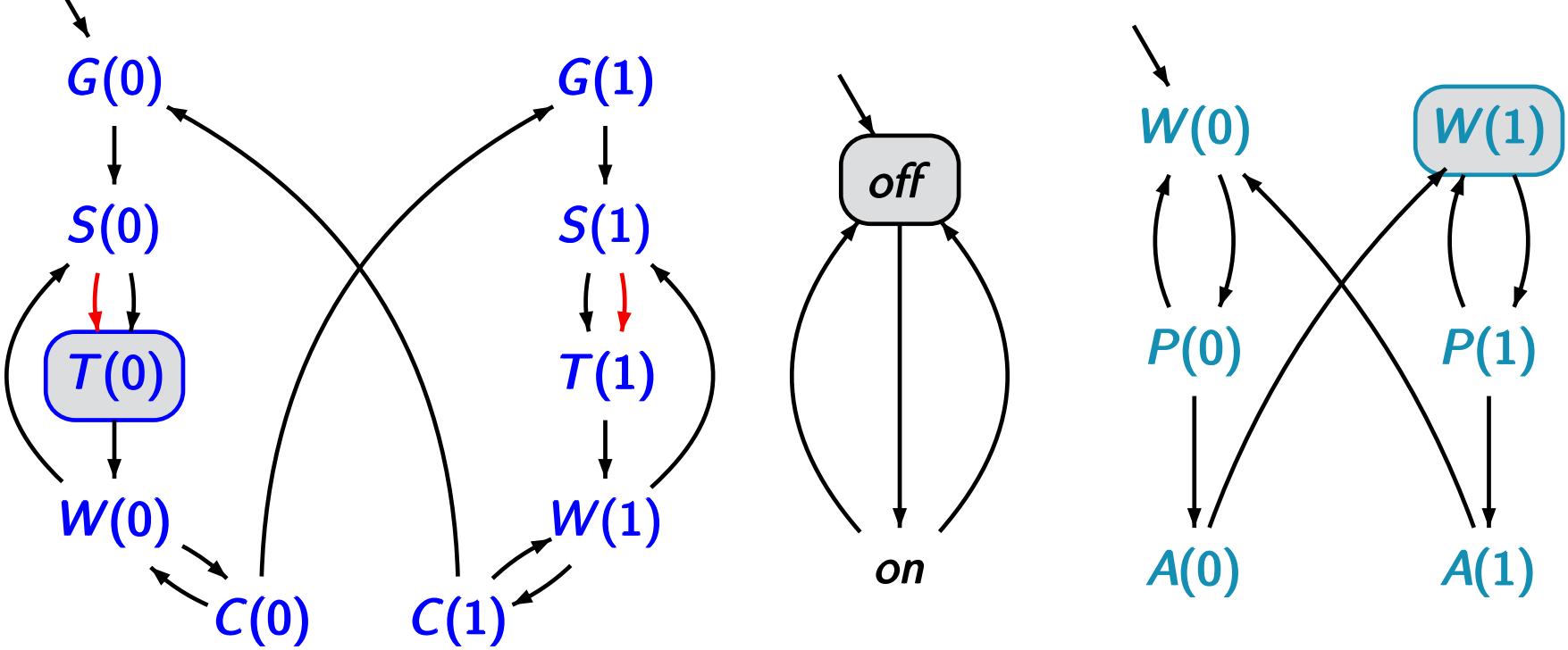
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	0 sent again
Timer_on(0)	off	Wait(0)	$c{=}00$	$d{=}\varepsilon$	
Timer_on(0)	off	Proc(0)	$c{=}0$	$d{=}\varepsilon$	message received
Timer_on(0)	off	Ack(0)	$c{=}0$	$d{=}\varepsilon$	



