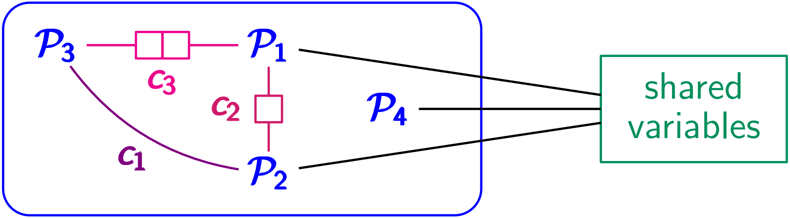
Channel systems

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** ↗ synchronous

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

Channel systems

representation of data-dependent parallel systems with

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

$\left.\begin{array}{l}\text{synchronous message passing}\\ \text{asynchronous message passing}\end{array}\right\}$ 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 $Dom(x)$

evaluation for a set Var of typed variables:

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

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 $\xi : Chan \rightarrow Values^*$

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 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

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

Channel system (CS)

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

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

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

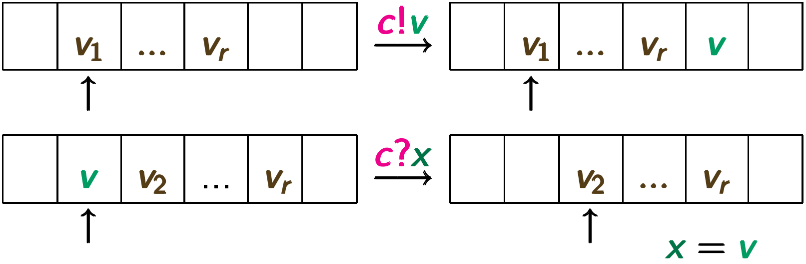
$\ell \overset{c!v}{\hookrightarrow}_i \ell'$ sending value v via channel c

$\ell \overset{c?x}{\hookrightarrow}_i \ell'$ 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 $c!v$	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 $\mathbf{0}$

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

TS-semantics of channel systems

channel system over $(\mathit{Var}, \mathit{Chan})$

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

$$\Downarrow$$

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

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

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

locations of $\mathcal{P}_1, \ldots, \mathcal{P}_n$

channel evaluation

variable valuation

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 $\ell_i \in Loc_i$, $\eta \in Eval(Var)$,
$\xi \in Eval(Chan)$

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

- interleaving rules for $\alpha \in Act_i$
- rules for message passing along channels

interleaving rule for actions $\alpha \in Act_i$:

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

$\uparrow$

does not affect the channel evaluation ξ

SOS-rules for asynchronous message passing

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

receiving a message:

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

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:

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

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

sending a message:

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

SOS-rules for synchronous message passing

for synchronous channel c:

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

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

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:

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

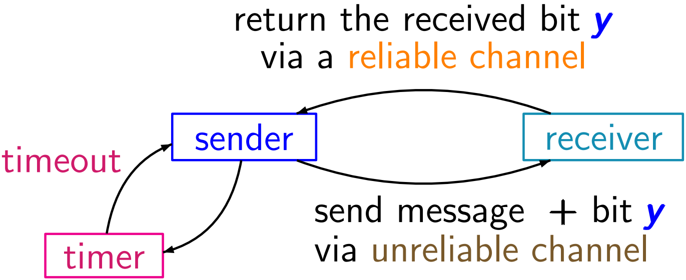
note: $2^{11}-1 = 1 + 2 + 2^2 + \ldots + 2^{10}$

... with an unbounded channel **?**

answer: ∞

Protocol for the sender

PC2.2-32



LOOP FOREVER

(1) send message + **bit** y and activate timer

(2) AWAIT timeout or acknowledgement DO

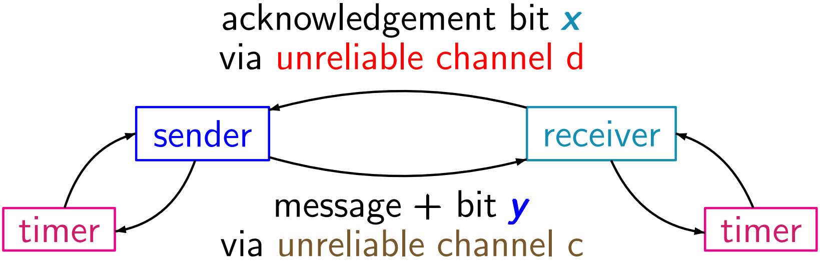
  IF timeout THEN goto (1)

