Lecture 4
Model Checking and Logic Synthesis

Nok Wongpiromsarn

Richard M. Murray   Ufuk Topcu

EECI-IGSC, 9 March 2020

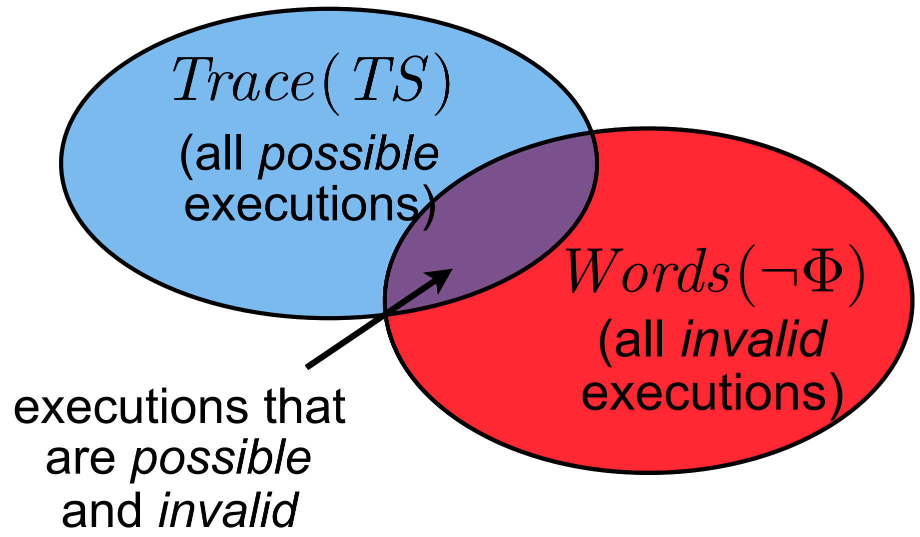
Outline

- Model checking: what it is, how it works, how it is used
- Computational complexity of model checking
- Closed system synthesis
- Examples using SPIN model checker

The basic idea behind model checking

Given:

- Transition system TS
- LTL formula Φ



Question: Does TS satisfy Φ, i.e.,

$$TS \models \Phi \text{ ?}$$

Answer (conceptual):

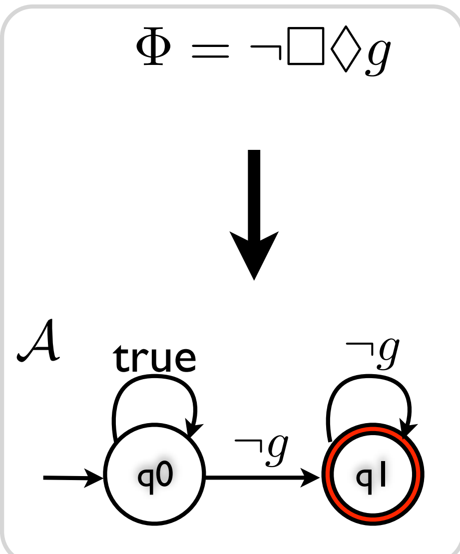
$TS \models \Phi$	[*TS* satisfies Φ]
$\Updownarrow$	
$Trace(TS) \subseteq Words(\Phi)$	[All executions of *TS* satisfy Φ]
$\Updownarrow$	
$Trace(TS) \cap Words(\neg\Phi) = \emptyset$	[No execution of *TS* violates Φ]

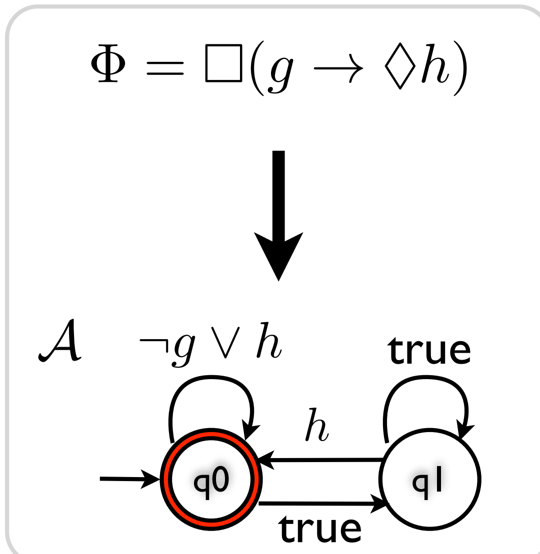
How to determine whether $Trace(TS) \cap Words(\neg\Phi) = \emptyset$ **?**

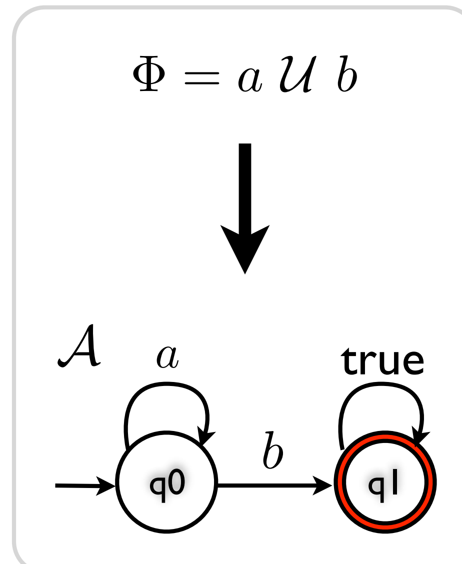
Preliminaries: LTL → Buchi automata

Theorem. *There exists an algorithm that takes an LTL formula Φ and returns a Büchi automaton $\mathcal{A}$ such that*

$$Words(\Phi) = \mathcal{L}_\omega(\mathcal{A})$$







A tool for constructing Buchi automata from LTL formulas: LTL2BA [http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/index.php]

Preliminaries: transition system ⊗ Buchi automaton

Transition system:

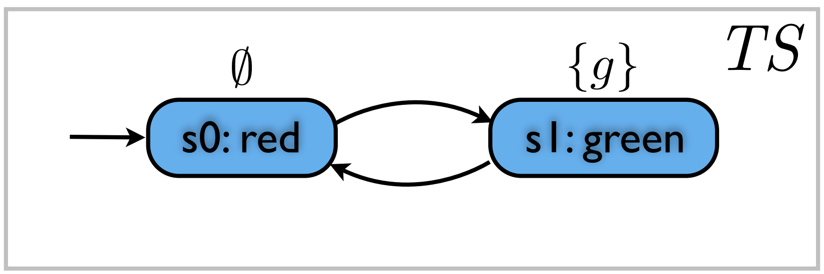
$$TS = (S, \mathrm{Act}, \rightarrow, I, \mathrm{AP}, L)$$

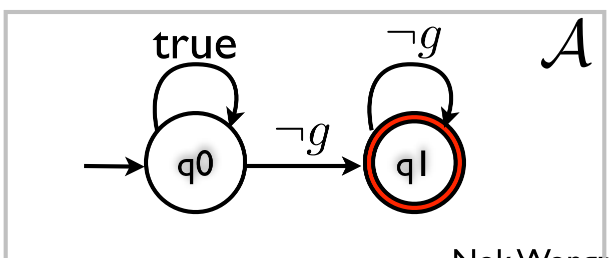
Nondeterministic Buchi automaton:

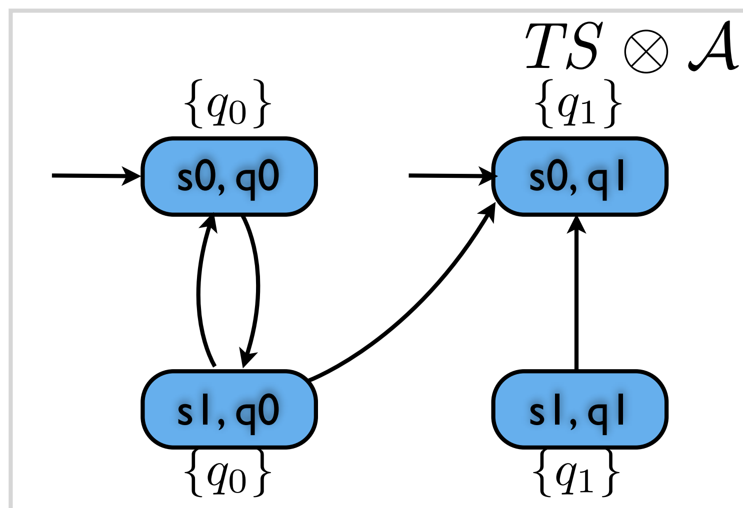
$$\mathcal{A} = (Q, 2^{\mathrm{AP}}, \delta, Q_0, F)$$

Define the product automaton: $TS \otimes \mathcal{A} = (S', \mathrm{Act}, \rightarrow', I', \mathrm{AP}', L')$, where

- $S' = S \times Q$
- $\forall s, t \in S, q, p \in Q$ with $s \xrightarrow{\alpha} t$ and $q \xrightarrow{L(t)} p$, there exists $\langle s, q \rangle \xrightarrow{\alpha}{}' \langle t, p \rangle$
- $I' = \{\langle s_0, q \rangle : s_0 \in I \text{ and } \exists q_0 \in Q_0 \text{ s.t. } q_0 \xrightarrow{L(s_0)} q\}$
- $AP' = Q$
- $L' : S \times Q \rightarrow 2^Q$ and $L'(\langle s, q \rangle) = \{q\}$







Preliminaries

Transition system: $TS = (S, \mathrm{Act}, \rightarrow, I, \mathrm{AP}, L)$

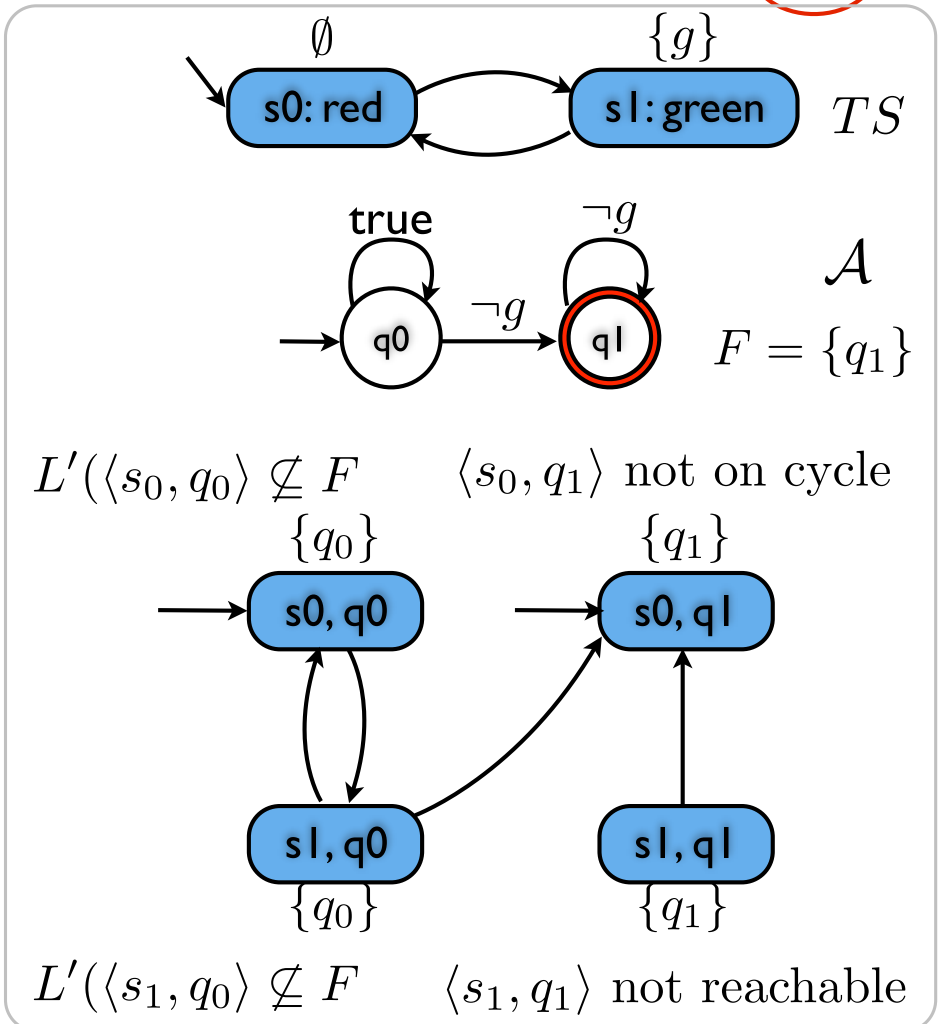
Nondeterministic Buchi automaton: $\mathcal{A} = (Q, 2^{\mathrm{AP}}, \delta, Q_0, F)$

not in *F*

Theorem: $Trace(TS) \cap \mathcal{L}_\omega(\mathcal{A}) \neq \emptyset \quad \Leftrightarrow \quad TS \otimes \mathcal{A} \not\models$ "eventually forever" $\neg F$

Proof idea (⇐): Pick a path π' in $TS \otimes \mathcal{A}$ s.t. $\pi' \not\models$ "eventually forever" $\neg F$, and let π be its projection to *TS*. Then,

- *trace*(π) ∈ *Trace*(*TS*) -- by definition of product
- *trace*(π) ∈ $\mathcal{L}_\omega(\mathcal{A})$ -- by hypothesis and by definition of product (*L*'(‹*s*,*q*›) = {*q*})



$TS \otimes \mathcal{A} \quad \not\models$ "eventually forever" $\neg F$

$\Updownarrow$

There exists a state *x* in $TS \otimes \mathcal{A}$

- *x* is reachable
- *L'*(*x*) ⊆ *F*
- *x* is on a directed cycle

} graph search, e.g., (nested) depth-first search

Putting together

Given:

- Transition system TS
- LTL formula Φ
- NBA $\mathcal{A}_{\neg\Phi}$ accepting $\neg\Phi$ with the set *F* of accepting states

$$TS \not\models \Phi$$

$$\Updownarrow$$

$$Trace(TS) \not\subseteq Words(\Phi)$$

$$\Updownarrow$$

$$Trace(TS) \cap Words(\neg\Phi) \neq \emptyset$$

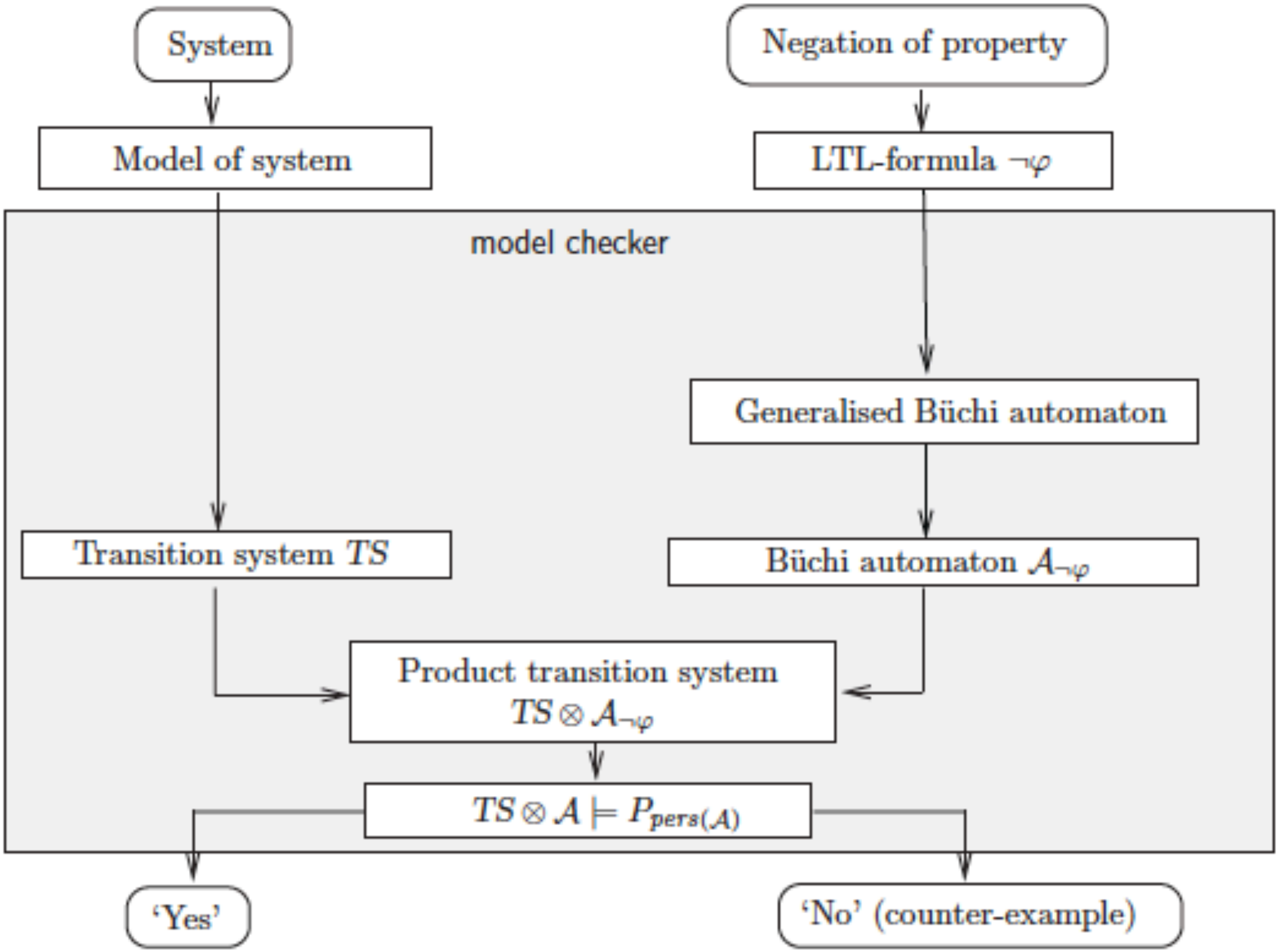
$$\Updownarrow$$

$$Trace(TS) \cap \mathcal{L}_\omega(\mathcal{A}_{\neg\Phi}) \neq \emptyset$$

$$\Updownarrow$$

$$TS \otimes \mathcal{A}_{\neg\Phi} \not\models \text{``eventually forever''} \neg F$$

The process flow of model checking



Efficient model checking tools automate the process: SPIN, nuSMV, TLC,...

Computational complexity of model checking

Transition system: $TS = (S, \mathrm{Act}, \rightarrow, I, \mathrm{AP}, L)$. Specification: Φ

Problem size:

$$\begin{pmatrix} \#\text{ of reachable} \\ \text{states in } TS \end{pmatrix} \times \begin{pmatrix} \#\text{ of states} \\ \text{in } \mathcal{A}_{\neg\Phi} \end{pmatrix} \times \begin{pmatrix} \text{size of one} \\ \text{state in bytes} \end{pmatrix}$$

$O(|S|)$ $\qquad$ $2^{O(|\neg\Phi|)}$ → "length" of $\neg\Phi$, e.g., # of operators in $\neg\Phi$

Potential reductions:

- Restrict the ranges of variables
- Use abstraction, separation of concerns, generalization
- Use compressed representation of the state space (e.g. BDD)
  - Used in symbolic model checkers, e.g., SMV, NuSMV
- **Partial order reduction** (avoid computing equivalent paths)

- Use separable properties, instead of large, combined ones

- Lossy compression, e.g., hash-compact and bitstate hashing
  - May result in incompleteness
- Lossless compression and alternate state representation methods
  - May increase time while reduce memory

"**On-the-fly**" construction of TS, $\mathcal{A}_{\neg\Phi}$ and the product automaton (while searching the automaton) to avoid constructing the complete state space

Time complexity of DFS: $O(\#\text{ of states} + \#\text{ of transitions in } TS \otimes A_{\neg\Phi})$

Closed system synthesis

Closed system: behaviors are generated purely by the system itself without any external influence

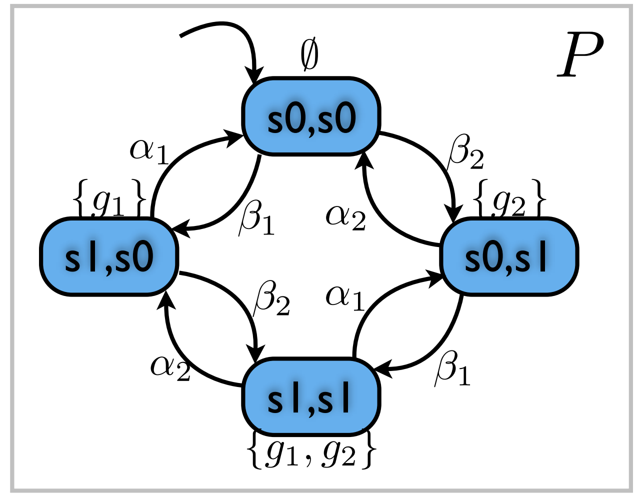
Given:

- A transition system *P*
- An LTL formula Φ

Compute: A path π of P such that

$$\pi \models \Phi$$

P: composition of two traffic lights



$$\Phi = \Box\neg(g_1 \wedge g_2) \wedge \Box\Diamond g_1 \wedge \Box\Diamond g_2$$

