Model-Checking Finite-State Systems for Temporal Logic Properties

Deepak D'Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

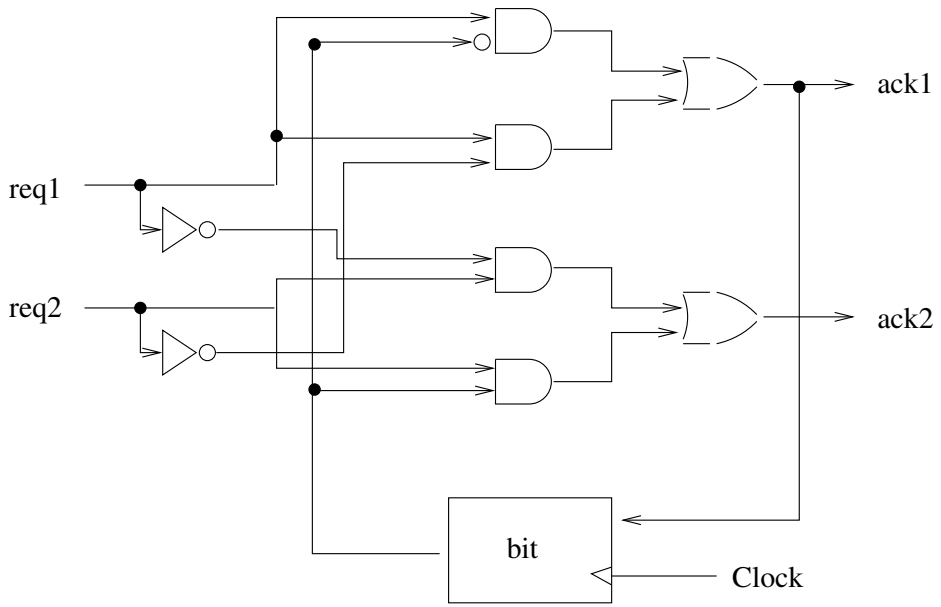
9 January 2018

Overview and Motivation

- Model systems as Transition Systems
- Specifying properties in Temporal Logic
- How we can check these properties algorithmically

Some example systems and properties: Arbiter Circuit

```
prio: context =
begin
  main: module =
  begin
     input req1, req2: boolean
     output ack1, ack2: boolean
     output bit: boolean
     definition
       ack1 = req1;
       ack2 = req2 AND (NOT req1)
     initialization
       bit = false;
     transition
       bit' = ack1
  end;
end
```



Does the circuit satisfy:

- mutual exclusion: "the bus is never granted simultaneously to *both* requesters".
- no starvation: "If Requester 2 asks for the bus continuously does

Some example systems and properties: Program

```
   ...
1. i := 0;
2. j := 0;
3. while (i < 100) {
4.   if (i = p)
5.     j := 1;
6.   i := i + 1;
7. }
   ...
```

Is the value of j always 1 when it the program exits the loop?

Some example systems and properties: Traffic Light

```
byte ctr = 0;
active proctype TrafficLight() {
  do
  :: if
     :: tick = false;
     :: tick = true;
     fi;
     if
     :: (status == GO) && (ctr == 3) && tick -> status = CHANGE; ctr = 0;
     :: (status == CHANGE) && (ctr == 1) && tick -> status = STOP; ctr = 0;
     :: (status == STOP) && (ctr == 3) && tick -> status = CHANGE; ctr = 0;
     :: else -> ctr = (tick -> (ctr + 1) % 4 :  ctr);
     fi;
     if
     :: status == GO -> light = GREEN;
     :: status == CHANGE -> light = AMBER;
     :: status == STOP -> light = RED;
     fi;
  od;
}
```

Whenever the light is RED does it becomes GREEN within 5 ticks?

Outline of this lecture

Transition systems: states

A state (over a set of variables *Var* with associated types) is a valuation for the variables in *Var*.
Thus a state is a map $s : Var \to Values$, that assigns to each variable x a value $s(x)$ in the domain of the type of x.

Example of a state

Consider $Var = \{loc, ctr\}$, with type of $loc = \{\texttt{sleep}, \texttt{try}, \texttt{crit}\}$, and type of $ctr = \mathbb{N}$.
Example state s: $\langle loc \mapsto \texttt{sleep},\ ctr \mapsto 2 \rangle$, depicted as:

$loc = sleep$

$ctr = 2$

Transition systems

A transition system is of the form $\mathcal{T} = (S, I, \rightarrow)$ where

- S is a set of states,
- $I \subseteq S$ is a set of initial states,
- $\rightarrow \subseteq S \times S$ is a transition relation.

