Systems Design Laboratory

Traffic Lights

Matteo Zavatteri

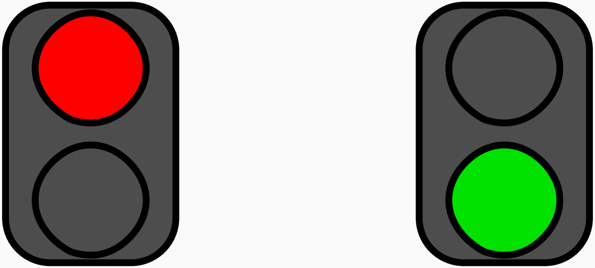
Department of Computer Science, University of Verona, ITALY

General Context

Main components:

- Two Red-Green Traffic Lights.
- A yellow car stream
- A blue car stream

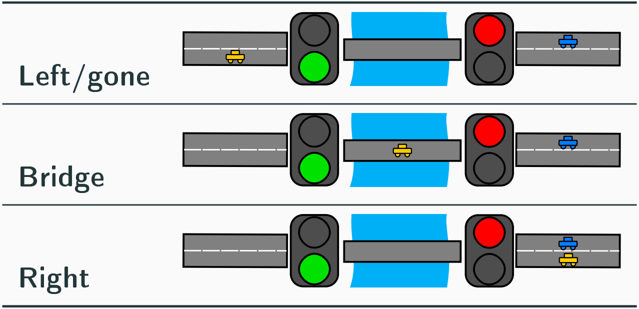
Traffic Lights



Each traffic light operates in two possible ways:

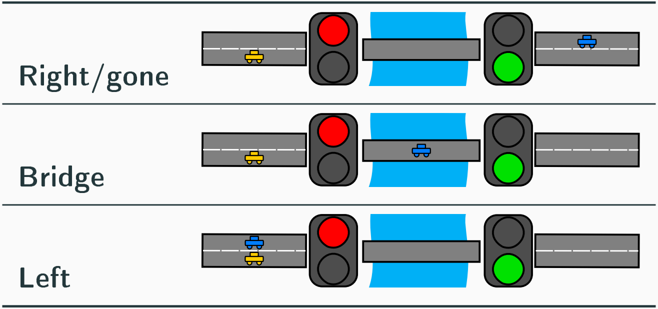
- Red Light
- Green Light

Yellow Car Stream



- A stream of single yellow cars going left to right
- When a car has green light, it can enter the bridge
- Once entered the bridge, the car can exit
- Once exited the bridge, the car can proceed disappearing from the right road segment with a new one appearing on the left
- Beside traffic light synchronization, there is no control on the entering/exiting the bridge of a car

Blue Car Stream



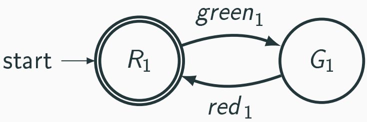
- A stream of single blue cars going right to left
- When a car has green light, it can enter the bridge
- Once entered the bridge, the car can exit
- Once exited the bridge, the car can proceed disappearing from the left road segment with a new one appearing on the right
- Beside traffic light synchronization, there is no control on the entering/exiting the bridge of a car

Traffic Light Automata

- States?
- Transitions?
- Event Controllability?

Traffic Light Automata

Automaton for Traffic Light 1



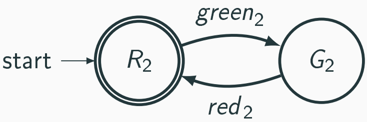
States:

- R_1 = Traffic Light 1 is red
- G_1 = Traffic Light 1 is green

Events:

- $green_1$ = Traffic Light 1 turns green
- red_1 = Traffic Light 1 turns red

Automaton for Traffic Light 2



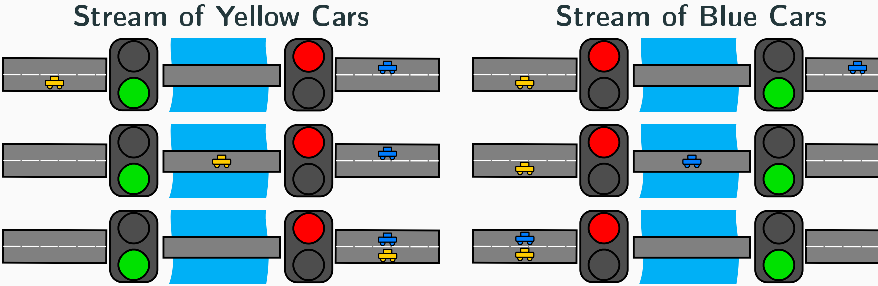
States:

- R_2 = Traffic Light 2 is red
- G_2 = Traffic Light 2 is green

Events:

- $green_2$ = Traffic Light 2 turns green
- red_2 = Traffic Light 2 turns red

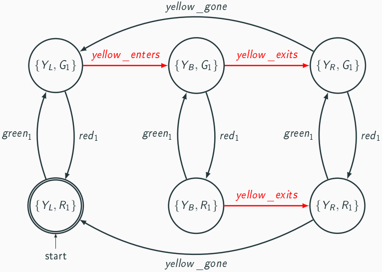
Stream of Cars Automata



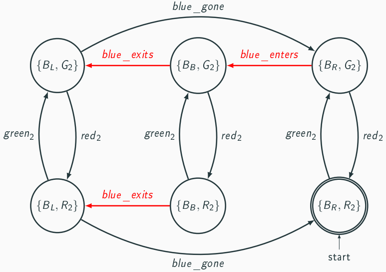
- States?
- Transitions?
- Event controllability?

