COMP6210 Automated Software Verification

Linear Temporal Logic

Pavel Naumov

Intended Learning Outcomes

By the end of these two lectures, you will be able to

- formalise simple temporal properties in Linear Temporal Logic (LTL)

LTL is the simplest and most widely used *temporal* specification language.

System Specifications

Some simple system properties:

- deadlock: a program is deadlocked when it cannot proceed
  - common in concurrent programs: threads waiting for each other
  - in a transition system model, this is seen as a state with no outgoing transitions
- basic propositions concerning values of the variables
  - e.g. $x \geq 1$

System properties involving *time*:

- system invariant: *all* reachable program states must satisfy it
- reachability: *eventually* a state with a given property is reached
- …

We will use **Linear** Temporal Logic to formalise temporal properties that must hold **on all paths** which start in an initial state.

Example: temporal properties for mutual exclusion

- **mutual exclusion**: at most one process in critical section *at any time* (i.e. *in every reachable state*)
- **starvation freedom** (enhanced version): *whenever* a process tries to enter its critical section, it will *eventually* succeed (*along every computation path*)
- **no strict sequencing**: processes need not enter their critical section in strict sequence (i.e. *there exists a computation path along which they don't*)

Recap on Propositional Logic

- assume a set *Prop* of atomic propositions
- syntax for formulas:

$$\phi, \psi ::= p \mid \texttt{tt} \mid \neg\phi \mid \phi \wedge \psi \qquad\qquad (p \in \mathit{Prop})$$

  Note: all other boolean operators are definable, e.g.

$$\texttt{ff} ::= \neg\texttt{tt}$$

$$\phi \vee \psi ::= \neg(\neg\phi \wedge \neg\psi)$$

$$\phi \rightarrow \psi ::= \neg(\phi \wedge \neg\psi)$$

- models are *valuations*, stating which propositions are true and which are not:

$$V : \mathit{Prop} \rightarrow \{\texttt{True}, \texttt{False}\}$$

- *semantics* (meaning) of formulas:
  - *V* defines the meaning of atomic propositions
  - truth tables define the meaning of boolean operators

Recap on Propositional Logic: Example

$Prop = \{red, amber, green\}$

- some formulas: red, $green$, $\neg amber$, $red \vee green$, $red \wedge amber$, $red \wedge green$, $red \rightarrow green$
- a model:

  $V(red) = \texttt{True}$, $V(amber) = \texttt{True}$, $V(green) = \texttt{False}$
- semantics:
  - red, $red \vee green$, $red \wedge amber$ – all `True` in the above model
  - $green$, $\neg amber$, $red \wedge green$, $red \rightarrow green$ – all `False` in the above model

Outline

Linear Temporal Logic (LTL)

Paths in Transition Systems

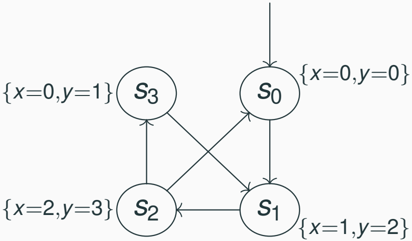
- a path in a transition system is a finite or infinite sequence of states

  $$\pi = s_0 s_1 s_2 \ldots$$

  with s_0 an initial state and $s_i \to s_{i+1}$ for all i.
- we write π^i for the suffix of π starting at s_i
  - $\pi^0 = \pi$
  - $\pi^1 = s_1 s_2 s_3 \ldots$
  - $\ldots$

Paths in Transition Systems – Example

- transition system:



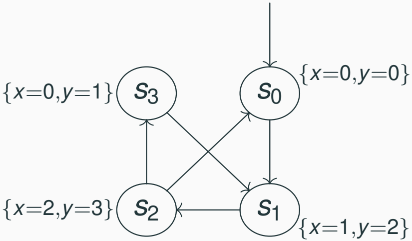
- as atomic propositions we take x=0, x=1, x=2, y=0, y=1, y=2, y=3
- some computation paths:

$$s_0 \to s_1 \to s_2 \to s_3 \to s_1 \to s_2 \to s_3 \to \dots$$
$$s_0 \to s_1 \to s_2 \to s_0 \to s_1 \to s_2 \to \dots$$
$$s_0 \to s_1 \to s_2 \to s_3 \to s_1 \to s_2 \to s_0 \to \dots$$

- LTL formulas must hold on *all* computation paths
  - e.g. "on every path, a state satisfying x=1 is eventually reached"

Paths in Transition Systems – Example

- transition system:



- as atomic propositions we take x=0, x=1, x=2, y=0, y=1, y=2, y=3
- some computation paths:

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$$

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$$

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_1 \rightarrow s_2 \rightarrow s_0 \rightarrow \dots$$

- LTL formulas must hold on *all* computation paths
  - e.g. "on every path, a state satisfying x=1 is eventually reached"

Syntax of LTL

Fix a set *Prop* of atomic propositions.