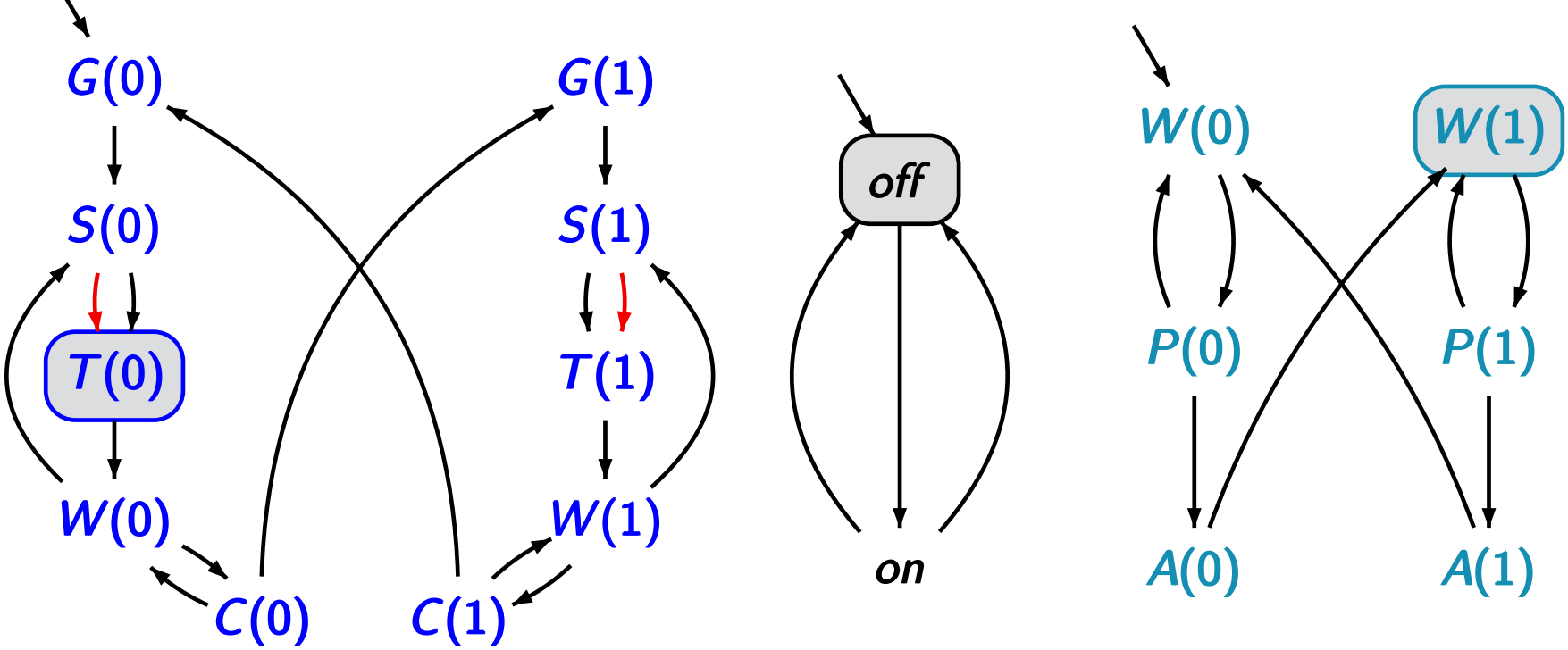
Generate(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	
Send(0)	off	Wait(0)	$c{=}\varepsilon$	$d{=}\varepsilon$	message 0 sent
Timer_on(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	0 sent again
Timer_on(0)	off	Wait(0)	$c{=}00$	$d{=}\varepsilon$	
Timer_on(0)	off	Proc(0)	$c{=}0$	$d{=}\varepsilon$	message received
Timer_on(0)	off	Ack(0)	$c{=}0$	$d{=}\varepsilon$	send ack via d



