Model Checking I
alias
Reactive Systems Verification

Luca Tesei

MSc in Computer Science, University of Camerino

Topics

- Parallelism and Communication
- Synchronous Message Passing
- Examples

Material

Reading:

Chapter 2 of the book, pages 47–53.

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.

Operators for parallelism and communication

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS
  (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  * description of subsystems by program graphs
  * interleaving $|||$ for program graphs
  * TS is obtained by "unfolding"
- synchronous message passing

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  * description of subsystems by program graphs
  * interleaving $|||$ for program graphs
  * TS is obtained by "unfolding"
- synchronous message passing
  * operator $\|_{Syn}$ for TS
  * interleaving for independent actions
  * synchronization over actions in Syn

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - ∗ description of subsystems by program graphs
  - ∗ interleaving $|||$ for program graphs
  - ∗ TS is obtained by "unfolding"
- synchronous message passing $\longleftarrow$ data abstract
  - ∗ operator $\|_{Syn}$ for TS
  - ∗ interleaving for independent actions
  - ∗ synchronization over actions in *Syn*

Operators for parallelism and communication

- true concurrency: interleaving operator $|||$ for TS (no communication, no dependencies)
- communication via shared variables
  - description of subsystems by program graphs
  - interleaving $|||$ for program graphs
  - TS is obtained by "unfolding"
- synchronous message passing $\longleftarrow$ data abstract
  - operator $\|_{Syn}$ for TS
  - interleaving for independent actions
  - synchronization over actions in Syn
- channel systems
  communication via shared variables + via channels
- synchronous product

Synchronous message passing

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \|_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

for modeling the concurrent execution of $\mathcal{T}_1$ and $\mathcal{T}_2$ with synchronization over all actions in Syn

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \|_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

handshaking (rendezvous) for all $\alpha \in Syn$:

Synchronous message passing

$\mathcal{T}_1 = (S_1, Act_1, \to_1, \ldots)$, $\mathcal{T}_2 = (S_2, Act_2, \to_2, \ldots)$ TS

$Syn \subseteq Act_1 \cap Act_2$ set of synchronization actions

composite transition system:

$$\mathcal{T}_1 \|_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \to, \ldots)$$

interleaving for all actions $\alpha \in Act_i \setminus Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

handshaking (rendezvous) for all $\alpha \in Syn$:

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

Mutual exclusion

by synchronous message passing

Mutual exclusion

by synchronous message passing using an arbiter

Mutual exclusion with an arbiter

protocol for process P_i

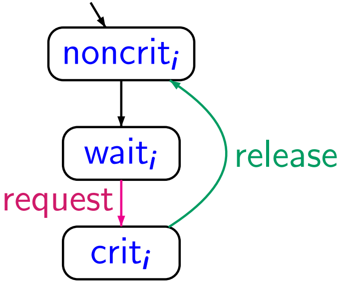
```
LOOP FOREVER DO
      noncritical actions
      request
      critical section
      release
      noncritical actions
OD
```

Mutual exclusion with an arbiter

protocol for process P_i

```
LOOP FOREVER DO
      noncritical actions
      request
      critical section
      release
      noncritical actions
OD
```

transition system $\mathcal{T}_i$

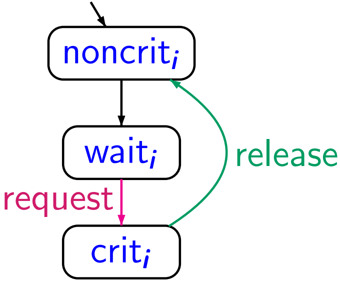


Mutual exclusion with an arbiter

protocol for process P_i

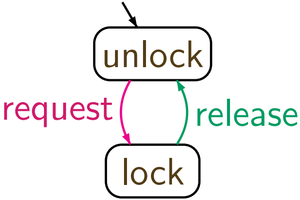
```
LOOP FOREVER DO
        noncritical actions
        request
        critical section
        release
        noncritical actions
OD
```

transition system $\mathcal{T}_i$



Arbiter:

selects nondeterministically a synchronization partner $\mathcal{T}_1$ or $\mathcal{T}_2$



Mutual exclusion with an arbiter

$(\mathcal{T}_1 \; ||| \; \mathcal{T}_2) \; \|_{\mathit{Syn}} \; \mathit{Arbiter}$ where $\mathit{Syn} = \{\mathsf{request}, \mathsf{release}\}$

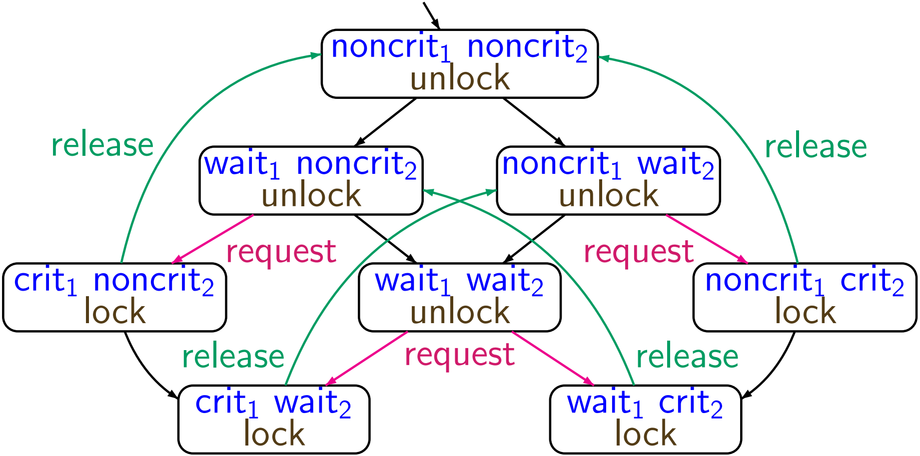
Mutual exclusion with an arbiter

$$(\mathcal{T}_1 \;|||\; \mathcal{T}_2) \;\|_{Syn}\; \mathit{Arbiter} \text{ where } \mathit{Syn} = \{\text{request}, \text{release}\}$$

