Lecture 2
Automata Theory

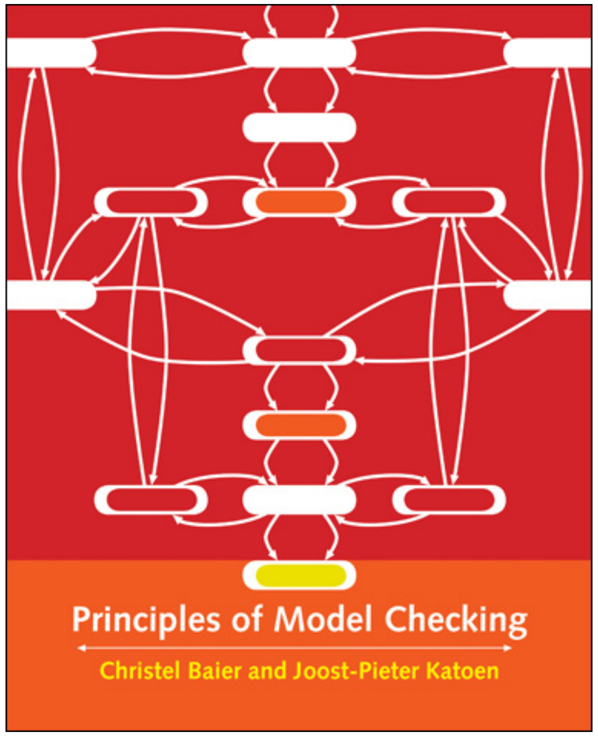
Ufuk Topcu

Nok Wongpiromsarn   Richard M. Murray

EECI, 14 May 2012

Outline

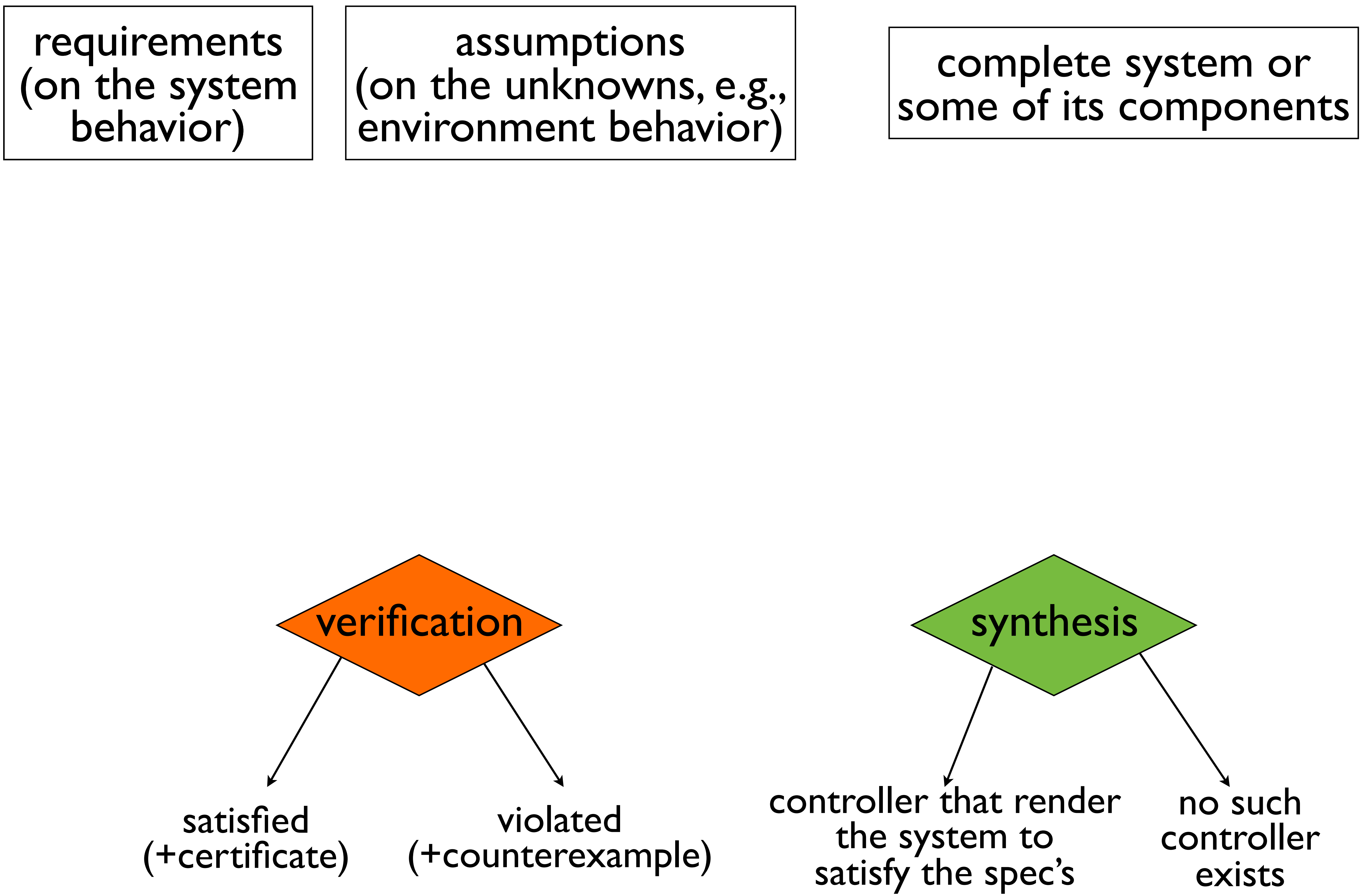
- Modeling (discrete) concurrent systems: transition systems, concurrency and interleaving
- Linear-time properties: invariants, safety and liveness properties



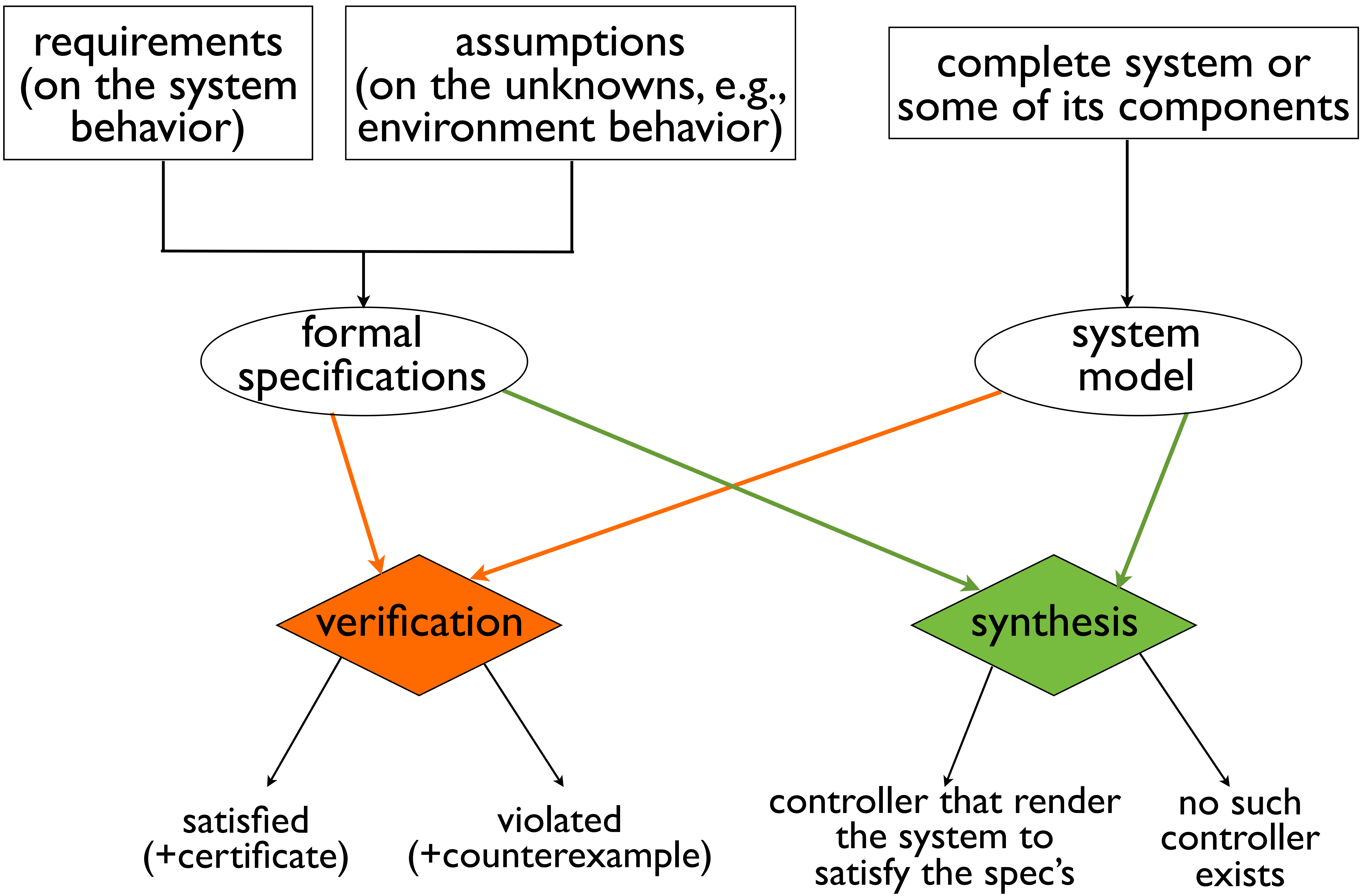
Principles of Model Checking, C. Baier and J.-P. Katoen, The MIT Press, 2008

Chapters 2.1, 2.2, 3.2-3.4

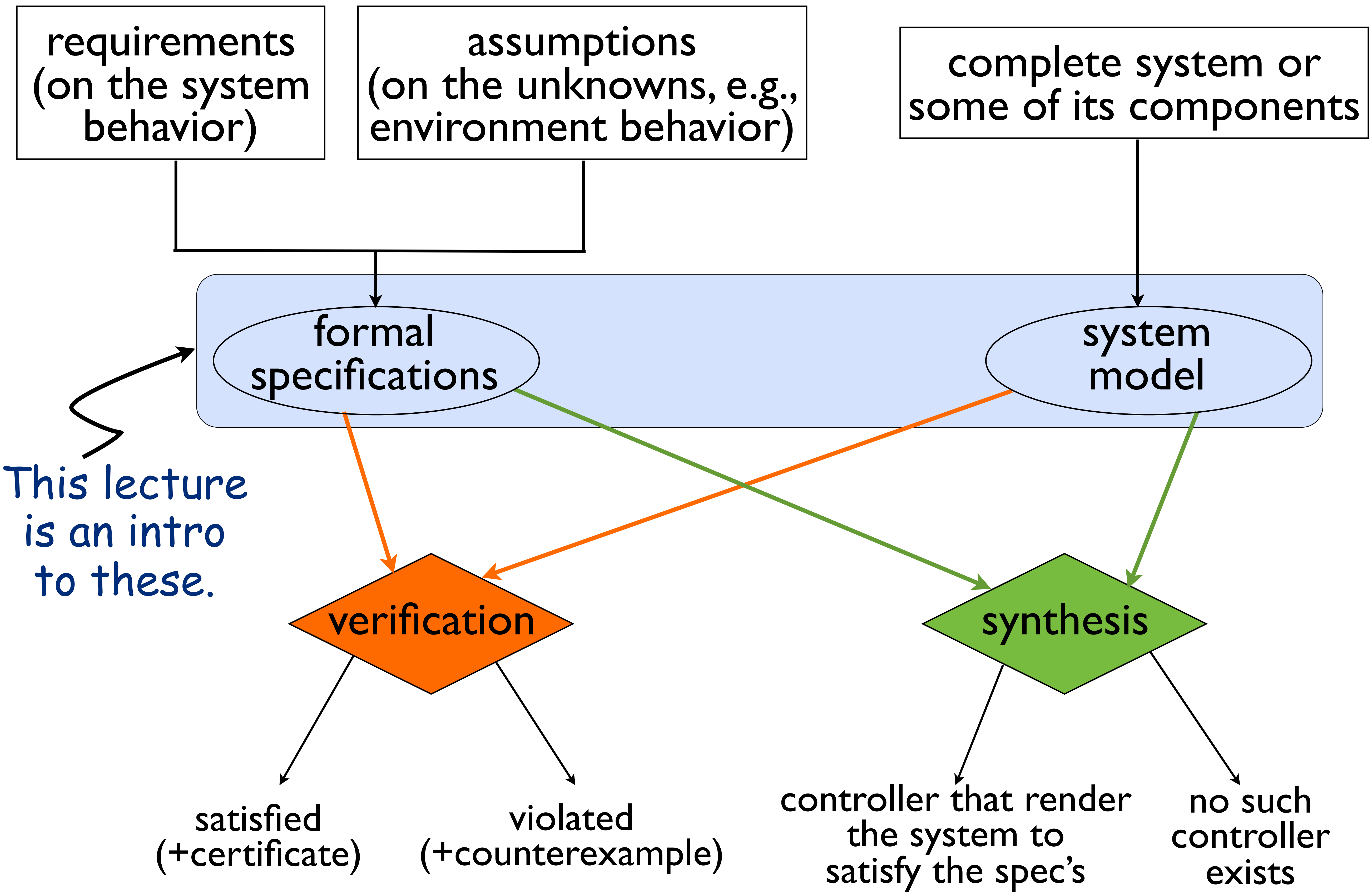
This short-course is on this picture applied to a particular class of systems/problems.