A run or execution of $\mathcal{T}$ is a (finite or infinite) sequence of states $s_0, s_1, s_2, \ldots$ such that

- $s_0 \in I$, and
- for each i, $s_i \rightarrow s_{i+1}$.

Example transition system: a mod-4 counter

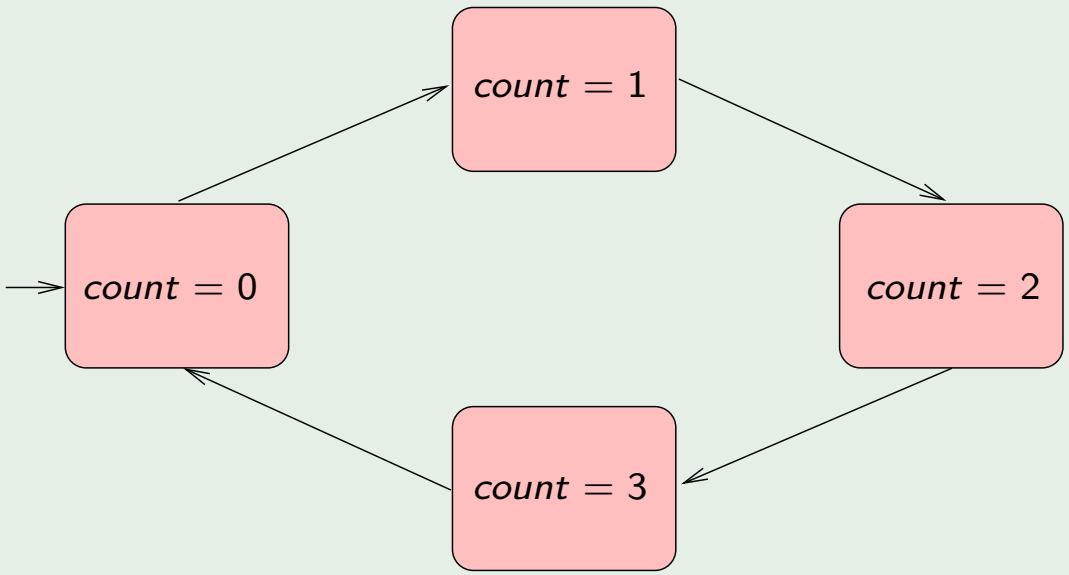
Transition system of a mod-4 counter

Here $Var = \{count\}$, with type of $count = \{0, 1, 2, 3\}$.

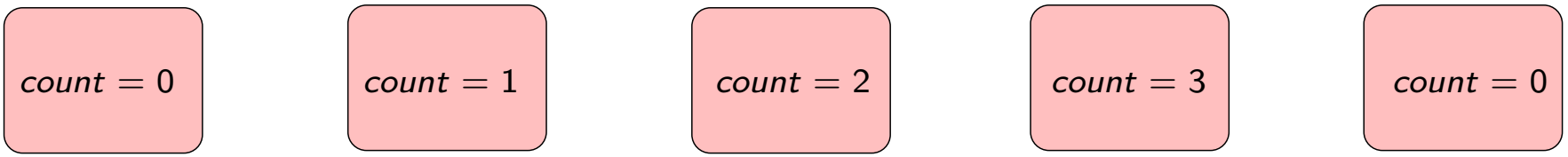
$$
\begin{array}{rlcl}
\mathcal{T} = & (S & = & \{\langle count \mapsto 0\rangle, \langle count \mapsto 1\rangle, \langle count \mapsto 2\rangle, \langle count \mapsto 3\rangle, \}, \\
& I & = & \{\langle count \mapsto 0\rangle\}, \\
& \rightarrow & = & \{(\langle count \mapsto 0\rangle, \langle count \mapsto 1\rangle), \\
& & & (\langle count \mapsto 1\rangle, \langle count \mapsto 2\rangle), \\
& & & (\langle count \mapsto 2\rangle, \langle count \mapsto 3\rangle), \\
& & & (\langle count \mapsto 3\rangle, \langle count \mapsto 0\rangle)\}).
\end{array}
$$

Example transition system: a mod-4 counter

Diagrammatic representation



Example run:



Overview of Spin

Spin is model-checking tool, in which we can

- Describe transition system models.
  - Suited for concurrent protocols, supports different synchronization constructs.
- Simulate them, explore paths in them.
- Describe desirable properties of the system in temporal logic.
- Check that the system satisfies these properties.
  - Proves that property is satisfied
  - Produces counter-examples (execution that violates property).