"pure" interleaving for TS

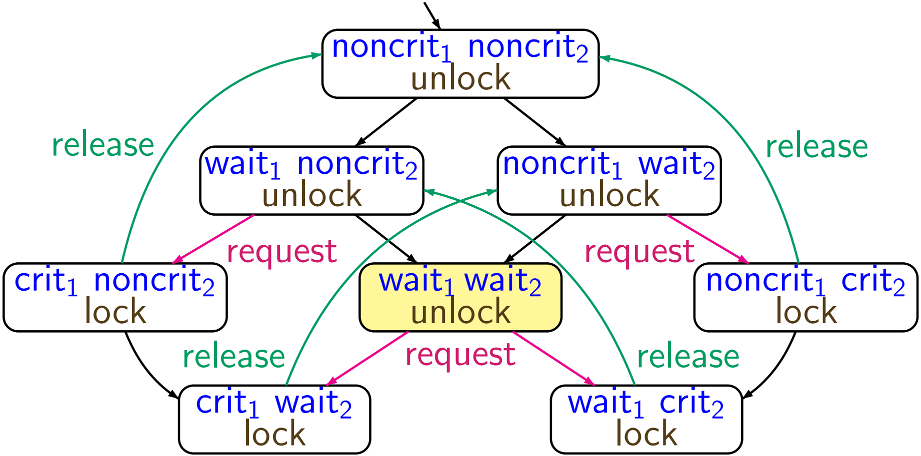
handshaking for actions request and release

$(\mathcal{T}_1 \,|||\, \mathcal{T}_2) \parallel_{Syn} \textit{Arbiter}$ where $Syn = \{\text{request}, \text{release}\}$



Mutual exclusion with an arbiter

$(\mathcal{T}_1 \,|||\, \mathcal{T}_2) \parallel_{Syn} \textit{Arbiter}$ where $Syn = \{\text{request}, \text{release}\}$



nondeterministic choice: who enters the critical section**?**

synchronization operator $\|_{Syn}$ for
three or more processes

Synchronous message passing

$$\left.\begin{array}{rcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \end{array}\right\} \text{transition systems}$$

$$\begin{array}{rcl} \vdots & & \vdots \end{array}$$

Synchronous message passing

$$\left.\begin{array}{lcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \end{array}\right\} \text{transition systems}$$

$$\begin{array}{lcl} \vdots & & \vdots \end{array}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\boxed{\begin{array}{l} \mathcal{T}_1 \,\|_{Syn}\, \mathcal{T}_2 \,\|_{Syn}\, \mathcal{T}_3 \,\|_{Syn}\, \mathcal{T}_4 \,\|_{Syn}\, \ldots \stackrel{\text{def}}{=} \\ \qquad \Big(\big((\mathcal{T}_1 \,\|_{Syn}\, \mathcal{T}_2) \,\|_{Syn} \mathcal{T}_3 \big) \,\|_{Syn}\, \mathcal{T}_4 \Big) \,\|_{Syn}\, \ldots \end{array}}$$

Synchronous message passing

$$\left.\begin{array}{lcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \quad\quad \vdots \end{array}\right\} \text{transition systems}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\boxed{\begin{array}{l} \mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \parallel_{Syn} \mathcal{T}_3 \parallel_{Syn} \mathcal{T}_4 \parallel_{Syn} \ldots \stackrel{\text{def}}{=} \\ \qquad \Big(\big((\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2) \parallel_{Syn} \mathcal{T}_3\big) \parallel_{Syn} \mathcal{T}_4\Big) \parallel_{Syn} \ldots \end{array}}$$

or any other order of paranthesis

Synchronous message passing

$$\left.\begin{array}{lcl} \mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\ \mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\ \mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\ \mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\ \vdots & & \qquad\vdots \end{array}\right\} \text{ transition systems}$$

for $Syn \subseteq Act_1 \cup Act_2 \cup Act_3 \cup Act_4 \cup \ldots$

$$\boxed{\begin{array}{l} \mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \parallel_{Syn} \mathcal{T}_3 \parallel_{Syn} \mathcal{T}_4 \parallel_{Syn} \ldots \stackrel{\text{def}}{=} \\ \qquad \Big(((\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2) \parallel_{Syn} \mathcal{T}_3) \parallel_{Syn} \mathcal{T}_4\Big) \parallel_{Syn} \ldots \end{array}}$$

where, e.g., $\mathcal{T}_1 \parallel_{Syn} \mathcal{T}_2 \stackrel{\text{def}}{=} \mathcal{T}_1 \parallel_{H} \mathcal{T}_2$

with $H = Syn \cap Act_1 \cap Act_2$

Parallel operator $\|$

$$\begin{array}{lcl}
\mathcal{T}_1 & = & (S_1, Act_1, \rightarrow_1, \ldots) \\
\mathcal{T}_2 & = & (S_2, Act_2, \rightarrow_2, \ldots) \\
\mathcal{T}_3 & = & (S_3, Act_3, \rightarrow_3, \ldots) \\
\mathcal{T}_4 & = & (S_4, Act_4, \rightarrow_4, \ldots) \\
\vdots & & \quad\quad\quad \vdots
\end{array}$$

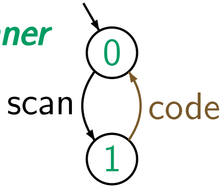
transition systems s.t. $Act_i \cap Act_j \cap Act_k = \emptyset$ if i, j, k are pairwise distinct

$$\boxed{\begin{array}{l} \mathcal{T}_1 \| \mathcal{T}_2 \| \mathcal{T}_3 \| \mathcal{T}_4 \| \ldots \overset{\text{def}}{=} \\ \quad (((\mathcal{T}_1 \|_{Syn_{1,2}} \mathcal{T}_2) \|_{Syn_{1,2,3}} \mathcal{T}_3) \|_{Syn_{1,2,3,4}} \mathcal{T}_4) \ldots \end{array}}$$

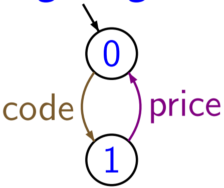
where
$$\begin{array}{lcl}
Syn_{1,2} & = & Act_1 \cap Act_2 \\
Syn_{1,2,3} & = & (Act_1 \cup Act_2) \cap Act_3 \\
Syn_{1,2,3,4} & = & (Act_1 \cup Act_2 \cup Act_3) \cap Act_4 \\
\vdots & & \quad\quad\quad \vdots
\end{array}$$

