Protocol Verification

A Brief Introduction to Model Checking and Temporal Logic

Andrés Goens (U. of Amsterdam)

SPLV 2024 @ Strathclyde

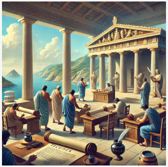
29.07.2024 - 02.08.2024

Motivation

Protocol Verification?

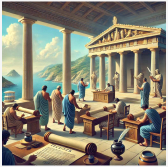
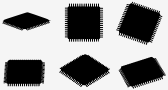
Protocols

Examples of protocols

- Distributed systems (e.g. paxos) 

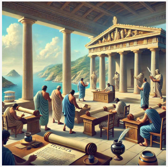
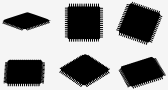
Protocols

Examples of protocols

- Distributed systems (e.g. paxos) 
- Hardware (e.g. cache coherence) 

Protocols

Examples of protocols

- Distributed systems (e.g. paxos) 
- Hardware (e.g. cache coherence) 
- Cryptographic protocols (e.g. TLS) 

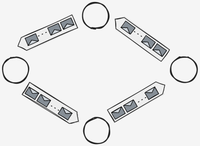
Verification

Examples of properties

- Fairness 

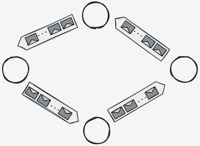
Verification

Examples of properties

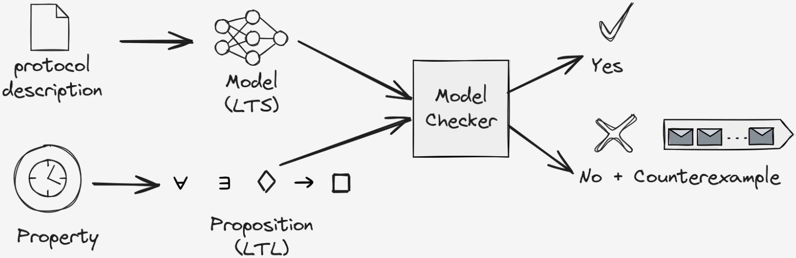
- Fairness 
- Deadlock-freedom 

Verification

Examples of properties

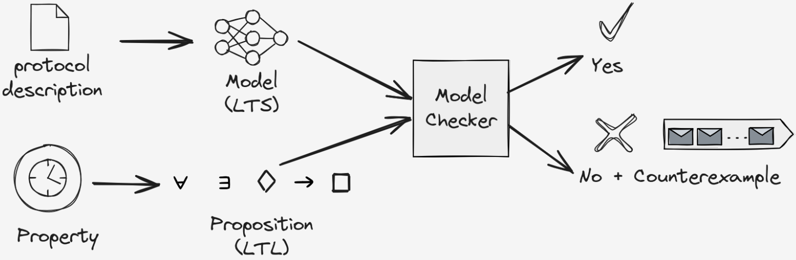
- Fairness 
- Deadlock-freedom 
- Safety 

What this course is about



Protocol Verification

What this course is about



This is what we'll cover

Overview of the course

What you *will* (hopefully) know by the end

- Labeled transition systems (LTS)
- Modeling languages (promela)
- (Propositional) Linear Temporal Logic (LTL)
- Examples!

Overview of the course

What you *will* (hopefully) know by the end

- Labeled transition systems (LTS)
- Modeling languages (promela)
- (Propositional) Linear Temporal Logic (LTL)
- Examples!

What you will *not* (necessarily) know by the end

- Other logics (e.g. CTL*, μ calculus)
- How model checking works internally (decision procedures)

Modelling Protocols

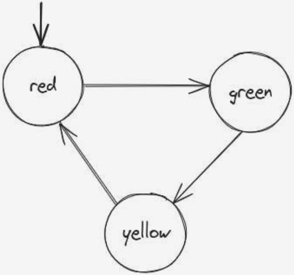
Labeled Transition Systems

Definition (Labeled Transition Systems)

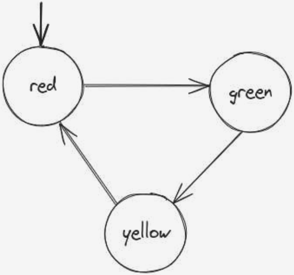
A labeled transition system is a tuple of the form $(S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$, where S is a set of states, $S_0 \subseteq S$ a subset of initial states, Act is a set (of actions), $\rightarrow \subseteq \mathrm{Act} \times S \times S$ is a (transition) relation, AP is a set (of atomic propositions) and $L : S \rightarrow \mathrm{Pow}(\mathrm{AP})$ is a (labeling) function.