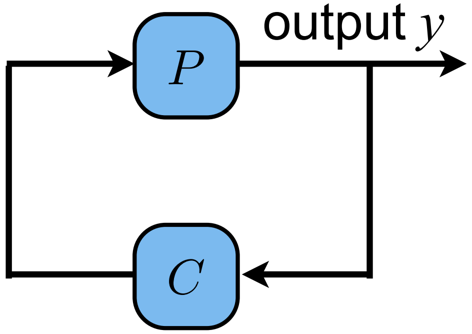
Sample paths of *P*:

$$\pi_1 = (\langle s_0 s_0\rangle\langle s_1 s_0\rangle\langle s_1 s_1\rangle\langle s_0 s_1\rangle)^\omega$$ ✗

$$\pi_2 = (\langle s_0 s_0\rangle\langle s_0 s_1\rangle)^\omega$$ ✗

$$\pi_3 = (\langle s_0 s_0\rangle\langle s_1 s_0\rangle\langle s_0 s_0\rangle\langle s_0 s_1\rangle)^\omega$$ ✓

Closed system synthesis--a "controls" interpretation



The controller *C* is a function $C : M \times S \rightarrow Act$ (memory domain)

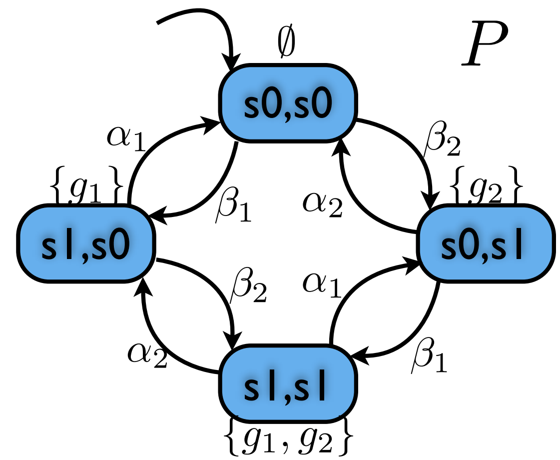
- The controller keeps some history of states
- It picks the next action for *P* such that the resulting path satisfies the specification Φ (i.e., *C* constrains the paths system can take.

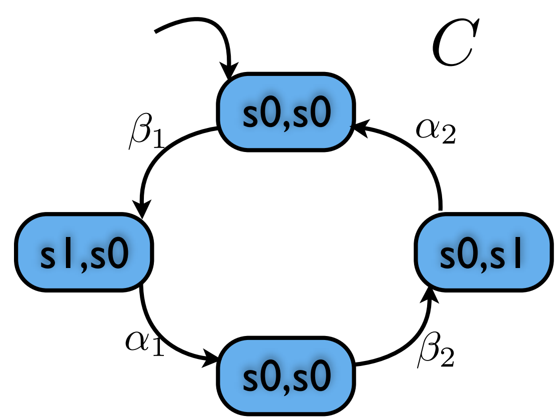
Let *M* be a sequence of length 1, i.e., the controller keeps only the previous state

$$\begin{aligned} C(\emptyset, \langle s_0 s_0 \rangle) &= \beta_1 \\ C(\langle s_0 s_1 \rangle, \langle s_0 s_0 \rangle) &= \beta_1 \\ C(\langle s_1 s_0 \rangle, \langle s_0 s_0 \rangle) &= \beta_2 \\ C(\langle s_0 s_0 \rangle, \langle s_1 s_0 \rangle) &= \alpha_1 \\ C(\langle s_0 s_0 \rangle, \langle s_0 s_1 \rangle) &= \alpha_2 \end{aligned}$$

$$\Rightarrow \pi = (\langle s_0 s_0 \rangle \langle s_1 s_0 \rangle \langle s_0 s_0 \rangle \langle s_0 s_1 \rangle)^\omega$$

and $\pi \models \Phi = \Box \neg (g_1 \wedge g_2) \wedge \Box \Diamond g_1 \wedge \Box \Diamond g_2$



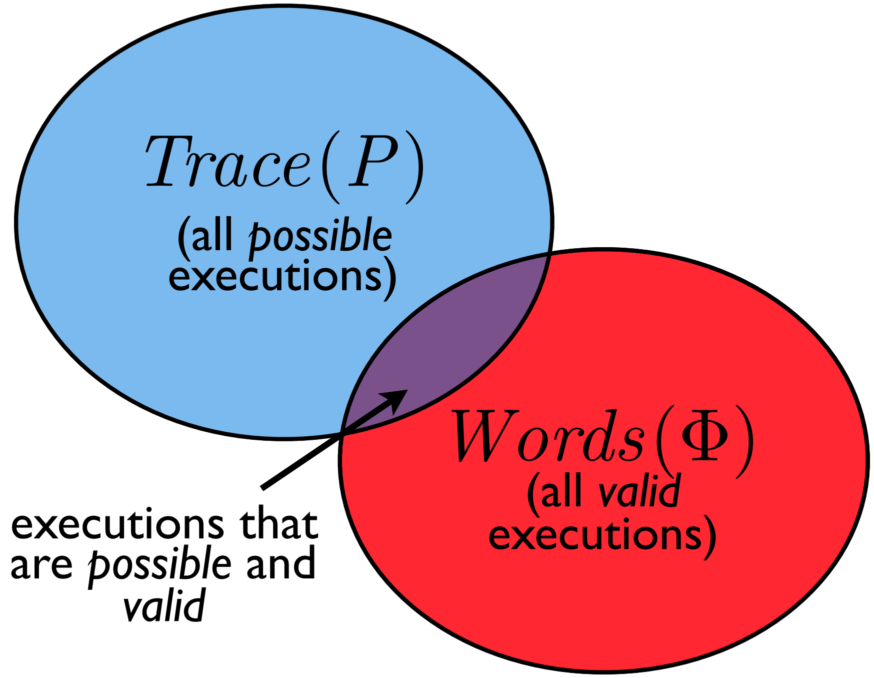


A solution approach

- Closed system synthesis can be formulated as a non-emptiness of the specification or satisfiability problem

$$\exists y \cdot \Phi(y)$$

- For synthesis problems, “interesting” behaviors are “good” behaviors (as opposed to verification problems where “interesting behaviors are “bad” behaviors)