Booking system in supermarket

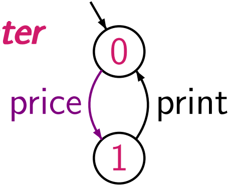
Scanner



Booking Program

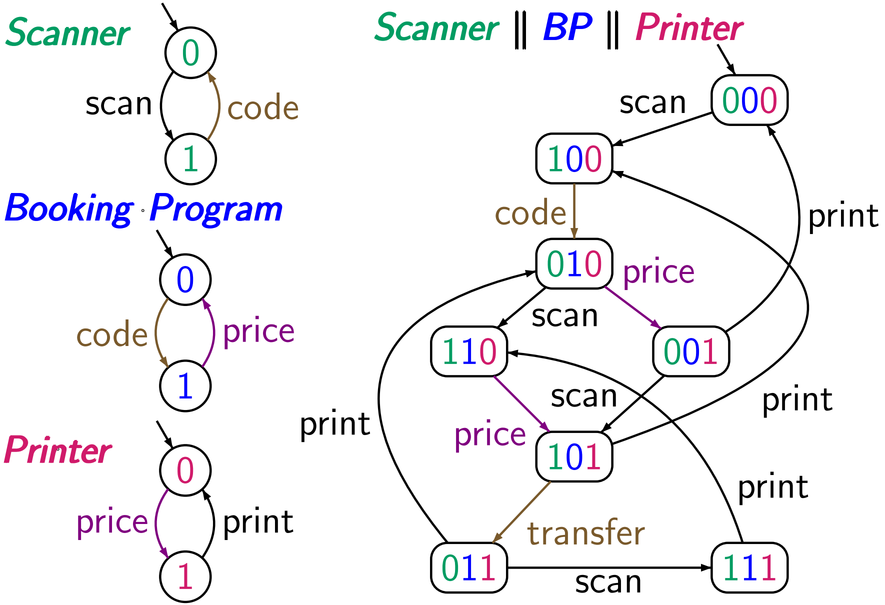


Printer

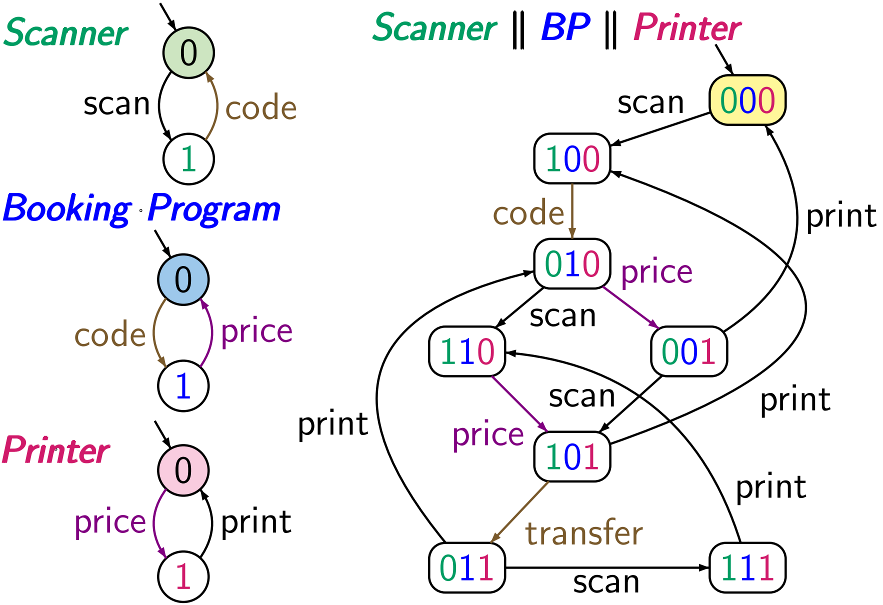


Scanner $\|$ *BP* $\|$ *Printer*

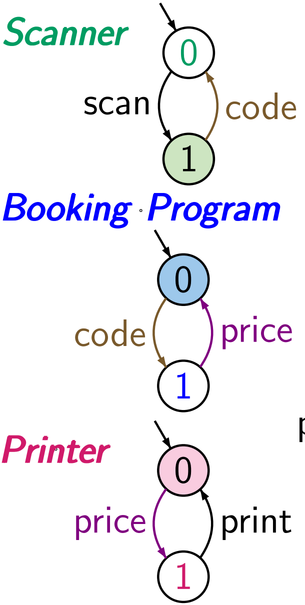
Booking system in supermarket

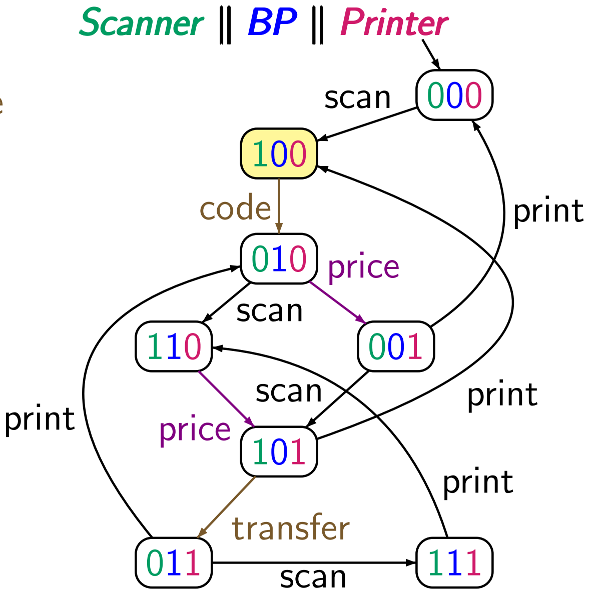


Booking system in supermarket

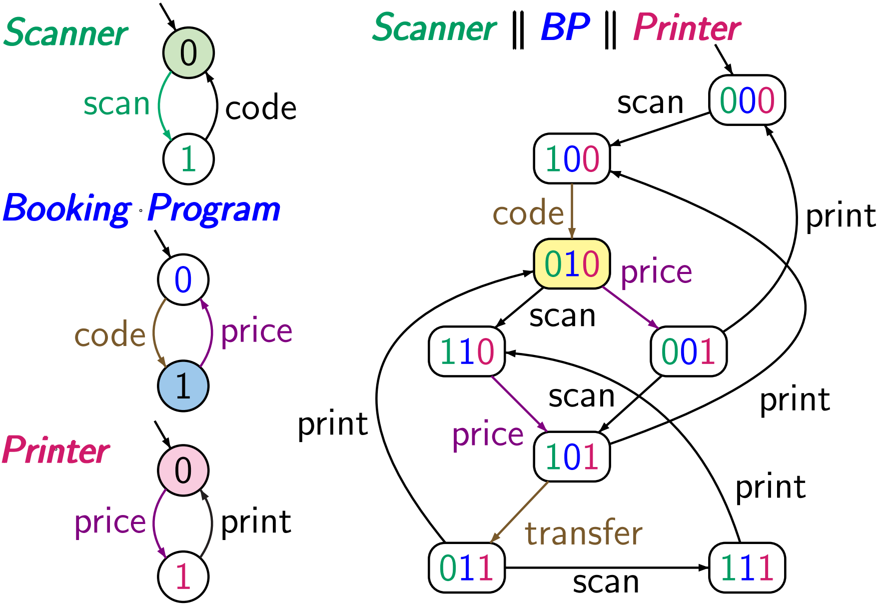


Booking system in supermarket

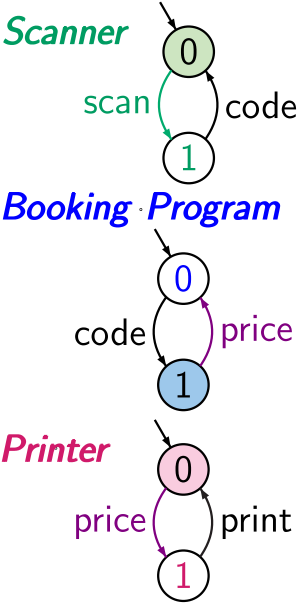