Example: Traffic Light

Example: Traffic Light

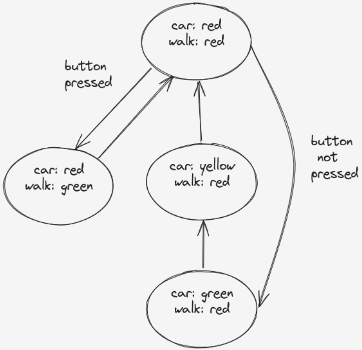


Example: Traffic Light

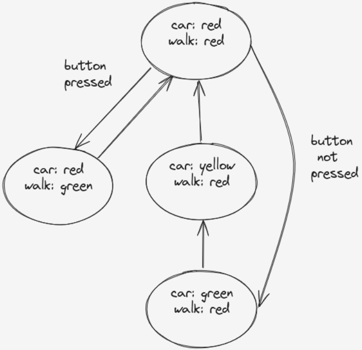


- $S = \{\text{red}, \text{green}, \text{yellow}\},\ S_0 = \text{red}$
- $\text{Act} = \{*\}$
- $\rightarrow = \{(*, \text{red}, \text{green}), (*, \text{green}, \text{yellow}), (*, \text{yellow}, \text{red})\}$
- $\text{AP} = L = \varnothing.$

Two Traffic Lights



Two Traffic Lights



- $\text{Act} = \{\epsilon, \text{button pressed}, \text{no button pressed}\}$
- $\text{AP} = \{\text{Pedestrians can go}, \text{Cars can go}\}$
- $L = \text{cars: red, walk: green} \mapsto \{\text{Pedestrians can go}\}, \ldots$

Interleaving

Two traffic lights $\nleftrightarrow$ One LTS

Interleaving

Two traffic lights $\nleftrightarrow$ One LTS

Definition (Interleaving)

Let $TS_i = (S_i, \mathrm{Act}_i, \rightarrow_i, S_{0,i}, \mathrm{AP}_i, L_i), i = 1, 2$ be two transition systems. We define the transition system $TS_1 \,|||\, TS_2 :=$ $(S_1 \times S_2, \mathrm{Act}_1 \times \mathrm{Act}_2, \rightarrow, S_{0,1} \times S_{0,2}, \mathrm{AP}_1 \cup \mathrm{AP}_2, L_1 \times L_2)$, where $L_1 \times L_2 : S_1 \times S_2 \rightarrow \mathrm{Pow}(\mathrm{AP}_1 \cup \mathrm{AP}_2)$ is defined as $(L_1 \times L_2)(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by

$$\frac{s_1 \rightarrow_1^\alpha s_1'}{(s_1, s_2) \rightarrow^\alpha (s_1', s_2)} \qquad \frac{s_2 \rightarrow_2^\alpha s_2'}{(s_1, s_2) \rightarrow^\alpha (s_1, s_2')} \; .$$

We call this construction the *interleaving* of TS_1 and TS_2.

Interleaving

Two traffic lights $\not\leftrightarrow$ One LTS

Definition (Interleaving)

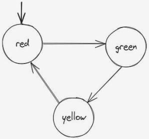
Let $TS_i = (S_i, \mathrm{Act}_i, \rightarrow_i, S_{0,i}, \mathrm{AP}_i, L_i), i = 1, 2$ be two transition systems. We define the transition system $TS_1 \,|||\, TS_2 :=$ $(S_1 \times S_2, \mathrm{Act}_1 \times \mathrm{Act}_2, \rightarrow, S_{0,1} \times S_{0,2}, \mathrm{AP}_1 \cup \mathrm{AP}_2, L_1 \times L_2)$, where $L_1 \times L_2 : S_1 \times S_2 \rightarrow \mathrm{Pow}(\mathrm{AP}_1 \cup \mathrm{AP}_2)$ is defined as $(L_1 \times L_2)(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$ and $\rightarrow$ is defined by

$$\frac{s_1 \rightarrow_1^\alpha s_1'}{(s_1, s_2) \rightarrow^\alpha (s_1', s_2)} \qquad \frac{s_2 \rightarrow_2^\alpha s_2'}{(s_1, s_2) \rightarrow^\alpha (s_1, s_2')} \ .$$

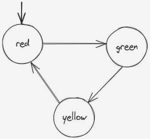
We call this construction the *interleaving* of TS_1 and TS_2.

