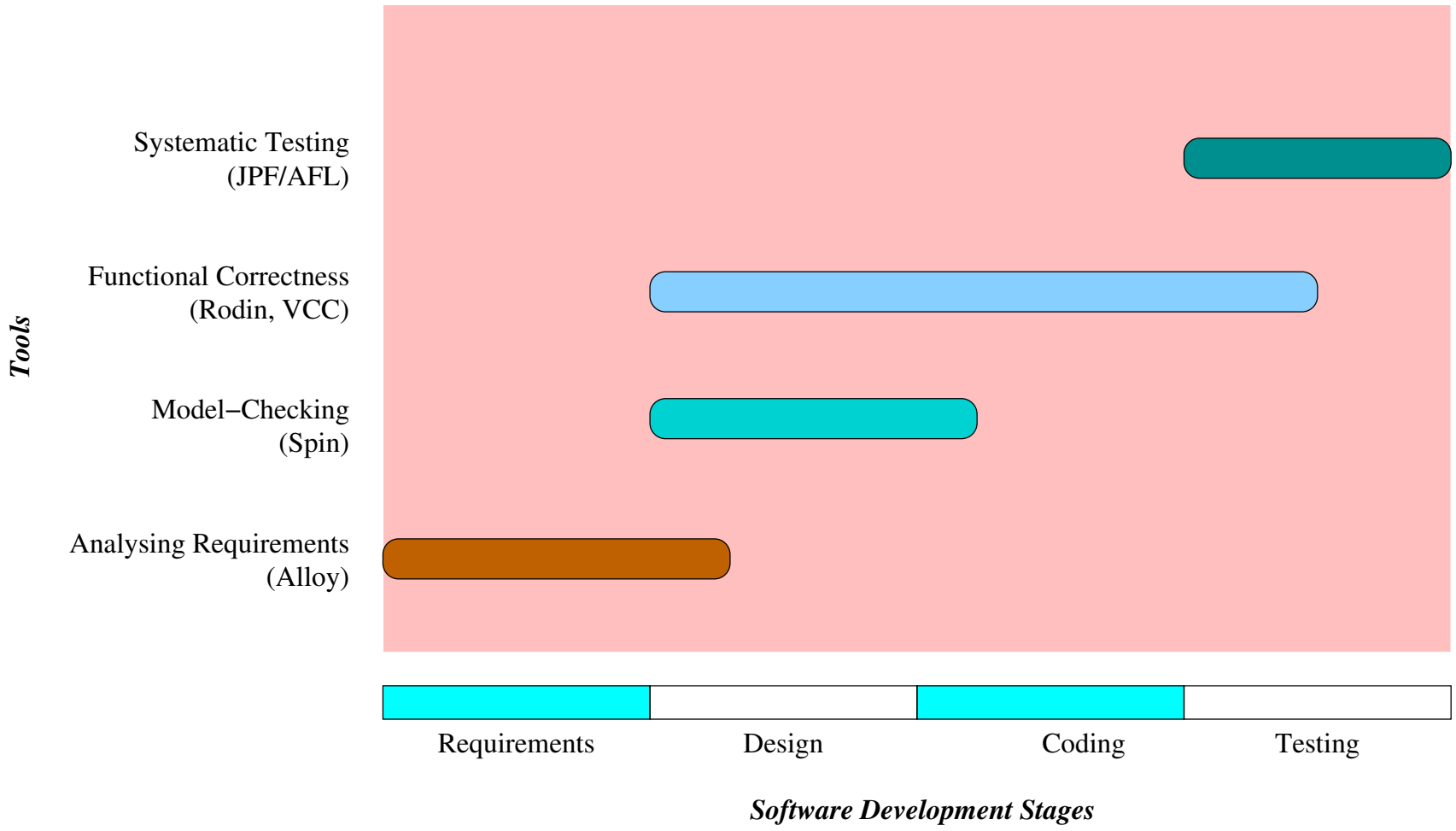
Introduction to Model-Checking using Spin

Deepak D'Souza

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

18, 23 January 2024

Methods and tools covered in this course



Model-checking using Spin: Plan of lectures

- Lecture 1 & 2: Intro to model-checking using Spin.
- Lecture 3 & 4: How LTL model checking works.