This short-course is on this picture applied to a particular class of systems/problems.



This short-course is on this picture applied to a particular class of systems/problems.



Finite transition system

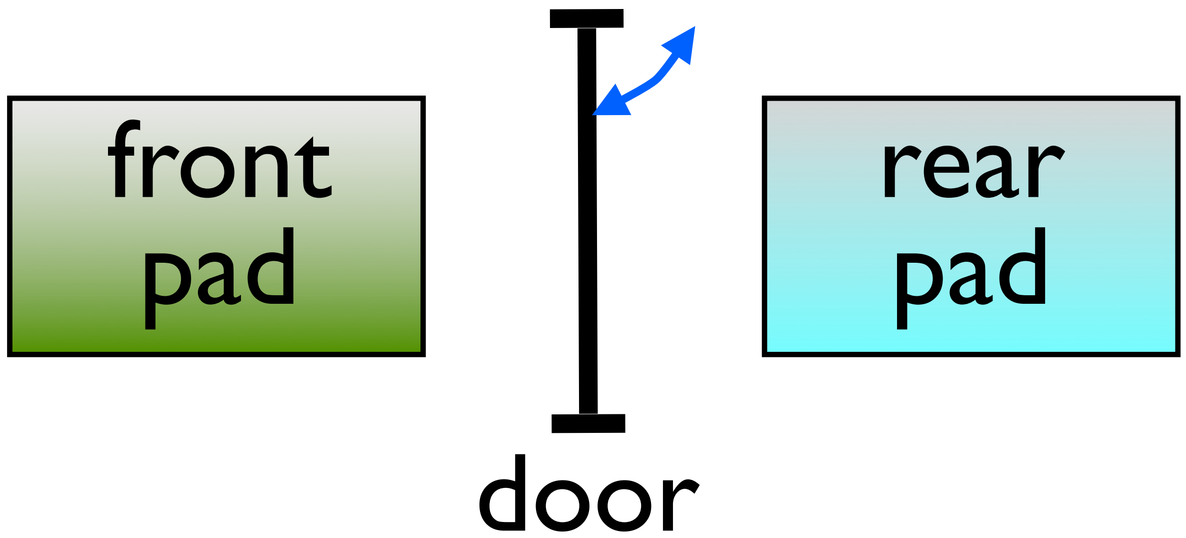
A *finite transition system* is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

- inputs,
- outputs, and
- internal states and transitions between the states.

Finite transition system

A *finite transition system* is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

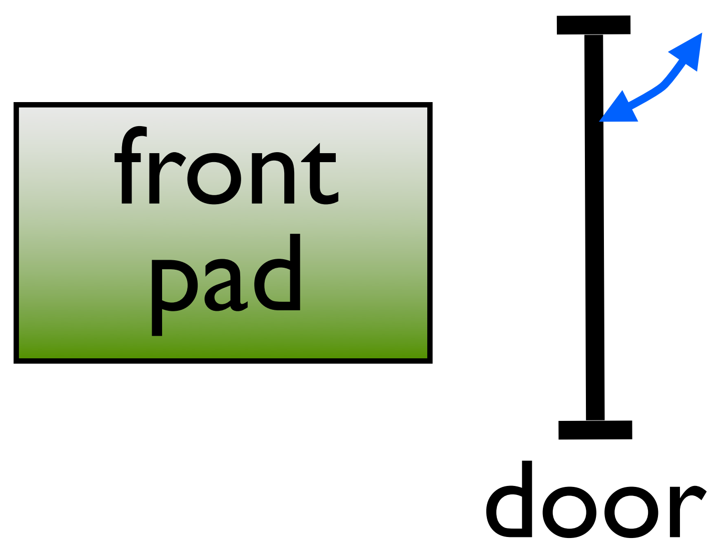
- inputs,
- outputs, and
- internal states and transitions between the states.

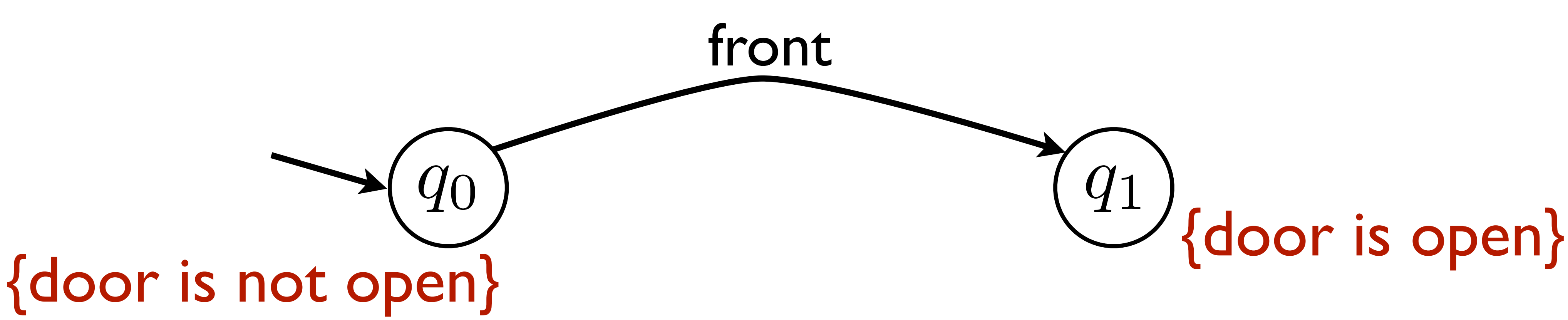


Finite transition system

A *finite transition system* is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

- inputs,
- outputs, and
- internal states and transitions between the states.

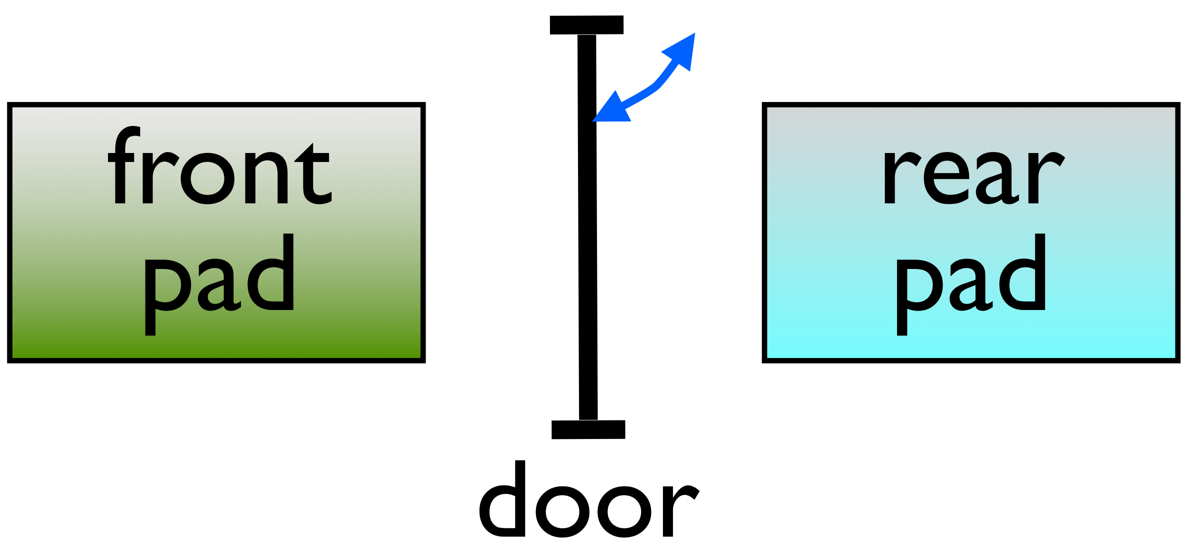


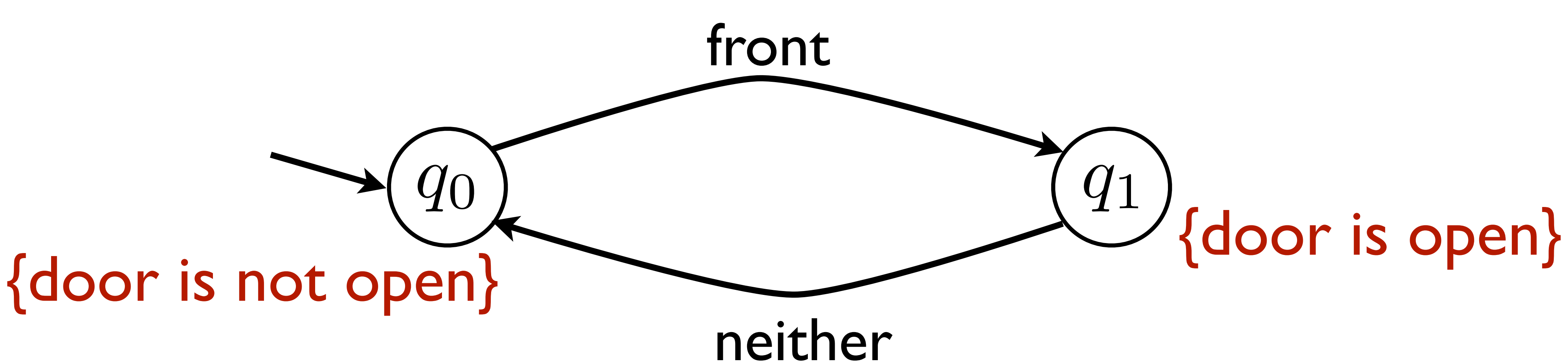


Finite transition system

A *finite transition system* is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

- inputs,
- outputs, and
- internal states and transitions between the states.