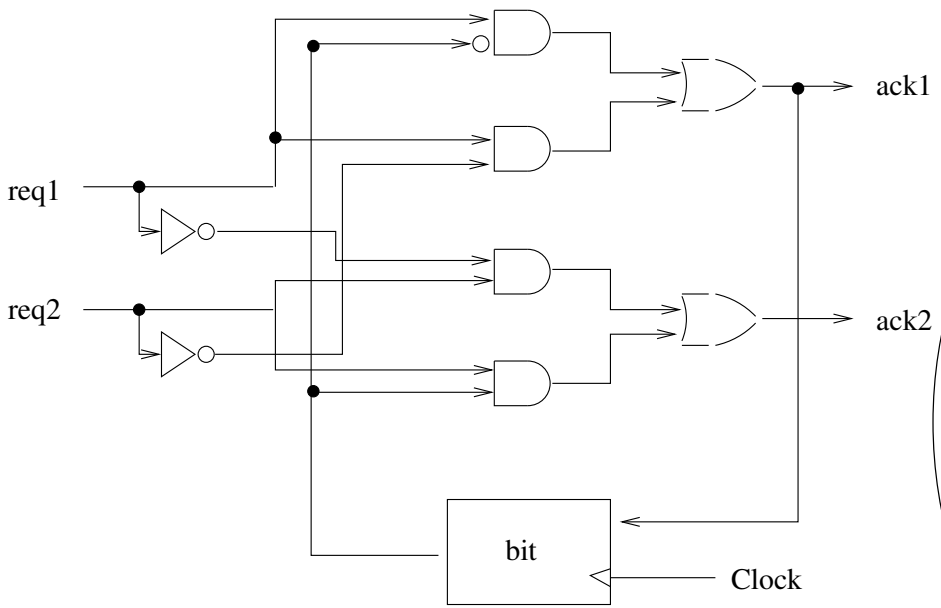
Mod-4 counter in Spin

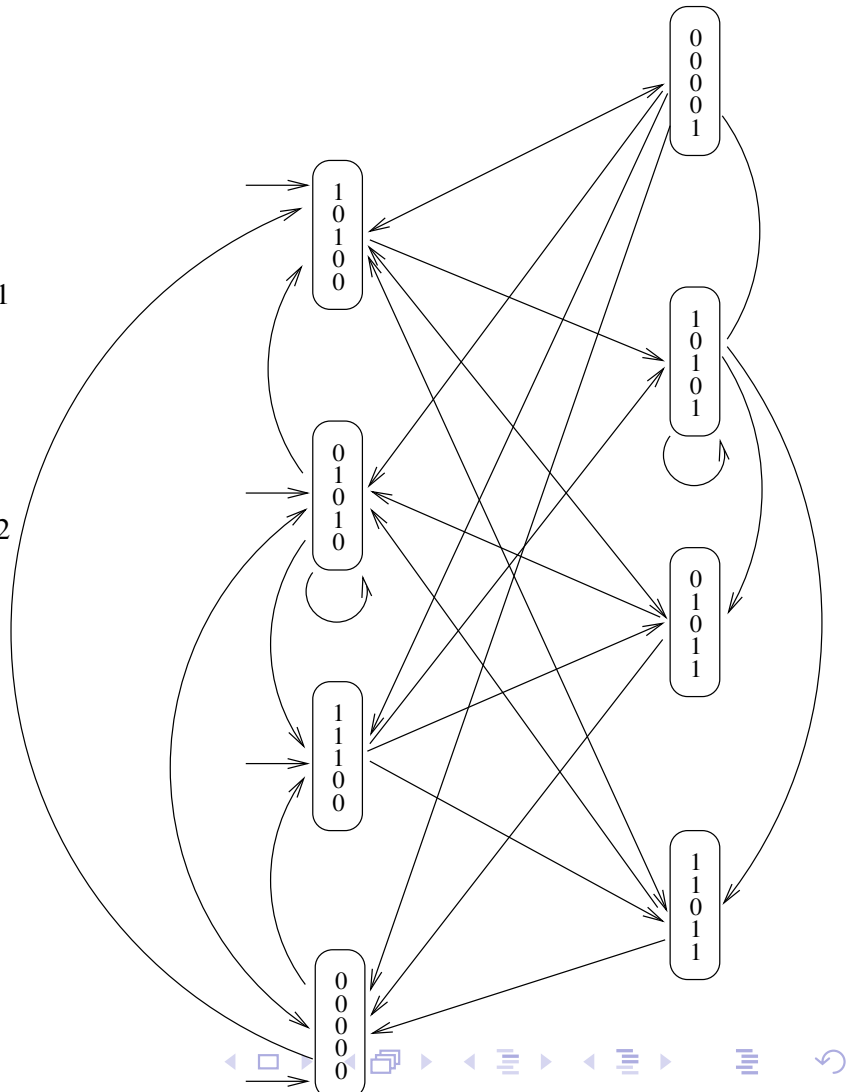
```
byte count = 0;

proctype counter() {
  do
  :: true -> count = (count + 1) % 4;
  od
}

init {
  run counter();
}
```

