# Embedded Systems Specification and Design

David Kendall

# Introduction

- Recap of Propositional Logic
- Linear-time Temporal Logic (LTL)
- Specifying properties with LTL
- LTL specification patterns
- LTL, ProMela and SPIN

# Propositional Logic: Recap

## Definition (Proposition, Propositional Variable)

A proposition is a statement that is either true or false.

A propositional variable is a variable that has one of two possible values: true and false. Propositional variables are used to represent propositions.

- Propositions
  - "It rained in Newcastle on 17th October 2008".
  - "Mike Ashley is an excellent football club chairman"
  - `n <= 1`, `len(buffer) == MAX_BUF_SIZE`
- Propositional variables
  - rain, ashley, $p$, $q$, ...
- Not propositions
  - "Pass the salt"
  - "Does Joe Kinnear know what he's doing?"

# Propositional Logic: Recap

## Definition (Logical Connective)

Compound propositions can be formed from simpler propositions using logical connectives, whose meaning can be given by a truth table.

| p | q | $\neg p$ | $p \vee q$ | $p \wedge q$ | $p \Rightarrow q$ | $p \Leftrightarrow q$ |
|---|---|---|---|---|---|---|
| false | false | true | false | false | true | true |
| false | true | true | true | false | true | false |
| true | false | false | true | false | false | false |
| true | true | false | true | true | true | true |

# Propositions and Transition Systems

- Modify our view of LTS with the idea that each state provides a valuation for a set of propositional variables. Ignore labels.

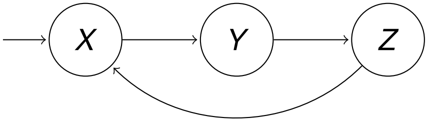
## Definition (Transition System)

A transition system is a tuple $(S, s^0, \longrightarrow, \mathit{Prop}, \mathcal{P})$, where

- $S$ is the set of states
- $s^0 \in S$ is the initial state
- $\longrightarrow \subseteq S \times S$ is the transition relation
- $\mathit{Prop}$ is the set of propositions
- $\mathcal{P} : S \rightarrow 2^{\mathit{Prop}}$ labels states with true propositions

# Propositions and Transition Systems

Consider the transition system *TE* given by




with $Prop = \{m\_is\_2, m\_is\_3, m\_is\_4, n\_is\_0, n\_is\_2, n\_is\_4\}$ and $\mathcal{P} = \{X \mapsto \{m\_is\_2, n\_is\_0\}, Y \mapsto \{m\_is\_3, n\_is\_2\}, Z \mapsto \{m\_is\_4, n\_is\_4\}\}$

# Propositions and Transition Systems

- *TE* may arise from the Promela model given by

```
active proctype P() {

  bool m_is_2 = true, m_is_3, m_is_4;
  bool n_is_0 = true, n_is_2, n_is_4;

  do
  :: m_is_2 ->
       d_step{m_is_2 = false; m_is_3 = true;
              n_is_0 = false; n_is_2 = true}
  :: m_is_3 ->
       d_step{m_is_3 = false; m_is_4 = true;
              n_is_2 = false; n_is_4 = true}
  :: m_is_4 ->
       d_step{m_is_4 = false; m_is_2 = true;
              n_is_4 = false; n_is_0 = true}
   od
}
```

# Propositions and Transition Systems

- or more naturally from

```
active proctype P() {
  byte m = 2;
  byte n = 0;

  do
  :: (n >= m) -> d_step{m = 2; n = 0}
  :: else     -> d_step{m = m + 1; n = n + 2}
  od
}
```

- so we will usually assume that the elements of *Prop* are simple propositions involving program variables.

# Traces

## Definition (Trace)

Given a transition system $T$, a trace of $T$ is a (possibly infinite) sequence of states $s_0, s_1, s_2, \ldots$ such that for $i > 0$, $s_{i-1} \longrightarrow s_i$.