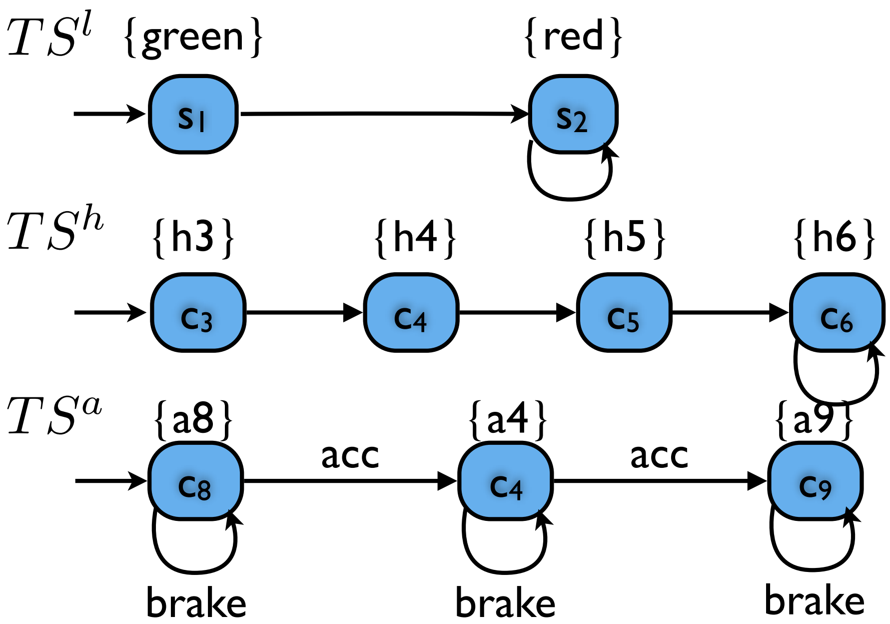


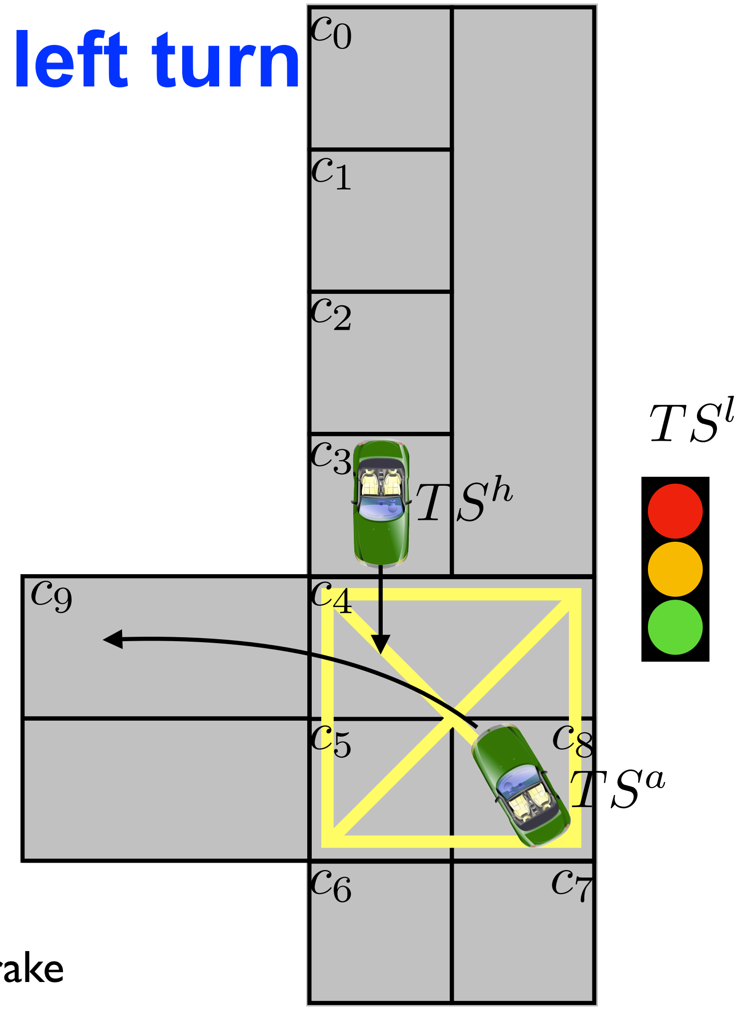
- Construct a verification model and claim that

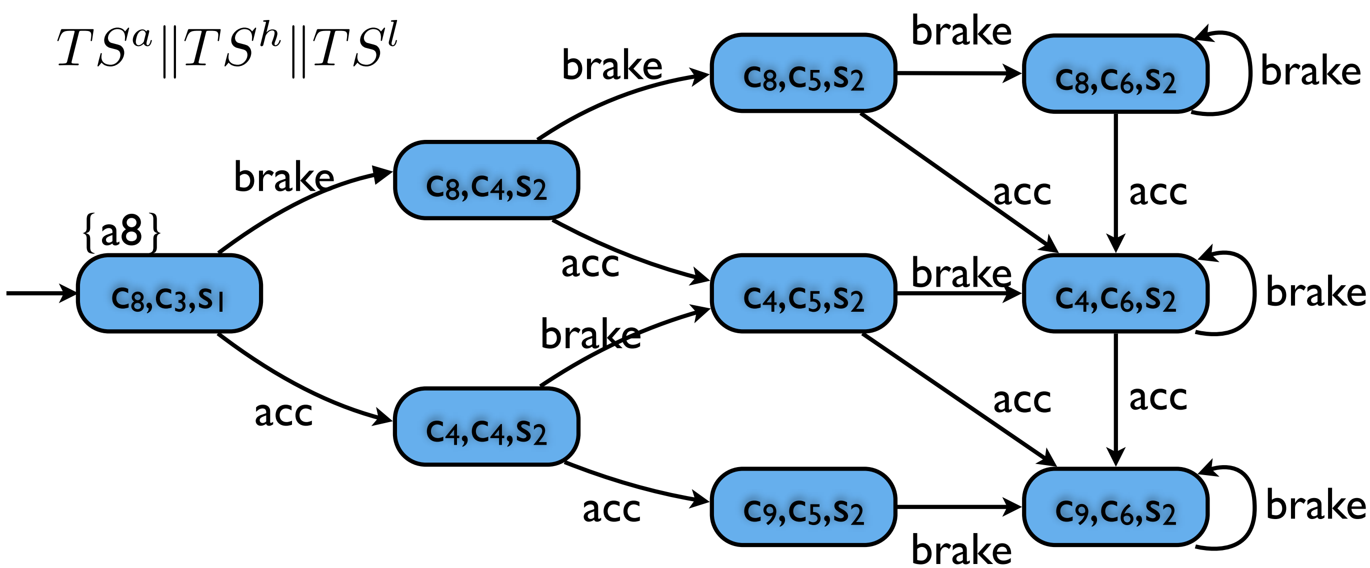
$$Trace(P) \cap Words(\Phi) = \emptyset$$

  - A counterexample provided in case of negative result is a path π of *P* that satisfies Φ
  - Positive result means $Trace(P) \cap Words(\Phi) = \emptyset$, i.e., a path π of *P* that satisfies Φ does not exist