Car Stream Automata

Automaton for Yellow Car Stream



Automaton for Blue Car Stream

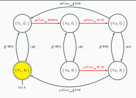
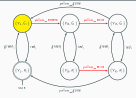
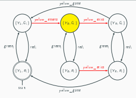
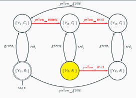
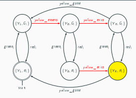
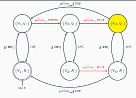


- Y_L: Yellow car is on the left
- Y_B: Yellow car is on the bridge
- Y_R: Yellow car is on the right
- R_1/G_1: Traffic Light 1 is red/green

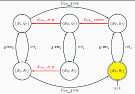
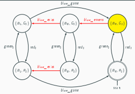
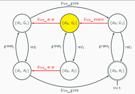
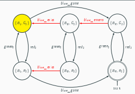
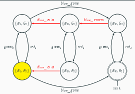
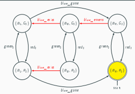
- B_L: Blue car is on the left
- B_B: Blue car is on the bridge
- B_R: Blue car is on the right
- R_2/G_2: Traffic Light 2 is red/green

Conceptually the states are pairs (Car Position, Traffic Light Status)

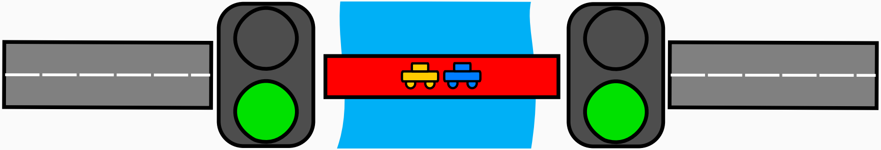
Yellow Car Stream Usecase Example

		Traffic Light 1 is red. Yellow car can't enter the bridge
		Traffic Light 1 turns green. Yellow car can enter the bridge
		Traffic Light 1 stays green. Yellow car enters the bridge
		Traffic Light 1 turns red. Yellow car is still on the bridge
		Traffic Light 1 stays red. Yellow car exits the bridge
		Traffic Light 1 turns green

Blue Car Stream Usecase Example

Traffic Light 2 is red. Blue car can't enter the bridge		
Traffic Light 2 turns green. Blue car can enter the bridge		
Traffic Light 2 stays green. Blue car enters the bridge		
Traffic Light 2 stays green. Blue car exits the bridge		
Traffic Light 2 turns red.		
Traffic Light 2 stays red. Blue car is gone (another one appears).		

Requirement 1: Traffic Lights must not be simultaneously green

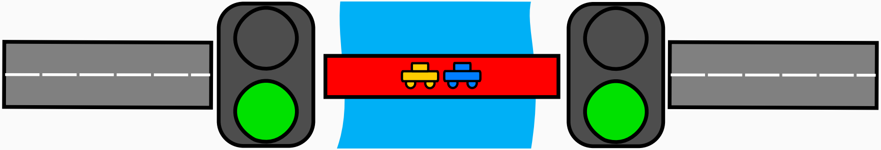


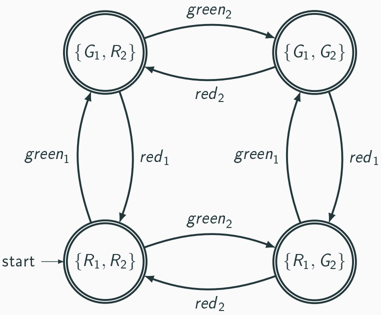
- States?
- Transitions?
- Event controllability?