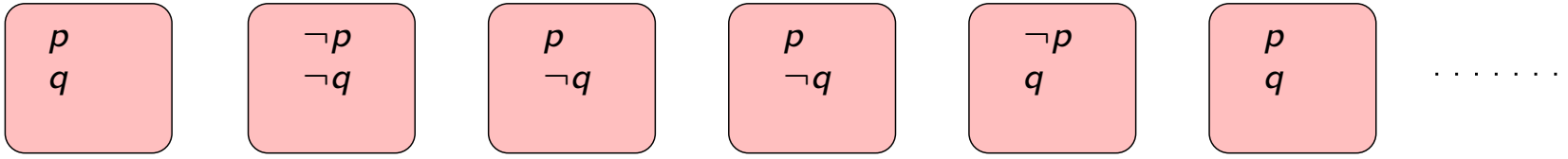
Examples: Transition System for Arbiter Circuit





Property specifications in Temporal Logic

- Linear-time Temporal Logic (LTL) proposed by Amir Pnueli in 1978 to specify properties of program executions.
- What can we say in LTL? An LTL formula describes a property of an infinite sequence of "states."
  - p: an atomic proposition p (like "$count = 2$" or "$tick = false$") holds in the current state.
  - Xp ("next p"): property p holds in the tail of the sequence starting from the next state.
  - Fp ("future p"): property p holds eventually at a future state.
  - Gp ("globally p"): property p holds henceforth (at all future states).
  - $U(p, q)$ ("p Until q"): property q holds eventually and p holds till then.



Syntax and semantics of LTL

Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid U(\varphi, \varphi).$$

Semantics: Given a finite sequence of states $w = s_0 s_1 \cdots s_n$, and a position $i \in \{0, 1, \ldots\ n\}$, we define the relation $w, i \models \varphi$ inductively as follows:

$$\begin{array}{lll}
w, i \models p & \text{iff} & p \text{ holds true in } s_i. \\
w, i \models \neg\varphi & \text{iff} & w, i \not\models \varphi. \\
w, i \models \varphi \vee \psi & \text{iff} & w, i \models \varphi \text{ or } w, i \models \psi. \\
w, i \models X\varphi & \text{iff} & i < n \text{ and } w, i+1 \models \varphi. \\
w, i \models U(\varphi, \psi) & \text{iff} & \exists j \leq n : \ i \leq j, \ w, j \models \psi, \text{ and} \\
 & & \forall k : i \leq k < j, \ w, k \models \varphi.
\end{array}$$

$F\varphi$ is shorthand for $U(\mathit{true}, \varphi)$, and $G\varphi$ is shorthand for $\neg(F\neg\varphi)$.

When a system model satisfies an LTL property

If $\mathcal{T}$ is a transition system and φ is an LTL formula with propositions that refer to values of variables in $\mathcal{T}$, then we say $\mathcal{T} \models \varphi$ (read "$\mathcal{T}$ satisfies φ") iff each infinite execution of $\mathcal{T}$ satisfies φ in its initial state.