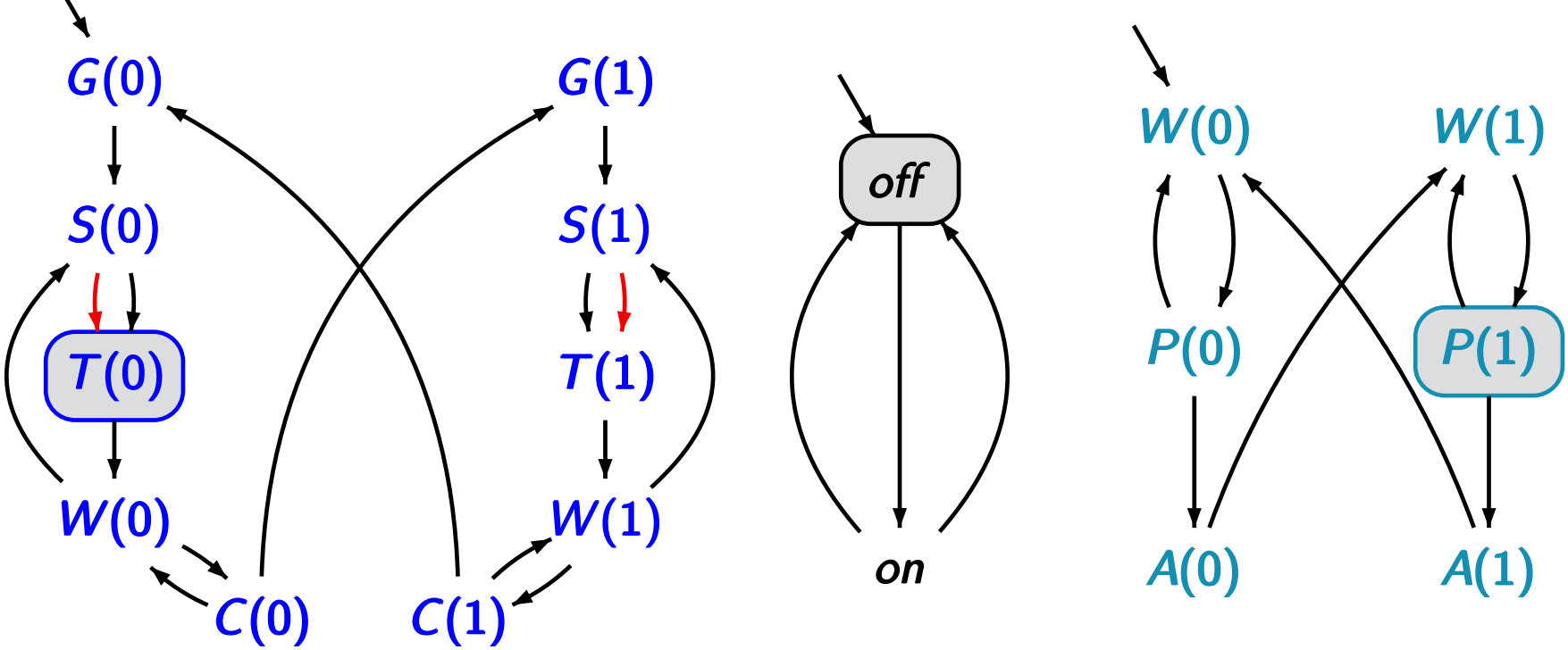
Wait(0)	on	Wait(0)	$c{=}0$	$d{=}\varepsilon$	timeout
Send(0)	off	Wait(0)	$c{=}0$	$d{=}\varepsilon$	0 sent again
Timer_on(0)	off	Wait(0)	$c{=}00$	$d{=}\varepsilon$	
Timer_on(0)	off	Proc(0)	$c{=}0$	$d{=}\varepsilon$	message received
Timer_on(0)	off	Ack(0)	$c{=}0$	$d{=}\varepsilon$	send ack via d
Timer_on(0)	off	Wait(1)	$c{=}0$	$d{=}0$	receiver changes its mode