(Recall that once a vehicle has green light, we can't prevent it from entering the bridge)

Requirement 1 - Attempt 1

Requirement 1: Traffic Lights must not be simultaneously green



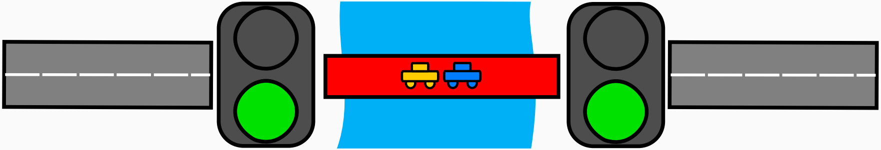


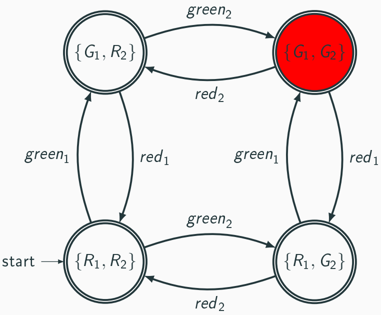
Step 1:

Traffic Light 1 $\|$ Traffic Light 2

Requirement 1 - Attempt 1

Requirement 1: Traffic Lights must not be simultaneously green



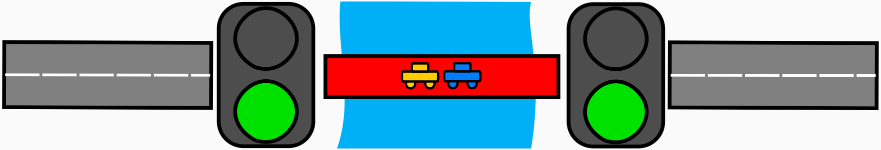


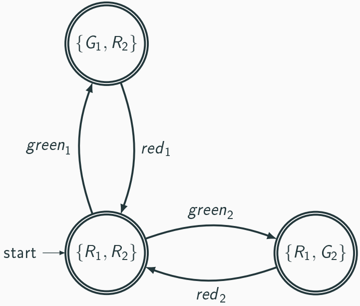
Step 2

Remove the state $\{G_1, G_2\}$

Requirement 1 - Attempt 1

Requirement 1: Traffic Lights must not be simultaneously green





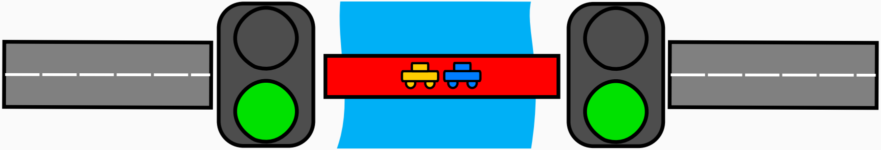
Correct requirement.

Can we avoid starting from

Traffic Light 1 $\parallel$ Traffic Light 2?

Requirement 1 - Attempt 2

Requirement 1: Traffic Lights must not be simultaneously green

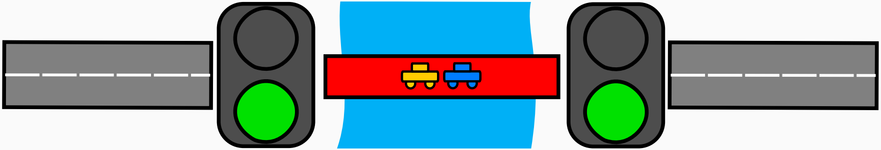


1A) Traffic Light 1 can turn green only if Traffic Light 2 is red

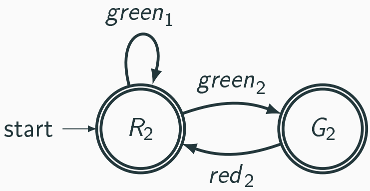
1B) Traffic Light 2 can turn green only if Traffic Light 1 is red

Requirement 1 - Attempt 2 - Decomposition

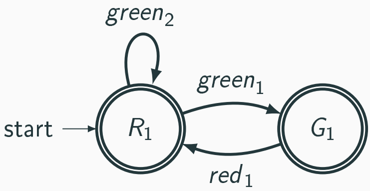
Requirement 1: Traffic Lights must not be simultaneously green



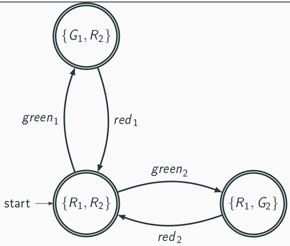
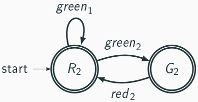
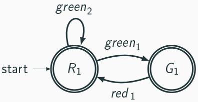
1A) Traffic Light 1 can turn green only if Traffic Light 2 is red



1B) Traffic Light 2 can turn green only if Traffic Light 1 is red



Automata for R_1 - Summary of Equivalent Versions

Version	Automaton	Modeling Intuition
Version 1		**From a modified copy of Traffic Light 1 $\|\|$ Traffic Light 2**
Version 2		**From modified copies of Traffic Lights 1 and 2 (each in isolation)**

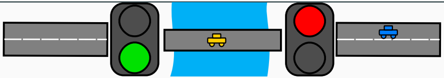
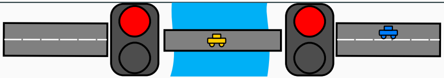
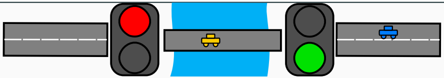
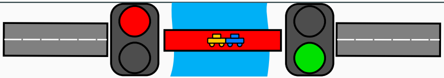
Homework: check if the parallel composition of the two automata in Version 2 results in the automaton of Version 1.