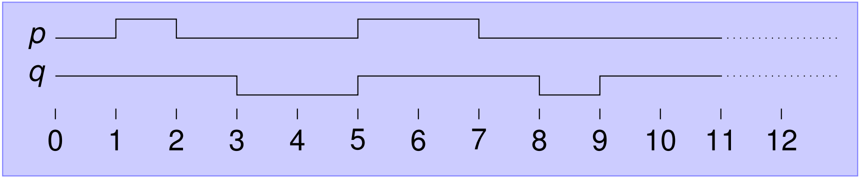
## Example

$s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, \ldots$ is a trace of the transition system $TE$ where

$$\begin{array}{ll} s_i = X, & \text{if } i \bmod 3 = 0 \\ s_i = Y, & \text{if } i \bmod 3 = 1 \\ s_i = Z, & \text{if } i \bmod 3 = 2 \end{array}$$

# Timing diagrams

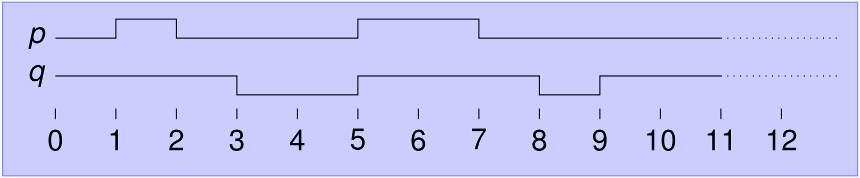
- Propositions are associated with the states in a trace.
- So they can change their truth value over time.
- Timing diagrams can be useful to show this change.




- Each propositional variable's value is shown as a line of the timing diagram.
- Line high – proposition is true.
- Line low – proposition is false.

# State formulas

- Now we can start to make formal statements about what happens in a trace.
- State formulas are the simplest statements about traces. They are propositional formulas constructed from *Prop* and the logical connectives; e.g. $n\_is\_4$, $m\_is\_3 \wedge n\_is\_2$, $p \Rightarrow q$, etc.
- For a trace $s$ and state formula $f$, we can ask whether $f$ is true at some state $s_i$ in $s$, if it is we write $s_i \models f$




For which $i$ do we have (a) $s_i \models p \wedge q$, (b) $s_i \models p \vee q$, and (c) $s_i \models p \Rightarrow q$?

# Temporal Connectives

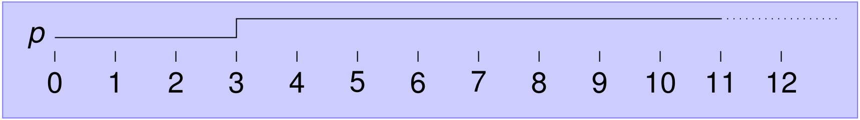
How can we say something about traces as they evolve over time?

- $\Box$ – **Always**: $\Box\phi$ is true at moment *i* if $\phi$ is true from moment *i* onwards
- $\Diamond$ – **Eventually**: $\Diamond\phi$ is true at moment *i* if $\phi$ is eventually true at *i* or some time later
- $\mathcal{U}$ – **Strong Until**: $\phi \, \mathcal{U} \, \psi$ is true at moment *i* if $\psi$ is eventually true and $\phi$ is true until then.
- $\mathcal{W}$ – **Weak Until**: Like strong until except we don't require that $\psi$ is eventually true

# Always ($\Box\phi$)

## Definition (Always)

$s_i \models \Box\phi$ iff $\forall j \geq i \bullet s_j \models \phi$

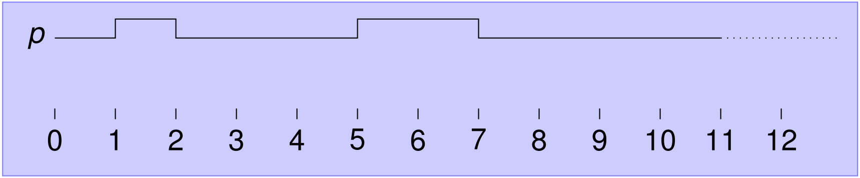



- $s_3 \models \Box p$
- $s_0 \not\models \Box p$

# Eventually ($\Diamond \phi$)

## Definition (Eventually)

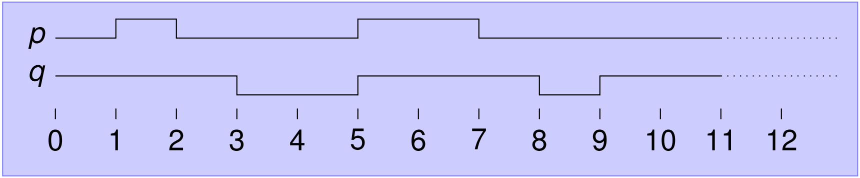
$s_i \models \Diamond \phi \text{ iff } \exists j \geq i \bullet s_j \models \phi$