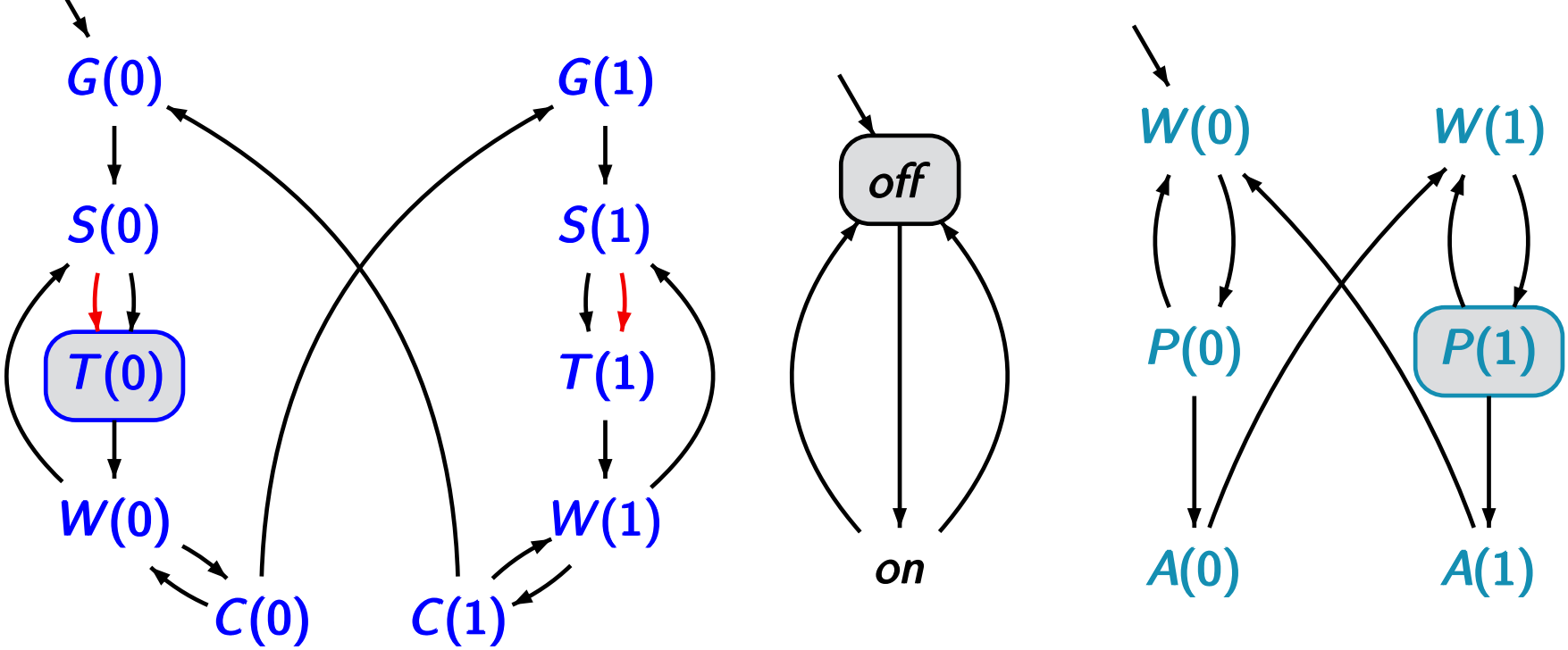
Timer_on(0) off Wait(1) $c=0$ $d=0$



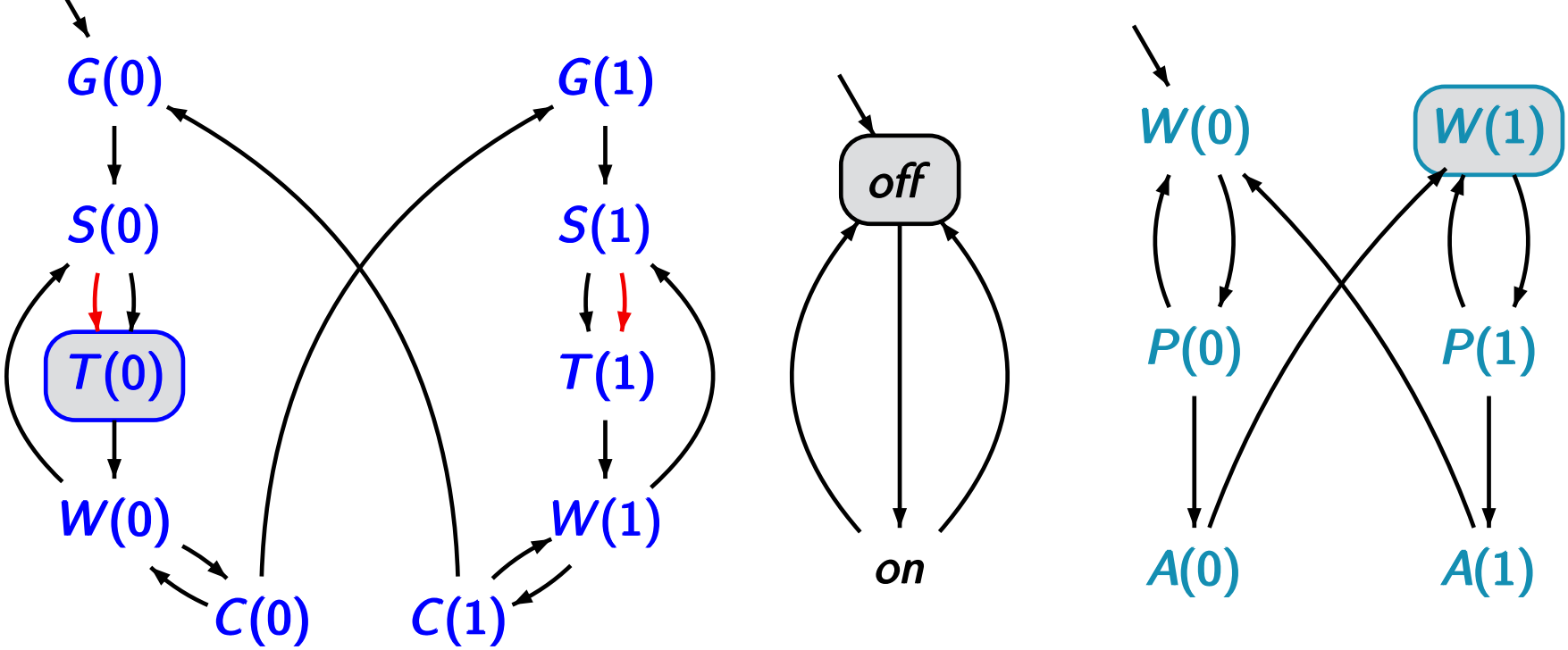
Timer_on(0) off Wait(1) $c=0$ $d=0$ receiver reads the same message again