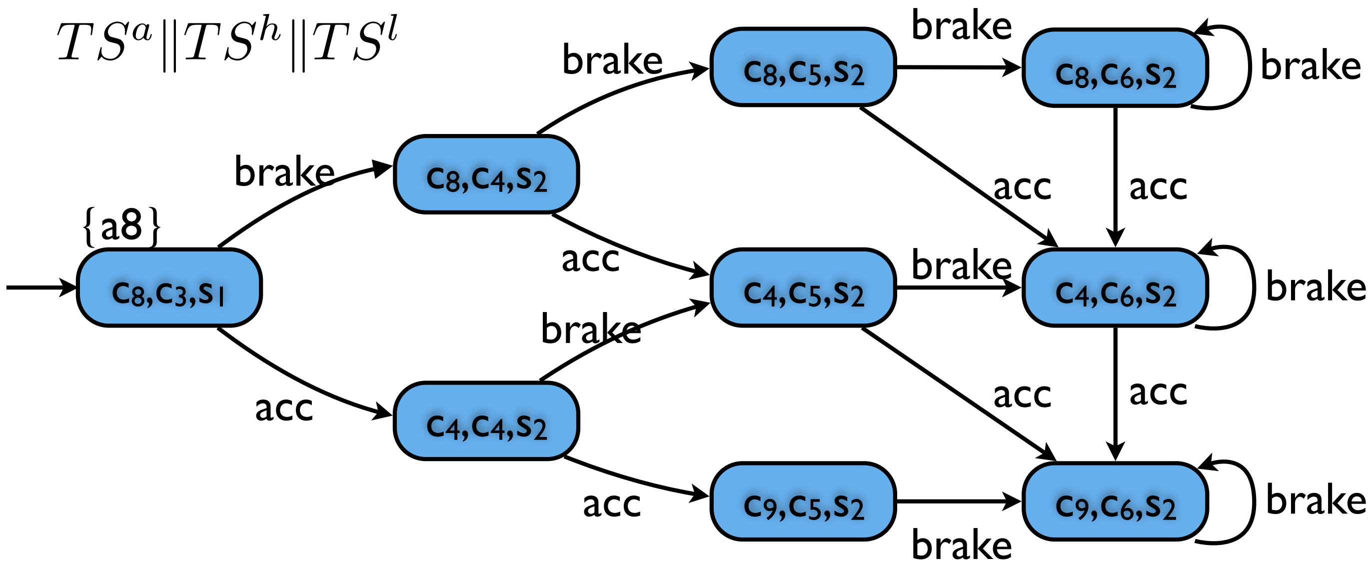
Example 1: unprotected left turn



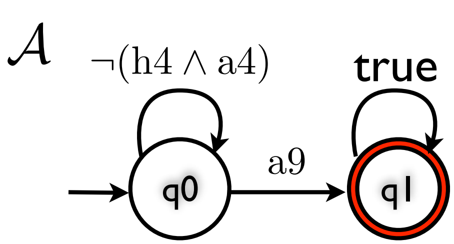


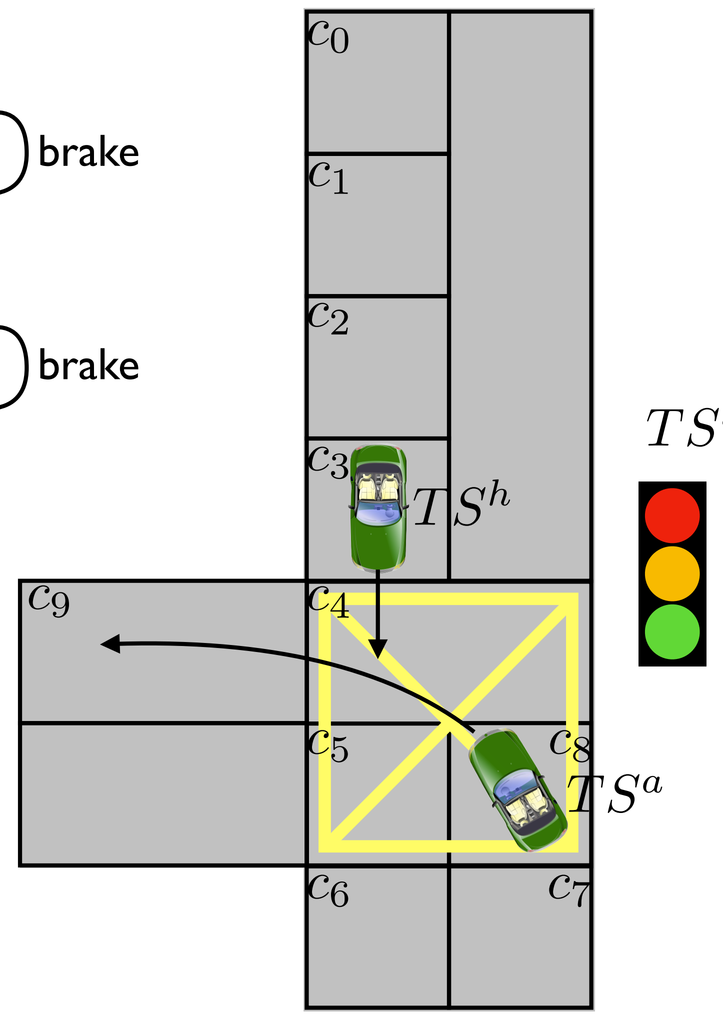


Example 1: unprotected left turn



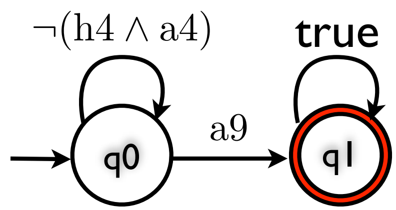
$$\varphi = \neg(\mathrm{h4} \wedge \mathrm{a4})\ \mathbf{U}\ \mathrm{a9}$$