LTL formulas are of two kinds:

- path formulas:
  - `tt`, p where p is an atomic proposition
  - if f and g are path formulas, then so are:

$\neg f$	not f
$f \wedge g$	f and g
$\mathbf{X}\, f$	at the ne**X**t point in time, f
$\mathbf{F}\, f$	at some point in the **F**uture, f
$\mathbf{G}\, f$	**G**lobally (at all future points) f
$f\, \mathbf{U}\, g$	f **U**ntil g

- state formulas:

$\mathbf{A}\, f$	along **A**ll computation paths, f holds

- **binding priorities:** unary operators ; $\mathbf{U}$; $\wedge$ and $\vee$; $\rightarrow$

Syntax of LTL

Fix a set *Prop* of atomic propositions.

LTL formulas are of two kinds:

- path formulas:
  - $\mathtt{tt}$, p where p is an atomic proposition
  - if f and g are path formulas, then so are:

$\neg f$	not f
$f \wedge g$	f and g
$\mathbf{X}\, f$	at the neXt point in time, f
$\mathbf{F}\, f$	at some point in the Future, f
$\mathbf{G}\, f$	Globally (at all future points) f
$f\, \mathbf{U}\, g$	f Until g

- state formulas:

$\mathbf{A}\, f$	along All computation paths, f holds

- **binding priorities:** unary operators ; $\mathbf{U}$; $\wedge$ and $\vee$; $\rightarrow$

Syntax of LTL

Fix a set *Prop* of atomic propositions.

LTL formulas are of two kinds:

- path formulas:
  - $\mathtt{tt}$, p where p is an atomic proposition
  - if f and g are path formulas, then so are:

$\neg f$	not f
$f \wedge g$	f and g
$\mathbf{X}\, f$	at the ne**X**t point in time, f
$\mathbf{F}\, f$	at some point in the **F**uture, f
$\mathbf{G}\, f$	**G**lobally (at all future points) f
$f\, \mathbf{U}\, g$	f **U**ntil g

- state formulas:

$\mathbf{A}\, f$	along **A**ll computation paths, f holds

- **binding priorities:** unary operators ; $\mathbf{U}$; $\wedge$ and $\vee$; $\rightarrow$

Semantics of LTL

Fix a transition system $M = (S, R, V)$.

The semantics of LTL defines:

- when a computation path π through M satisfies a *path formula* f,

  Notation: $\pi \models f$

- when a state s of M satisfies a *state formula* $\mathbf{A}\, f$.

  Notation: $s \models \mathbf{A}\, f$

Meaning of Temporal Operators (Pictorially)

Let $\pi = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots$ be a computation path.

π^0	$\pi^0 = s_0 \rightarrow \ldots$	$\pi^1 = s_1 \rightarrow \ldots$	...	$\pi^i = s_i \rightarrow \ldots$	...
X f	...	f	...	...	...
F f	...	...	...	f	...
G f	f	f	f	f	f
f **U** g	f	f	f	g	...

Note:

- f here is a path formula, so it is itself interpreted over *paths* !!

Semantics of LTL (Cont'd)

- Given $\pi = s_0 \to s_1 \to s_2 \to \dots$, recall $\pi^i = s_i \to s_{i+1} \to \dots$.
- Now define when a path formula f holds in a path π:

$\pi \models \mathtt{tt}$

$\pi \models p$ iff $p \in V(s_0)$

$\pi \models \neg f$ iff $\pi \models f$ does not hold

$\pi \models f \wedge g$ iff $\pi \models f$ and $\pi \models g$

$\pi \models \mathbf{X}\, f$ iff $\pi^1 \models f$

$\pi \models \mathbf{F}\, f$ iff there exists i s.t. $\pi^i \models f$

$\pi \models \mathbf{G}\, f$ iff $\pi^i \models f$ for all $i \geq 0$

$\pi \models f\, \mathbf{U}\, g$ iff there exists i s.t. $\pi^0 \models f, \dots, \pi^{i-1} \models f, \pi^i \models g$

- Finally, define when a state formula $\mathbf{A}\, f$ holds *in a state* $s \in S$:

$s \models \mathbf{A}\, f$ iff $\pi \models f$ for all paths π starting in s

Semantics of LTL (Cont'd)