Outline of this lecture

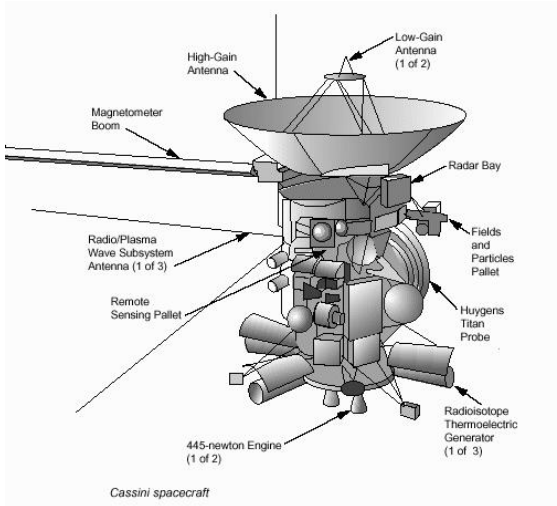
Overview of Spin

Spin is a 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).

Systems where Spin was successfully used

- Software control of Flood control barrier in The Netherlands. Verified the control algorithms.
- Call Processing logic of PathStar telephone switch: extraction of the model and verification of properties related to call-waiting/forwarding etc.
- Mission-Critical Software in Mars Rover and other space missions. Verification of resource (including motors) manager, hand-off protocols, etc.



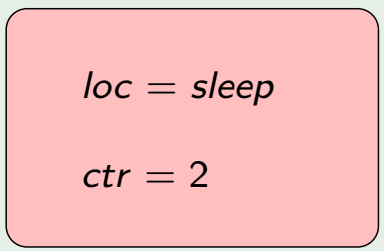
More details at `spinroot.com/spin/success.html`.

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:



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

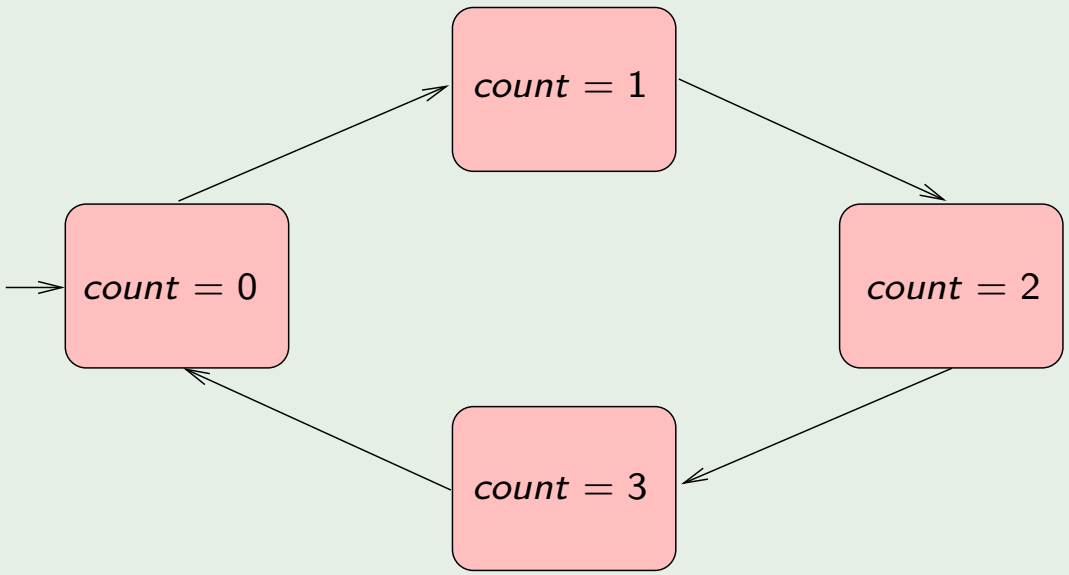
Transition system of a mod-4 counter

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

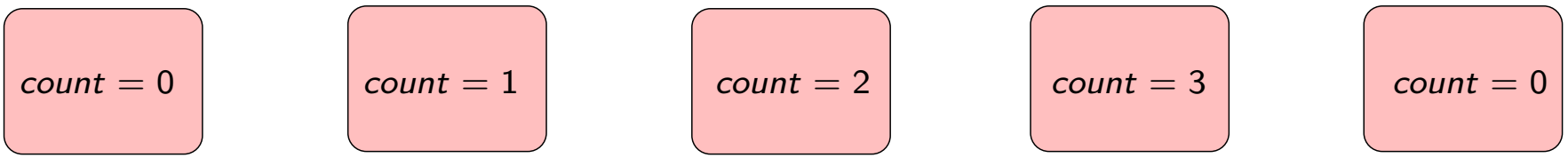
$$
\begin{array}{rlll}
\mathcal{T} = & (S & = & \{\langle count \mapsto 0\rangle, \langle count \mapsto 1\rangle, \langle count \mapsto 2\rangle, \langle count \mapsto 3\rangle\}, \\
& I & = & \{\langle count \mapsto 0\rangle\}, \\
& \rightarrow & = & \{(\langle count \mapsto 0\rangle, \langle count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

Example transition system: a mod-4 counter

Diagrammatic representation



Example run:

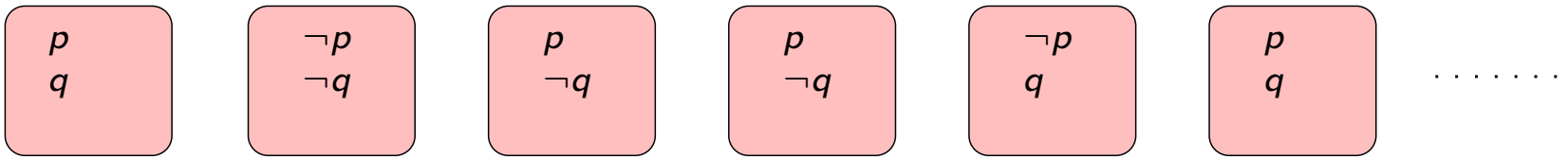


Mod-4 counter in Spin

```
byte count = 0;

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

init {
  run counter();
}
```

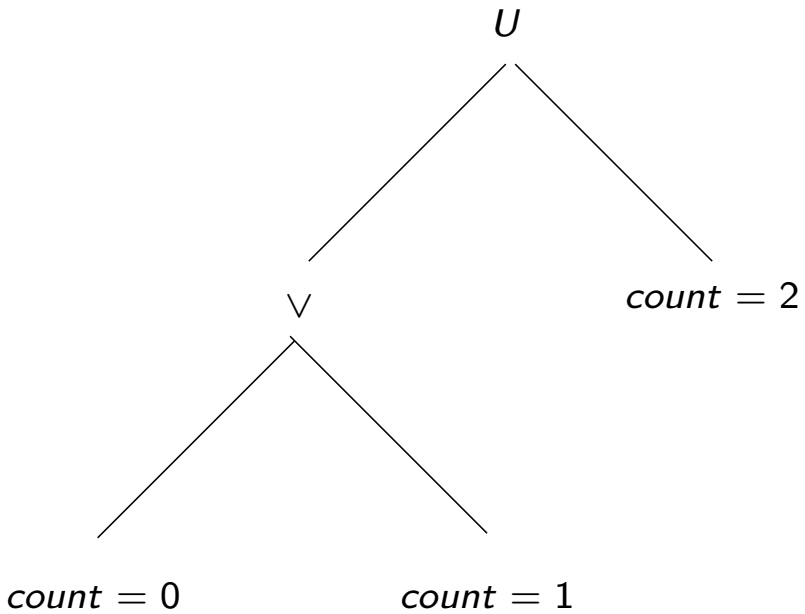
Property specifications in Temporal Logic

- 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.