Example properties for counter model

```
byte count = 0;

proctype counter() {
  do
  :: true -> count = (count + 1) % 4;
              assert (count <= 3);
  od
}

init {
  run counter();
}

ltl prop1 { [](count <= 3) };
ltl inc { []((count == 1) -> X(count == 2)) }
ltl prop3 { ((count == 0) || (count == 1)) U (count == 2));
ltl prop4 { [](count == 0) };
```

Traffic light model in Spin

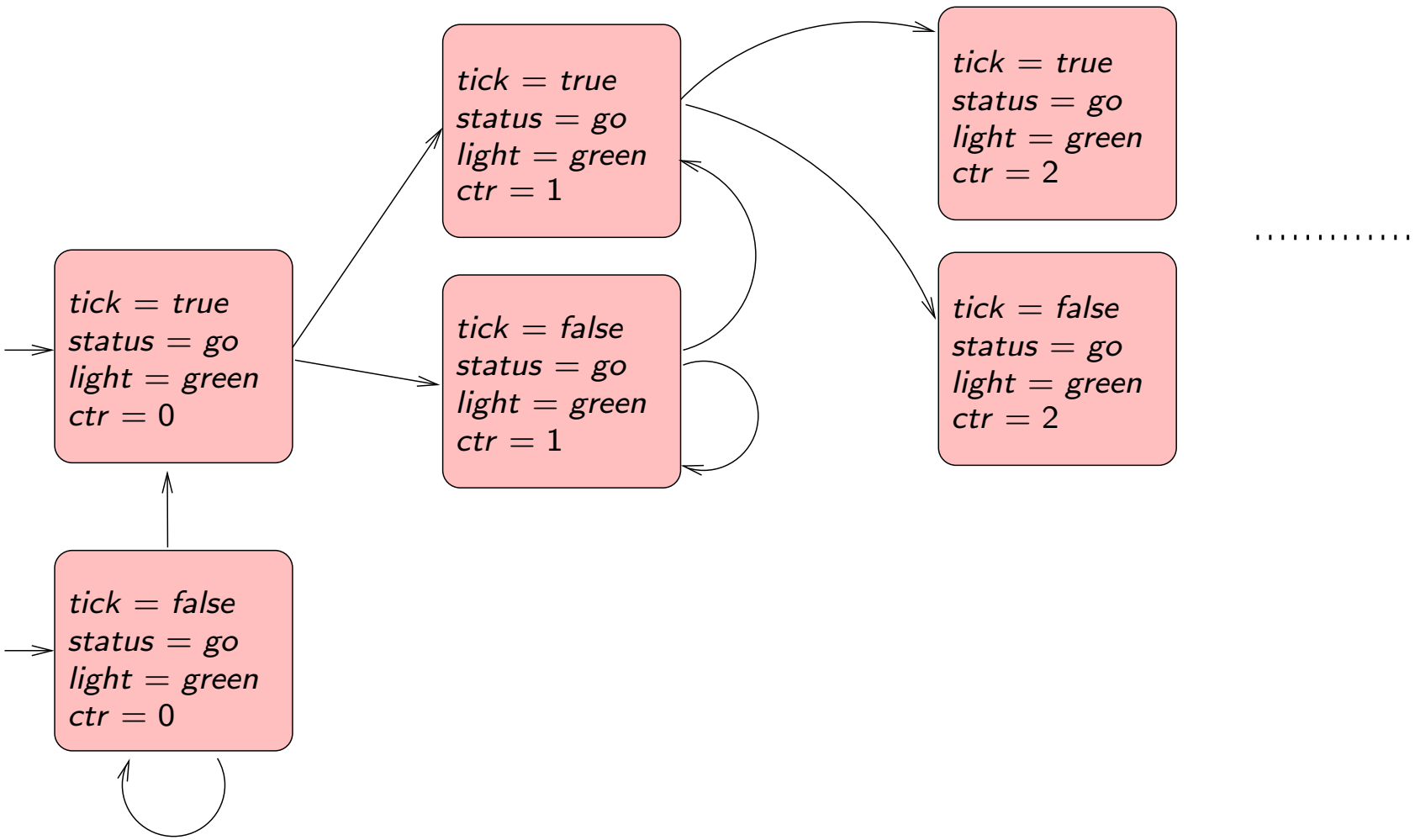
```
mtype = { GREEN, AMBER, RED };
mtype = { GO, CHANGE, STOP };

bool tick = false;
mtype status = GO;
mtype light = GREEN;
byte ctr = 0;

active proctype TrafficLight() {
  do
  :: if
     :: tick = false;
     :: tick = true;
     fi;

     if
     :: (status == GO) && (ctr == 3) && tick -> status = CHANGE; ctr = 0;
     :: (status == CHANGE) && (ctr == 1) && tick -> status = STOP; ctr = 0;
     :: (status == STOP) && (ctr == 3) && tick -> status = CHANGE; ctr = 0;
     :: else -> ctr = (tick -> (ctr + 1) % 4 :  ctr);
     fi;
     if
    :: status == GO -> light = GREEN;
    :: status == CHANGE -> light = AMBER;
    :: status == STOP -> light = RED;
     fi;
  od;
}
ltl liveness { [](light == RED) -> <>(light == GREEN)) };
ltl sequence { [](light == RED) U ((light == AMBER) U (light == GREEN))) };
```

