Verification by Model Checking

Readings: Chapter 3 of LICS2

EECS4315 Z:
Mission-Critical Systems
Winter 2023

CHEN-WEI WANG

Motivation for Formal Verification

- ***Safety-Critical Systems***
  e.g., shutdown system of a nuclear power plant
- ***Mission-Critical Systems***
  e.g., mass-produced computer chips
- ***Formal verification*** of the *correctness* of critical systems can prevent loss of fortune or even lives.
- Formal verification consists of:
  1. ***Systems***: Need a **specification** language for modelling abstractions.
  2. ***Properties***: Need a **specification** language for expressing (e.g., safety, temporal) concerns.
  3. ***Verification***: Need a **systematic method** for establishing that a system satisfies the desired properties.
- The **earlier** errors are caught in the course of system development, the **cheaper** it is to rectify.
  - e.g., Much cheaper to catch an error in the design phase than recalling defected products after release.

Example of Formal Verification

Pentium FDIV bug: https://en.wikipedia.org/wiki/Pentium_FDIV_bug

*The Pentium FDIV bug is a hardware bug affecting the **floating-point unit (FPU)** of the early Intel Pentium processors. Because of the bug, the processor would return incorrect binary floating point results when dividing certain pairs of high-precision numbers.*

*In December 1994, Intel **recalled** the defective processors ... In its 1994 annual report, Intel said it incurred "**a $475 million pre-tax charge** ... to recover replacement and write-off of these microprocessors."*

*In the aftermath of the **bug** and subsequent **recall**, there was a marked increase in the use of formal verification of hardware floating point operations across the **semiconductor industry**. Prompted by the discovery of the bug, a technique ... called "word-level **model checking**" was developed in 1996. Intel went on to use **formal verification** extensively in the development of later CPU architectures. In the development of the Pentium 4, symbolic trajectory evaluation and **theorem proving** were used to **find a number of bugs that could have led to a similar recall incident** had they gone undetected.*

Classification of Verification Methods

- **Degree of Automation**: Automatic, Interactive, or Manual
- **~~Model~~Check-based vs. Proof-based**
  - ***Proof***-based:
    - The ***system*** (abstractly) described as a set of formulas Γ
    - ***Properties*** specified as a set of formulas ϕ
    - ***Prove*** (automatically or interactively) that $\Gamma \vdash \phi$ [*undecidable*]
      i.e., Γ can be derived to ϕ (via ***inference rules***).
  - ***Check***-based:
    - The ***system*** (abstractly) described as a *finite* model $\mathbb{M}$
    - ***Properties*** specified as a set of formulas ϕ
    - ***Decide*** (automatically) that $\mathbb{M} \vDash \phi$ [*decidable*, *algorithmic*]
      i.e., Traversal of $\mathbb{M}$'s ***state graph*** shows that ϕ is satisfied.
- **Domain of Application**
  - Hardware vs. Software
  - Sequential vs. Concurrent
  - Reactive vs. Terminating
- **Pre-development vs. Post-development**

Verification via Model Checking

- Automatic, Check-based
- Intended for ***reactive***, ***concurrent*** systems
  - ***Reactivity***:
    Continuous reaction to stimuli from the environment
    e.g., communication protocols, operating systems, embedded systems, etc.
  - ***Concurrency***:
    Simultaneous execution of (independent or inter-dependent) system units, each of which evolving its own states
- ***Testing*** of concurrent, reactive systems is hard:
  - Many scenarios are **non-reproducible**.
  - Hard to **systematically** cover all important interactions
  - E. W. Dijkstra: ***Program testing can be used to show the presence of bugs, but never to show their absence!***
- Originated as a ***post***-development method
- But should be used as ***pre***-development method to save cost

Model Checking: Temporal Logic

- **System**
  - A system model $\mathbb{M}$ is a ***labeled transition system (LTS)*** with a (large) number of states and transitions between states.
  - A ***model*** of an actual physical system ***abstracts away*** details that are irrelevant to the ***properties*** to be checked.
- **Properties**
  - *Temporal logic (TL)* incorporates the notion of ***timing***.
  - A TL formula ϕ is **not** statically true or false.
  - Instead, the truth of a TL formula ϕ depends on where the SUV **dynamically** evolves into (by following transitions).
- **Verification**
  - A computer program, called a *model checker*, takes as inputs $\mathbb{M}$ and ϕ, and **decides** if $\mathbb{M} \models \phi$
    - **Yes** $\Rightarrow$ All ***reachable*** states of $\mathbb{M}$ satisfy ϕ.
    - **No** $\Rightarrow$ An ***error trace***, leading to a state satisfying $\neg\phi$, is generated. This facilitates debugging through reproducing a problematic scenario.
    - **Unknown** $\Rightarrow$ The checker runs out of memory due to ***state explosion***.