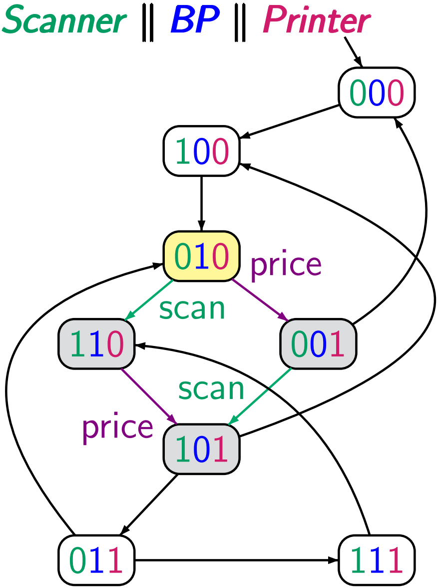


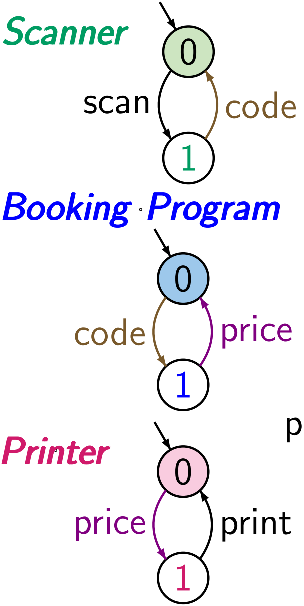
Booking system in supermarket



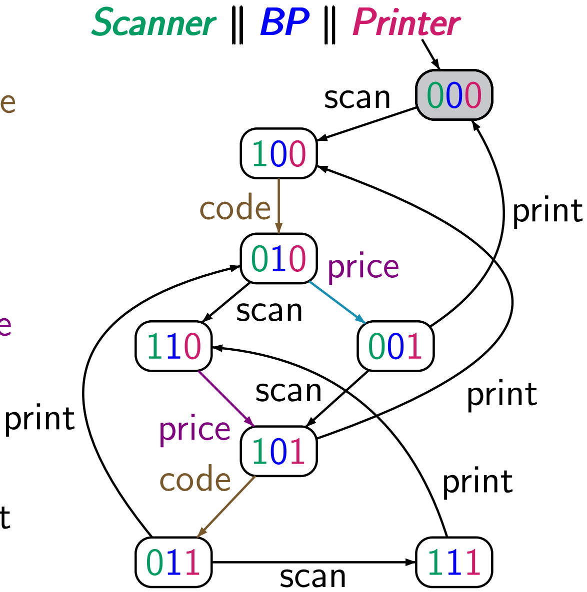
Interleaving





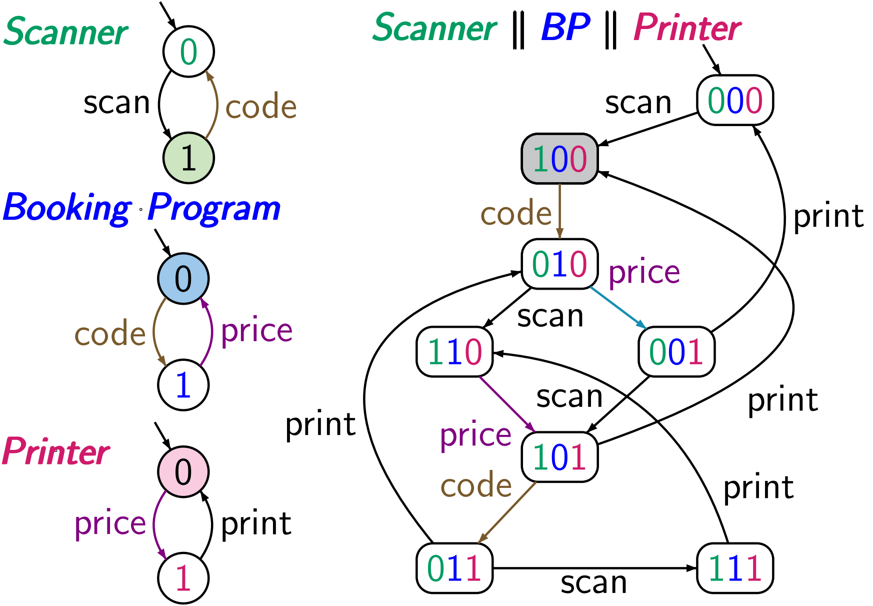


Scanner
0
scan
code
1
Booking Program
0
code
price
1
Printer
0
price
print
1

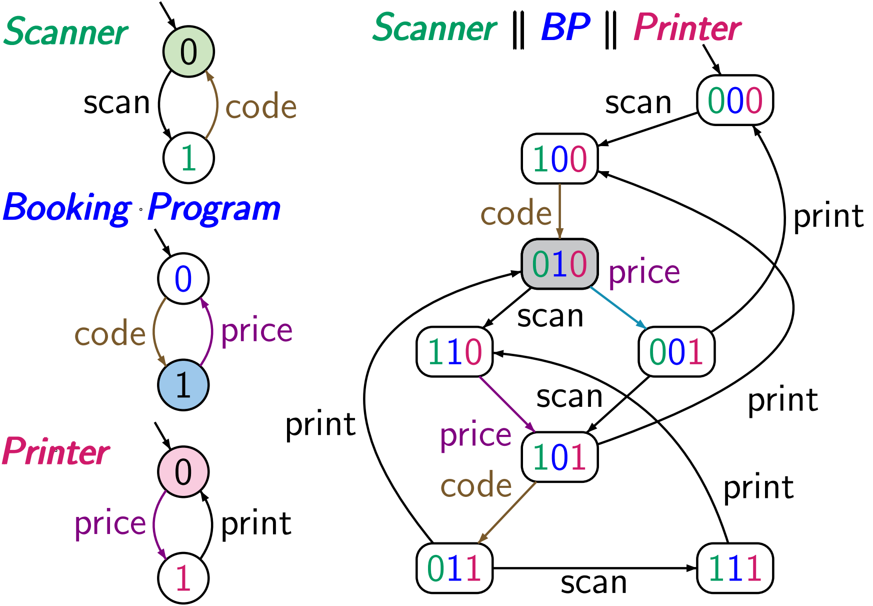


Scanner ‖ BP ‖ Printer
000
scan
100
code
010
price
scan
110
001
print
print
scan
price
101
print
code
print
011
111
scan