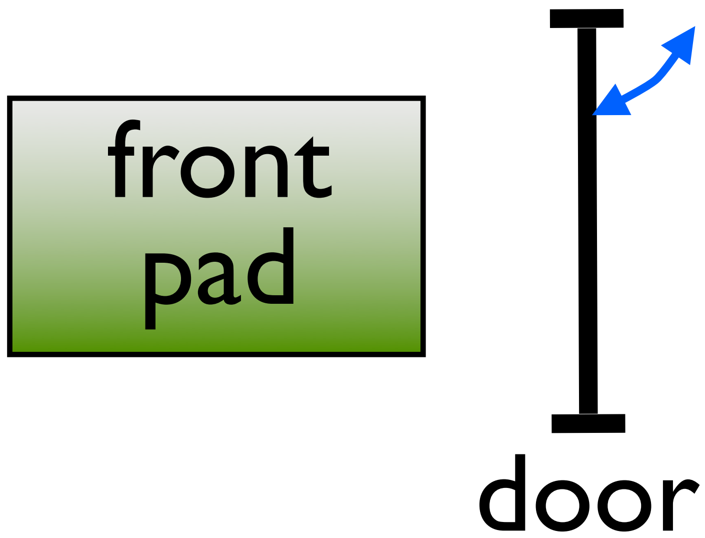


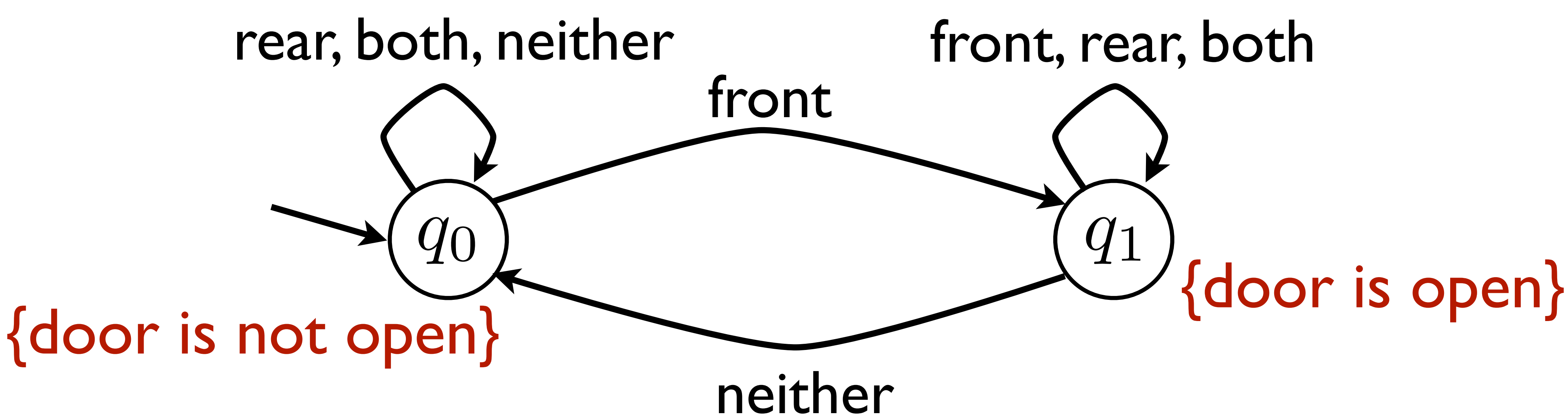


Finite transition system

A *finite transition system* is a mathematical description of the behavior of systems, plants, controllers or environments with finite (discrete)

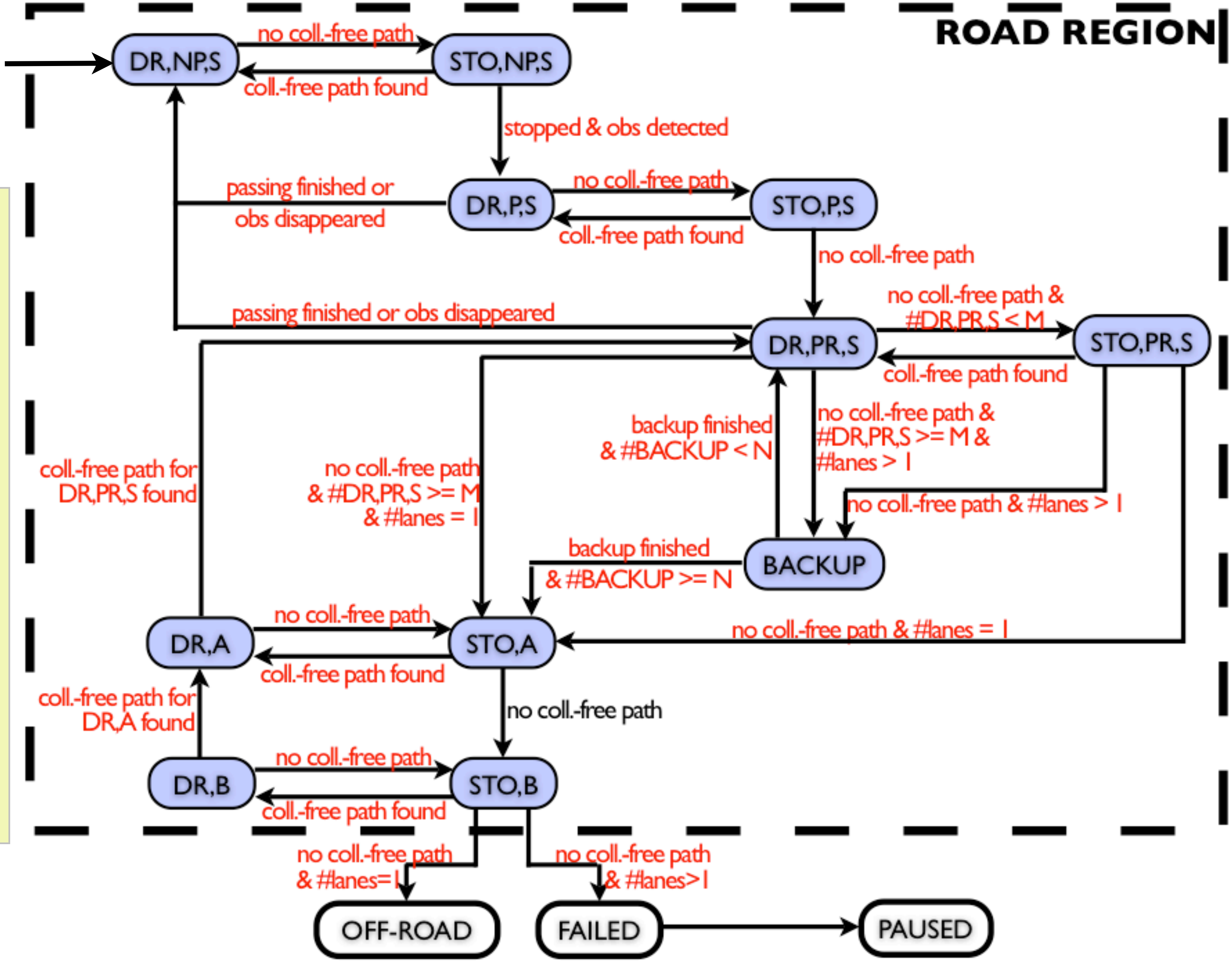
- inputs,
- outputs, and
- internal states and transitions between the states.





Finite transition system

Example: Traffic logic planner in Alice.

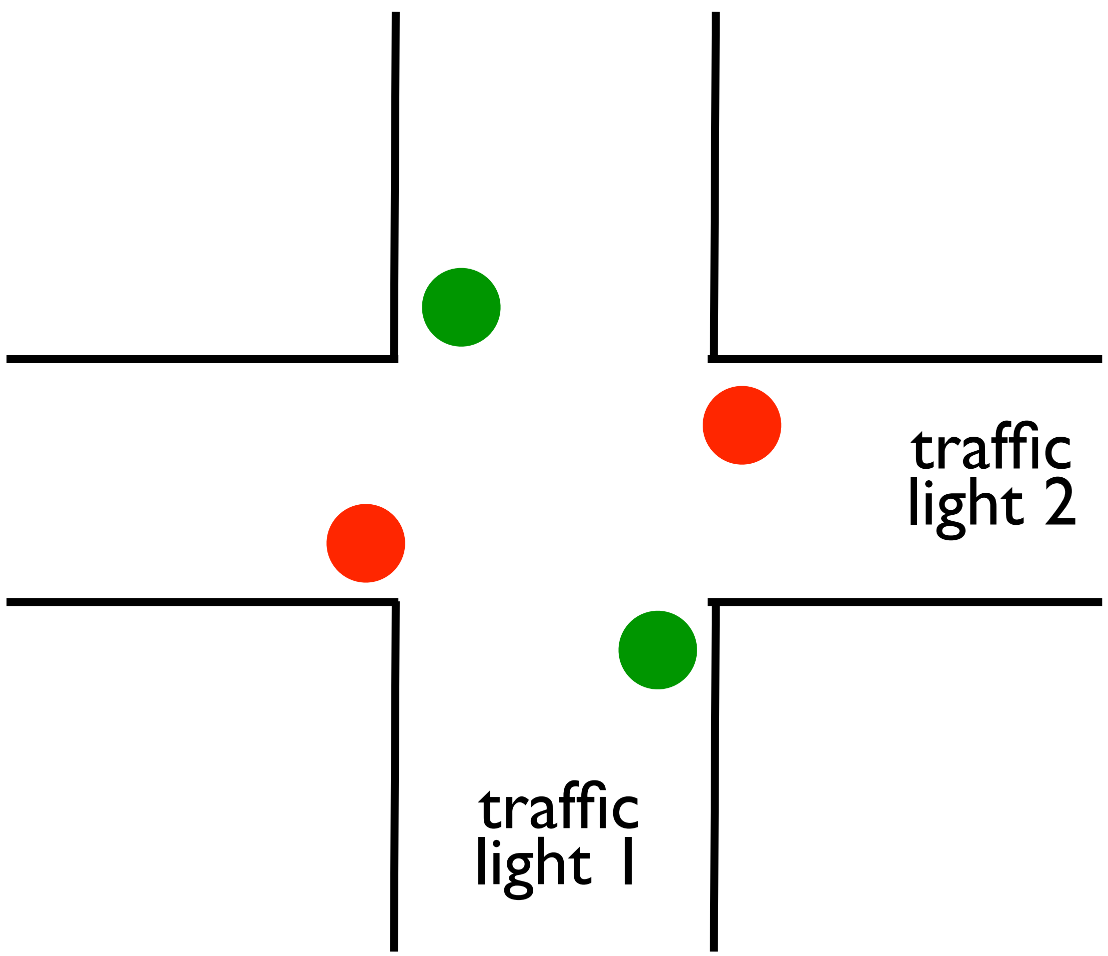


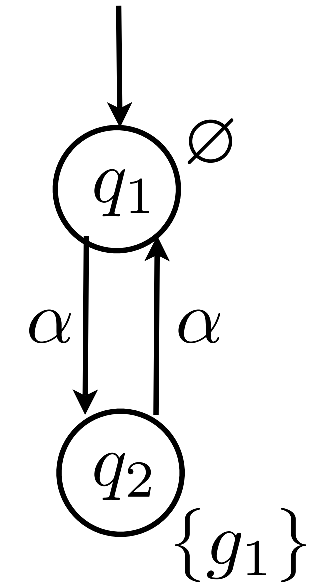
Partial nomenclature:

DR = drive.
STO = stop.
NP = no passing, no reversing.
P = passing, no reversing.
PR = passing, reversing allowed.
S = safe clearance with obstacle.
A = aggressive clearance with obstacle.
B = no clearance with obstacle.

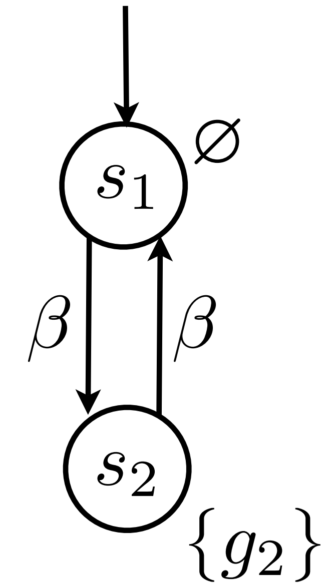
Finite transition system

Example: Traffic lights.