Problem

Yet, car crashes are not completely avoided even if both traffic lights are prevented from turning simultaneously green

Can you spot the problem?

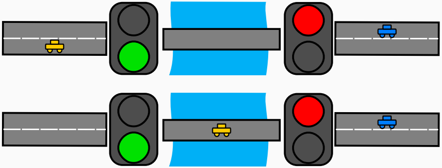
An Unforeseen Scenario

Traffic Light 1 turns green	
Yellow car enters the bridge	
Traffic Light 1 turns red	
Traffic Light 2 turns green	
Blue car enters the bridge	

Requirement 2

Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

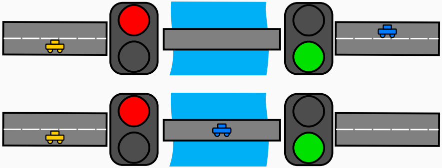
(Yellow car)



Traffic Light 2 cannot turn green

Traffic Light 2 can turn green

(Blue car)



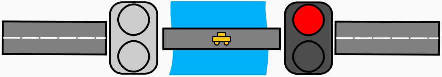
Traffic Light 1 cannot turn green

Traffic Light 1 can turn green

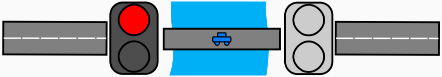
Requirement 2

Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

⇓



2A) ...

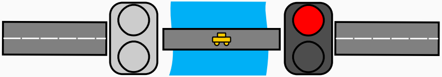


2B) ...

Requirement 2

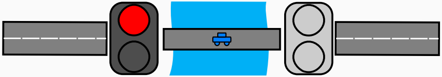
Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

⇓



2A) Traffic Light 2 can turn green only if there is no yellow car on the bridge

...



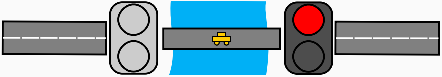
2B) Traffic Light 1 can turn green only if there is no blue car on the bridge

...

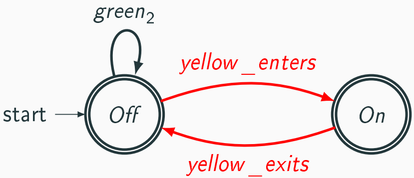
Requirement 2 - Decomposition

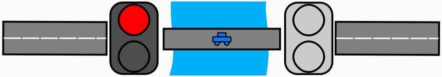
Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

⇓

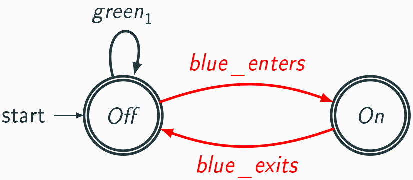


2A) Traffic Light 2 can turn green only if there is no yellow car on the bridge





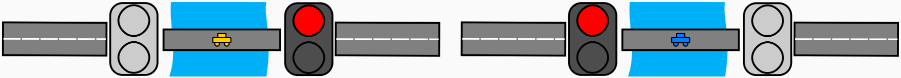
2B) Traffic Light 1 can turn green only if there is no blue car on the bridge



Requirement 2

Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

$\Downarrow$

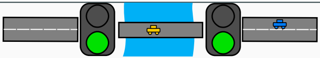
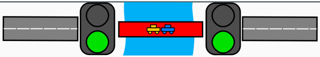


Question: Does R_2 in isolation guarantees to avoid car crashes?

Is R_2 enough to avoid car crashed?

Requirement 2: A Traffic Light can turn green only if there is no car on the bridge coming from the opposite direction

Question: Does R_2 in isolation guarantees to avoid car crashes?

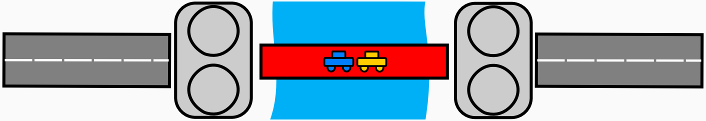
$G \| R_2$	Description
	Traffic Light 1 turns green
	Traffic Light 2 turns green
	Yellow car enters the bridge
	Blue car enters the bridge

No! Since R_1 does not hold, we can turn green both traffic lights before having cars on the bridge (and the problem is still there).

Alternative to Requirements 1 and 2: Right or wrong?

Instead of having R_1 and R_2. Consider this requirement.

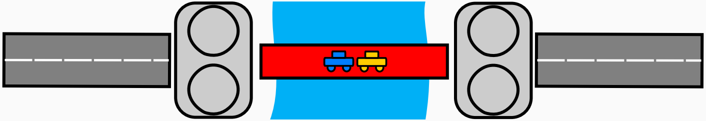
Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.



Does this requirement have the same effect on the plant of requirements 1 and 2 together?