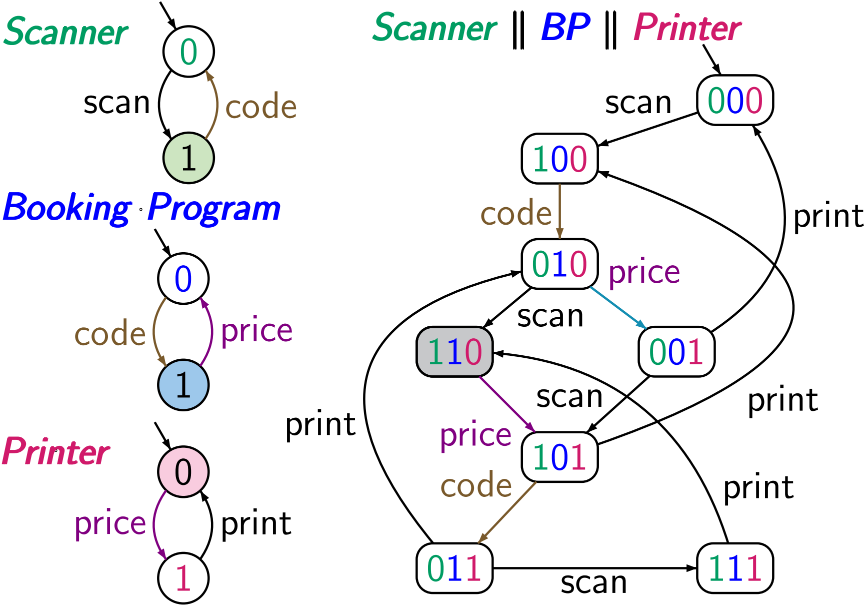
Booking system in supermarket



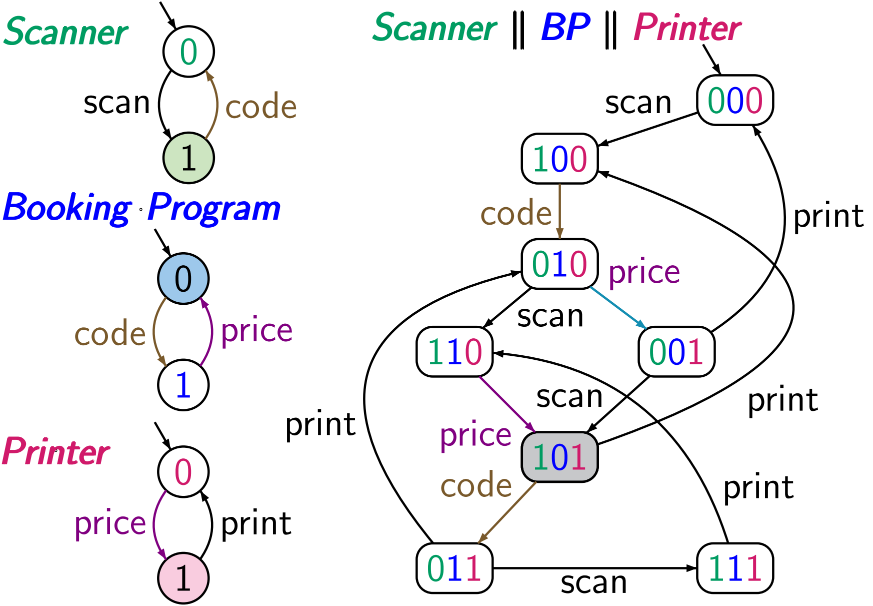
Booking system in supermarket



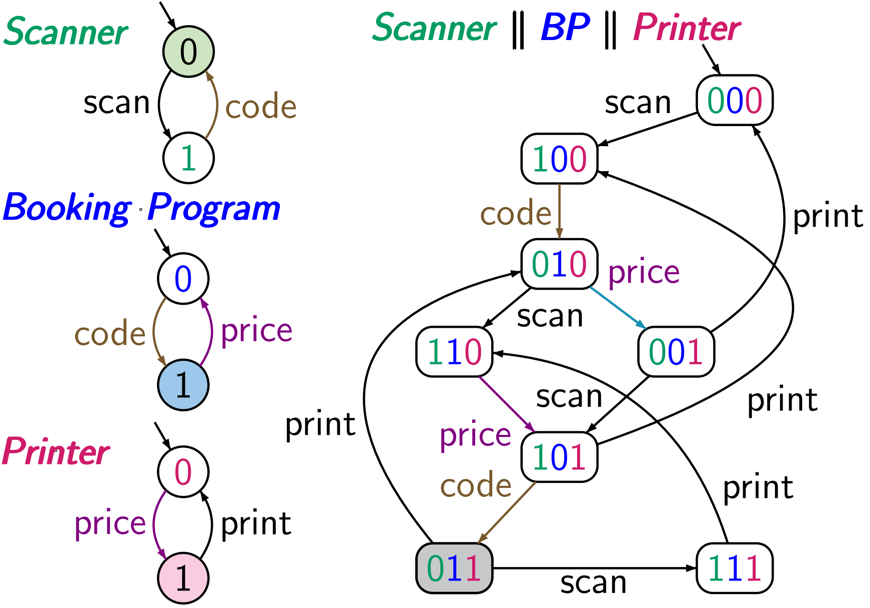
Booking system in supermarket



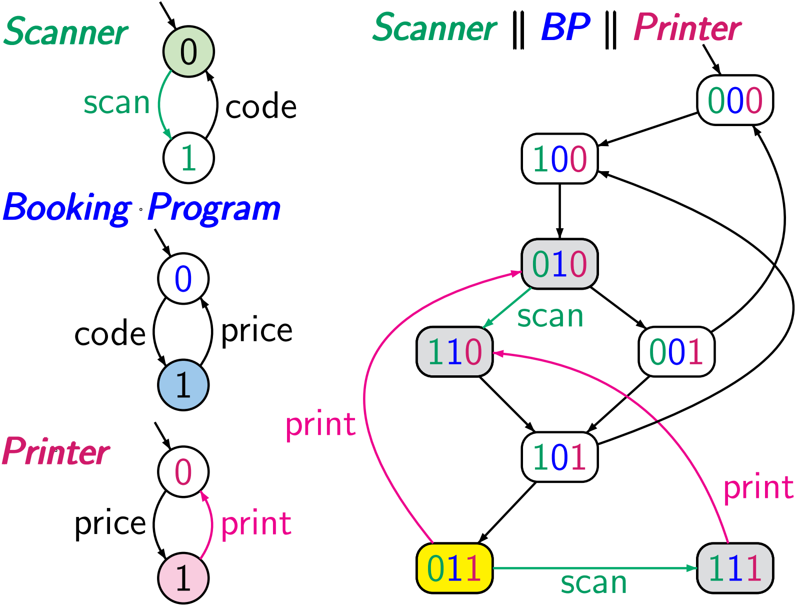
Booking system in supermarket