       ELSE turn off timer; $y := \neg y$

  FI

OD

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; $y := \neg y$

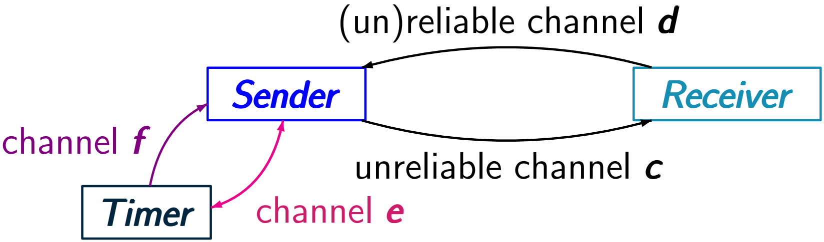
ELSE ignore x

FI

FI

OD

Alternating bit protocol (ABP)

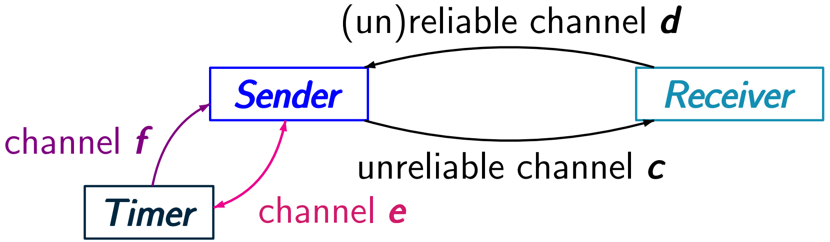


channel system: $\big[\, \textit{Sender} \mid \textit{Timer} \mid \textit{Receiver} \,\big]$

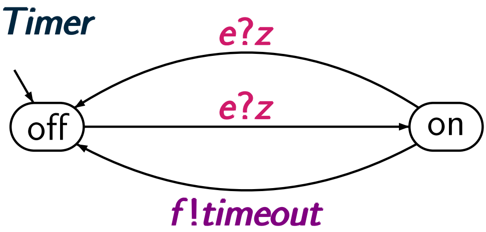
synchronous message passing between
Timer and *Sender* $\longleftarrow$ channels *e* and *f*

asynchronous message passing between
Receiver and *Sender* $\longleftarrow$ channels *c*, *d*

Alternating bit protocol (ABP)



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



actions of *Sender*:

$\vdots$

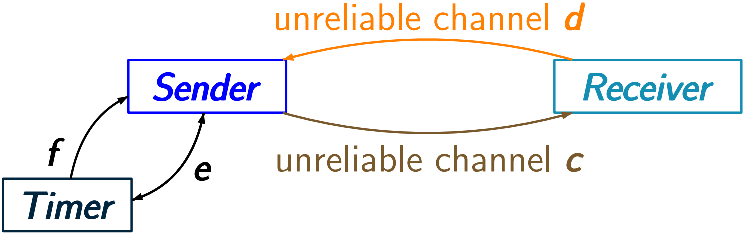
$e!\mathit{timer\_on}$

$e!\mathit{timer\_off}$

$f?z'$

$\vdots$

Program graph for the sender

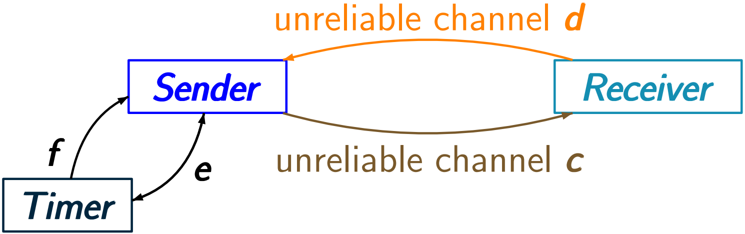


specify the sender by a program graph using

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

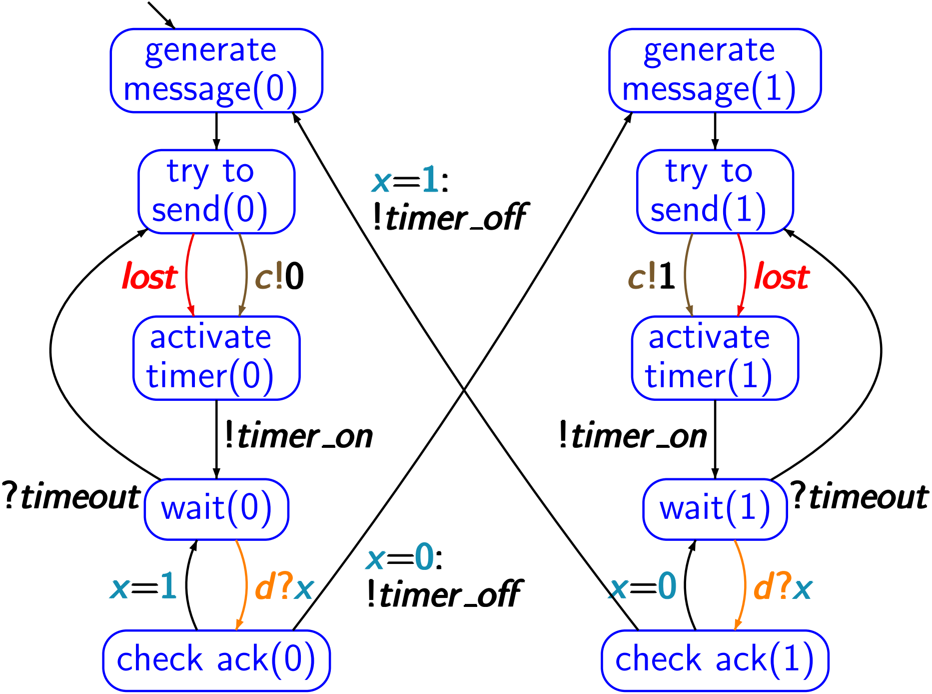
simply write **!*timeout***
?*timer_on*
?*timer_off*

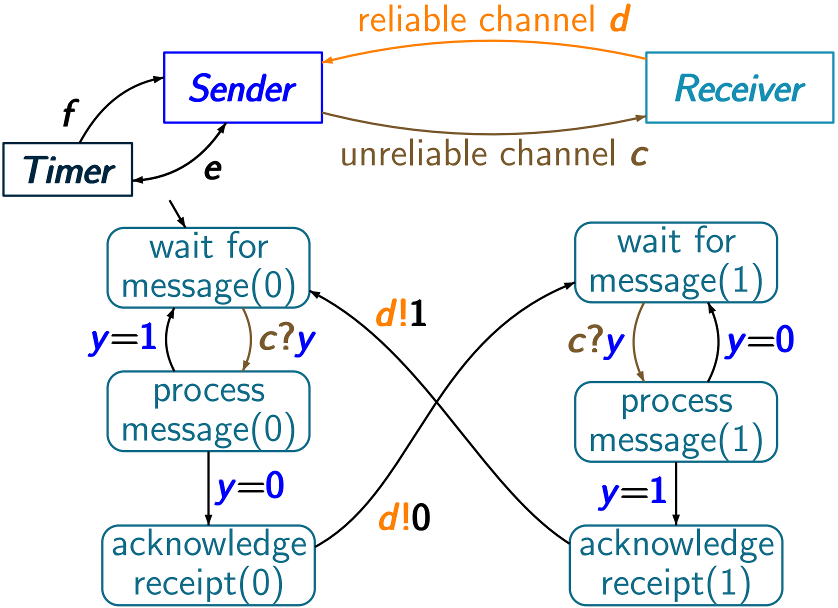
Program graph for the sender



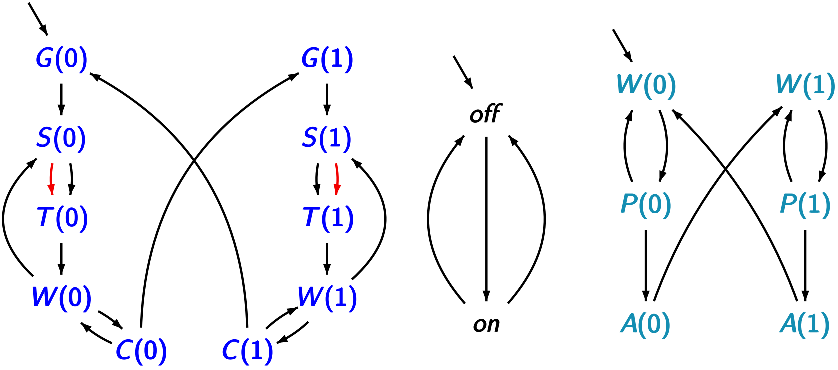
specify the sender by a program graph using