Requirements $R'_{1,2}$ - Attempt 1

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

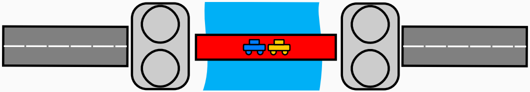


Such a requirement should:

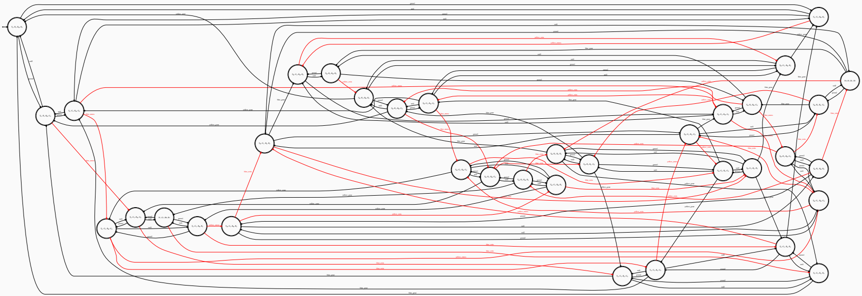
- **no longer be designed from copies of traffic lights**
- **reasonably be designed from the combinations of car positions**

Requirements $R'_{1,2}$ - Attempt 1

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.



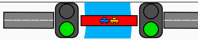
Step 1: Compute the parallel composition of the car stream automata. Mark all states.



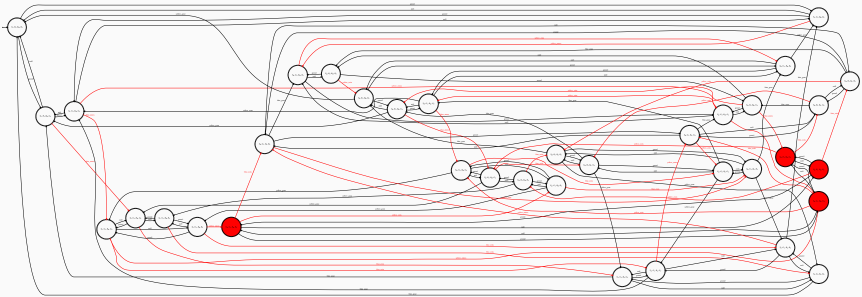
$6 \times 6 = 36$ **states,** 132 **transitions. Why so big? What kind of composition is it?**

Requirements $R'_{1,2}$ - Attempt 1

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

Usecase 1	Usecase 2	Usecase 3	Usecase 4
			

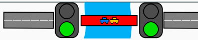
Step 2: Find all states where a yellow and a blue car are on the bridge together.



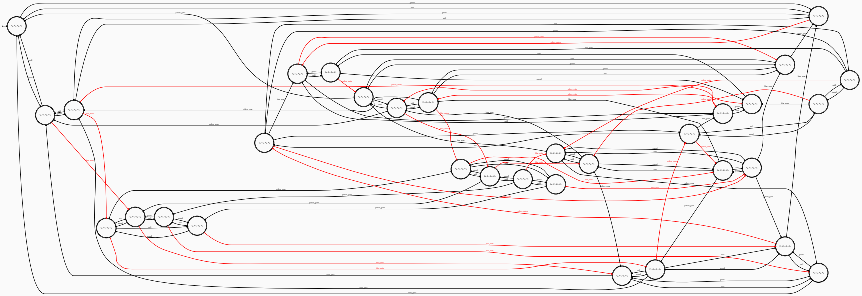
Clearly 4 states. Why?

Alternative to Requirements 1 and 2: Right or wrong?

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

Usecase 1	Usecase 2	Usecase 3	Usecase 4
			

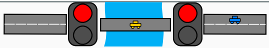
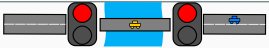
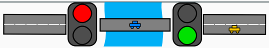
Step 3: Remove those illegal states.



Final requirement: 32 states, 112 transitions.

Alternative to Requirements 1 and 2: Right or wrong?

Question: $G \| R_1 \| R_2 \equiv G \| R'_{1,2}$**?**

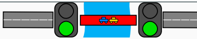
$G \| R_1 \| R_2$	$G \| R'_{1,2}$	**Description**
		**Traffic Light 1 turns green**
		**Yellow car enters the bridge**
		**Traffic Light 1 turns red**
		**Yellow car exits the bridge**
		**Traffic Light 2 turns green**
		**Blue car enters the bridge**
		**Blue car exits the bridge**
Disabled by R_1		**Traffic Light 1 turns green**

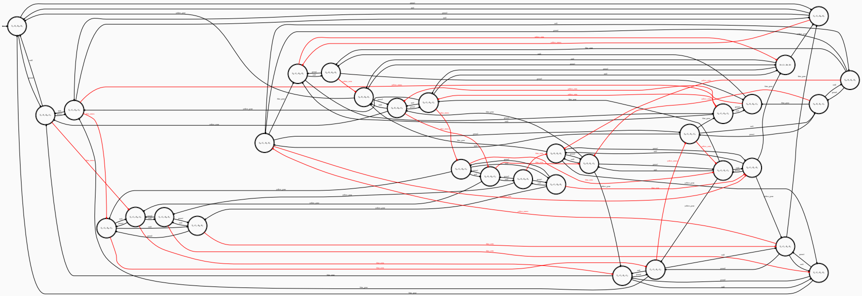
Wrong! $G \| R_1 \| R_2 \not\equiv G \| R'_{1,2}$. **The problem is that** R_1 **does not hold in** $R'_{1,2}$.