- Given $\pi = s_0 \to s_1 \to s_2 \to \ldots$, recall $\pi^i = s_i \to s_{i+1} \to \ldots$.
- Now define when a path formula f holds in a path π:

$$
\begin{array}{llll}
\pi \models \texttt{tt} & & \\
\pi \models p & \text{iff} & p \in V(s_0) \\
\pi \models \neg f & \text{iff} & \pi \models f \quad \text{does not hold} \\
\pi \models f \wedge g & \text{iff} & \pi \models f \text{ and } \pi \models g \\
\pi \models \mathbf{X}\, f & \text{iff} & \pi^1 \models f \\
\pi \models \mathbf{F}\, f & \text{iff} & \text{there exists } i \text{ s.t. } \pi^i \models f \\
\pi \models \mathbf{G}\, f & \text{iff} & \pi^i \models f \text{ for all } i \geq 0 \\
\pi \models f\, \mathbf{U}\, g & \text{iff} & \text{there exists } i \text{ s.t. } \pi^0 \models f, \ldots, \pi^{i-1} \models f, \pi^i \models g
\end{array}
$$

- Finally, define when a state formula $\mathbf{A}\, f$ holds *in a state* $s \in S$:

$$s \models \mathbf{A}\, f \quad \text{iff} \quad \pi \models f \quad \text{for all paths } \pi \text{ starting in } s$$

Semantics of LTL (Cont'd)

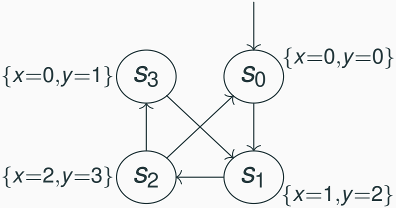
- Given $\pi = s_0 \to s_1 \to s_2 \to \ldots$, recall $\pi^i = s_i \to s_{i+1} \to \ldots$.
- Now define when a path formula f holds in a path π:

$$
\begin{array}{lll}
\pi \models \texttt{tt} & & \\
\pi \models p & \text{iff} & p \in V(s_0) \\
\pi \models \neg f & \text{iff} & \pi \models f \quad \text{does not hold} \\
\pi \models f \wedge g & \text{iff} & \pi \models f \text{ and } \pi \models g \\
\pi \models \mathbf{X}\, f & \text{iff} & \pi^1 \models f \\
\pi \models \mathbf{F}\, f & \text{iff} & \text{there exists } i \text{ s.t. } \pi^i \models f \\
\pi \models \mathbf{G}\, f & \text{iff} & \pi^i \models f \text{ for all } i \geq 0 \\
\pi \models f\, \mathbf{U}\, g & \text{iff} & \text{there exists } i \text{ s.t. } \pi^0 \models f, \ldots, \pi^{i-1} \models f, \pi^i \models g
\end{array}
$$

- Finally, define when a state formula $\mathbf{A}\, f$ holds *in a state* $s \in S$:

$$s \models \mathbf{A}\, f \ \text{ iff } \ \pi \models f \quad \text{for all paths } \pi \text{ starting in } s$$

Semantics of LTL – Example



$s_0 \to s_1 \to s_2 \to s_0 \to \ldots \models \mathbf{X}\,(y = 2)$

$s_0 \to s_1 \to s_2 \to s_3 \to \ldots \models \mathbf{F}\,(y = 1)$

$s_0 \to s_1 \to s_2 \to s_3 \to \ldots \models (x = 0)$

$s_0 \to s_1 \to s_2 \to s_3 \to \ldots \not\models \mathbf{G}\,(x = 0)$

$s_0 \to s_1 \to s_2 \to s_3 \to \ldots \models \mathbf{G}\,((x = 0) \vee (x = 1) \vee (x = 2))$

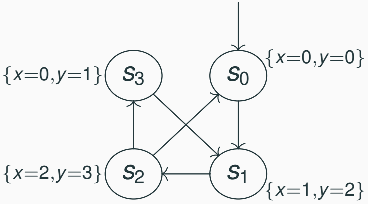
$s_0 \to s_1 \to s_2 \to s_0 \to \ldots \not\models \mathbf{F}\,\mathbf{G}\,(x = 0)$

$s_0 \to s_1 \to s_2 \to s_0 \to \ldots \models ((x = 0) \vee (x = 1))\,\mathbf{U}\,(x = 2)$

$\pi \models \mathbf{G}\,\mathbf{F}\,(x = 0)$ for any π

Note: $\pi \models \mathbf{G}\,\mathbf{F}\,f$ iff f holds infinitely often along π.

Semantics of LTL – Example



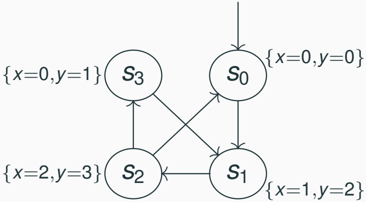
$s_0 \models \mathbf{A}\,((x = 0) \wedge (y = 0))$
$s_0 \models \mathbf{A}\ \mathbf{X}\,(x = 1)$
$s_0 \models \mathbf{A}\ \mathbf{X}\ \mathbf{X}\,((x = 2) \wedge (y = 3))$
$s_0 \models \mathbf{A}\ \mathbf{F}\,(x = 2)$