Linear-Time Temporal Logic (LTL)

- *LTL (Linear-time Temoral Logic)* has connectives/operators which allow us to refer to the ***future***.
- Two features of *LTL*:
  - ***(Computation) Path***:
    Time is modelled as an ***infinite*** sequence of states.
  - ***Undetermined Future***:
    Alternative paths exist, one of which being the "actual" path.

LTL: Syntax in CFG (1)

ϕ	::=	$\top$	[*true*]
	|	$\bot$	[*false*]
	|	p	[`propositional atom`]
	|	$(\neg\phi)$	[`logical negation`]
	|	$(\phi \wedge \phi)$	[`logical conjunction`]
	|	$(\phi \vee \phi)$	[`logical disjunction`]
	|	$(\phi \Rightarrow \phi)$	[`logical implication`]
	|	$(\mathbf{X}\,\phi)$	[`neXt state`]
	|	$(\mathbf{F}\,\phi)$	[`some Future state`]
	|	$(\mathbf{G}\,\phi)$	[`all future states (Globally)`]
	|	$(\phi\,\mathbf{U}\,\phi)$	[`Until`]
	|	$(\phi\,\mathbf{W}\,\phi)$	[`Weak-until`]
	|	$(\phi\,\mathbf{R}\,\phi)$	[`Release`]

p denotes ***atomic***, propositional statements
e.g., Printer `ltr2` is available.
e.g., Reading of sensor `s3` exceeds some threshold.
e.g., The sudoku board is filled out with a correct solution.

LTL: Syntax in CFG (2)

ϕ	$::=$	$\top$	[*true*]
	$\mid$	$\bot$	[*false*]
	$\mid$	p	[propositional atom]
	$\mid$	$(\neg\phi)$	[logical negation]
	$\mid$	$(\phi \wedge \phi)$	[logical conjunction]
	$\mid$	$(\phi \vee \phi)$	[logical disjunction]
	$\mid$	$(\phi \Rightarrow \phi)$	[logical implication]
	$\mid$	$(\mathbf{X}\,\phi)$	[ne**X**t state]
	$\mid$	$(\mathbf{F}\,\phi)$	[some **F**uture state]
	$\mid$	$(\mathbf{G}\,\phi)$	[all future states (**G**lobally)]
	$\mid$	$(\phi\,\mathbf{U}\,\phi)$	[**U**ntil]
	$\mid$	$(\phi\,\mathbf{W}\,\phi)$	[**W**eak-until]
	$\mid$	$(\phi\,\mathbf{R}\,\phi)$	[**R**elease]

$\forall$ and $\exists$ are embedded in defining the ***temporal*** connectives.
Universe of disclosure: Set of alternative (computation) ***paths***

LTL: Syntax in CFG (3)

ϕ	$::=$	$\top$	[*true*]
	$\mid$	$\bot$	[*false*]
	$\mid$	p	[propositional atom]
	$\mid$	$(\neg\phi)$	[logical negation]
	$\mid$	$(\phi \wedge \phi)$	[logical conjunction]
	$\mid$	$(\phi \vee \phi)$	[logical disjunction]
	$\mid$	$(\phi \Rightarrow \phi)$	[logical implication]
	$\mid$	$(\mathbf{X}\,\phi)$	[ne**X**t state]
	$\mid$	$(\mathbf{F}\,\phi)$	[some **F**uture state]
	$\mid$	$(\mathbf{G}\,\phi)$	[all future states (**G**lobally)]
	$\mid$	$(\phi\,\mathbf{U}\,\phi)$	[**U**ntil]
	$\mid$	$(\phi\,\mathbf{W}\,\phi)$	[**W**eak-until]
	$\mid$	$(\phi\,\mathbf{R}\,\phi)$	[**R**elease]

- ***Temporal*** connectives bind **tighter** than ***logical*** ones.
- Unary ***temporal*** connectives bind **tighter** than binary ones.
  - Use parentheses to force the intended order of evaluation.
  - Use a ***parse tree***, a ***LMD***, or a ***RMD*** to verify the order of evaluation.