- 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)
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)


reliable channel d
Sender
Receiver
unreliable channel c
Timer
f
e
wait for message(0)
wait for message(1)
process message(0)
process message(1)
acknowledge receipt(0)
acknowledge receipt(1)
y=1
$c?y$
y=0
$d!1$
$d!0$
$c?y$
y=0
y=1

Alternating bit protocol (ABP)



number of states in the TS:

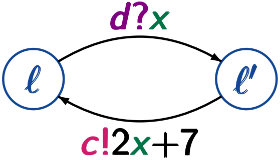
$\mathbf{10} \cdot 2 \cdot \mathbf{6} \cdot$ #channel evaluations

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

Variants of channel systems

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

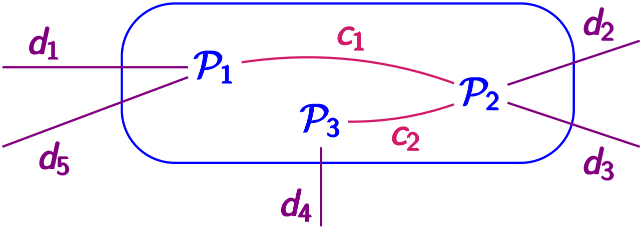
  e.g.



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

$\rightsquigarrow$ more compact TS-representations

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



Summary: parallel operators

(pure) interleaving for TS $\mathcal{T}_1 \ ||| \ \mathcal{T}_2$

- only concurrency, no communication
- not applicable for competing systems

synchronous message passing for TS $\mathcal{T}_1 \|_{Syn} \mathcal{T}_2$

- interleaving for concurrent actions
- synchronization via actions in Syn

interleaving for program graphs $\mathcal{P}_1 \ ||| \ \mathcal{P}_2$

- interleaving for concurrent actions
- communication via shared variables

channel systems: open $\mathcal{P}_1 | \ldots | \mathcal{P}_n$ or closed $[\mathcal{P}_1 | \ldots | \mathcal{P}_n]$

- interleaving, shared variables, message passing

Parallel operators

(pure) interleaving for TS $\mathcal{T}_1 \;|||\; \mathcal{T}_2$

- only concurrency, no communication

synchronous message passing for TS $\mathcal{T}_1 \,\|_{Syn}\, \mathcal{T}_2$

- interleaving, synchronization via actions in Syn

interleaving for program graphs $\mathcal{P}_1 \;|||\; \mathcal{P}_2$

- interleaving, shared variables

channel systems: open $\mathcal{P}_1 \,|\, \ldots \,|\mathcal{P}_n$ or closed $[\mathcal{P}_1 \,|\, \ldots \,|\mathcal{P}_n]$

- interleaving, shared variables
- synchronous and asynchronous message passing

synchronous product for TS $\mathcal{T}_1 \otimes \mathcal{T}_2$

- no interleaving, "pure" synchronization

Synchronous product

for parallel systems with fully synchronized processes

$$\left.\begin{array}{l} \mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, \ldots) \\ \mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, \ldots) \end{array}\right\} \text{two TS}$$

synchronous product:

$$\mathcal{T}_1 \otimes \mathcal{T}_2 \; = \; (S_1 \times S_2, Act, \longrightarrow, \ldots)$$

where the action set Act is given by a function

$$Act_1 \times Act_2 \longrightarrow Act, \quad (\alpha, \beta) \mapsto \alpha * \beta$$

action name for the concurrent execution of α and β

if action names are irrelevant: $Act_1 = Act_2 = Act = \{\tau\}$

Synchronous product

for parallel systems with fully synchronized processes

$$\left.\begin{array}{l}\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, \ldots) \\ \mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, \ldots)\end{array}\right\} \text{ two TS}$$