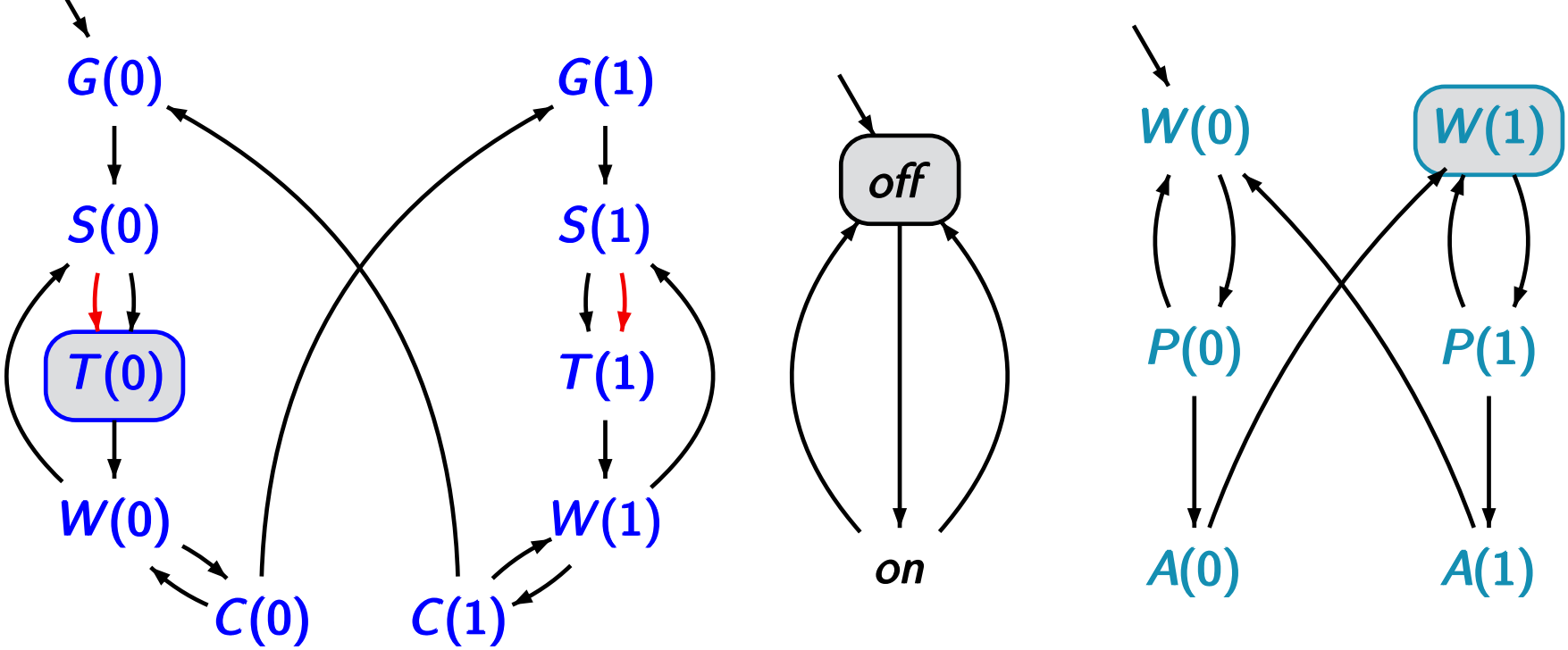
Timer_on(0)	off	Wait(1)	$c{=}0$	$d{=}0$	receiver reads the same message again
Timer_on(0)	off	Proc(1)	$c{=}\varepsilon$	$d{=}0$	