LTL: Symbols of Unary Temporal Operators

Temporal Connective	Letter	Symbol
Next	**X**	$\bigcirc$
Future/Eventually	**F**	$\Diamond$
Global/Henceforth	**G**	$\Box$

Practical Knowledge about Parsing

- A ***context-free grammar (CFG)*** g
  - defines, **recursively**, **all** (typically an infinite number of) possible strings that can be ***derived*** from it.
  - contains both ***terminals***/***tokens*** (base cases) and ***non-terminals***/***variables*** (recursive cases)
- Given an input string s, to show that $s \in L(g)$, we can either:
  - **Draw** a ***parse tree (PT)*** of s, based on g, where:
    - All ***internal nodes*** (i.e., roots of subtrees) are ϕ (non-terminals).
    - All ***external nodes*** (a.k.a. leaves) are characters of s.
  - **Perform** a ***left-most derivation (LMD)***, by starting with ϕ (the ***start variable***) and continuing to substitute the leftmost non-terminal, until **no** non-terminals remain.
  - **Perform** a ***right-most derivation (RMD)***, by starting with ϕ (the ***start variable***) and continuing to substitute the rightmost non-terminal, until **no** non-terminals remain.
- PTs, LMDs, and RMDs are legitimate, and equivalent, ways for showing ***interpretations*** of a valid LTL formula string.

LTL: Exercises on Parsing Formulas

- Draw and compare the ***parse trees*** of:

  $\mathbf{F}\ p \land \mathbf{G}\ q \Rightarrow p\,\mathbf{U}\,r$
  vs. $\mathbf{F}\ (p \land \mathbf{G}\ q \Rightarrow p\,\mathbf{U}\,r)$
  vs. $\mathbf{F}\ p \land (\mathbf{G}\ q \Rightarrow p\,\mathbf{U}\,r)$
  vs. $\mathbf{F}\ p \land ((\mathbf{G}\ q \Rightarrow p)\,\mathbf{U}\,r)$
- The above formulas are all ***derivable*** from the grammar of LTL.
  - Show using the ***LMD*** (Left-Most Derivations)
  - Show using the ***RMD*** (Right-Most Derivations)

LTL Formulas: More Exercises

Draw the ***parser trees*** for:

$$(\ \mathbf{F}(p \Rightarrow \mathbf{G}\,r) \lor ((\neg q)\,\mathbf{U}\,p)\ )$$

vs. $\mathbf{F}\,p \Rightarrow \mathbf{G}\,r \lor \neg q\,\mathbf{U}\,p$

vs. $\mathbf{F}(\ (p \Rightarrow \mathbf{G}\,r) \lor (\neg q\,\mathbf{U}\,p)\ )$

LTL Formulas: Subformulas

- Given an LTL formula ϕ, its ***subformulas*** are all those whose ***parse trees (rooted at ϕ)*** are subtrees of ϕ's parse tree.

  e.g., Enumerate all subformula of $(\ \mathbf{F}(p \Rightarrow \mathbf{G}\,r) \lor ((\neg q)\,\mathbf{U}\,p)\ )$.
  - p, q, r,
  - $\mathbf{G}\,r$, $p \Rightarrow (\mathbf{G}\,r)$, $\mathbf{F}(p \Rightarrow (\mathbf{G}\,r))$,
  - $\neg q$, $(\neg q)\,\mathbf{U}\,p$, $\mathbf{F}(p \Rightarrow (\mathbf{G}\,r)) \lor (\neg q)\,\mathbf{U}\,p$
  - $(\ \mathbf{F}(p \Rightarrow \mathbf{G}\,r) \lor ((\neg q)\,\mathbf{U}\,p)\ )$

LTL Semantics: Labelled Transition Systems (LTS)

- **Definition**. Given that P is a set of atoms/propositions of concern, a ***transition system*** $\mathbb{M}$ is a ***formal model*** represented as a triple $\mathbb{M} = (S, \longrightarrow, L)$:
  - S
    A finite set of ***states***
  - $\longrightarrow : S \leftrightarrow S$
    A ***transition relation*** on S
  - $L : S \to \mathbb{P}(P)$
    A ***labelling function*** mapping each state to its satisfying atoms

  Assumption. No state of the system can ***deadlock***:

  From any state, it's always possible to make progress (by taking a transition).

$$\forall s \bullet s \in S \Rightarrow (\ \exists s' \bullet s' \in S \land (s, s') \in \longrightarrow\ )$$