Illustrating Semantics of an LTL formula

Example formula: $(count = 0 \vee count = 1)\, U\, (count = 2)$



Syntax and semantics of LTL

Syntax:

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

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:

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

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

When a system model satisfies an LTL property

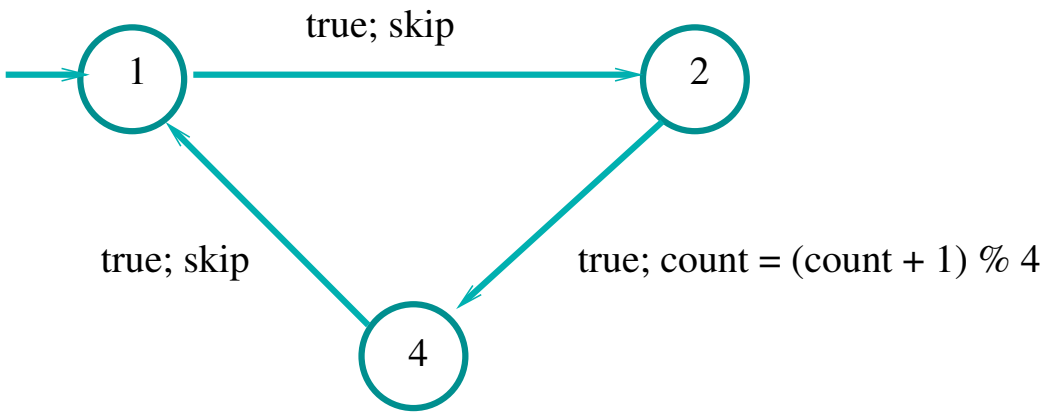
If $\mathcal{T}$ is a transition system and φ is an LTL formula with propositions that refer to values of variables in $\mathcal{T}$, then we say $\mathcal{T} \models \varphi$ (read "$\mathcal{T}$ satisfies φ") iff each infinite execution of $\mathcal{T}$ satisfies φ in its initial state.

Example properties for counter model

```
byte count = 0;

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

Transition System generated by Spin

Extended transition system generated by Spin:

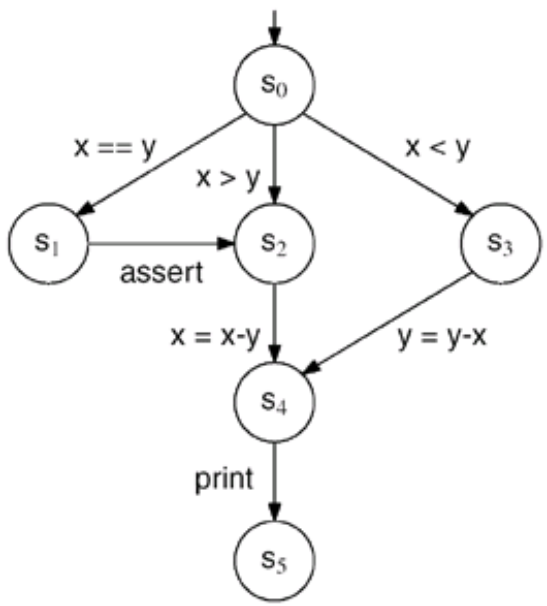


Corresponding transition system:

Transition System generated by Spin: Example from Spin Primer and Ref book