Timer_on(0)	off	Wait(1)	$c{=}0$	$d{=}0$	receiver reads the same message again
Timer_on(0)	off	Proc(1)	$c{=}\varepsilon$	$d{=}0$	receiver discards the message

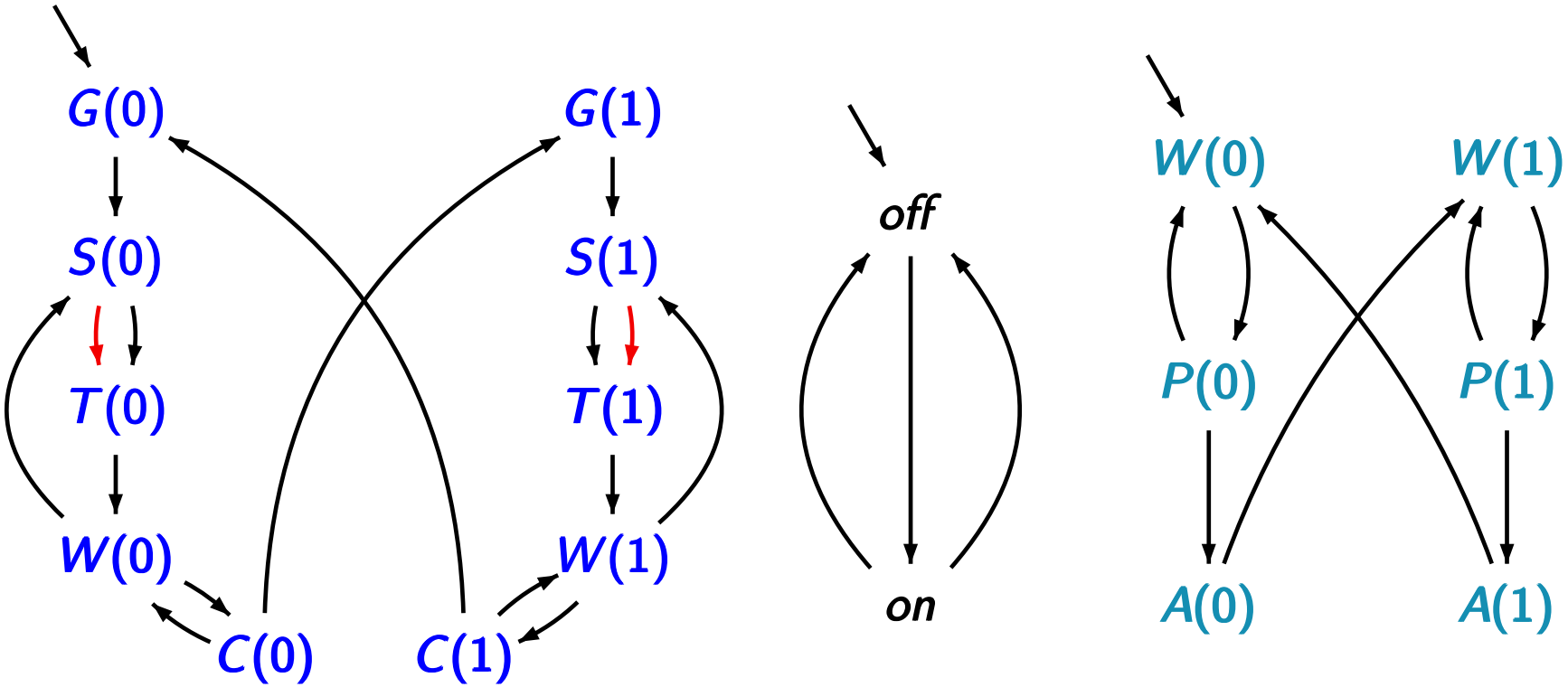


Timer_on(0)	off	Wait(1)	$c{=}0$	$d{=}0$	receiver reads the same message again
Timer_on(0)	off	Proc(1)	$c{=}\varepsilon$	$d{=}0$	receiver discards the message