LTL Semantics: Example of LTS

- We may visual a transition system $\mathbb{M}$ using a ***directed graph***:
  - Nodes/Vertices denote ***states***.
  - Edges/Arcs denote ***transitions***.
- **Exercises** Consider the system with a counter c with the following assumption:

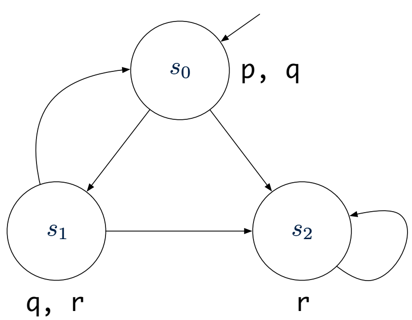
$$0 \leq c \leq 3$$

Say c is initialized 0 and may be incremented (via a transition *inc*, enabled when $c < 3$) or decremented (via a transition *dec*, enabled when $c > 0$).

  - **Draw** a ***state graph*** of this system.
  - **Formulate** the state graph as an ***LTS*** (via a triple $(S, \longrightarrow, L)$).

    Assume: Set P of atoms is: $\{\ c \geq 1, c \leq 1\ \}$

LTL Semantics: More Example of LTS



$\mathbb{M} = (S, \longrightarrow, L)$:

- $S = \{s_0, s_1, s_2\}$
- $\longrightarrow = \{(s_0, s_1), (s_0, s_2), (s_1, s_0), (s_1, s_2), (s_2, s_2)\}$
- $L = \{(s_0, \{p, q\}), (s_1, \{q, r\}), (s_2, \{r\})\}$

LTL Semantics: Paths

Definition. A ***path*** in a model $\mathbb{M} = (S, \longrightarrow, L)$ is an ***infinite sequence of states*** $s_i \in S$, where $i \geq 1$, such that $s_i \longrightarrow s_{i+1}$.

- We write the path, starting at the ***initial state*** s_1, as

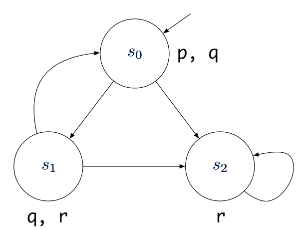
$$s_1 \longrightarrow s_2 \longrightarrow \ldots$$

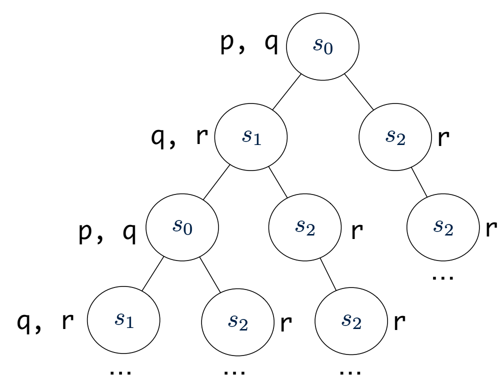
- **Note.** s_1 in the above path pattern denotes the first, initial state of the path, but in general, the actual name of the initial state may cause confusion, e.g., s_0.
- A ***path*** $\pi = s_1 \longrightarrow s_2 \longrightarrow \ldots$ represents a ***possible future*** of $\mathbb{M}$.
- We write π^{i} for the ***suffix*** of path π: a path starting from state s_i.
  e.g., $\pi^3 = s_3 \longrightarrow s_4 \longrightarrow \ldots$
  e.g., $\pi^1 = \pi$

LTL Semantics: All Possible Paths

Given a state s, we represent **all** possible ***(computation) paths*** as a ***computation tree*** by *unwinding* the transitions.

e.g.





LTL Semantics: Path Satisfaction (1)

Definition. Given a ***model*** $\mathbb{M} = (S, \longrightarrow, L)$ and a ***path*** $\pi = s_1 \longrightarrow \dots$ in $\mathbb{M}$, whether or not path π satisfies an ***LTL formula*** is defined by the *satisfaction relation* $\models$ as follows:

$$
\begin{array}{lllll}
\pi & \models & p & \iff & p \in L(s_1) \\
\pi & \models & \top & & \\
\pi & \not\models & \bot & & \\
\pi & \models & \neg\phi & \iff & \neg(\pi \models \phi) \\
\pi & \models & \phi_1 \wedge \phi_2 & \iff & \pi \models \phi_1 \wedge \pi \models \phi_2 \\
\pi & \models & \phi_1 \vee \phi_2 & \iff & \pi \models \phi_1 \vee \pi \models \phi_2 \\
\pi & \models & \phi_1 \Rightarrow \phi_2 & \iff & \pi \models \phi_1 \Rightarrow \pi \models \phi_2
\end{array}
$$

Tips. To evaluate $\pi \models \phi_1 \wedge \phi_2$ (and similarly for $\neg$, $\vee$, $\Rightarrow$):

- If ϕ_1 and ϕ_2 are sophisticated, decompose it to $\pi \models \phi_1$ and $\pi \models \phi_2$.
- Otherwise, directly evaluate $\phi_1 \wedge \phi_2$ on s_1.