$s_1 \models \mathbf{A}\ \mathbf{G}\,\neg((x = 1) \wedge (y = 3))$
$s_1 \not\models \mathbf{A}\ \mathbf{G}\,(x = 1)$
$s_1 \models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\,(x = 0)$
$s_0 \not\models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\,((x = 0) \wedge (y = 1))$
$s_0 \models \mathbf{A}\,(\mathbf{G}\ \mathbf{F}\,(y = 1) \rightarrow \mathbf{G}\ \mathbf{F}\,((x = 0) \wedge (y = 1)))$
$s_1 \not\models \mathbf{A}\ (((y = 2) \vee (y = 3))\ \mathbf{U}\,(y = 0))$
$s_1 \models \mathbf{A}\ (((y = 2) \vee (y = 3))\ \mathbf{U}\,(y = 0) \vee (y = 1))$
$s_0 \models \mathbf{A}\ \mathbf{X}\ (((y = 2) \vee (y = 3))\ \mathbf{U}\,((y = 0) \vee (y = 1)))$

Note:

- $s \models \mathbf{A}\ \mathbf{G}\ f$ iff f holds in all states reachable from s (incl. s)
- $s \models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\ f$ iff f holds infinitely often on every path from s

Semantics of LTL – Example



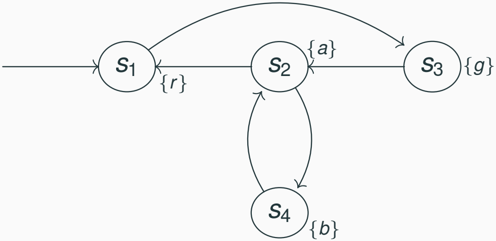
$s_0 \models \mathbf{A}\,((x = 0) \wedge (y = 0))$
$s_0 \models \mathbf{A}\ \mathbf{X}\,(x = 1)$
$s_0 \models \mathbf{A}\ \mathbf{X}\ \mathbf{X}\,((x = 2) \wedge (y = 3))$
$s_0 \models \mathbf{A}\ \mathbf{F}\,(x = 2)$

$s_1 \models \mathbf{A}\ \mathbf{G}\,\neg((x = 1) \wedge (y = 3))$
$s_1 \not\models \mathbf{A}\ \mathbf{G}\,(x = 1)$
$s_1 \models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\,(x = 0)$
$s_0 \not\models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\,((x = 0) \wedge (y = 1))$
$s_0 \models \mathbf{A}\,(\mathbf{G}\ \mathbf{F}\,(y = 1) \rightarrow \mathbf{G}\ \mathbf{F}\,((x = 0) \wedge (y = 1)))$
$s_1 \not\models \mathbf{A}\ (((y = 2) \vee (y = 3))\,\mathbf{U}\,(y = 0))$
$s_1 \models \mathbf{A}\ (((y = 2) \vee (y = 3))\,\mathbf{U}\,(y = 0) \vee (y = 1))$
$s_0 \models \mathbf{A}\ \mathbf{X}\ (((y = 2) \vee (y = 3))\,\mathbf{U}\,((y = 0) \vee (y = 1)))$

Note:

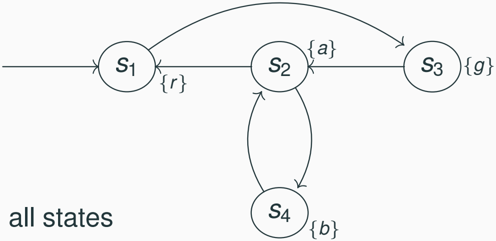
- $s \models \mathbf{A}\ \mathbf{G}\,f$ iff f holds in all states reachable from s (incl. s)
- $s \models \mathbf{A}\ \mathbf{G}\ \mathbf{F}\,f$ iff f holds infinitely often on every path from s

Another Example: Blinking Traffic Lights



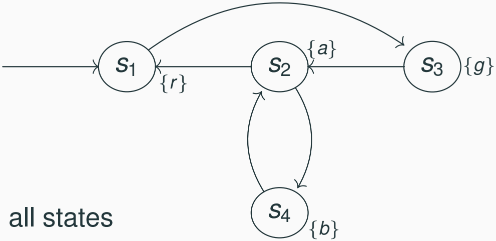
- $\mathbf{A}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{F}\,r$
- $\mathbf{A}\,\mathbf{F}\,g$
- $\mathbf{A}\,\mathbf{G}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



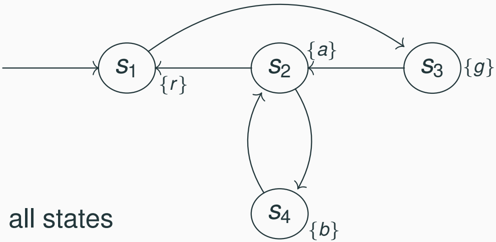
- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$
- $\mathbf{A}\,\mathbf{F}\,g$
- $\mathbf{A}\,\mathbf{G}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