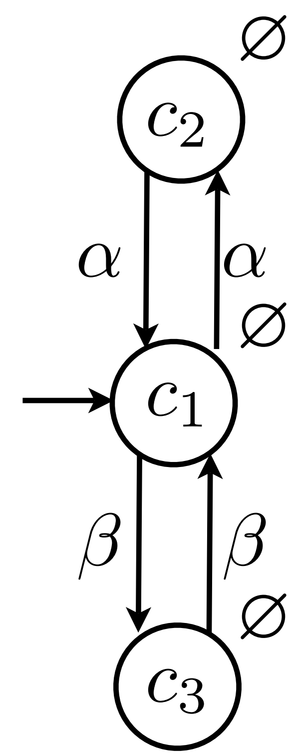




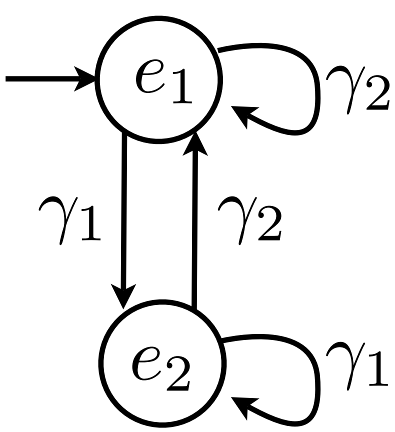
traffic light 1



traffic light 2



controller



environment

Preliminaries

A ***proposition*** is a statement that can be either true or false, but not both.

Examples:

- "Traffic light is green" is a proposition.
- "The front pad is occupied" is a proposition.
- "Is the front pad?" is not a proposition.

Preliminaries

A ***proposition*** is a statement that can be either true or false, but not both.

Examples:

- "Traffic light is green" is a proposition.
- "The front pad is occupied" is a proposition.
- "Is the front pad?" is <u>not</u> a proposition.

An ***atomic proposition*** is one whose truth or falsity does not depend on the truth or falsity of any other proposition.

Examples:

- All propositions above are atomic propositions.
- "If traffic light is green, the car can drive" is <u>not</u> an atomic proposition.

Preliminaries

A ***proposition*** is a statement that can be either true or false, but not both.

Examples:
- "Traffic light is green" is a proposition.
- "The front pad is occupied" is a proposition.
- "Is the front pad?" is <u>not</u> a proposition.

An ***atomic proposition*** is one whose truth or falsity does not depend on the truth or falsity of any other proposition.

Examples:
- All propositions above are atomic propositions.
- "If traffic light is green, the car can drive" is <u>not</u> an atomic proposition.

For notational brevity, use propositional variables to abbreviate propositions. For example,

$$p \equiv \text{Traffic light is green}$$

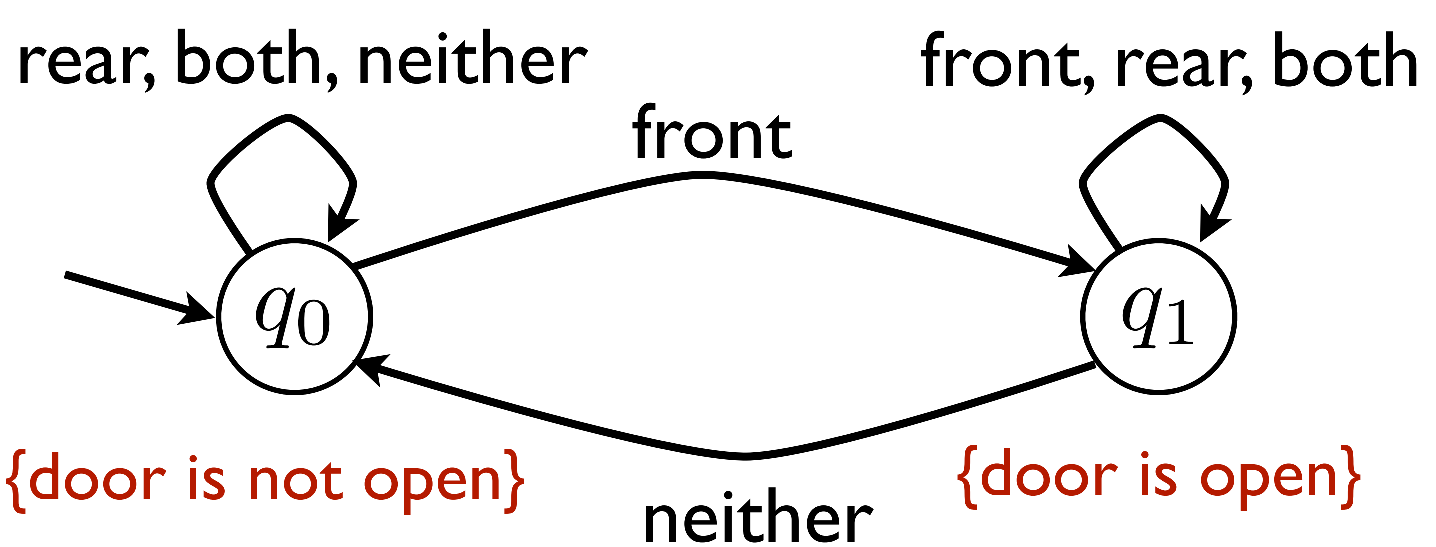
$$q \equiv \text{Front pad is occupied}$$

Finite transition system

A transition system TS is a tuple $TS = (S, Act, \to, I, AP, L)$, where

- S is a set of states,
- Act is a set of actions,
- $\to \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions,
- $L : S \to 2^{AP}$ is a labeling function, and

TS is called finite if S, Act, and AP are finite.



example

$S = \{q_0, q_1\}$
$Act = \{rear, front, both, neither\}$
$\to = \{(q_0, front, q_1), (q_1, neither, q_0),$
$\qquad (q_1, rear, q_1), \ldots\}$
$I = \{q_0\}$
$L(q_0) = \{door\ is\ not\ open\}$
$L(q_1) = \{door\ is\ open\}$

Finite transition system

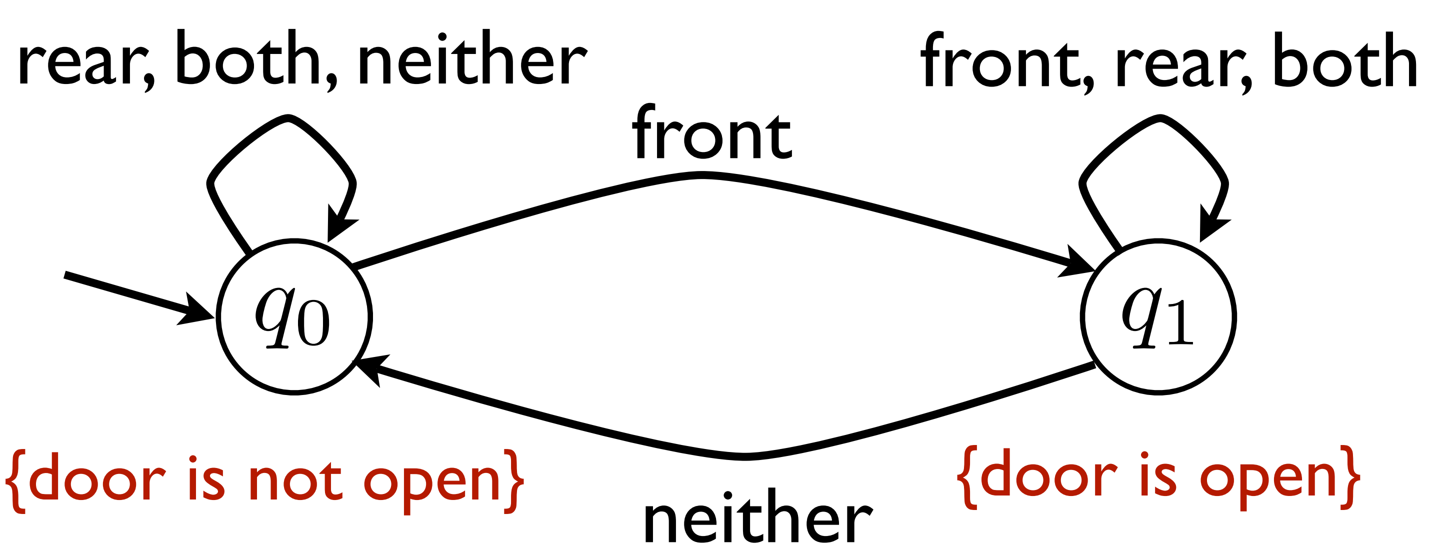
A transition system TS is a tuple $TS = (S, Act, \rightarrow, I, AP, L)$, where

- S is a set of states,
- Act is a set of actions,
- $\rightarrow \subseteq S \times Act \times S$ is a transition relation,
- $I \subseteq S$ is a set of initial states,
- AP is a set of atomic propositions,
- $L : S \rightarrow 2^{AP}$ is a labeling function, and

TS is called finite if S, Act, and AP are finite.

- AP depends on the characteristics of the system of interest.
- For state s, $L(s)$ is the set of atomic propositions that are satisfied at s.
- Labels model outputs or observables.
- Actions model inputs or "communication."

example



$S = \{q_0, q_1\}$
$Act = \{rear, front, both, neither\}$
$\rightarrow = \{(q_0, front, q_1), (q_1, neither, q_0),$
$(q_1, rear, q_1), \ldots\}$
$I = \{q_0\}$
$L(q_0) = \{door\ is\ not\ open\}$
$L(q_1) = \{door\ is\ open\}$

Propositional logic

From "Specifying Systems" by L. Lamport: Propositional logic is the math of the Boolean values, true and false, and the operators $\neg, \wedge, \vee, \rightarrow$

Given finite set AP of atomic propositions, the set of propositional logic formulas is inductively defined by:

- true is a formula;
- any $a \in AP$ is a formula;
- if ϕ_1, ϕ_2, and ϕ are formulas, so are $\neg\phi$ and $\phi_1 \wedge \phi_2$; and
- nothing else is a formula.

Propositional logic

From "Specifying Systems" by L. Lamport: Propositional logic is the math of the Boolean values, true and false, and the operators $\neg, \wedge, \vee, \rightarrow$

Given finite set AP of atomic propositions, the set of propositional logic formulas is inductively defined by:

- true is a formula;
- any $a \in AP$ is a formula;
- if ϕ_1, ϕ_2, and ϕ are formulas, so are $\neg\phi$ and $\phi_1 \wedge \phi_2$; and
- nothing else is a formula.

<u>Notation</u>

- Connectives:

$\neg$ (negation), $\wedge$ (and)

$\vee$ (or), $\rightarrow$ (implies)

- 1 for "true" and 0 for "false."

Example propositional logic formulas obtained by applying the above four rules:

$$\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$$

$$\phi_1 \rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$$

Propositional logic

From "Specifying Systems" by L. Lamport: Propositional logic is the math of the Boolean values, true and false, and the operators $\neg, \wedge, \vee, \rightarrow$

Given finite set AP of atomic propositions, the set of propositional logic formulas is inductively defined by:

- true is a formula;
- any $a \in AP$ is a formula;
- if ϕ_1, ϕ_2, and ϕ are formulas, so are $\neg\phi$ and $\phi_1 \wedge \phi_2$; and
- nothing else is a formula.

<u>Notation</u>

•Connectives:

$\neg$ (negation), $\wedge$ (and)

$\vee$ (or), $\rightarrow$ (implies)

•1 for "true" and 0 for "false."

Example propositional logic formulas obtained by applying the above four rules:

$$\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$$

$$\phi_1 \rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$$

The *evaluation function* $\mu : AP \rightarrow \{0, 1\}$ assigns a truth value to each $a \in AP$.

The truth value $\mu(\Phi)$ of a formula Φ is determined by substituting the values for the atomic propositions specified by μ.