Homework: find other usecases (i.e., executions, traces) violating R_1.

Essentiality of $R'_{1,2}$

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

Usecase 1	Usecase 2	Usecase 3	Usecase 4
			

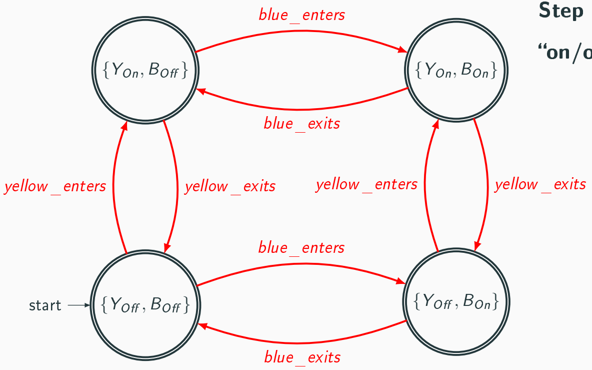


Can we simplify it?

Requirement $R'_{1,2}$ - Attempt 2

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

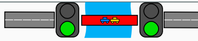
Usecase 1	Usecase 2	Usecase 3	Usecase 4
			

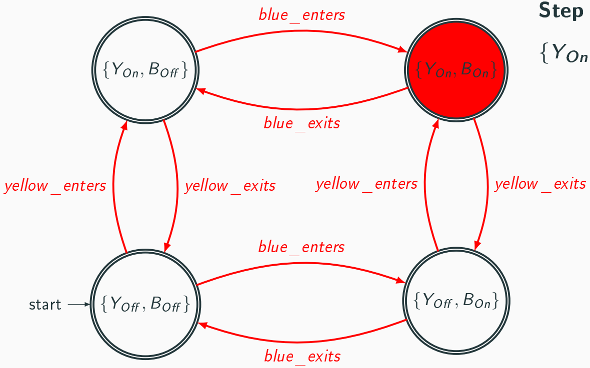


Step 1: Concurrent behavior of “on/off bridge” automata

Requirement $R'_{1,2}$ - Attempt 2

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

Usecase 1	Usecase 2	Usecase 3	Usecase 4
			

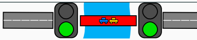


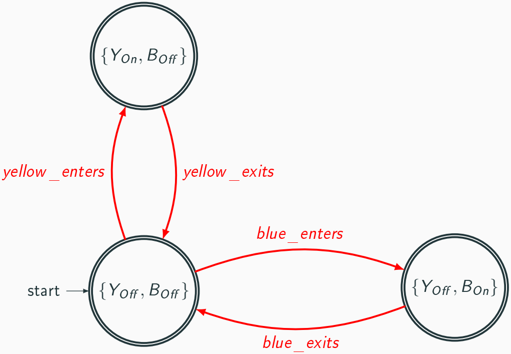
Step 2: Remove the illegal state

$\{Y_{On}, B_{On}\}$

Requirement $R'_{1,2}$ - Attempt 2

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.

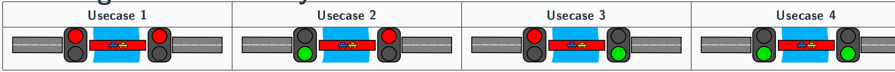
Usecase 1	Usecase 2	Usecase 3	Usecase 4
			



Correct. Can we avoid starting from the concurrent behavior of “on/off bridge” automata?

Requirement $R'_{1,2}$ - Attempt 3 - Decomposition

Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.



$R'_{1,2}A$) **A yellow car can enter the bridge only if there is no blue car on it**

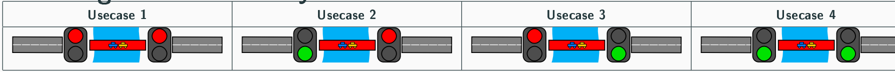
...

$R'_{1,2}B$) **A blue car can enter the bridge only if there is no yellow car on it**

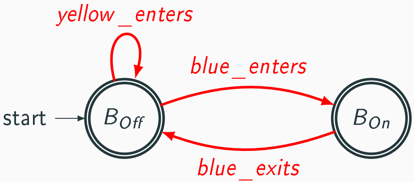
...