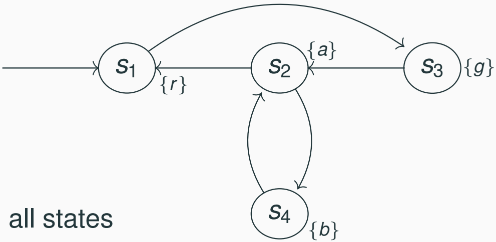
- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$ holds in state s_1 only
- $\mathbf{A}\,\mathbf{F}\,g$
- $\mathbf{A}\,\mathbf{G}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



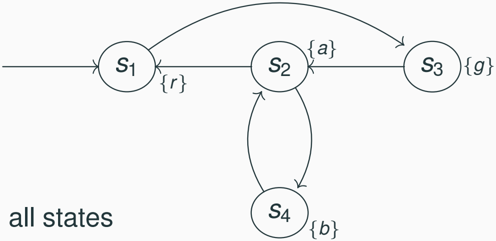
- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$ holds in state s_1 only
- $\mathbf{A}\,\mathbf{F}\,g$ holds in states s_1 and s_3
- $\mathbf{A}\,\mathbf{G}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



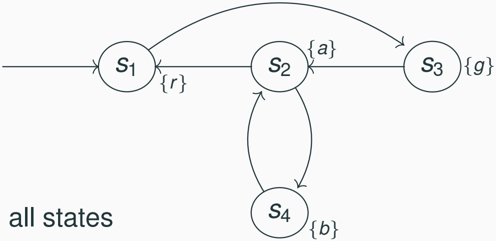
- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$ holds in state s_1 only
- $\mathbf{A}\,\mathbf{F}\,g$ holds in states s_1 and s_3
- $\mathbf{A}\,\mathbf{G}\,a$ holds in no state
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



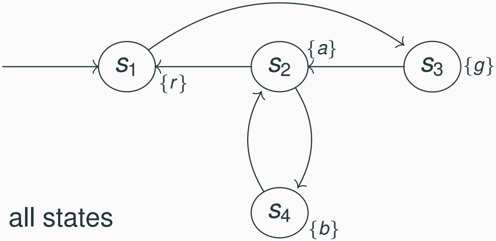
- $\mathbf{A}\ \mathbf{F}\, a$ holds in all states
- $\mathbf{A}\ \mathbf{F}\, r$ holds in state s_1 only
- $\mathbf{A}\ \mathbf{F}\, g$ holds in states s_1 and s_3
- $\mathbf{A}\ \mathbf{G}\, a$ holds in no state
- $\mathbf{A}\ \mathbf{G}\ \mathbf{F}\, a$ holds in all states
- $\mathbf{A}\ \mathbf{G}\ \mathbf{F}\, r$
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



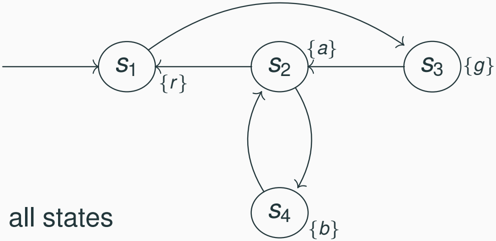
- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$ holds in state s_1 only
- $\mathbf{A}\,\mathbf{F}\,g$ holds in states s_1 and s_3
- $\mathbf{A}\,\mathbf{G}\,a$ holds in no state
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$ holds in no state
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



- $\mathbf{A}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{F}\,r$ holds in state s_1 only
- $\mathbf{A}\,\mathbf{F}\,g$ holds in states s_1 and s_3
- $\mathbf{A}\,\mathbf{G}\,a$ holds in no state
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,a$ holds in all states
- $\mathbf{A}\,\mathbf{G}\,\mathbf{F}\,r$ holds in no state
- $\mathbf{A}\,(b\,\mathbf{U}\,\neg b)$ holds in all states
- $\mathbf{A}\,(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$

Another Example: Blinking Traffic Lights



- **A F** a holds in all states
- **A F** r holds in state s_1 only
- **A F** g holds in states s_1 and s_3
- **A G** a holds in no state
- **A G F** a holds in all states
- **A G F** r holds in no state
- **A** $(b\,\mathbf{U}\,\neg b)$ holds in all states
- **A** $(g\,\mathbf{U}\,(a\,\mathbf{U}\,r))$ holds in state s_1 only

Some LTL Patterns

- invariance (always):

  "*p* remains invariantly true throughout every path"

- guarantee (eventually):

  "*p* will eventually become true in every path"

- stability (non-progress):

  "there is a point in every path where *p* will become invariantly true"

- recurrence (progress):

  "if *p* happens to be false at any given point in a path, it is always guaranteed to become true again later"

  Same as: "*p* holds infinitely often"