Given: $AP = \{a, b, c\}$, $\mu(a) = 0$ and $\mu(b) = \mu(c) = 1$.

$$\Phi_1 = (a \wedge \neg b) \vee c, \quad \mu(\Phi_1) = 1$$

$$\Phi_2 = (a \wedge \neg b) \wedge c, \quad \mu(\Phi_2) = 0$$

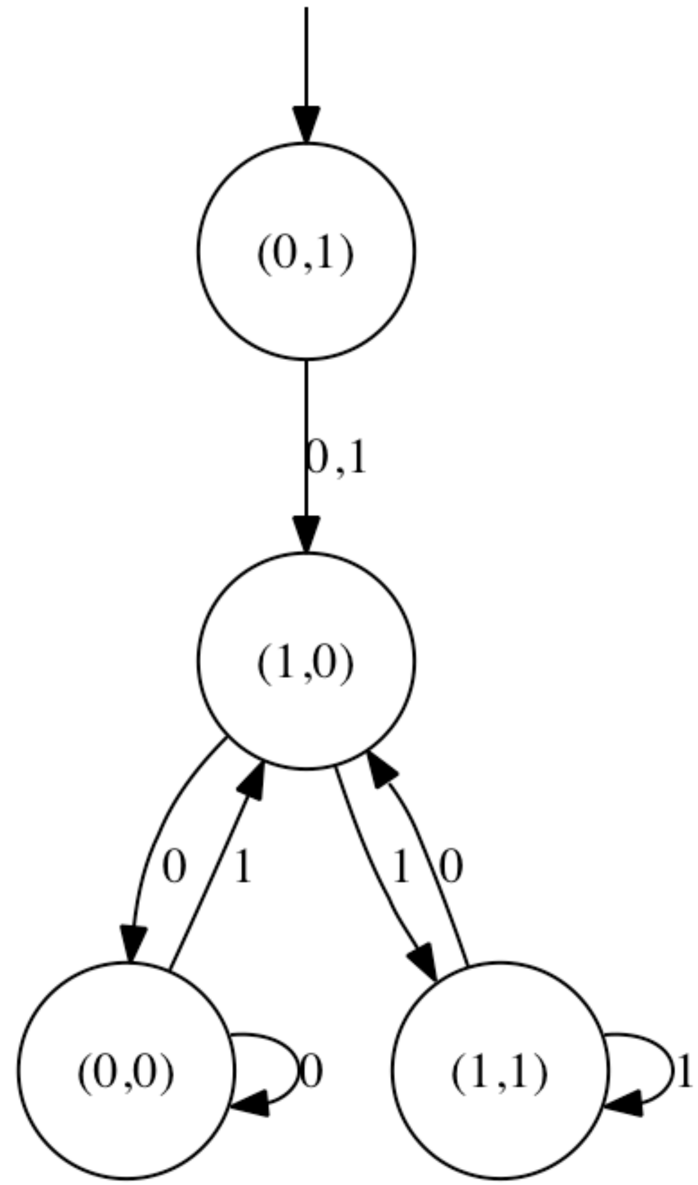
Logical dynamical system as a finite transition system

$$x_1[k+1] = x_2[k] \vee u[k], \qquad x_1[0] = 0,$$
$$x_2[k+1] = x_1[k] \wedge u[k], \qquad x_2[0] = 1,$$
$$y[k] = x_1[k] \oplus x_2[k]$$

$\phi_1 \oplus \phi_2 := (\neg\phi_1 \wedge \phi_2) \vee (\phi_1 \wedge \neg\phi_2)$

XOR (exclusive or) gives true only if exactly one of the operands is true.

$$S = \{0, 1\}^2$$
$$Act = \{0, 1\}$$
$$I = \{(0, 1)\}$$
$$AP = \{y\}$$
$$L(x_1, x_2) = \begin{cases} \{y\} \text{ (indicating 1) if } x_1 \oplus x_2 = 1 \\ \emptyset \text{ (indicating 0) otherwise} \end{cases}$$



Concurrent systems

Systems in which multiple tasks can be executed at the same time potentially with inter-task communication and resource sharing.

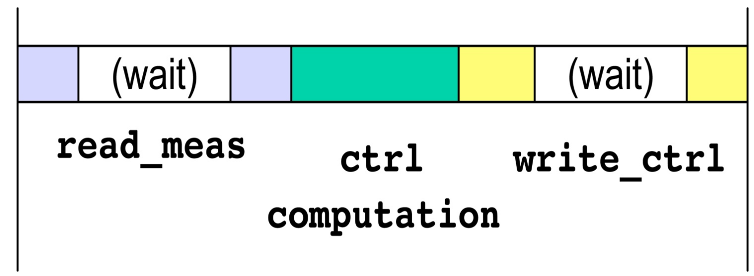
Concurrent systems

Systems in which multiple tasks can be executed at the same time potentially with inter-task communication and resource sharing.

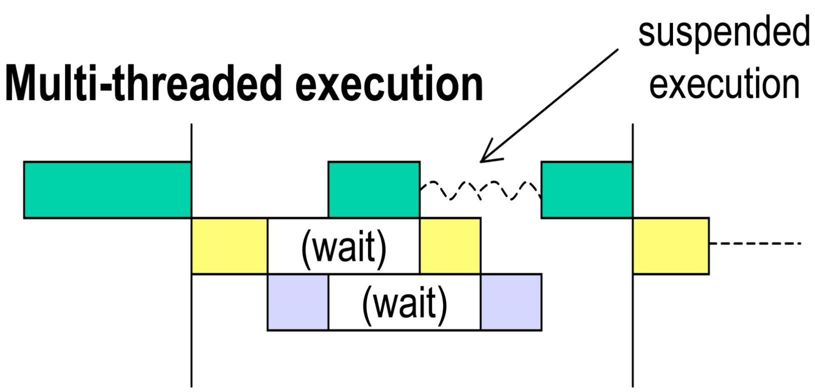
<u>Example</u>: multi-threaded control

- Separate code into independent threads
- Switch between threads, allowing each to run simultaneously
- Potential problems: deadlocks, race conditions

Single threaded execution



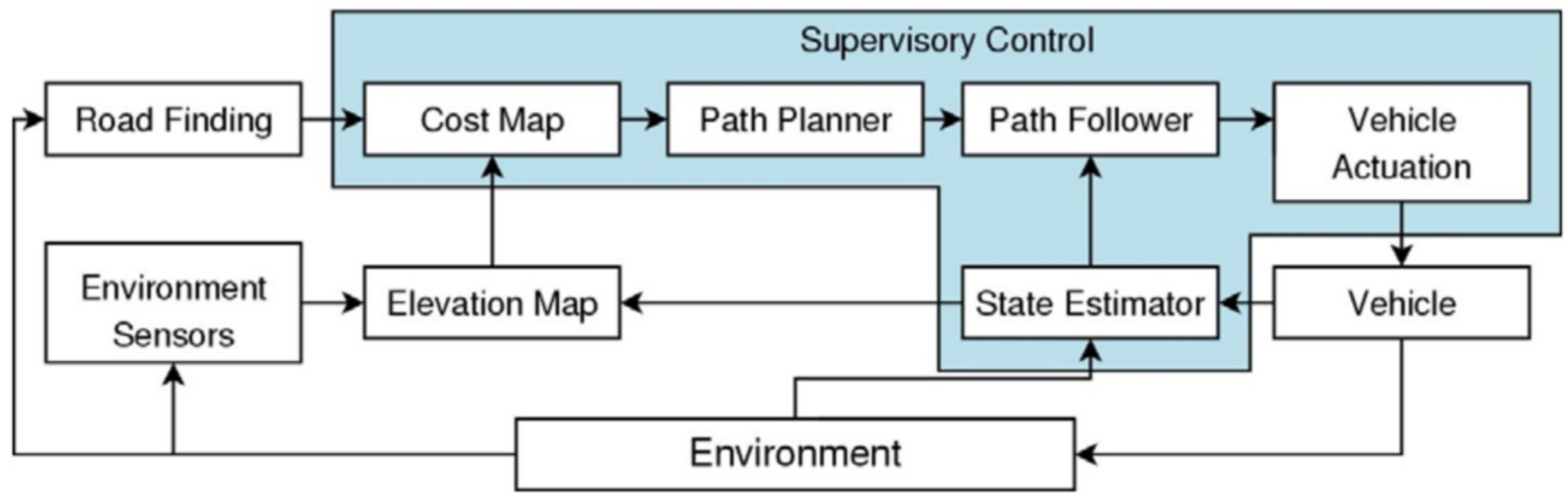
Multi-threaded execution



Modes of communication between the subsystems:

- hand-shaking (leads to synchrony)
- changing the values of shared variables (leads to asynchrony)

Thread Usage in Alice (DGC05)



Module	Threads
adrive (actuation)	19
trajFollower	10
astate (state estimator)	10
plannerModule	4
fusionMapper	16

Module	Threads
ladarFeeder (5)	8
stereoFeeder (2)	7
road (road follower)	5
superCon	3
DBS	3

* doesn't count heartbeat and logging threads

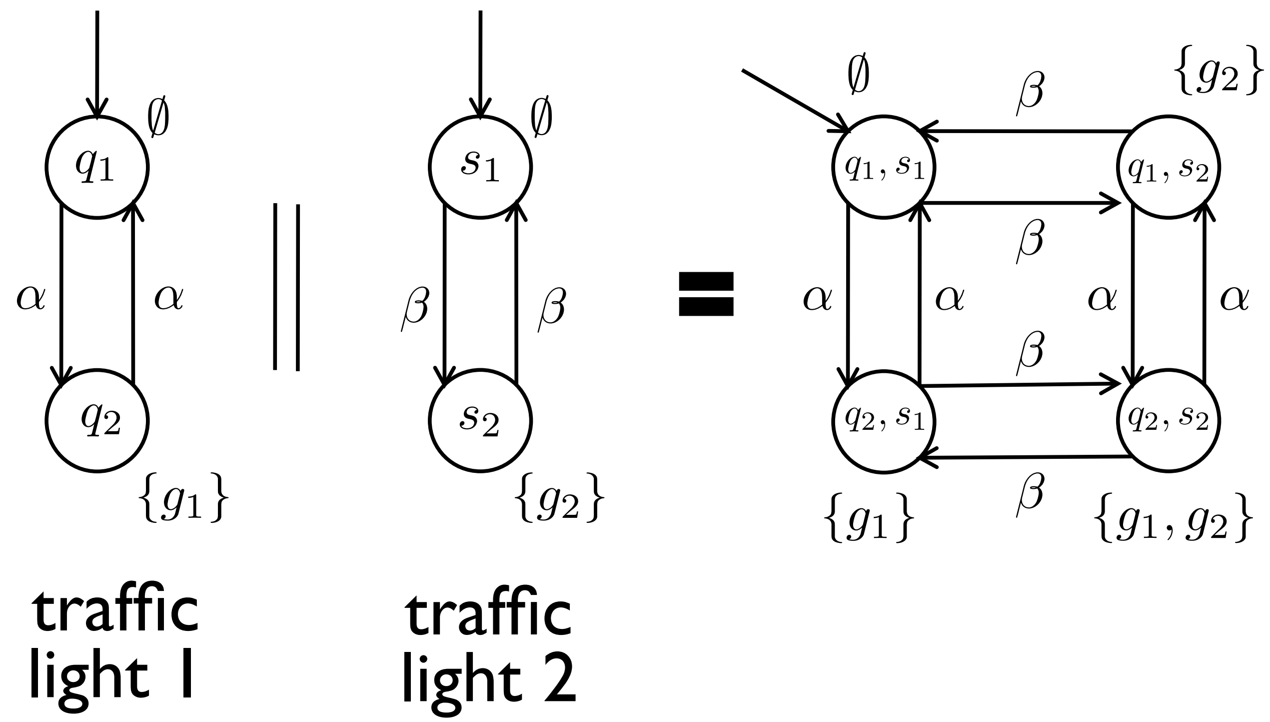
Composition of transition systems (by handshaking)

Let $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1 || TS_2$ is the transition system defined by

$$TS_1 || TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rules:

- If $\alpha \in Act_1 \cap Act_2$, $s_1 \xrightarrow{\alpha}_1 s_1'$, and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle$.
- If $\alpha \in Act_1 \setminus Act_2$ and $s_1 \xrightarrow{\alpha}_1 s_1'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$.
- If $\alpha \in Act_2 \setminus Act_1$ and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$.