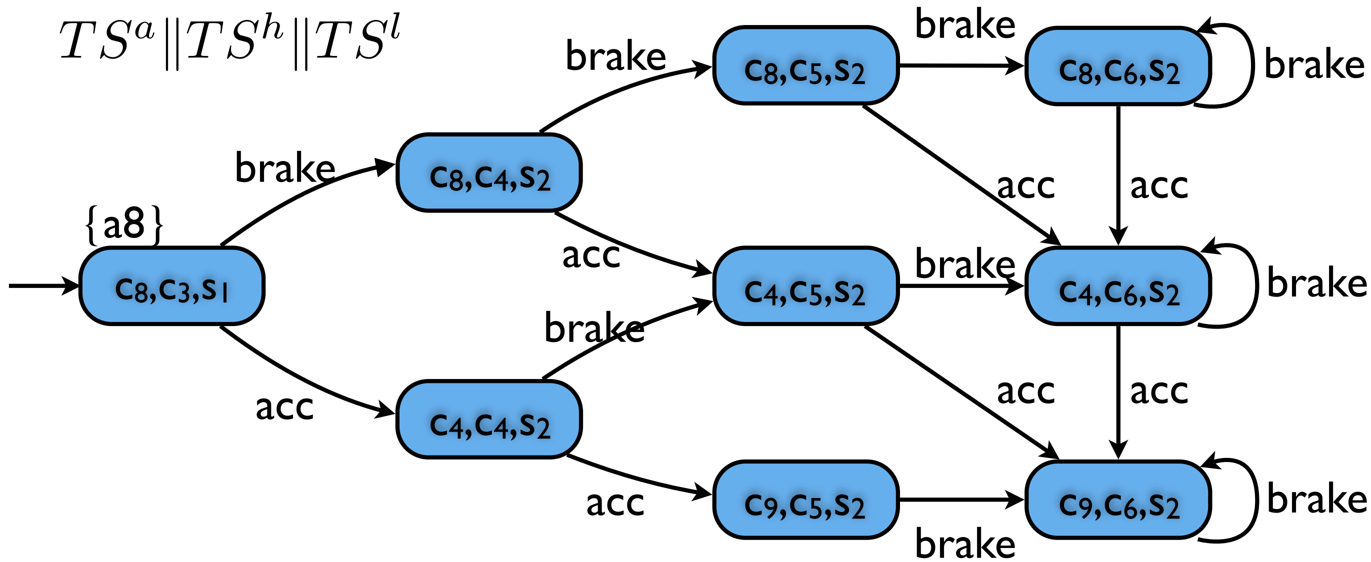


Unprotected left turn: product automaton

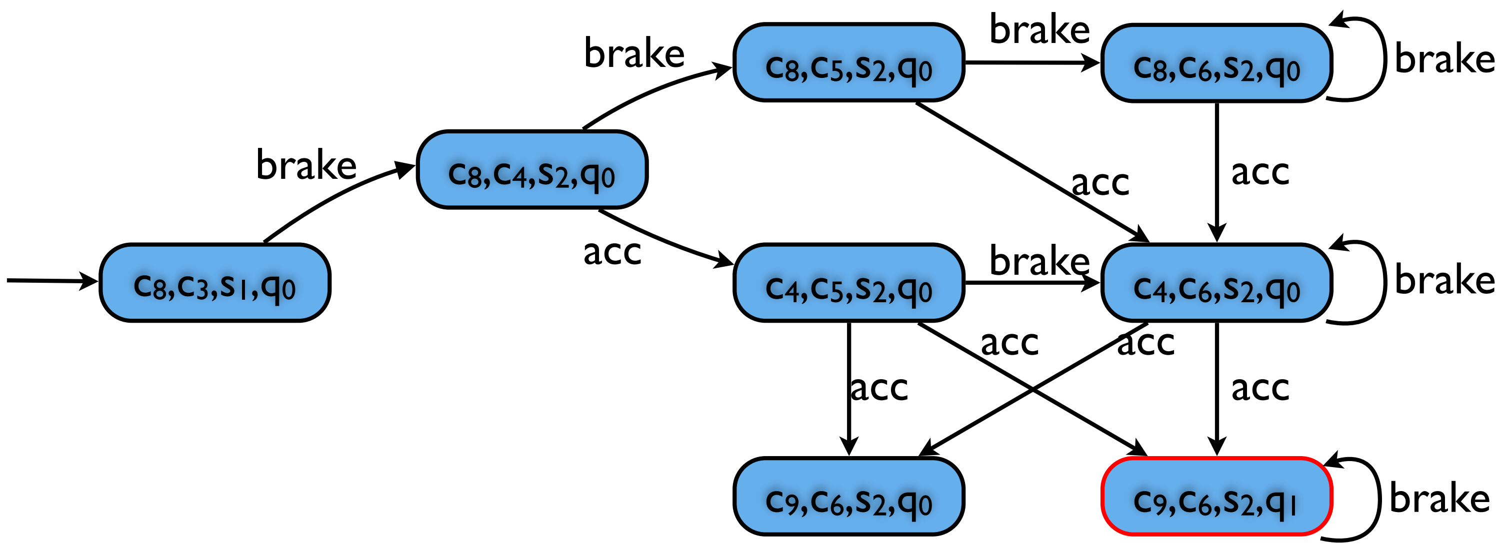
$\mathcal{A}$



$TS^a \| TS^h \| TS^l$

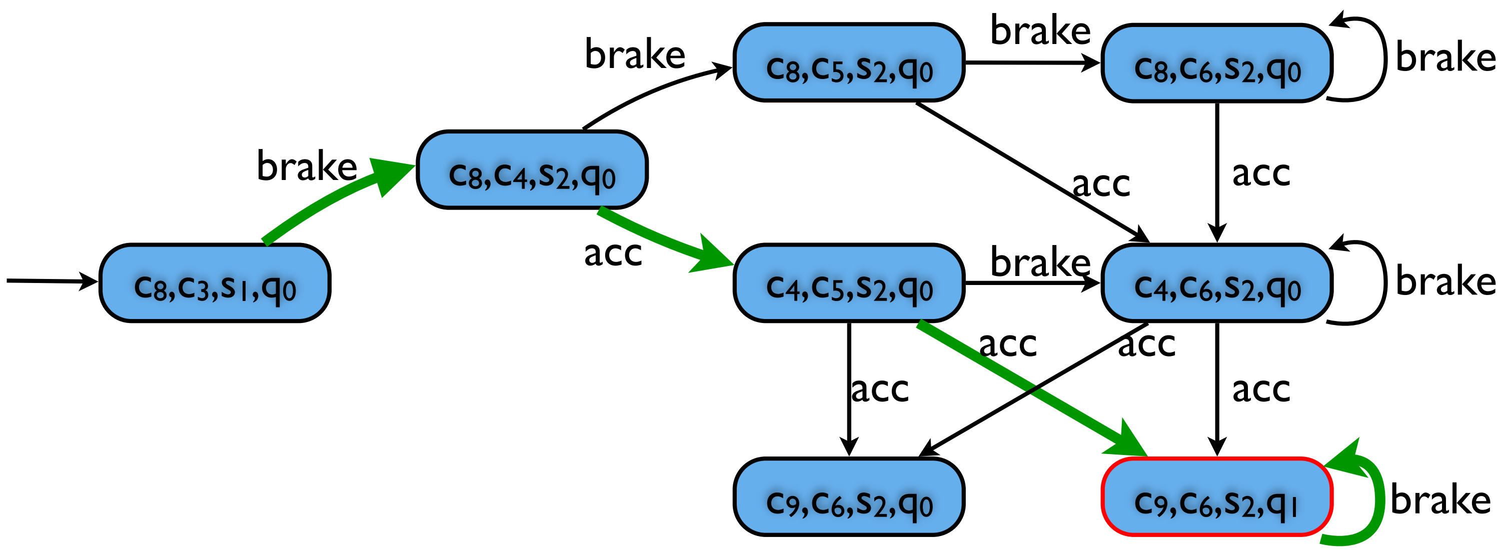


$(TS^a \| TS^h \| TS^l) \otimes \mathcal{A}$



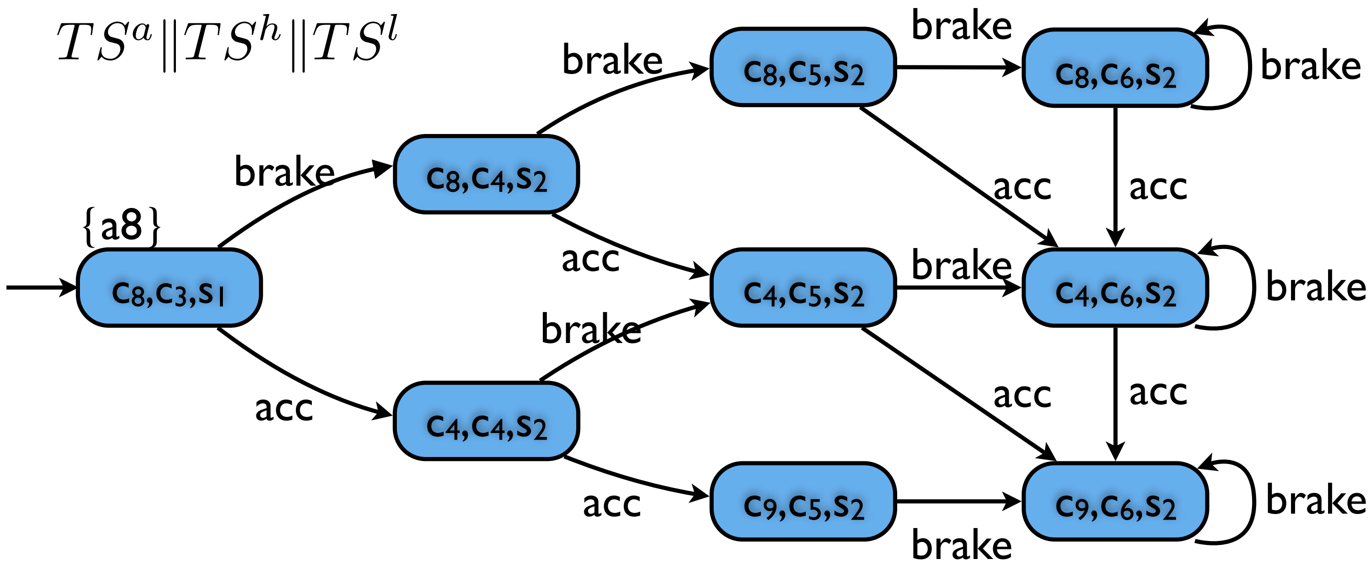
Unprotected left turn: solution

$(TS^a \| TS^h \| TS^l) \otimes \mathcal{A}$

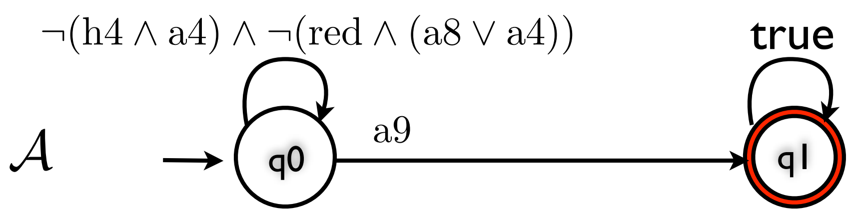


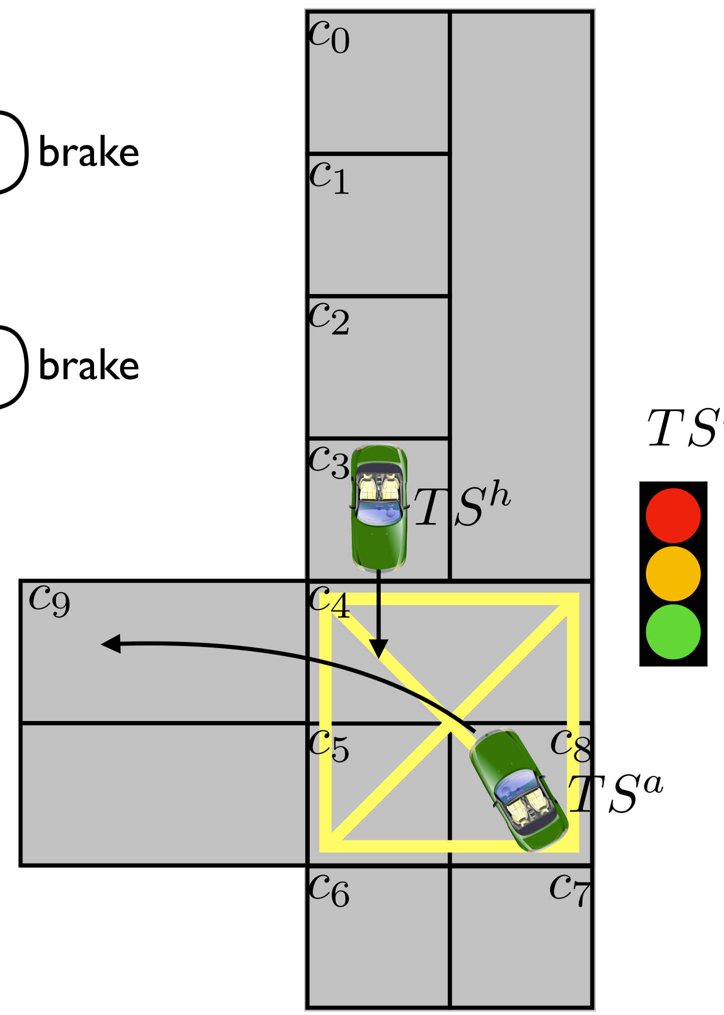
$$\pi = (c_8, c_3, s_1)(c_8, c_4, s_2)(c_4, c_5, s_2)(c_9, c_6, s_2)^\omega$$