Transition system for traffic light (partial)



Exercise

1. Which of the properties below are true of the traffic light model?

```
G((light = red) => F(light = green));

G((light = red) U ((light = amber) U (light = green)));
```

Exercise

1. Which of the properties below are true of the traffic light model?

```
G((light = red) => F(light = green));

G((light = red) U ((light = amber) U (light = green)));
```

2. Fix model based on error trail found by Spin.

Exercise

1. Which of the properties below are true of the traffic light model?

```
G((light = red) => F(light = green));

G((light = red) U ((light = amber) U (light = green)));
```

2. Fix model based on error trail found by Spin.
3. Give modified properties that the system satisfies.

Syntax and semantics of LTL

Syntax:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid X\varphi \mid \varphi U \varphi.$$

Semantics: Given an infinite sequence of states $w = s_0 s_1 \cdots$, and a position $i \in \{0, 1, \ldots\}$, we define the relation $w, i \models \varphi$ inductively as follows:

$$\begin{array}{lll}
w, i \models p & \text{iff} & p \text{ holds true in } s_i. \\
w, i \models \neg\varphi & \text{iff} & w, i \not\models \varphi. \\
w, i \models \varphi \vee \psi & \text{iff} & w, i \models \varphi \text{ or } w, i \models \psi. \\
w, i \models X\varphi & \text{iff} & w, i+1 \models \varphi. \\
w, i \models \varphi U \psi & \text{iff} & \exists j : \ i \leq j, \ w, j \models \psi, \text{ and} \\
 & & \forall k : i \leq k < j, \ w, k \models \varphi.
\end{array}$$

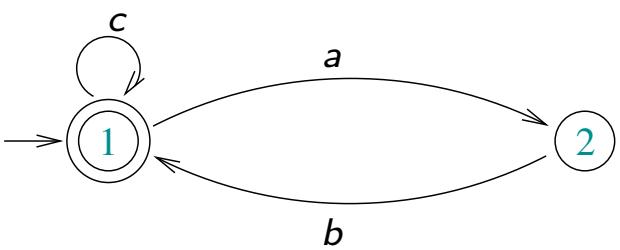
$F\varphi$ is shorthand for $true U \varphi$, and $G\varphi$ is shorthand for $\neg(F\neg\varphi)$.

When a system model satisfies an LTL property

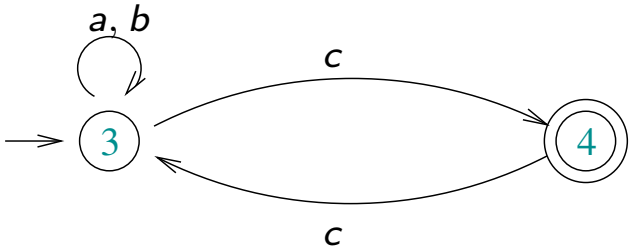
If $\mathcal{T}$ is a transition system and φ is an LTL formula with propositions that refer to values of variables in $\mathcal{T}$, then we say $\mathcal{T} \models \varphi$ (read "$\mathcal{T}$ satisfies φ") iff each infinite execution of $\mathcal{T}$ satisfies φ in the initial position.

Model-Checking Algo: Idea

Can we give an algorithm to decide if $L(\mathcal{A}) \subseteq L(B)$?