Note that this means the two TS are *independent*

Example: Intearleaving

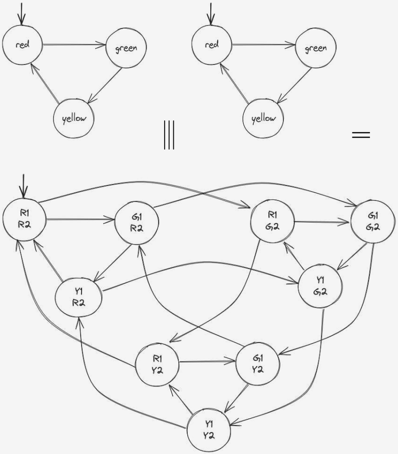


$|||$



$=$

Example: Intearleaving



Parallel Composition

Definition (Handshake)

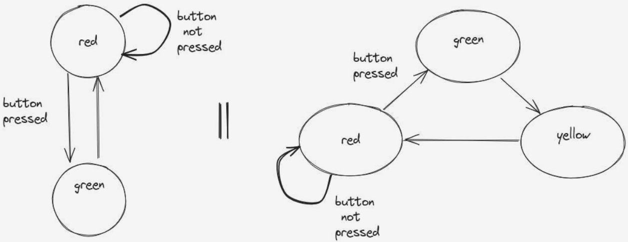
Let $TS_i = (S_i, \mathrm{Act}_i, \rightarrow_i, S_{0,i}, \mathrm{AP}_i, L_i), i = 1, 2$ be two transition systems and $H \subseteq \mathrm{Act}_1 \cap \mathrm{Act}_2$. We define the transition system $TS_1 \parallel_H TS_2 := (S_1 \times S_2, \mathrm{Act}_1 \times \mathrm{Act}_2, \rightarrow$ $, S_{0,1} \times S_{0,2}, \mathrm{AP}_1 \cup \mathrm{AP}_2, L_1 \times L_2)$, where $\rightarrow$ is defined by:

$$\frac{s_1 \rightarrow_1^\alpha s_1' \quad \alpha \notin H}{(s_1, s_2) \rightarrow^\alpha (s_1', s_2)} \qquad \frac{s_2 \rightarrow_1^\alpha s_2' \quad \alpha \notin H}{(s_1, s_2) \rightarrow^\alpha (s_1, s_2')}$$

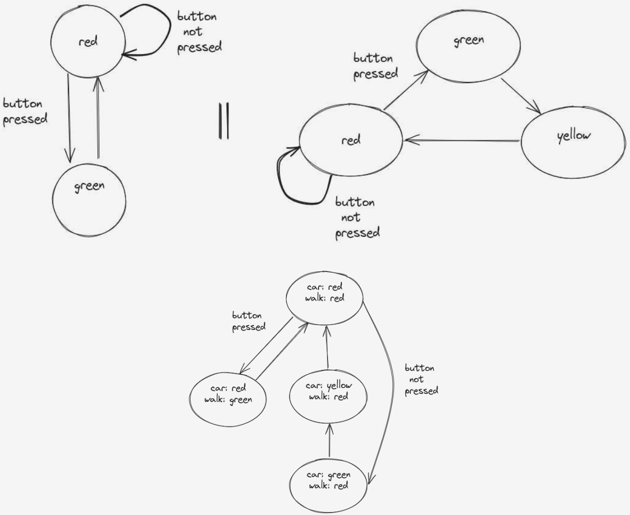
$$\frac{s_1 \rightarrow_1^\alpha s_1' \quad s_1 \rightarrow_1^\alpha s_1' \quad \alpha \in H}{(s_1, s_2) \rightarrow^\alpha (s_1', s_2')}$$

We call this the *parallel composition with handshake H*. When $H = \mathrm{Act}_1 \cap \mathrm{Act}_2$, we omit H.