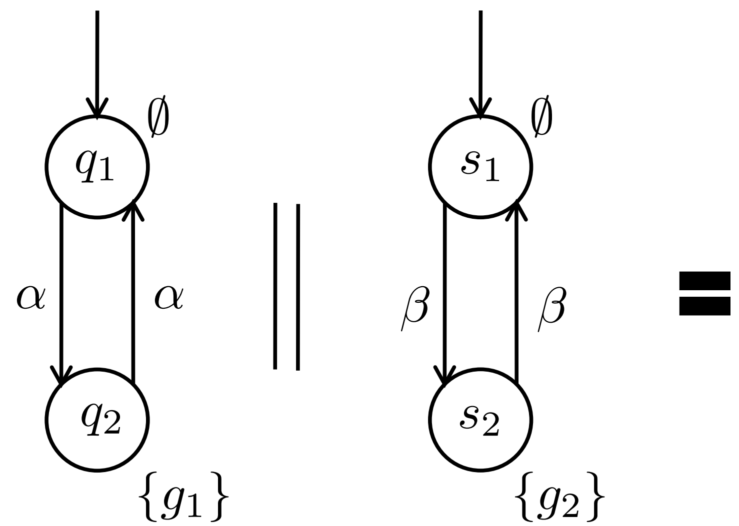
Composition of transition systems (by handshaking)

Let $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1 || TS_2$ is the transition system defined by

$$TS_1 || TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rules:

- If $\alpha \in Act_1 \cap Act_2$, $s_1 \xrightarrow{\alpha}_1 s_1'$, and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle$.
- If $\alpha \in Act_1 \setminus Act_2$ and $s_1 \xrightarrow{\alpha}_1 s_1'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$.
- If $\alpha \in Act_2 \setminus Act_1$ and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$.



traffic light 1

traffic light 2

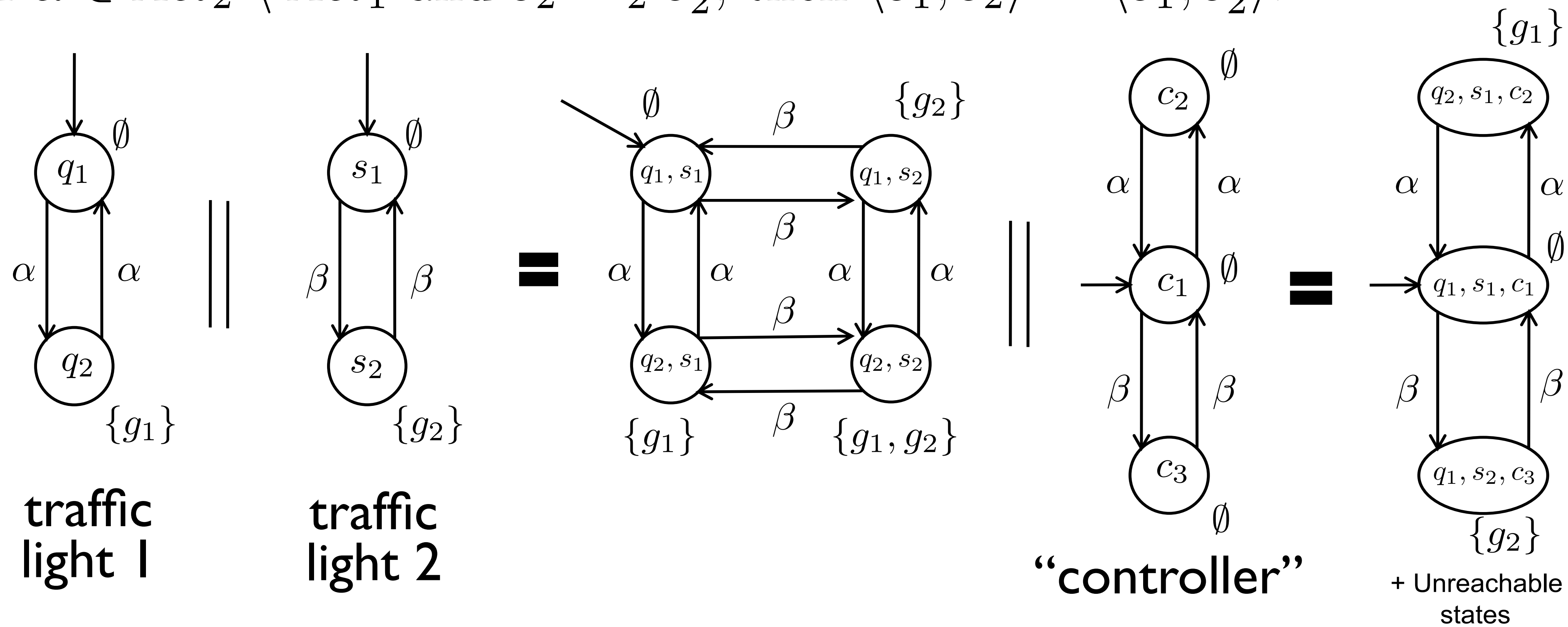
Composition of transition systems (by handshaking)

Let $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1)$ and $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2)$ be transition systems. Their parallel composition, $TS_1 || TS_2$ is the transition system defined by

$$TS_1 || TS_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, I_1 \times I_2, AP_1 \cup AP_2, L)$$

where $L(\langle s_1, s_2 \rangle) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by the following rules:

- If $\alpha \in Act_1 \cap Act_2$, $s_1 \xrightarrow{\alpha}_1 s_1'$, and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle$.
- If $\alpha \in Act_1 \setminus Act_2$ and $s_1 \xrightarrow{\alpha}_1 s_1'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$.
- If $\alpha \in Act_2 \setminus Act_1$ and $s_2 \xrightarrow{\alpha}_2 s_2'$, then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$.

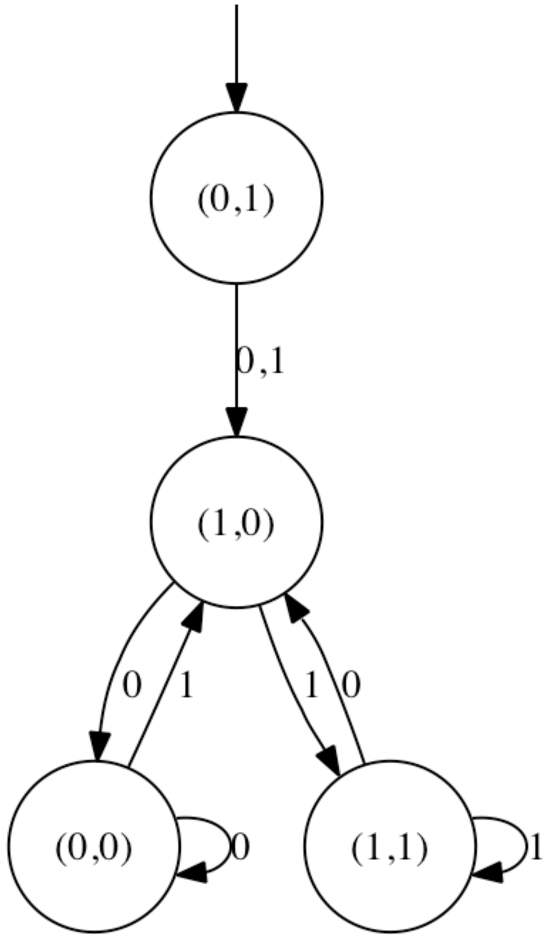


Paths of a finite transition system

Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$.
For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state s is *terminal* iff $Post(s)$ is empty.

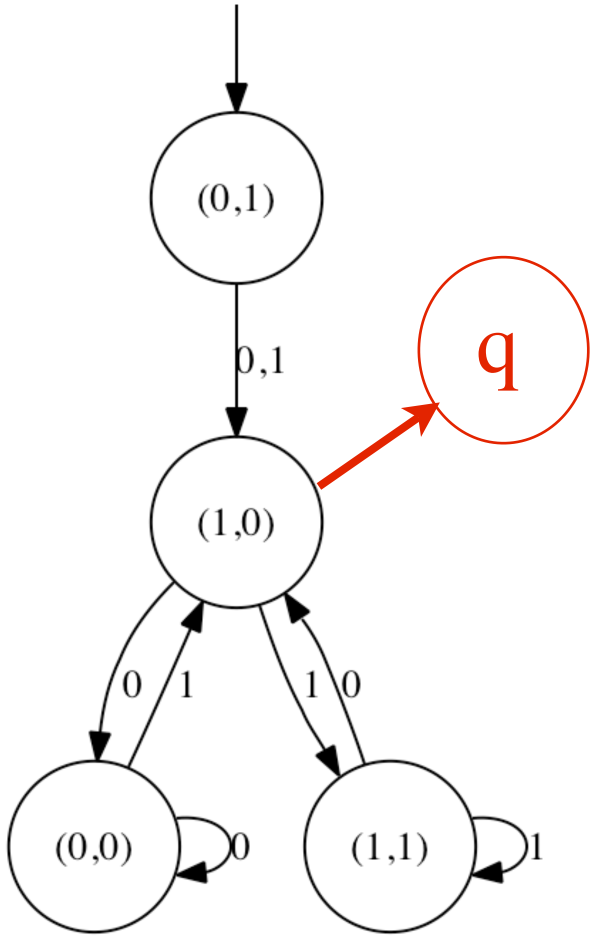


Paths of a finite transition system

Given a transition system $TS = (S, Act, \to, I, AP, L)$.
For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state s is *terminal* iff $Post(s)$ is empty.



Paths of a finite transition system

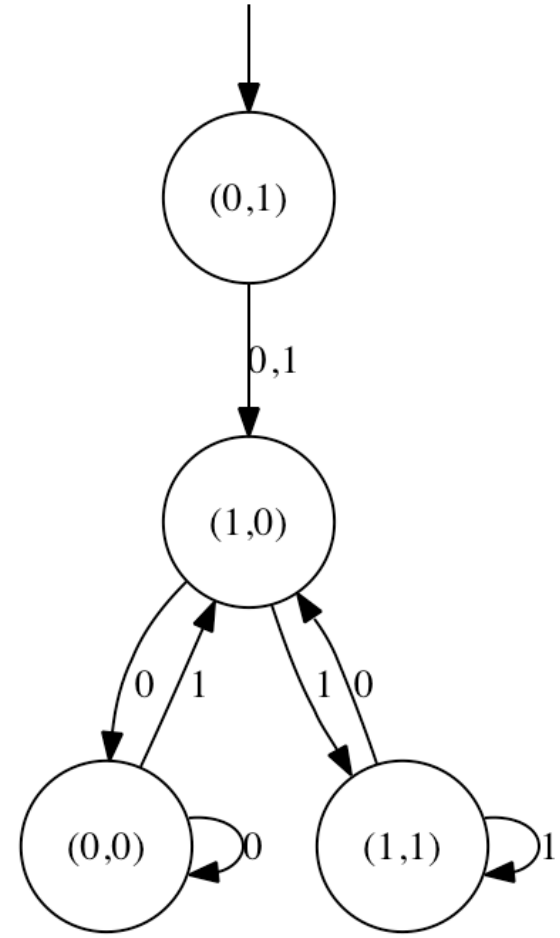
Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$.
For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.

- A state s is *terminal* iff $Post(s)$ is empty.