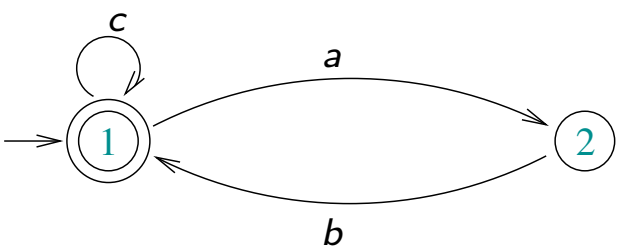
$\mathcal{A}$



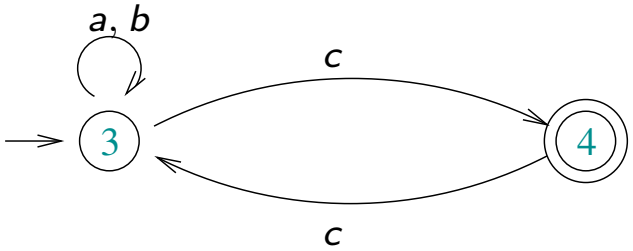
$\mathcal{B}$

Model-Checking Algo: Idea

Can we give an algorithm to decide if $L(\mathcal{A}) \subseteq L(B)$?

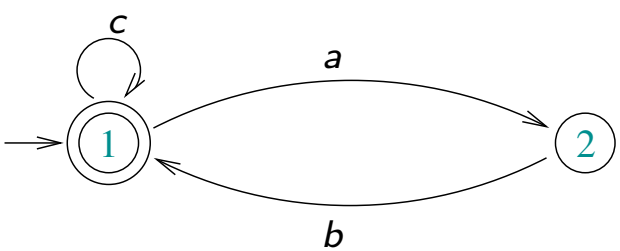


$\mathcal{A}$

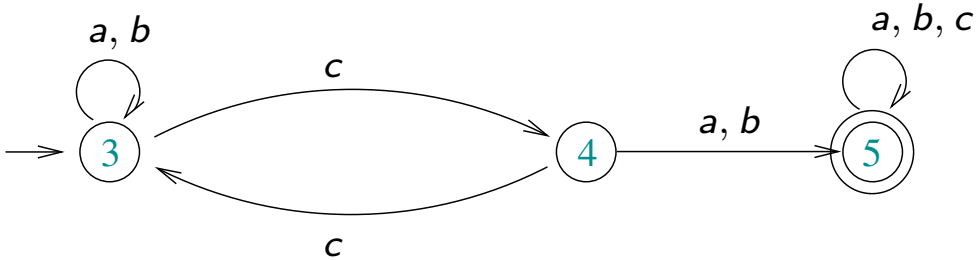


$\mathcal{B}$

First complement $\mathcal{B}$:



$\mathcal{A}$



$\overline{\mathcal{B}}$

Then construct the "product" of $\mathcal{A}$ and $\overline{\mathcal{B}}$:

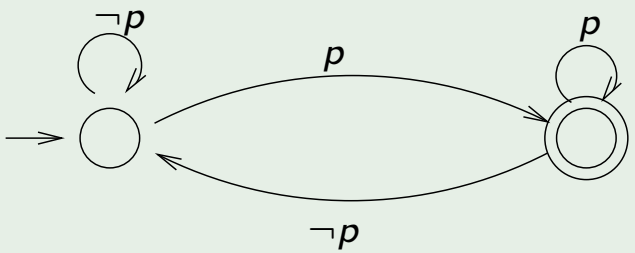
LTL models as sequences of propositional valuations

- LTL can be interpreted over a sequence of valuations to the propositions used in the formula.
  - E.g. In the formula `G((count = 1) => X(count = 2))`, `count = 1` and `count = 2` are the only propositions (say p and q), and a state can be viewed as a valuation to these propositions
- Example propositional valuation: $\langle p \mapsto \mathit{true},\ q \mapsto \mathit{false} \rangle$.
- We represent such a valuation as simply $\{p\}$ (that is the subset of propositions that are true).
- Further use a propositional formula (like $p \vee q$) to represent *sets* of propositional valuations, namely those in which the formula is true.
  - E.g. $p \vee q$ represents the 3 valuations $\{p, q\}$, $\{p\}$, and $\{q, \}$.

Compiling LTL properties into automata

Every LTL property φ over a set of propositions P can be expressed in the form of a (Büchi) automaton $\mathcal{A}_\varphi$ over the the alphabet 2^P, that accepts precisely the models of φ.
Some examples over set of propositions $P = \{p, q\}$. The label "$\neg p$" is short for the set of labels $\{q\}$ and $\{\}$.