Some LTL Patterns

- invariance (always): **A G** p

  "p remains invariantly true throughout every path"

- guarantee (eventually):

  "p will eventually become true in every path"

- stability (non-progress):

  "there is a point in every path where p will become invariantly true"

- recurrence (progress):

  "if p happens to be false at any given point in a path, it is always guaranteed to become true again later"

  Same as: "p holds infinitely often"

Some LTL Patterns

- invariance (always): **A G** *p*

  "*p* remains invariantly true throughout every path"

- guarantee (eventually): **A F** *p*

  "*p* will eventually become true in every path"

- stability (non-progress):

  "there is a point in every path where *p* will become invariantly true"

- recurrence (progress):

  "if *p* happens to be false at any given point in a path, it is always guaranteed to become true again later"

  Same as: "*p* holds infinitely often"

Some LTL Patterns

- invariance (always): **A G** p

  "p remains invariantly true throughout every path"

- guarantee (eventually): **A F** p

  "p will eventually become true in every path"

- stability (non-progress): **A F G** p

  "there is a point in every path where p will become invariantly true"

- recurrence (progress):

  "if p happens to be false at any given point in a path, it is always guaranteed to become true again later"

  Same as: "p holds infinitely often"

Some LTL Patterns

- invariance (always): $\mathbf{A}\ \mathbf{G}\ p$

  "p remains invariantly true throughout every path"

- guarantee (eventually): $\mathbf{A}\ \mathbf{F}\ p$

  "p will eventually become true in every path"

- stability (non-progress): $\mathbf{A}\ \mathbf{F}\ \mathbf{G}\ p$

  "there is a point in every path where p will become invariantly true"

- recurrence (progress): $\mathbf{A}\ \mathbf{G}\ \mathbf{F}\ p$

  "if p happens to be false at any given point in a path, it is always guaranteed to become true again later"

  Same as: "p holds infinitely often"

Some LTL Patterns (Cont'd)

- response:

  "any state satisfying p is eventually followed by a state satisfying q"

- precedence:

  "from any state satisfying p, the system will continuously satisfy property q until property r becomes true"

- correlation:

  "if p holds at some point in the future, so does q"

Some LTL Patterns (Cont'd)

- response: $\mathbf{A}\ \mathbf{G}\,(p \rightarrow \mathbf{F}\,q)$

  "any state satisfying p is eventually followed by a state satisfying q"

- precedence:

  "from any state satisfying p, the system will continuously satisfy property q until property r becomes true"

- correlation:

  "if p holds at some point in the future, so does q"

Some LTL Patterns (Cont'd)

- response: $\mathbf{A}\ \mathbf{G}\,(p \rightarrow \mathbf{F}\,q)$

  "any state satisfying p is eventually followed by a state satisfying q"

- precedence: $\mathbf{A}\ \mathbf{G}\,(p \rightarrow q\,\mathbf{U}\,r)$

  "from any state satisfying p, the system will continuously satisfy property q until property r becomes true"

- correlation:

  "if p holds at some point in the future, so does q"

Some LTL Patterns (Cont'd)

- response: $\mathbf{A}\ \mathbf{G}\,(p \rightarrow\ \mathbf{F}\,q)$

  "any state satisfying p is eventually followed by a state satisfying q"

- precedence: $\mathbf{A}\ \mathbf{G}\,(p \rightarrow q\,\mathbf{U}\,r)$

  "from any state satisfying p, the system will continuously satisfy property q until property r becomes true"

- correlation: $\mathbf{A}\,(\,\mathbf{F}\,p \rightarrow\ \mathbf{F}\,q)$

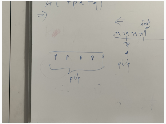
  "if p holds at some point in the future, so does q"

  不代表p和q的顺序
  区别于 AG(Fp–>Fq)

Some Interesting Equivalences

F(p且q是同时的意思)

$$
\begin{aligned}
\neg \mathbf{G}\, p &\equiv \mathbf{F} \neg p \\
\mathbf{G}\, p &\equiv \mathbf{G}\, \mathbf{G}\, p \\
\mathbf{F}\, p &\equiv \mathbf{F}\, \mathbf{F}\, p \\
\mathbf{G}\, (p \wedge q) &\equiv \mathbf{G}\, p \wedge \mathbf{G}\, q \\
\mathbf{F}\, (p \vee q) &\equiv \mathbf{F}\, p \vee \mathbf{F}\, q \\
\mathbf{G}\, \mathbf{F}\, (p \vee q) &\equiv \mathbf{G}\, \mathbf{F}\, p \vee \mathbf{G}\, \mathbf{F}\, q \\
p\, \mathbf{U}\, q &\equiv p\, \mathbf{U}\, (p\, \mathbf{U}\, q) \\
p\, \mathbf{U}\, q &\equiv (p\, \mathbf{U}\, q)\, \mathbf{U}\, q
\end{aligned}
$$