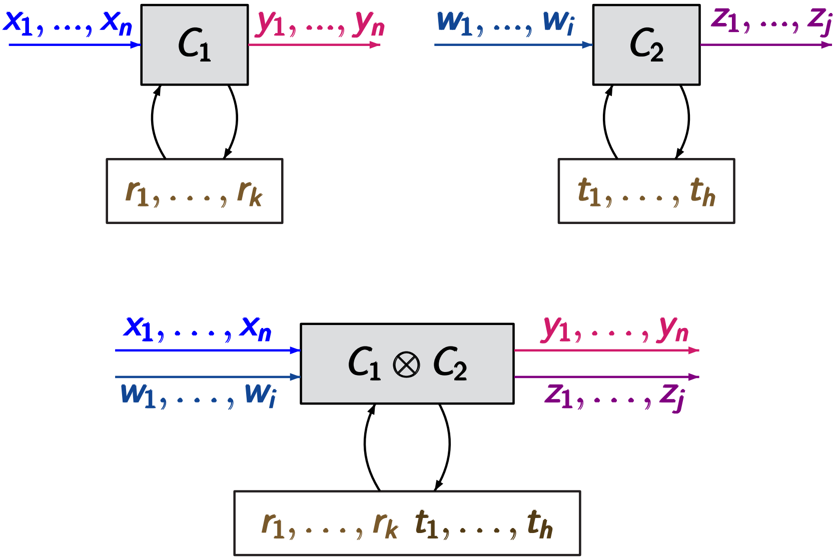
synchronous product:

$$\mathcal{T}_1 \otimes \mathcal{T}_2 \;=\; (S_1 \times S_2, Act, \longrightarrow, \ldots)$$

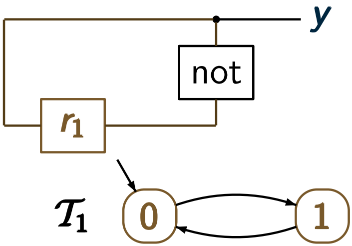
transition relation $\longrightarrow$:

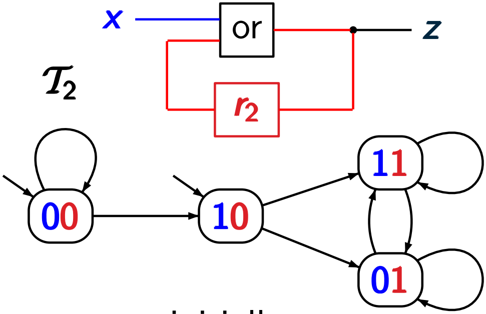
$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \;\wedge\; s_2 \xrightarrow{\beta}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha * \beta} \langle s_1', s_2' \rangle}$$