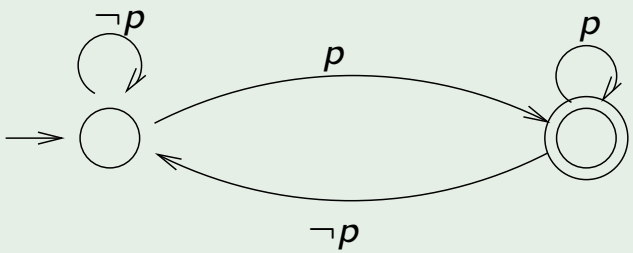
Automaton for $G(F(p))$



Compiling LTL properties into automata

Every LTL property φ over a set of propositions P can be expressed in the form of a (Büchi) automaton $\mathcal{A}_\varphi$ over the the alphabet 2^P, that accepts precisely the models of φ.
Some examples over set of propositions $P = \{p, q\}$. The label "$\neg p$" is short for the set of labels $\{q\}$ and $\{\}$.

Automaton for $G(F(p))$

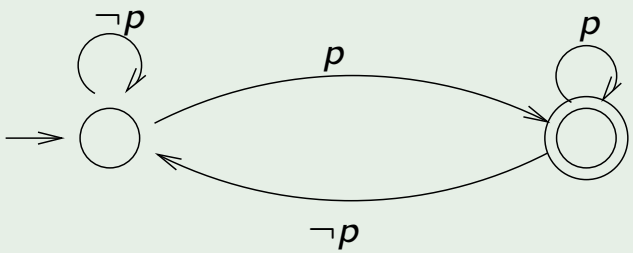


Automaton for pUq

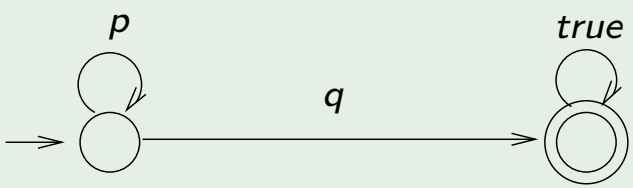
Compiling LTL properties into automata

Every LTL property φ over a set of propositions P can be expressed in the form of a (Büchi) automaton $\mathcal{A}_\varphi$ over the the alphabet 2^P, that accepts precisely the models of φ.
Some examples over set of propositions $P = \{p, q\}$. The label "$\neg p$" is short for the set of labels $\{q\}$ and $\{\}$.

Automaton for $G(F(p))$



Automaton for pUq



Model-checking LTL properties

Given a transition system $\mathcal{T}$ and an LTL property φ over a set of propositions P, we want to know whether $\mathcal{T} \models \varphi$ (i.e. do all infinite executions of $\mathcal{T}$ satisfy φ?).

- Compile given property φ into an automaton $\mathcal{A}_{\neg\varphi}$ accepting precisely the models of $\neg\varphi$.
- Take the "product" of $\mathcal{T}$ and $\mathcal{A}_{\neg\varphi}$. (Pair states t of $\mathcal{T}$ and A of $\mathcal{A}_{\neg\varphi}$ together iff the set of propositions p true in t is exactly $A \cap P$.)
- Look for an "accepting" path in this product.
- If such a path exists, this is a counter-example to the claim that $\mathcal{T}$ satisfies the property φ.
- If no such path exists, then $\mathcal{T}$ satisfies φ.

Exercise

Check if the arbiter model satisfies

$$G(req2 \implies F\, ack2)$$

- Construct a formula automaton that describes the models of the given formula.
- Construct the product of the arbiter transition system and formula automaton.
- Describe your counter-example if any.
- Use Spin to model the arbiter and assert the above property, and model-check it, and describe the counter-example reported by Spin.

Exercise

If p is the proposition "$count \neq 2$" then check if the mod-4 counter transition system satisfies the formula $\neg(pU\neg p)$.

- Construct the product of the mod-4 counter transition system and formula automaton for $pU\neg p$.
- Describe your counter-example if any.

Resources

Spin webpage: `http://spinroot.com/`
Current version: Spin v6.4.6
Useful documentation:

- Spin documentation (tutorial, reference manual, etc): `http://spinroot.com/spin/Man/`.
- Material for other topics:
  - Textbook by Huth and Ryan, *Logic in Computer Science*: Specifications, semantics, and model-checking techniques for LTL.