Back to Mutual Exclusion

Atomic propositions:

c_0 , c_1	(critical state)
n_0 , n_1	(non-critical state)
t_0 , t_1	(trying to enter critical state)

- mutual exclusion: at most one process in critical section *at any time*

Back to Mutual Exclusion

Atomic propositions:

c_0 , c_1	(critical state)
n_0 , n_1	(non-critical state)
t_0 , t_1	(trying to enter critical state)

- mutual exclusion: at most one process in critical section *at any time*

  $$\mathbf{A}\ \mathbf{G}\,\neg(c_0 \wedge c_1)$$

Back to Mutual Exclusion

Atomic propositions:

c_0 , c_1	(critical state)
n_0 , n_1	(non-critical state)
t_0 , t_1	(trying to enter critical state)

- mutual exclusion: at most one process in critical section *at any time*

  $$\mathbf{A}\ \mathbf{G}\,\neg(c_0 \wedge c_1)$$

- absence of starvation: *whenever* a process tries to enter its critical section, it will *eventually* succeed

Back to Mutual Exclusion

Atomic propositions:

c_0 , c_1	(critical state)
n_0 , n_1	(non-critical state)
t_0 , t_1	(trying to enter critical state)

- mutual exclusion: at most one process in critical section *at any time*

  $$\mathbf{A}\ \mathbf{G}\, \neg(c_0 \wedge c_1)$$

- absence of starvation: *whenever* a process tries to enter its critical section, it will *eventually* succeed

  $$\mathbf{A}\ \mathbf{G}\ ((t_0 \rightarrow \mathbf{F}\, c_0) \wedge (t_1 \rightarrow \mathbf{F}\, c_1))$$

- **no strict sequencing:** processes need not enter their critical section in strict sequence

Back to Mutual Exclusion

Atomic propositions:

c_0 , c_1	(critical state)
n_0 , n_1	(non-critical state)
t_0 , t_1	(trying to enter critical state)

- mutual exclusion: at most one process in critical section *at any time*

  $$\mathbf{A}\ \mathbf{G}\,\neg(c_0 \wedge c_1)$$

- absence of starvation: *whenever* a process tries to enter its critical section, it will *eventually* succeed

  $$\mathbf{A}\ \mathbf{G}\ ((t_0 \rightarrow \mathbf{F}\,c_0) \wedge (t_1 \rightarrow \mathbf{F}\,c_1))$$

- **no strict sequencing:** processes need not enter their critical section in strict sequence
  - can only express the negation of this property ...
  - ...but this is sufficient, since a counter-example to strict sequencing is proof for non-strict sequencing !

Strict Sequencing in LTL

- define Weak Until operator: $f\,\mathbf{W}\,g ::= \mathbf{G}\,f \;\vee\; f\,\mathbf{U}\,g$
- strict sequencing (assuming mutual exclusion holds!):

$$\mathbf{A}\,(\,\mathbf{G}\,(c_0 \rightarrow c_0\,\mathbf{W}\,(\neg c_0 \wedge \neg c_0\,\mathbf{W}\,c_1))\;\;\wedge\;\;\mathbf{G}\,(\ldots)\,) \qquad \times$$

Strict Sequencing in LTL

- define **Weak Until** operator: $f\,\mathbf{W}\,g ::= \mathbf{G}\,f \;\vee\; f\,\mathbf{U}\,g$
- strict sequencing (assuming mutual exclusion holds!):

$$\mathbf{A}\,(\,\mathbf{G}\,(c_0 \rightarrow c_0\,\mathbf{W}\,(\neg c_0 \wedge \neg c_0\,\mathbf{W}\,c_1)) \quad \wedge \quad \mathbf{G}\,(\ldots)\,) \qquad \times$$

Strict Sequencing in LTL

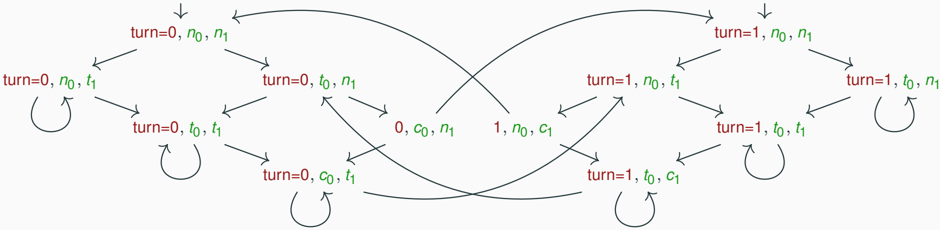
强Until，后续条件一定会发生
弱until，后续条件不一定发生

- define **Weak Until** operator: $f \, \mathbf{W} \, g ::= \mathbf{G} \, f \;\vee\; f \, \mathbf{U} \, g$
- strict sequencing (assuming mutual exclusion holds!):

$$\mathbf{A} \, (\, \mathbf{G} \, (c_0 \rightarrow c_0 \, \mathbf{W} \, (\neg c_0 \wedge \neg c_0 \, \mathbf{W} \, c_1 \,)) \quad \wedge \quad \mathbf{G} \, (\ldots) \,) \qquad \times$$