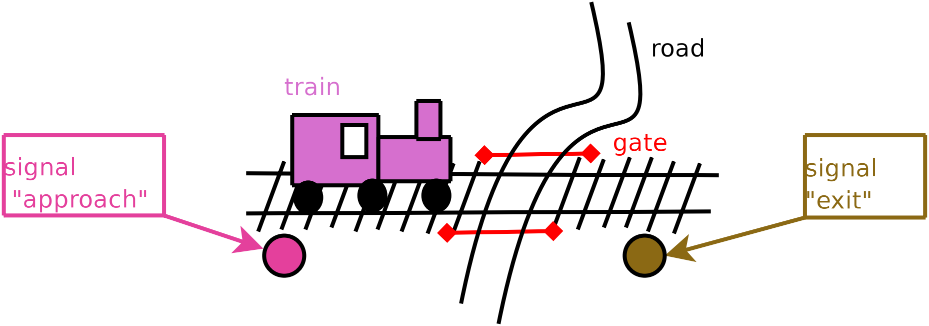
Booking system in supermarket



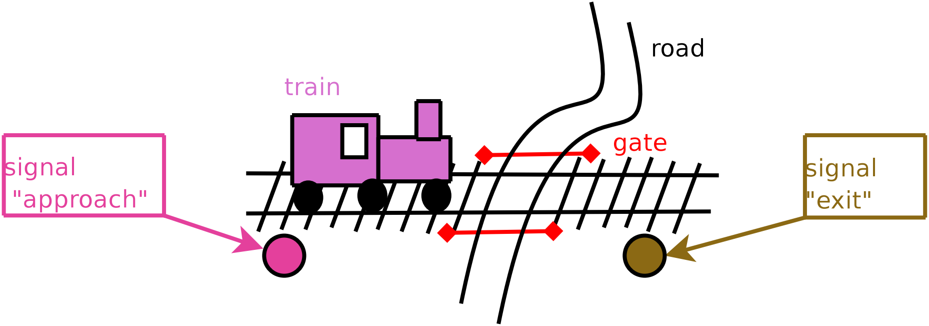
Interleaving



Railroad crossing



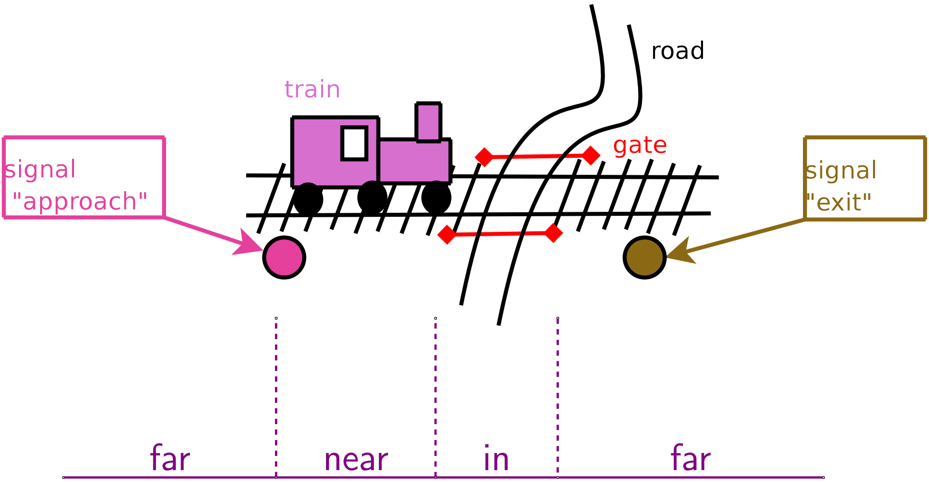
Railroad crossing



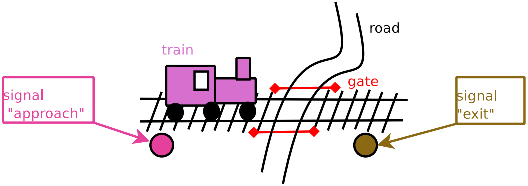
modeling by a transition system with **3** processes:

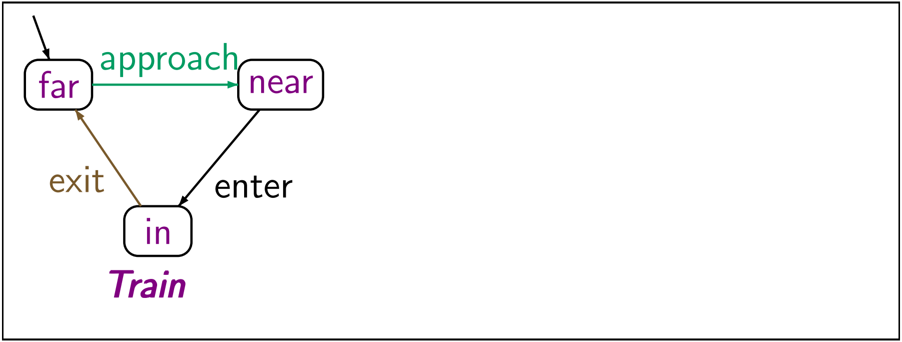
Train ‖ *Controller* ‖ *Gate*

Modeling the behavior of trains

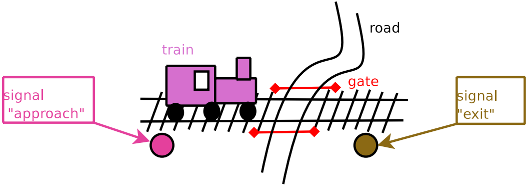


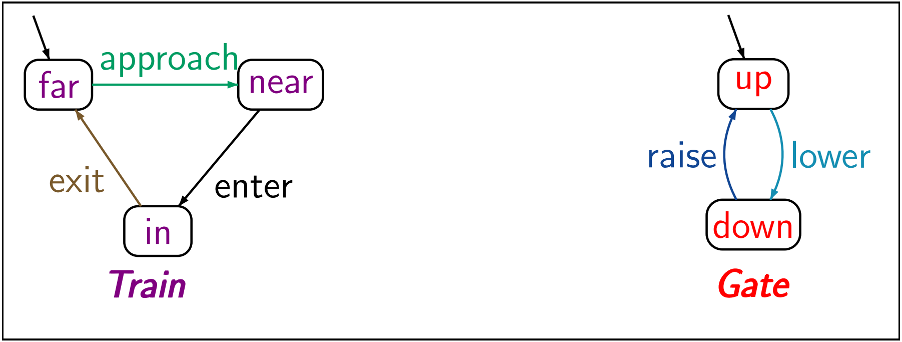
TS for the trains



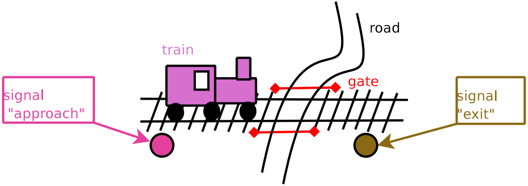


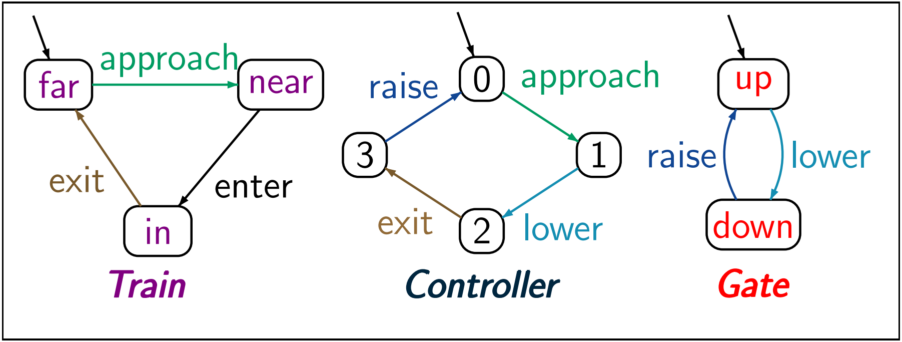
TS for the trains and gate



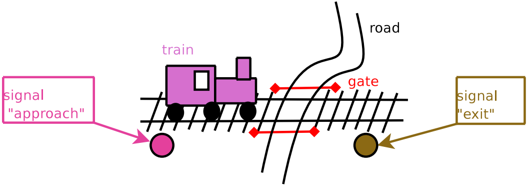


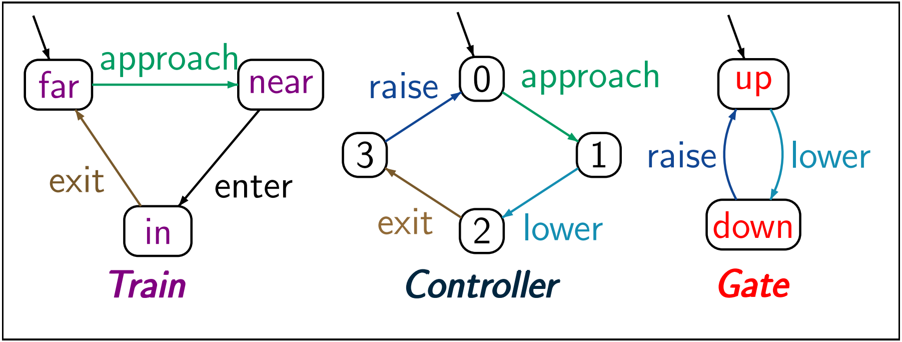
TS for the trains, controller and gate





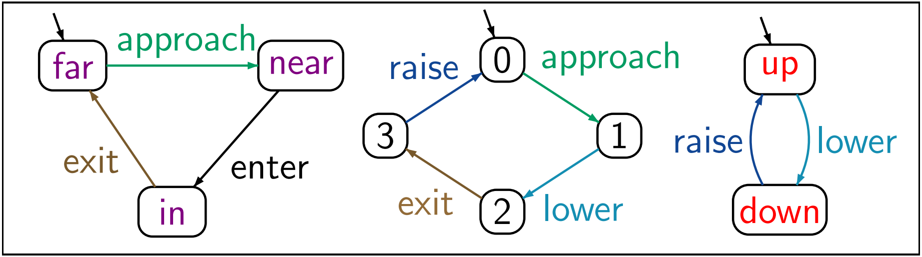
TS for the trains, controller and gate





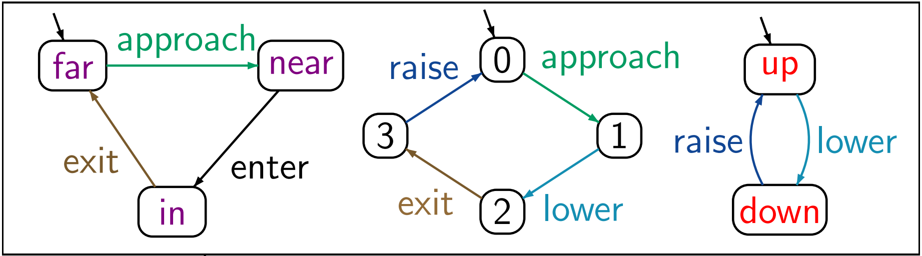
transition system *Train* ∥ *Controller* ∥ *Gate*