```
active proctype not_euclid(int x, y)
{
  if
  :: (x > y) -> L: x = x - y
  :: (x < y) -> y = y - x
  :: (x == y) -> assert (x != y); goto L
  fi;
  printf(";%d\n", x)
}
```

Corresponding transition system:



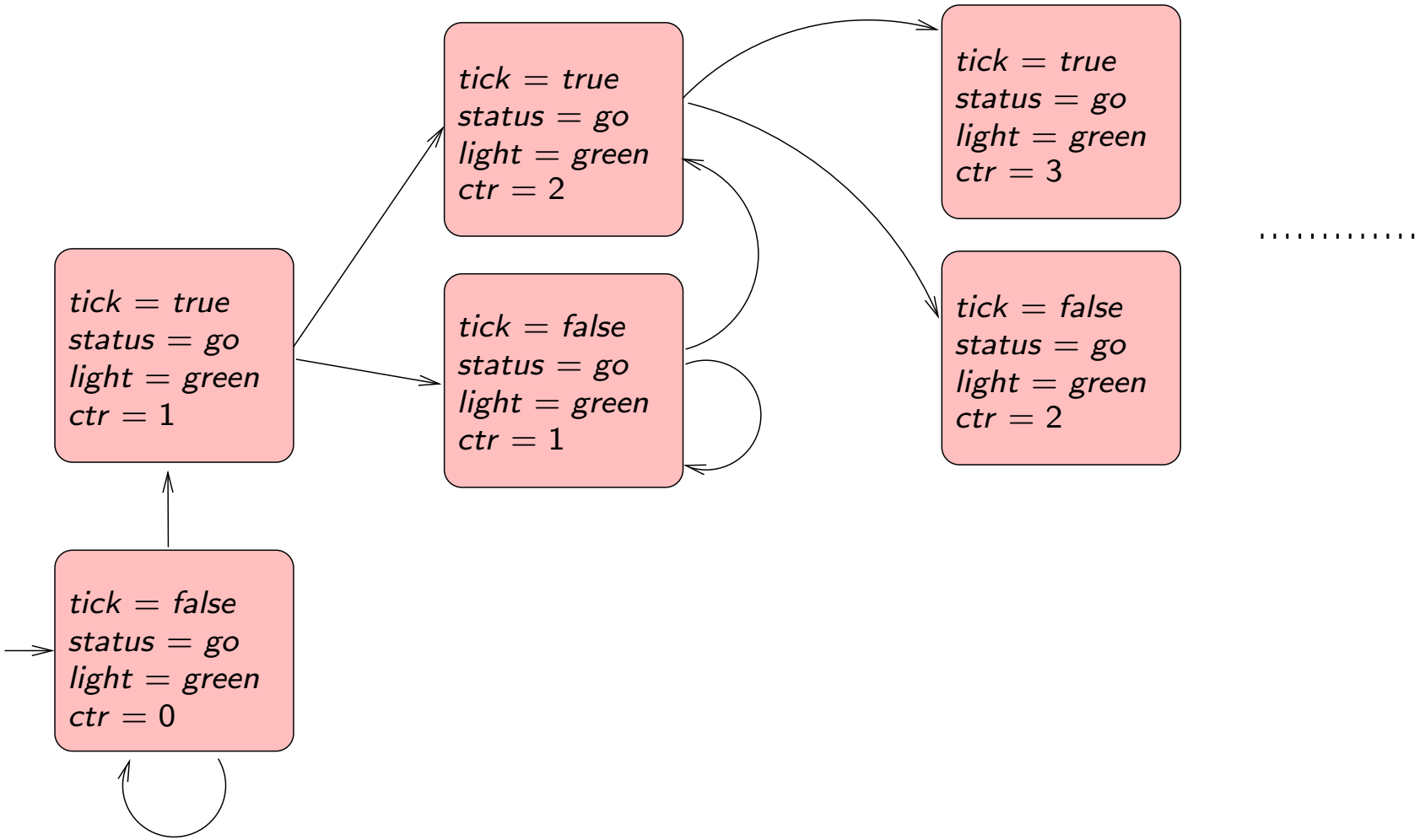
Example with inputs: Traffic light model

"Stop" says the red light, "Go" says the green.
"Change" says the amber light, blinking in between.
That's what they say, and that's what they mean.
We all must obey them, even the Queen!

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
  :: atomic {
     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.

Some example models

- Simple example modelling concurrency (`race.pml`, `inc-dec-lock.pml`)
- Example using channels for communication (`prod-con.pml`)
- Example of post-facto use of Spin (Detecting races in FreeRTOS)

Spin resources and other material

Spin webpage: `http://spinroot.com/`
Current version: Spin v6.5.1
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.