- A sequence of states, either finite $\pi = s_0 s_1 s_2 \ldots s_n$ or infinite $\pi = s_0 s_1 s_2 \ldots$, is a *path fragment* if $s_{i+1} \in Post(s_i), \; \forall i \geq 0$.



$$(0,1) \xrightarrow{,1} (1,0) \xrightarrow{1} (1,1) \xrightarrow{1} (1,1) \xrightarrow{0} \cdots$$

$$(1,0) \xrightarrow{0} (0,0) \xrightarrow{0} (0,0) \xrightarrow{1} (1,0) \xrightarrow{0} \cdots$$

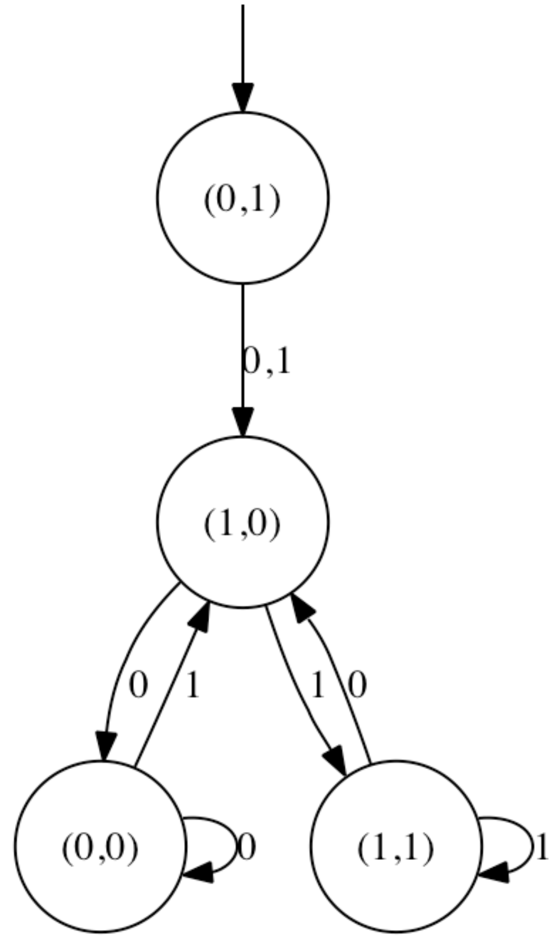
$$(0,1) \xrightarrow{,1} (1,0) \xrightarrow{1} (1,1).$$

Paths of a finite transition system

Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$.
For $s \in S$,

$$Post(s) := \left\{ s' \in S : \exists a \in Act \text{ s.t. } s \xrightarrow{a} s' \right\}$$

- Example: $Post((0,0)) = \{(0,0),(1,0)\}$.
- A state s is *terminal* iff $Post(s)$ is empty.
- A sequence of states, either finite $\pi = s_0 s_1 s_2 \ldots s_n$ or infinite $\pi = s_0 s_1 s_2 \ldots$, is a *path fragment* if $s_{i+1} \in Post(s_i), \; \forall i \geq 0$.



- A *path* is a path fragment s.t. $s_0 \in I$ and it is
  - either finite with terminal s_n
  - or infinite.
- Denote the set of paths in *TS* by $Path(TS)$.

a path:

$$(0,1) \xrightarrow{,1} (1,0) \xrightarrow{1} (1,1) \xrightarrow{1} (1,1) \xrightarrow{0} \cdots$$

not a path:

$$(1,0) \xrightarrow{0} (0,0) \xrightarrow{0} (0,0) \xrightarrow{1} (1,0) \xrightarrow{0} \cdots$$

not a path:

$$(0,1) \xrightarrow{,1} (1,0) \xrightarrow{1} (1,1).$$

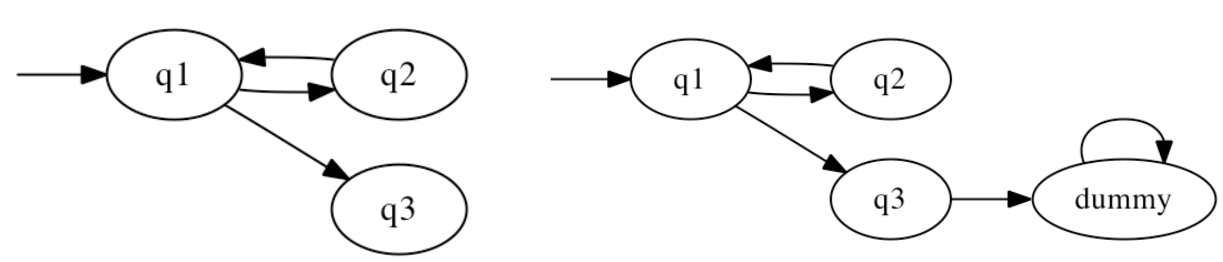
Traces of a finite transition system

Consider a finite transition system

$$TS = (S, Act, \rightarrow, I, AP, L)$$

with no terminal states (wlog).

Equivalent FSMs w/ and w/o terminal state



The *trace* of an infinite path fragment $\pi = s_0 s_1 s_2 \ldots$ is defined by

$$trace(\pi) = L(s_0)L(s_1)L(s_2)\ldots$$

sequence of sets of atomic propositions that are valid in the states along the path

The set, $Traces(TS)$, of traces of TS is defined by

$$Traces(TS) = \{trace(\pi) : \pi \in Paths(TS)\}$$

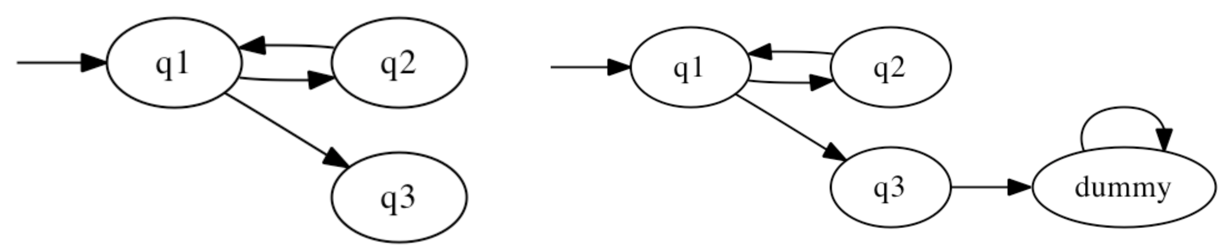
Traces of a finite transition system

Consider a finite transition system

$$TS = (S, Act, \rightarrow, I, AP, L)$$

with no terminal states (wlog).

Equivalent FSMs w/ and w/o terminal state



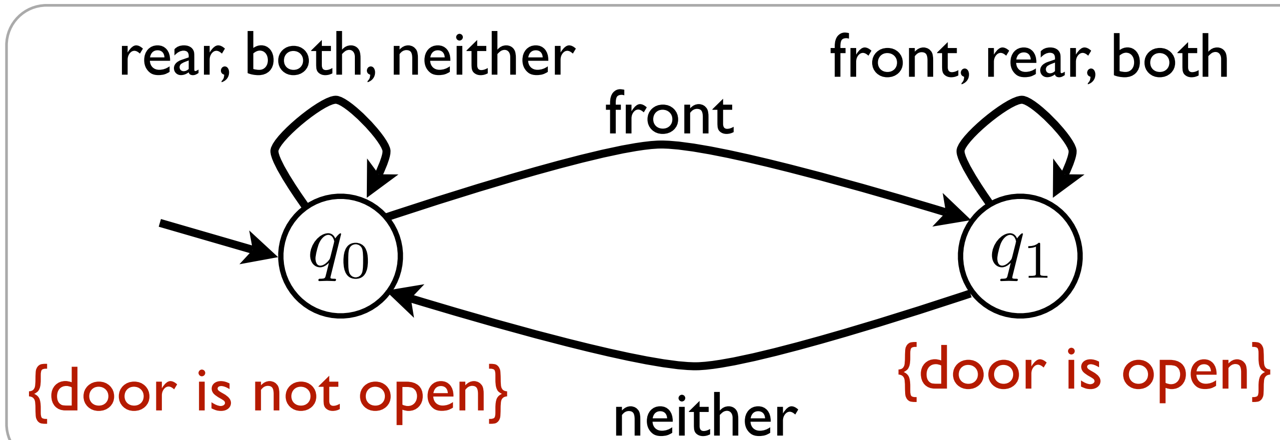
The *trace* of an infinite path fragment $\pi = s_0 s_1 s_2 \ldots$ is defined by

$$trace(\pi) = L(s_0)L(s_1)L(s_2)\ldots$$

sequence of sets of atomic propositions that are valid in the states along the path

The set, $Traces(TS)$, of traces of TS is defined by

$$Traces(TS) = \{trace(\pi) : \pi \in Paths(TS)\}$$



Actions: $f, f, n, b, f, f, b, \ldots$

Path: $q_0 q_1 q_1 q_0 q_0 q_1 q_1 q_1 \ldots$

Trace: $\neg o, o, o, \neg o, \neg o, o, o, o, \ldots$

(with some abuse of notation)

Linear-time properties

A linear-time (LT) property P over atomic propositions in AP is a set of infinite sequences over 2^{AP}.

Let P be an LT property over AP and $TS = (S, Act, \to, I, AP, L)$ be a transition system.

TS satisfies P, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$.

Linear-time properties

A linear-time (LT) property P over atomic propositions in AP is a set of infinite sequences over 2^{AP}.

Let P be an LT property over AP and $TS = (S, Act, \to, I, AP, L)$ be a transition system.

TS satisfies P, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$.

traces of TS

admissible, desired, undesired, etc. behavior

Linear-time properties

A linear-time (LT) property P over atomic propositions in AP is a set of infinite sequences over 2^{AP}.

Let P be an LT property over AP and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

TS satisfies P, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$.

traces of *TS*

admissible, desired, undesired, etc. behavior

<u>Example</u>: $AP = \{red1, green1, red2, green2\}$

P1 = "The first light is infinitely often green."

[$A_0 A_1 A_2 \ldots$ with $green1 \in A_i \subseteq 2^{AP}$ holds for infinitely many i]

√ $\{r1, g2\}\{g1, r2\}\{r1, g2\}\{g1, r2\}\ldots$
√ $\emptyset\{g1\}\emptyset\{g1\}\emptyset\{g1\}\emptyset\ldots$
√ $\{g1, g2\}\{g1, g2\}\{g1, g2\}\ldots$
× $\{r1, g2\}\{r1g1\}\emptyset\emptyset\ldots$

P2 = "The lights are never both green simultaneously."

[$A_0 A_1 A_2 \ldots$ with $green1 \notin A_i$ or $green2 \notin A_i$, for all $i \geq 0$]

Linear-time properties

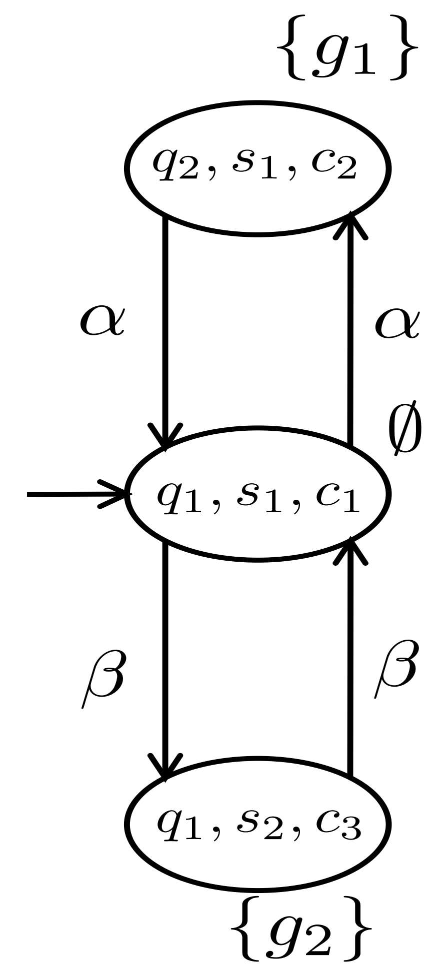
A linear-time (LT) property P over atomic propositions in AP is a set of infinite sequences over 2^{AP}.