Timer_on(0)	off	Wait(1)	$c{=}\varepsilon$	$d{=}0$	



Timer_on(0)	off	Wait(1)	$c{=}0$	$d{=}0$	receiver reads the same message again
Timer_on(0)	off	Proc(1)	$c{=}\varepsilon$	$d{=}0$	receiver discards the message
Timer_on(0)	off	Wait(1)	$c{=}\varepsilon$	$d{=}0$	

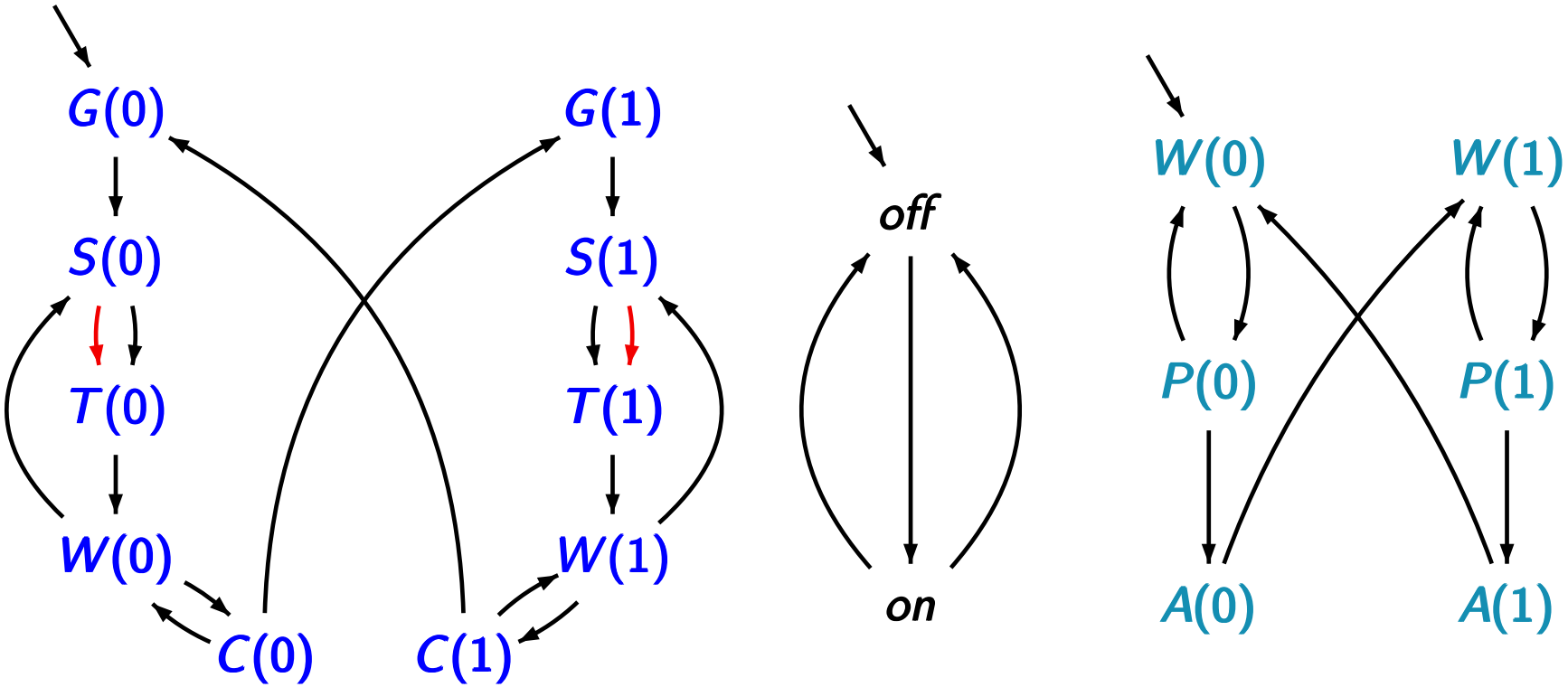
Alternating bit protocol (ABP)