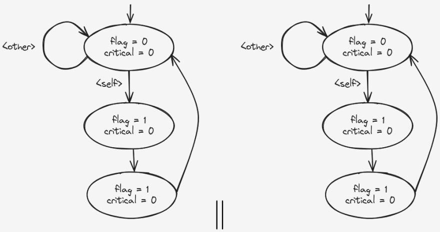
Two Traffic Lights, revisited



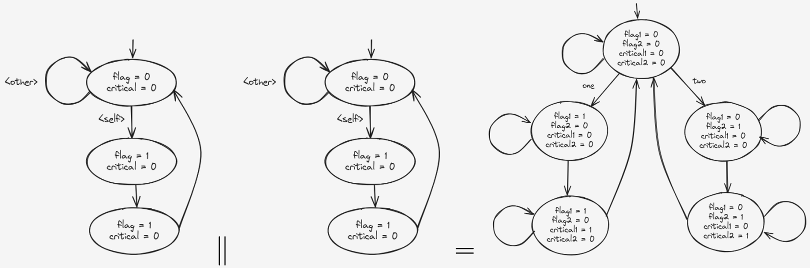
Two Traffic Lights, revisited



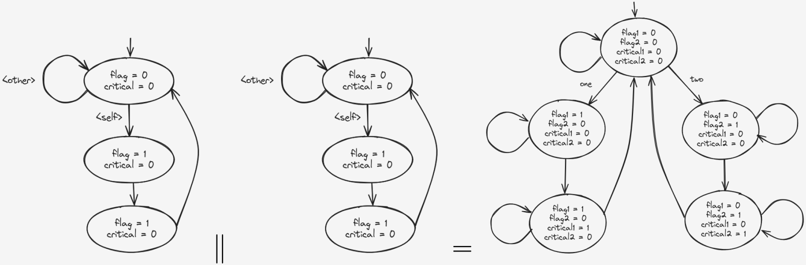
Concurrency: Message Passing



Concurrency: Message Passing

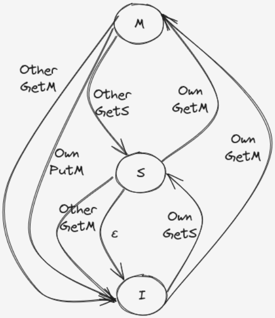


Concurrency: Message Passing

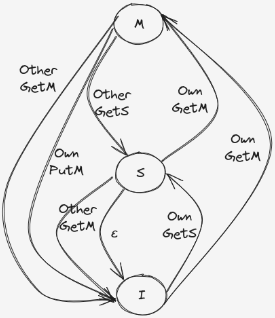


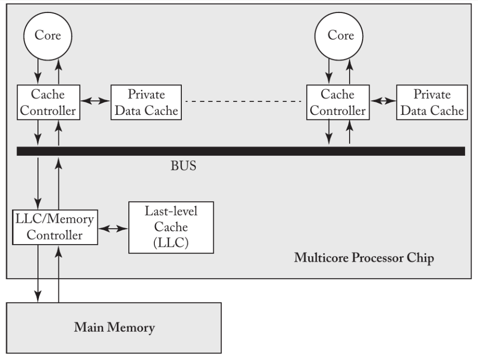
Assumption: atomicity of read-modify-writes here. Reasonable?

MSI Cache Coherency Protocol



MSI Cache Coherency Protocol





Source: Nagarajan, Vijay, et al. A primer on memory consistency and cache coherence. Springer Nature, 2020.

State Graph

- TS $\neq$ Graphs

State Graph

- TS $\neq$ Graphs
- Visualization (graphs): very useful!

State Graph

- TS $\neq$ Graphs
- Visualization (graphs): very useful!

Definition (Predecessors/Successors)

Let $TS = (S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$ be a transition system. For $s \in S, \alpha \in \mathrm{Act}$, we define $\mathrm{Post}(s, \alpha) := \{s' \in S \mid s \rightarrow^{\alpha} s'\}, \mathrm{Post}(s) := \bigcup_{\alpha \in \mathrm{Act}} \mathrm{Post}(s, \alpha)$ as the successors of s, and similarly Pre for the predecessors.

State Graph

- TS $\neq$ Graphs
- Visualization (graphs): very useful!

Definition (Predecessors/Successors)

Let $TS = (S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$ be a transition system. For $s \in S, \alpha \in \mathrm{Act}$, we define $\mathrm{Post}(s, \alpha) := \{s' \in S \mid s \rightarrow^{\alpha} s'\}, \mathrm{Post}(s) := \bigcup_{\alpha \in \mathrm{Act}} \mathrm{Post}(s, \alpha)$ as the successors of s, and similarly Pre for the predecessors.