- $s_0 \models \Diamond p$
- $s_6 \models \Diamond p$
- $s_7 \not\models \Diamond p$

# Exercises

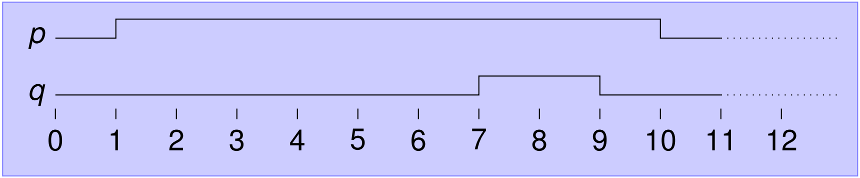
- Which of the following claims are true of the trace shown in the diagram below?
  1. $s_9 \models \Box q$
  2. $s_9 \models \Box \neg\, p$
  3. $s_0 \models \Box q$
  4. $s_0 \models \Box(p \Rightarrow q)$
  5. $s_9 \models \Diamond p$
  6. $s_0 \models \Diamond q$
  7. $s_0 \models \Diamond \neg\, (p \Rightarrow q)$



# Strong Until ($\phi\,\mathcal{U}\,\psi$)

## Definition (Strong Until)

$s_i \models \phi\,\mathcal{U}\,\psi$ iff $\exists\, k \geq i \bullet s_k \models \psi \wedge \forall\, j \mid i \leq j < k \bullet s_j \models \phi$




- $s_2 \models p\,\mathcal{U}\,q$
- $s_7 \models p\,\mathcal{U}\,q$
- $s_0 \not\models p\,\mathcal{U}\,q$
- $s_9 \not\models p\,\mathcal{U}\,q$

# Weak Until ($\phi \, \mathcal{W} \, \psi$)

- Define **weak until** in the style of the other definitions.
- Define **weak until** using the other temporal connectives.

# Classifying temporal properties

- **Safety** – "Nothing bad happens"
  $\Box\neg(\mathtt{in\_cs1} \wedge \mathtt{in\_cs2})$
- **Liveness** – "Something good happens"
  $\Box(\mathtt{request} \Rightarrow \Diamond\mathtt{receive})$
- **Fairness** – "Computation is fair"
  $\Box\Diamond\mathtt{coffee\_ordered} \Rightarrow \Box\Diamond\mathtt{coffee\_delivered}$

# Specification Patterns

- It is easier to write specifications using temporal logic rather than automata.
- But it is still not easy.
- Specification patterns can help – just as design patterns can help in programming.
- Alavi, H. et al., SPEC Patterns, 2005

# Main specification patterns for LTL

- Invariance ($\Box\neg\, P$)
  $\Box\neg\,(\texttt{systemState} == \texttt{DANGEROUS})$
  $\Box\neg\,((\texttt{crossing} == \texttt{OCCUPIED}) \wedge (\texttt{gate} == \texttt{OPEN}))$
- Bounded Response ($\Box(P \Rightarrow \Diamond Q)$)
  $\Box(\texttt{waitingForService} \Rightarrow \Diamond\texttt{serviceReceived})$
  $\Box(\texttt{facingObstacle} \Rightarrow \Diamond(\texttt{distanceFromObstacle} >= 5))$
- Nearly all of the specifications that you write will be either an *invariance* property or a *bounded response* property.
- Much more practice in writing LTL specifications to come in the lab sessions.

# Acknowledgements

These slides are based on material in Barland, I.; Vardi, M.; Greiner, J. *Model Checking Concurrent Programs*, Connexions Web site. http://cnx.org/content/col10294/1.3/, Oct 27, 2005.