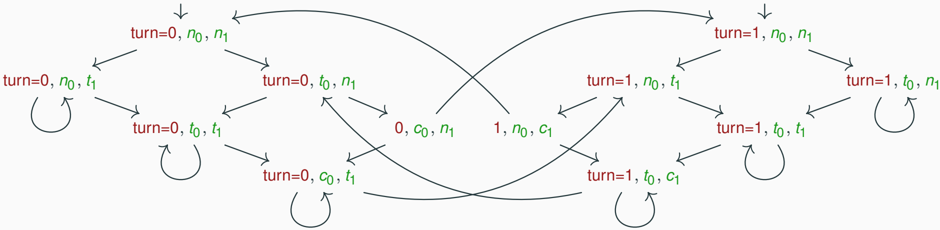
c0 在c1发生之后不能再次发生
c0 不一定发生

Mutual Exclusion: Checking Correctness



- $\mathbf{A}\ \mathbf{G}\, \neg(c_0 \wedge c_1)$

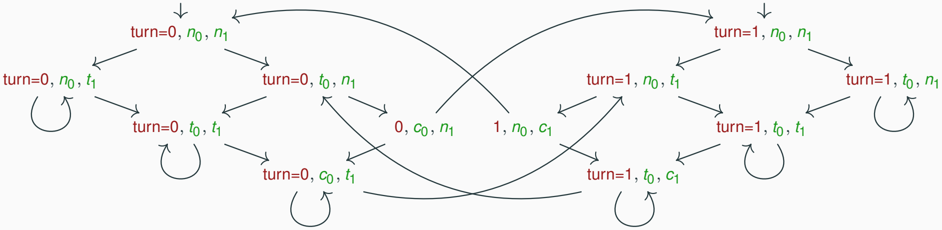
Mutual Exclusion: Checking Correctness



- $\mathbf{A\ G}\,\neg(c_0 \wedge c_1)$ ✓

  Need to check that $\neg(c_0 \wedge c_1)$ is true at all states reachable from the initial states.

Mutual Exclusion: Checking Correctness

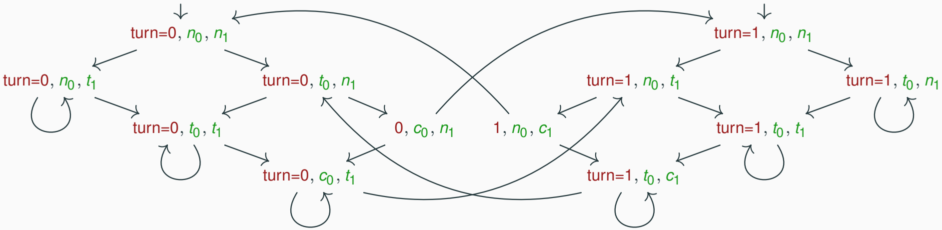


- $\mathbf{A}\ \mathbf{G}\, \neg(c_0 \wedge c_1)$ ✓

  Need to check that $\neg(c_0 \wedge c_1)$ is true at all states reachable from the initial states.

- $\mathbf{A}\ \mathbf{G}\ ((t_0 \rightarrow \mathbf{F}\, c_0) \wedge (t_1 \rightarrow \mathbf{F}\, c_1))$

Mutual Exclusion: Checking Correctness



- $\mathbf{A}\ \mathbf{G}\, \neg(c_0 \wedge c_1)$ ✓

  Need to check that $\neg(c_0 \wedge c_1)$ is true at all states reachable from the initial states.

- $\mathbf{A}\ \mathbf{G}\ ((t_0 \to \mathbf{F}\, c_0) \wedge (t_1 \to \mathbf{F}\, c_1))$ self loop × ✓

  Need fairness assumptions for the property to hold.

  加前提（additional assumpution）可以变为真

Exercise

Assume the following atomic formulas:
start, *ready*, *requested*, *acknowledged*, *enabled*, *running*, *deadlock*.

Specify the following correctness properties in LTL:

1. It is impossible to reach a state where *start* holds but *ready* does not hold. AG(start−>ready)
2. Whenever a request occurs, it will eventually be acknowledged. AG(requested−>F acknowledged)
3. If a process is enabled infinitely often along a path, then it runs infinitely often along that path. A(GF enabled−>GF runs)
4. Whatever happens, *deadlock* will eventually occur. AF deadlock
5. From any state, it is possible to reach a *ready* state.
   AGF ready

AGF(enable->GF runs)

AG(G enable->GF runs)

AGF(enable U runs)