Let P be an LT property over AP and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

TS satisfies P, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$.

traces of *TS*

admissible, desired, undesired, etc. behavior



+ Unreachable states

<u>Example</u>: $AP = \{red1, green1, red2, green2\}$

P1 = "The first light is infinitely often green."

$[A_0 A_1 A_2 \ldots$ with $green1 \in A_i \subseteq 2^{AP}$ holds for infinitely many $i]$

√ $\{r1, g2\}\{g1, r2\}\{r1, g2\}\{g1, r2\} \ldots$
√ $\emptyset\{g1\}\emptyset\{g1\}\emptyset\{g1\}\emptyset \ldots$
√ $\{g1, g2\}\{g1, g2\}\{g1, g2\} \ldots$
× $\{r1, g2\}\{r1 g1\}\emptyset\emptyset \ldots$

P2 = "The lights are never both green simultaneously."

$[A_0 A_1 A_2 \ldots$ with $green1 \notin A_i$ or $green2 \notin A_i$, for all $i \geq 0]$

Linear-time properties

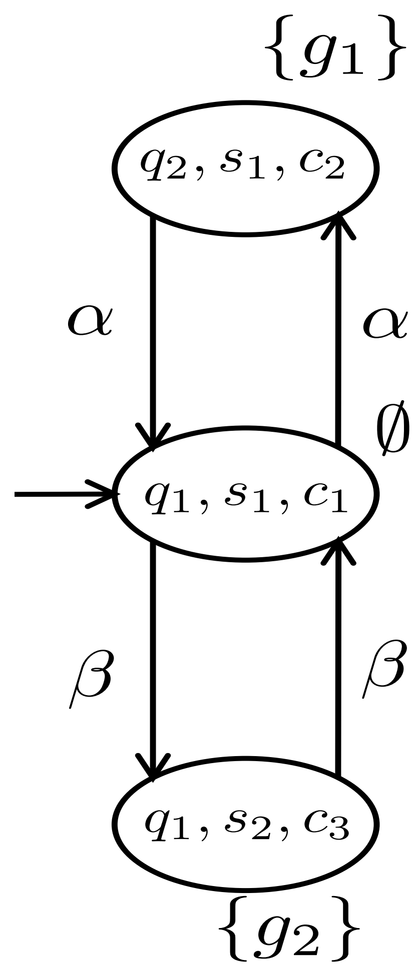
A linear-time (LT) property P over atomic propositions in AP is a set of infinite sequences over 2^{AP}.

Let P be an LT property over AP and $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

TS satisfies P, denoted as $TS \models P$, iff $Traces(TS) \subseteq P$.

traces of TS

admissible, desired, undesired, etc. behavior



<u>Example</u>: $AP = \{red1, green1, red2, green2\}$

P1 = "The first light is infinitely often green."

[$A_0 A_1 A_2 \ldots$ with $green1 \in A_i \subseteq 2^{AP}$ holds for infinitely many i]

√ $\{r1, g2\}\{g1, r2\}\{r1, g2\}\{g1, r2\} \ldots$
√ $\emptyset\{g1\}\emptyset\{g1\}\emptyset\{g1\}\emptyset \ldots$
√ $\{g1, g2\}\{g1, g2\}\{g1, g2\} \ldots$
× $\{r1, g2\}\{r1g1\}\emptyset\emptyset \ldots$

P2 = "The lights are never both green simultaneously."

[$A_0 A_1 A_2 \ldots$ with $green1 \notin A_i$ or $green2 \notin A_i$, for all $i \geq 0$]

The transition system satisfies P2, but it does not satisfy P1.

Invariants

An LT property P_Φ over AP is an *invariant* with respect to a propositional logic formula Φ over AP if

$$P_\Phi = \{A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \ : \ A_j \models \Phi \ \forall j \geq 0\}.$$

Invariants

Notation: repeat infinitely many times

An LT property P_Φ over AP is an *invariant* with respect to a propositional logic formula Φ over AP if

$$P_\Phi = \{A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \ : \ A_j \models \Phi \ \forall j \geq 0\}.$$

For $A \subseteq AP$, let the evaluation μ_A be the characteristic function of A.

$A \models \Phi$ iff $\mu_A(\Phi) = 1$

Invariants

Notation: repeat infinitely many times

An LT property P_Φ over AP is an *invariant* with respect to a propositional logic formula Φ over AP if

$$P_\Phi = \{A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \; : \; A_j \models \Phi \; \forall j \geq 0\}.$$

For $A \subseteq AP$, let the evaluation μ_A be the characteristic function of A.

$A \models \Phi$ iff $\mu_A(\Phi) = 1$

<u>Example</u>: The LT property "the lights are never both green simultaneously" is an invariant with respect to $\Phi = \neg green1 \vee \neg green2$.

Invariants

An LT property P_Φ over AP is an *invariant* with respect to a propositional logic formula Φ over AP if

Notation: repeat infinitely many times

$$P_\Phi = \{A_0 A_1 A_2 \ldots \in (2^{AP})^\omega \;:\; A_j \models \Phi \;\forall j \geq 0\}.$$

For $A \subseteq AP$, let the evaluation μ_A be the characteristic function of A.

$A \models \Phi$ iff $\mu_A(\Phi) = 1$

Example: The LT property "the lights are never both green simultaneously" is an invariant with respect to $\Phi = \neg green1 \vee \neg green2$.

Given TS, Φ, and P_Φ, $TS \models P_\Phi$?

The following four statements are equivalent.

1. $TS \models P_\Phi$
2. $trace(\pi) \in P_\Phi, \; \forall \pi \in Path(TS)$
3. $L(s) \models \Phi, \; \forall s \in S$ on a path of TS
4. $L(s) \models \Phi, \; \forall s \in Reach(TS)$

A state s is reachable if there exists an execution fragment s.t. $s_0 \in I$ and

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n = s$$

$Reach(TS)$: set of reachable states in TS

Invariants are state properties. That is, for verification, find the reachable states and check Φ.

Safety properties

An LT property P_{safe} is a *safety* property if for all words $\sigma \in (2^{AP})^\omega \backslash P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of σ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies P_{safe}.

Safety properties

An LT property P_{safe} is a *safety* property if for all words $\sigma \in (2^{AP})^\omega \backslash P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of σ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies P_{safe}.

Example: *AP* = {red, green, yellow}

- "At least one of the lights is always on" is a safety property.

$$\{\sigma = A_0 A_1 \ldots \ : \ A_j \subseteq AP \wedge A_j \neq \emptyset\}$$

Bad prefixes: finite words that contain $\emptyset$.

- "Two lights are never on at the same time" is a safety property.

$$\{\sigma = A_0 A_1 \ldots \ : \ A_j \subseteq AP \wedge card(A_j) \leq 1\}$$

Bad prefixes: finite words that contain {red,green}, {red,yellow}, and so on.

Safety properties

An LT property P_{safe} is a *safety* property if for all words $\sigma \in (2^{AP})^\omega \backslash P_{safe}$ there exists a finite prefix $\hat{\sigma}$ of σ s.t.

$$P_{safe} \cap \{\sigma' \in (2^{AP})^\omega : \hat{\sigma} \text{ is a finite prefix of } \sigma'\} = \emptyset.$$

Bad things have happened in the bad prefix $\hat{\sigma}$. Hence, no infinite word that starts with $\hat{\sigma}$ satisfies P_{safe}.

<u>Example</u>: *AP* = {red, green, yellow}

- "At least one of the lights is always on" is a safety property.

$\{\sigma = A_0 A_1 \ldots \;\; : \;\; A_j \subseteq AP \wedge A_j \neq \emptyset\}$

Bad prefixes: finite words that contain $\emptyset$.

- "Two lights are never on at the same time" is a safety property.

$\{\sigma = A_0 A_1 \ldots \;\; : \;\; A_j \subseteq AP \wedge card(A_j) \leq 1\}$

Bad prefixes: finite words that contain {red,green}, {red,yellow}, and so on.

Any invariant is a safety property. There are safety properties that are not invariant.

<u>Example</u>: *AP* = {red, yellow}

"Each red is immediately preceded by a yellow" is a safety property, but not invariant (because it is not a state property).

Sample bad prefixes:

$\emptyset\emptyset\{r\}$

$\{y\}\{y\}\{r\}\{r\}\emptyset\{r\}$

Liveness properties

An LT property P is a liveness property if and only if for each finite word w of 2^{AP} there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying w$\sigma \in P$.

Example: Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will *eventually* turn green
- First light will turn green *infinitely often*

Liveness properties

An LT property P is a liveness property if and only if for each finite word w of 2^{AP} there exists an infinite word $\sigma \in (2^{AP})^{\omega}$ satisfying w$\sigma \in P$.

Example: Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will *eventually* turn green
- First light will turn green *infinitely often*

Use of liveness properties:

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.

Liveness properties

An LT property P is a liveness property if and only if for each finite word w of 2^{AP} there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying w$\sigma \in P$.

Example: Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will *eventually* turn green
- First light will turn green *infinitely often*

Use of liveness properties:

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.

Example: Is the following a safety or liveness property?

"the first light is eventually green
after it is initially red three time instances in a row"

Liveness properties

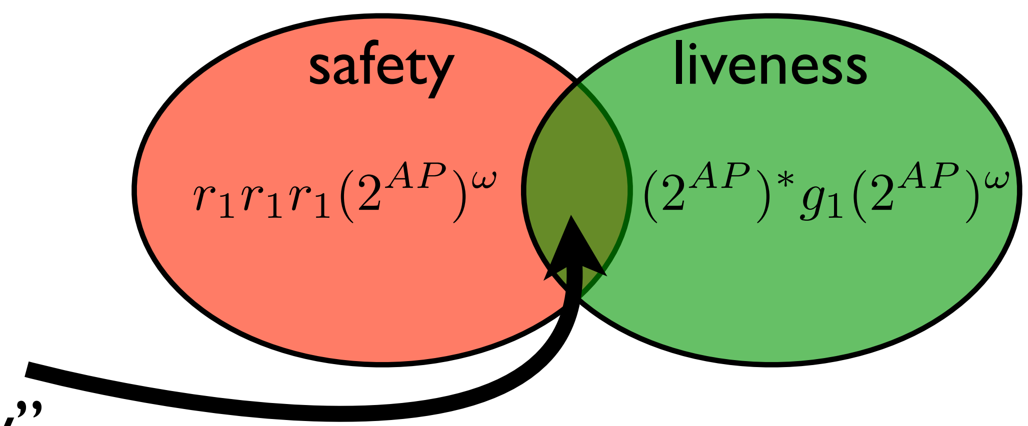
An LT property P is a liveness property if and only if for each finite word w of 2^{AP} there exists an infinite word $\sigma \in (2^{AP})^\omega$ satisfying w$\sigma \in P$.

<u>Example</u>: Two traffic lights with $AP = \{red1, green1, red2, green2\}$

- First light will *eventually* turn green
- First light will turn green *infinitely often*

<u>Use of liveness properties</u>:

- specify the absence of (undesired) infinite loops or progress toward a goal.
- rule out executions that cannot realistically occur (fairness), e.g., in an asynchronous execution, every process is activate infinitely often.



<u>Example</u>: Is the following a safety or liveness property?

"the first light is eventually green
after it is initially red three time instances in a row"

<u>Answer</u>: It is a combination of a safety and a liveness property.

- Liveness: any finite word can be extended by an infinite word $A_0 A_1 A_2 \ldots$ with $green1 \in A_j$ for some $j \geq 0$.
- Safety: any finite word $A_0 A_1 A_2$ with $red1 \notin A_i$ for any $i \in \{0, 1, 2\}$ is a bad prefix.

<u>Invariant</u>	<u>Safety</u>	<u>Liveness</u>
state condition	something bad never happens	something good will happen eventually
violated at individual states	any infinite run violating the property has a finite prefix	violated only by infinite runs
verification: find the reachable states and check the invariant condition	verification: ?	verification: ?