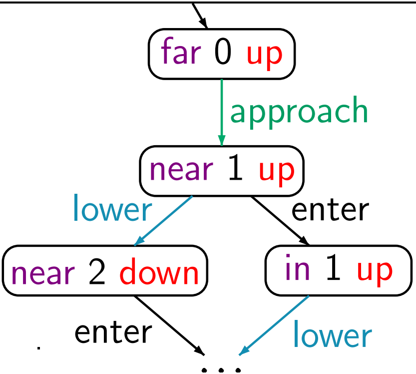
TS for railroad crossing



reachable fragment
of the transition system
Train $\|$ ***Controller*** $\|$ ***Gate***

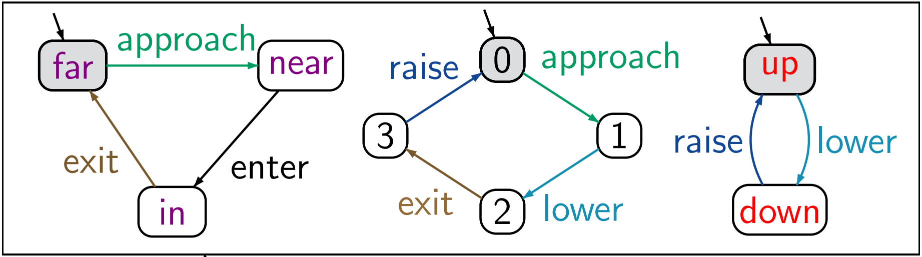
TS for railroad crossing

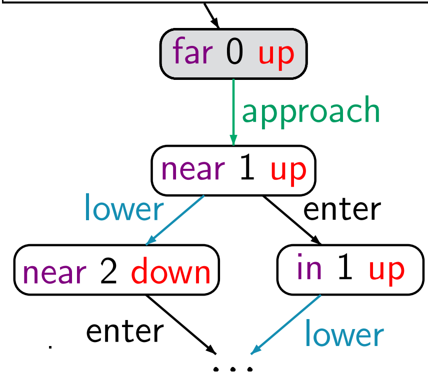




reachable fragment
of the transition system
Train* || *Controller* || *Gate

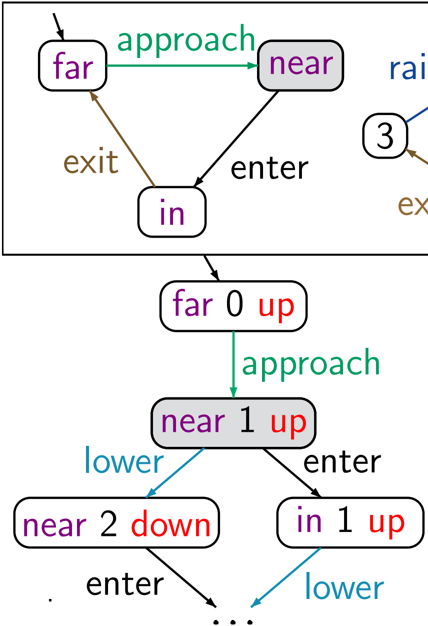
TS for railroad crossing





reachable fragment
of the transition system
Train || ***Controller*** || ***Gate***

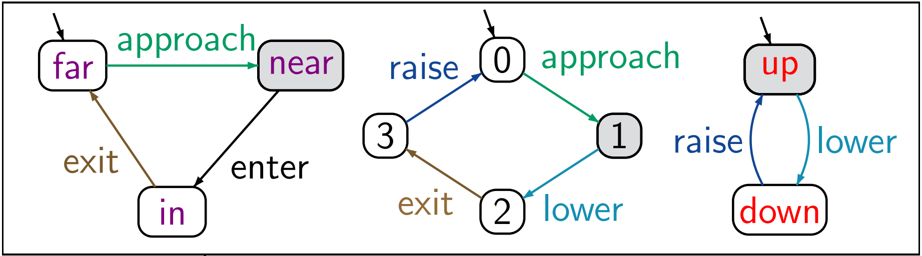
TS for railroad crossing

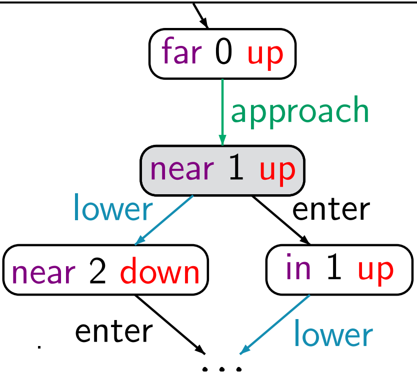


reachable fragment of the transition system

Train* || *Controller* || *Gate

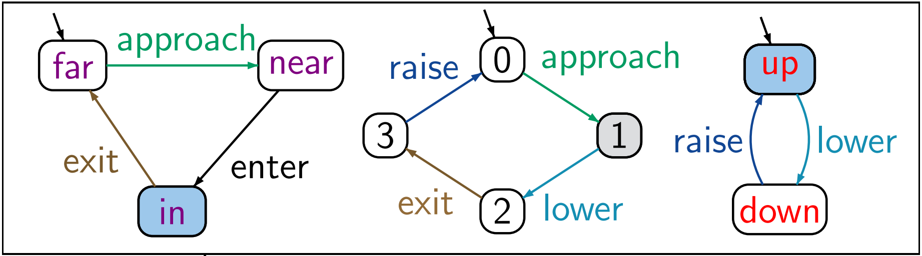
TS for railroad crossing

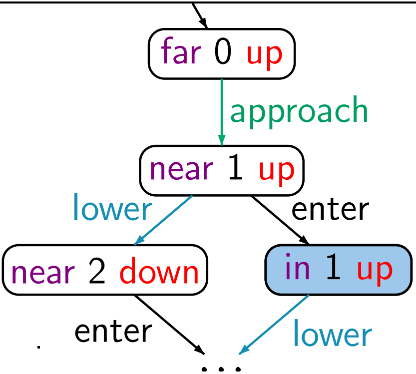




race between train and gate

TS for railroad crossing





gate is open,
while train is
crossing the road

TS for railroad crossing