Example 2: unprotected left turn with traffic light rule



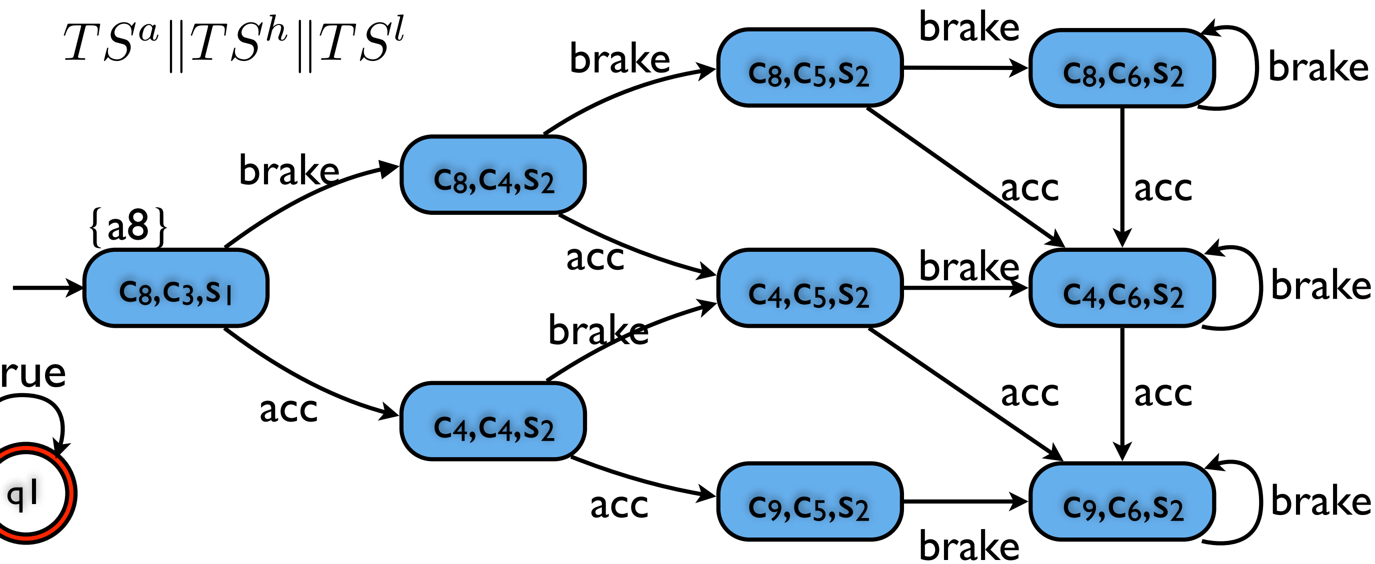
$$\varphi = \Big(\neg(\mathrm{h4} \wedge \mathrm{a4}) \wedge \neg(\mathrm{red} \wedge (\mathrm{a8} \vee \mathrm{a4})\Big) \ \mathbf{U} \ \mathrm{a9}$$



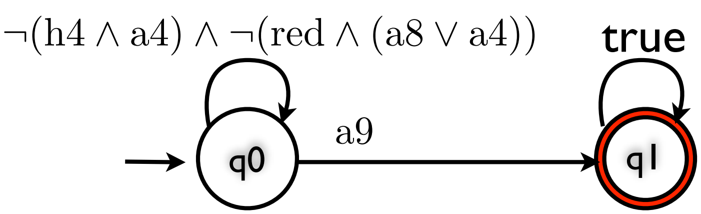


Unprotected left turn with traffic light rule: product automaton

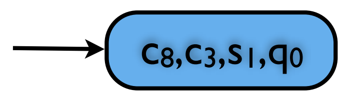
$TS^a \| TS^h \| TS^l$



$\mathcal{A}$



$(TS^a \| TS^h \| TS^l) \otimes \mathcal{A}$



No feasible controller!