Requirement $R'_{1,2}$ - Attempt 3

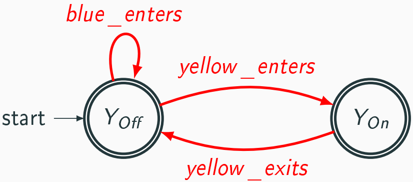
Requirement $R'_{1,2}$: There are never a yellow car and a blue car on the bridge simultaneously.



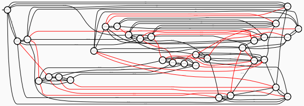
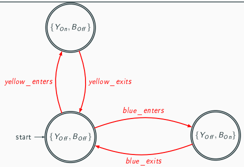
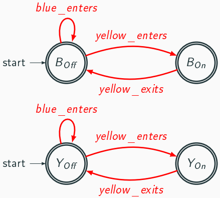
$R'_{1,2}A$) A yellow car can enter the bridge only if there is no blue car on it



$R'_{1,2}B$) A blue car can enter the bridge only if there is no yellow car on it



Automata for $R'_{1,2}$ - Summary of Equivalent Versions

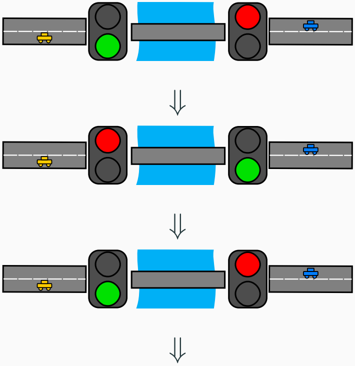
Version	Automaton	Modeling Intuition
Version 1		From a modified copy of YellowCarStream ∥ BlueCarStream
Version 2		From a modification of "On/Off bridge" automaton for yellow and blue cars (concurrent)
Version 3		From a modification of "On/Off bridge" automaton for yellow and blue cars (in isolation)

Homework: note the modeling similarities of $R'_{1,2}$ (version 2) with R_1 (version 1); of $R'_{1,2}$ (version 3) with R_1 (version 2) and R_2.

Requirement 3

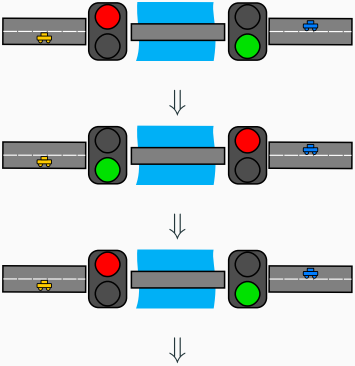
Requirement 3: Green Lights must alternate.

If Traffic Light 1 turns green first



. . .

If Traffic Light 2 turns green first



. . .

Requirement 3 - Attempt 1

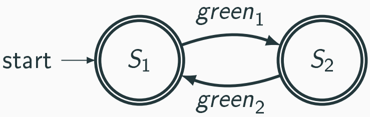
Requirement 3: Green Lights must alternate.

If Traffic Light 1 turns green first

⇓

...

Requirement R_{3A}

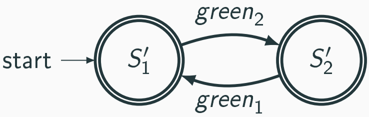


If Traffic Light 2 turns green first

⇓

...

Requirement R_{3B}



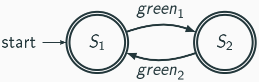
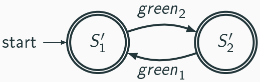
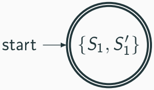
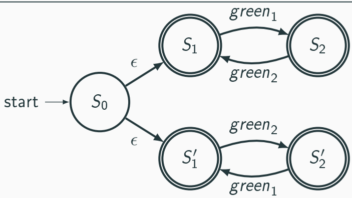
Not certaintly an AND of the two automata.

We need the UNION of these two automata.

Requirement 3: Green Lights must alternate.

If Traffic Light 1 turns green first

If Traffic Light 2 turns green first

Requirement R_{3A}	**Requirement R_{3B}**
	
Requirement $R_{3A} \wedge R_{3B} := R_{3A} \| R_{3B} = R_{3A} \times R_{3B}$	**Requirement $R_{3A} \vee R_{3B}$**
	

Homework: synthesize a supervisor that (also) takes into consideration requirement $R_{3A} \wedge R_{3B}$. What effect does it have on the plant?

Requirement 3 - Attempt 1 - Nondeterministic

Requirement 3: Green Lights must alternate.

If Traffic Light 1 turns green first

$\Downarrow$

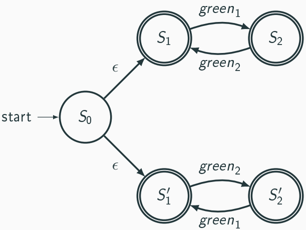
...