Synchronous product for composing circuits



Synchronous product: example





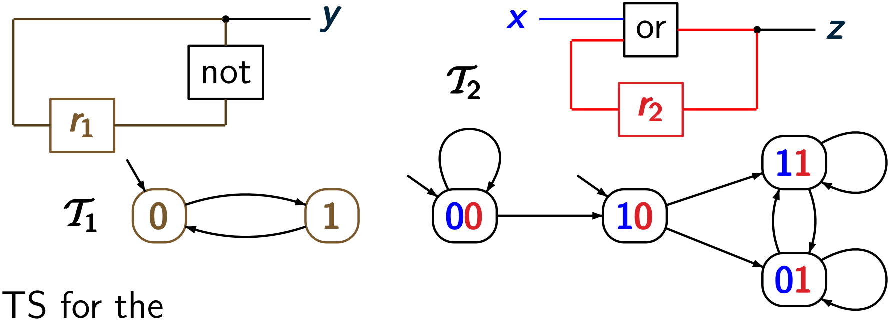
initially:
$r_1 = 0$

transition function:
$\delta_{r_1} = \neg r_1$

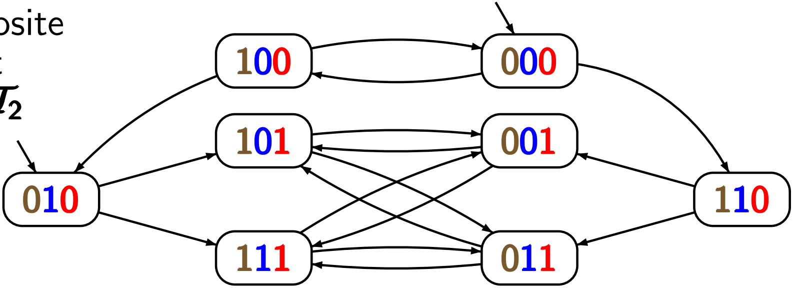
initially:
$r_2 = 0$

transition function:
$\delta_{r_2} = r_2 \vee x$

Synchronous product: example



TS for the composite circuit $\mathcal{T}_1 \otimes \mathcal{T}_2$



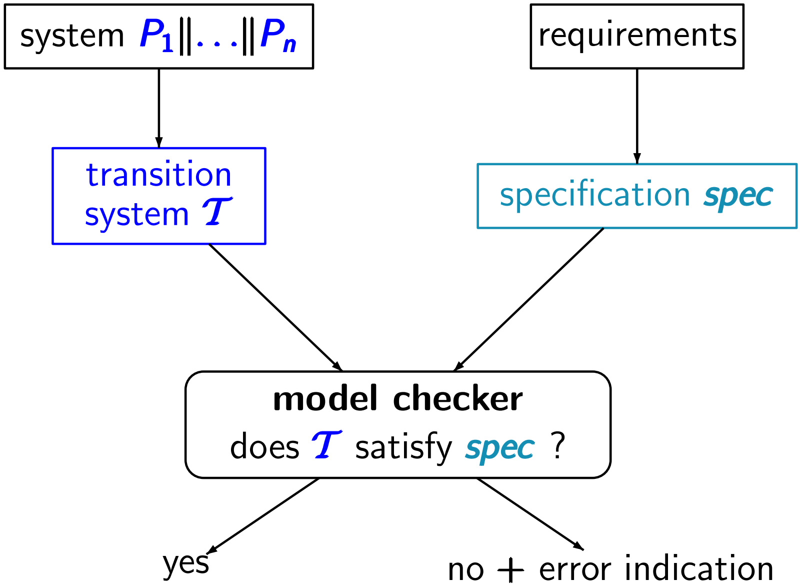
TS for reactive systems can be enormously large

- infinite for systems with
  - ∗ variables of infinite domains, e.g., $\mathbb{N}$
  - ∗ infinite data structures, e.g., stacks, queues, lists,...
- if finite: exponential growth in
  - ∗ number of parallel components,
    e.g., state space of $\mathcal{T}_1 \parallel \ldots \parallel \mathcal{T}_n$ is $S_1 \times \ldots \times S_n$
  - ∗ number of variables and channels

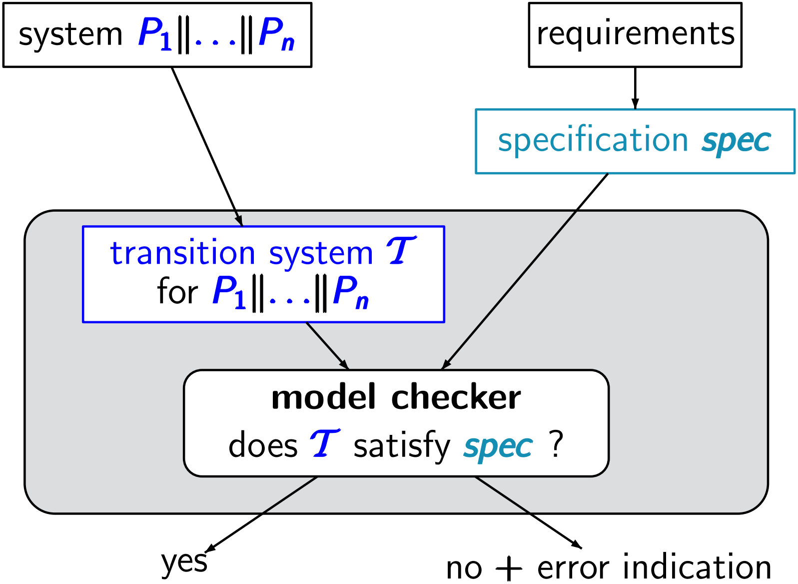
e.g., for channel systems: size of the state space is

$$|Loc_1| \cdot \ldots \cdot |Loc_n| \cdot \prod_{x \in Var} |Dom(x)| \cdot \prod_{c \in Chan} |Dom(c)|^{cap(c)}$$

Model checking



Model checking



Model checking