number of states in the TS:

$$\mathbf{10} \cdot 2 \cdot \mathbf{6} \cdot \#\text{channel evaluations}$$

Alternating bit protocol (ABP)



number of states in the TS:

$\mathbf{10} \cdot \mathbf{2} \cdot \mathbf{6} \cdot \#\text{channel evaluations}$

$> \mathbf{10^8}$ for FIFOs with capacity **10**

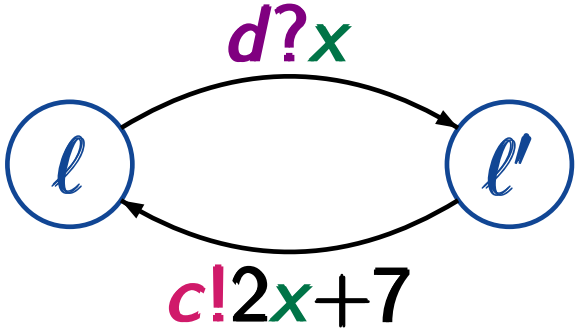
Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$

Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$
- generalized sending instructions $c!expr$ instead of $c!v$

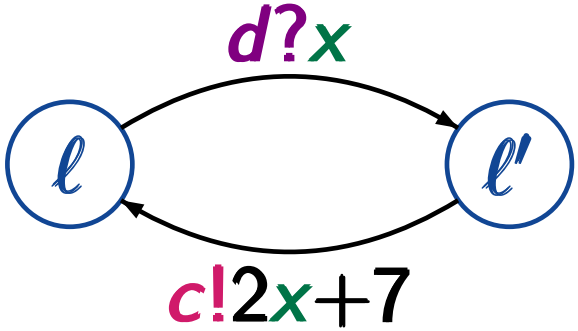
e.g.



Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$
- generalized sending instructions $c!expr$ instead of $c!v$

e.g.

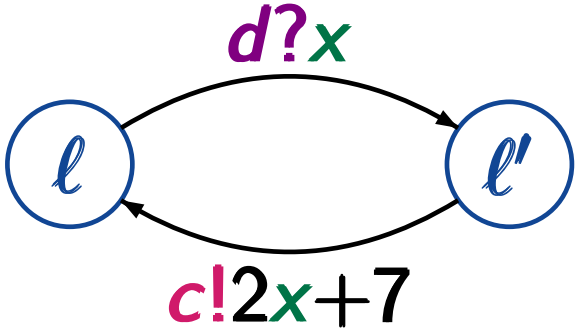


- communication as conditions $\ell \xrightarrow{c?x:\alpha} \ell'$

Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$
- generalized sending instructions $c!expr$ instead of $c!v$

e.g.



- communication as conditions $\ell \xrightarrow{c?x:\alpha} \ell'$

$\rightsquigarrow$ more compact TS-representations

Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$
- generalized sending instructions $c!expr$ instead of $c!v$
- communication as conditions $\ell \xrightarrow{c?x:\alpha} \ell'$
- *open* channel systems $\mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_n$
  instead of *closed* channel systems $[\mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_n]$

Variants of channel systems

- conditional communication actions $\ell \xrightarrow{g:c?x} \ell'$
- generalized sending instructions $c!expr$ instead of $c!v$
- communication as conditions $\ell \xrightarrow{c?x:\alpha} \ell'$
- *open* channel systems $\mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_n$
  instead of *closed* channel systems $[\mathcal{P}_1 \,|\, \ldots \,|\, \mathcal{P}_n]$