If Traffic Light 2 turns green first

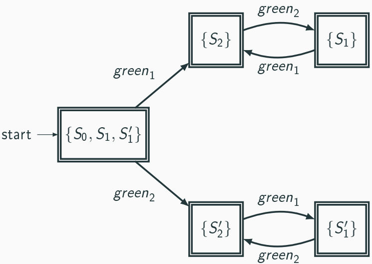
$\Downarrow$

...

NFA



DFA



Requirement 3 - Attempt 2 - Deterministic

Requirement 3: Green Lights must alternate.

$\Downarrow$

3A) If Traffic Light 1 turns green, then Traffic Light 2 must turn green at least once before Traffic Light 1 turns green again.

3B) Whenever Traffic Light 2 turns green, then Traffic Light 1 must turn green at least once before Traffic Light 2 turns green again.

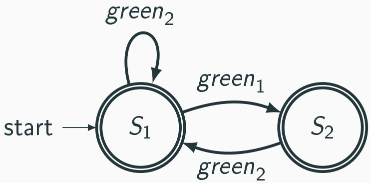
If Traffic Light $i = 1, 2$ turns green, then Traffic Light $(i \mod 2) + 1$ must turn green at least once before Traffic Light i turns green again.

Requirement 3 - Attempt 2 - Deterministic

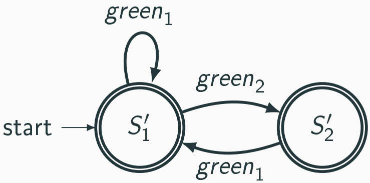
Requirement 3: Green Lights must alternate.

$\Downarrow$

3A) If Traffic Light 1 turns green, then Traffic Light 2 must turn green at least once before Traffic Light 1 turns green again.



3B) Whenever Traffic Light 2 turns green, then Traffic Light 1 must turn green at least once before Traffic Light 2 turns green again.

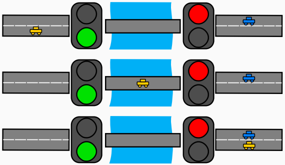


If Traffic Light $i = 1, 2$ **turns green, then Traffic Light** $(i \mod 2) + 1$ **must turn green at least once before Traffic Light** i **turns green again.**

Requirement 4

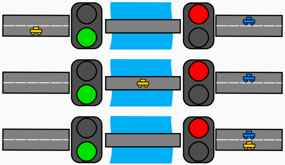
Requirement 4: Whenever Traffic Light 1 turns green, then 2 to 4 yellow cars traverse (i.e., exit) the bridge before Traffic Light 1 turns red again

1st car



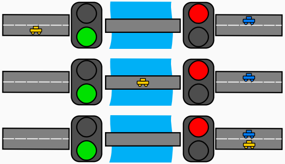
Traffic Light 1 cannot turn red

2nd car



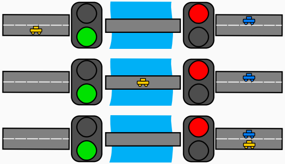
Traffic Light 1 can turn red

3rd car



Traffic Light 1 can turn red

4th car

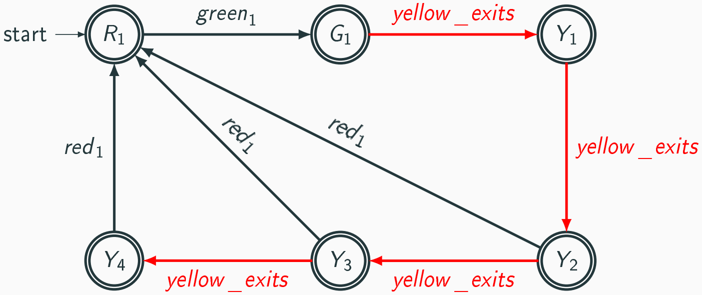


Traffic Light 1 must turn red

What about the automaton?

Requirement 4

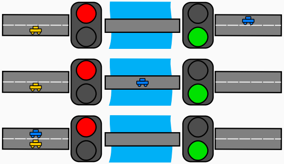
Requirement 4: Whenever Traffic Light 1 turns green, then 2 to 4 yellow cars traverse (i.e., exit) the bridge before Traffic Light 1 turns red again



Requirement 5

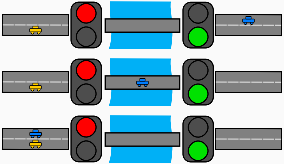
Requirement 5: Whenever Traffic Light 2 turns green, then 1 to 3 blue cars traverse (i.e., exit) the bridge before Traffic Light 2 turns red again

1st car



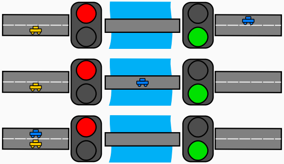
Traffic Light 2 can turn red

2nd car



Traffic Light 2 can turn red

3rd car



Traffic Light 2 must turn red

What about this automaton?

Requirement 5

Requirement 5: If Traffic Light 2 turns green, then 1 to 3 blue cars traverse (i.e., exit) the bridge before Traffic Light 2 turns red again