Definition (State Graph)

Let $TS = (S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$ be a transition system. We call the directed graph $G(TS) = (S, E)$ the state graph of TS, where $E = \{s, s' \in S \times S \mid s \in S, s' \in \mathrm{Post}(s)\}$

Path Fragments

Definition (Path fragments)

Let $TS = (S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$ be a transition system. A sequence $\pi = \pi_0\pi_1\pi_2 \ldots \in (S)_{\mathbb{N}}$ is called a *path fragment* if $\pi_{i+1} \in \mathrm{Post}(\pi_i) \forall i \in \mathbb{N}$. It is called *finite* if it is a finite sequence $(\pi_i)_{i=0}^{N}$instead.
For a path fragment π, we denote the i-th element by $\pi[i]$ and similarly the sub-sequence $(\pi_k)_{k=i}^{j}$ by $\pi[i..j]$

Path Fragments

Definition (Path fragments)

Let $TS = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L)$ be a transition system. A sequence $\pi = \pi_0\pi_1\pi_2 \ldots \in (S)_{\mathbb{N}}$ is called a *path fragment* if $\pi_{i+1} \in \text{Post}(\pi_i)\forall i \in \mathbb{N}$. It is called *finite* if it is a finite sequence $(\pi_i)_{i=0}^{N}$instead.
For a path fragment π, we denote the i-th element by $\pi[i]$ and similarly the sub-sequence $(\pi_k)_{k=i}^{j}$ by $\pi[i..j]$

Sequences of transitions = path framgents through the state graph

Paths

Definition (Initial path fragment)

A path fragment π is called *initial*, if it starts at an initial statei, i.e. $\pi_0 \in S_0$.

Paths

Definition (Initial path fragment)

A path fragment π is called *initial*, if it starts at an initial statei, i.e. $\pi_0 \in S_0$.

Definition (Maximal path fragment)

A path fragment π is called a maximal, if it is not a proper prefix $\pi \subsetneq \pi'$ of another path fragment π', i.e. it cannot be extended.

Paths

Definition (Initial path fragment)

A path fragment π is called *initial*, if it starts at an initial statei, i.e. $\pi_0 \in S_0$.

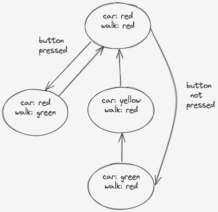
Definition (Maximal path fragment)

A path fragment π is called a maximal, if it is not a proper prefix $\pi \sqsubsetneq \pi'$ of another path fragment π', i.e. it cannot be extended.

Definition (Path)

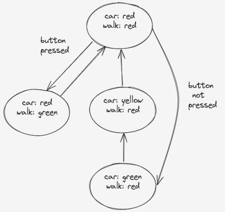
A path fragment π is called a *path* if it is initial and maximal.

Example: Paths in Traffic Light



A Typical Traffic Light in the UK?

Example: Paths in Traffic Light



A Typical Traffic Light in the UK?

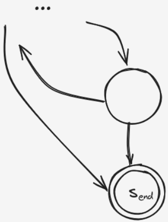
Non-example

Finite vs Infinite Paths

finite path fragments can be extended to infinite ones, but...

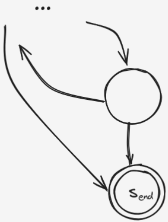
Finite vs Infinite Paths

finite path fragments can be extended to infinite ones, but...



Finite vs Infinite Paths

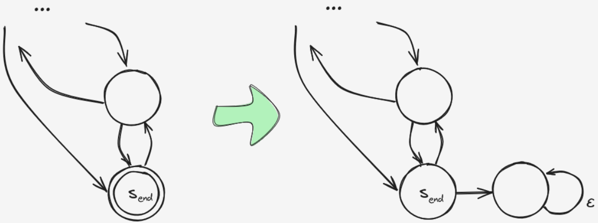
finite path fragments can be extended to infinite ones, but...



$$\text{Post}(s) = \varnothing$$

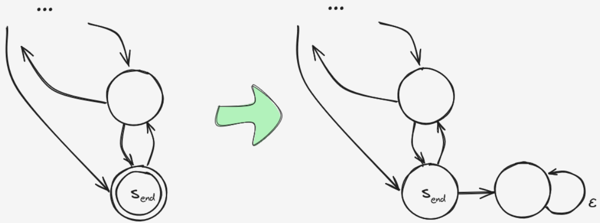
End States

Modeling end states with infinite paths



End States

Modeling end states with infinite paths



Assumption

For the rest of this course we assume no end states s with $\mathrm{Post}(s) = \emptyset$.