LTL Semantics: Path Satisfaction (2.1)

Definition. Given a ***model*** $\mathbb{M} = (S, \longrightarrow, L)$ and a ***path*** $\pi = s_1 \longrightarrow \dots$ in $\mathbb{M}$, whether or not path π satisfies an ***LTL formula*** is defined by the *satisfaction relation* $\models$ as follows:

$$
\begin{array}{lllll}
\pi & \models & \mathbf{X}\,\phi & \iff & \pi^2 \models \phi \\
\pi & \models & \mathbf{G}\,\phi & \iff & (\ \forall i \bullet i \geq 1 \Rightarrow \pi^i \models \phi\ ) \\
\pi & \models & \mathbf{F}\,\phi & \iff & (\ \exists i \bullet i \geq 1 \wedge \pi^i \models \phi\ )
\end{array}
$$

LTL Semantics: Model Satisfaction (1)

- **Definition**. Given:
  - a model $\mathbb{M} = (S, \longrightarrow, L)$
  - a state $s \in S$
  - an LTL formula ϕ

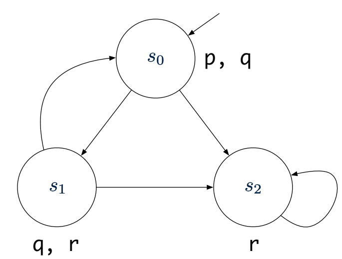
  $\boxed{\mathbb{M}, s \models \phi}$ if and only if for **every** path π of $\mathbb{M}$ starting at s, $\pi \models \phi$.

$$\mathbb{M}, s \models \phi \iff (\ \forall \pi \bullet (\ \pi = s \longrightarrow \dots\ ) \Rightarrow \pi \models \phi\ )$$

- When the model $\mathbb{M}$ is clear from the context, we write: $\boxed{s \models \phi}$.

LTL Semantics: Model Satisfaction (2.1)

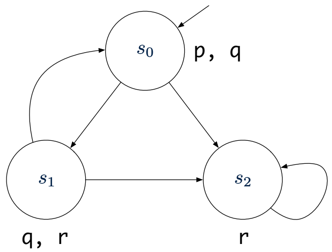
Consider the following system model:



- $s_0 \models \top$ [*true*]
- $s_0 \not\models \bot$ [*true*]
- $s_0 \models p \wedge q$ [*true*]
- $s_0 \models r$ [*false*]

LTL Semantics: Model Satisfaction (2.2)

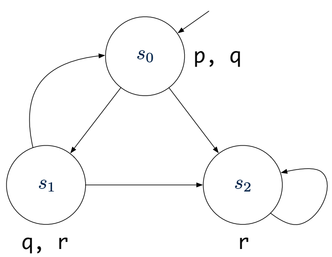
Consider the following system model:



- $s_0 \models \mathbf{X}\, q$ [*false*]
  Witness Path: $s_0 \longrightarrow \boxed{s_2} \longrightarrow s_2 \cdots \not\models \mathbf{X}\, q$
- $s_0 \models \mathbf{X}\, r$ [*true*]
- $s_0 \models \mathbf{X}(q \land r)$ [*false*]
  Witness Path: $s_0 \longrightarrow \boxed{s_2} \longrightarrow s_2 \cdots \not\models \mathbf{X}(q \land r)$
- $s_0 \models \mathbf{X}(q \Rightarrow r)$ [*true*]

LTL Semantics: Model Satisfaction (2.3)

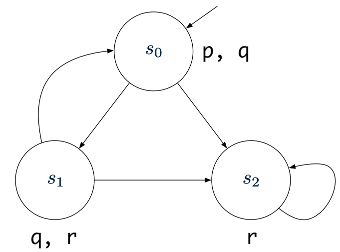
Consider the following system model:



- $s_0 \models \mathbf{G}\neg(p \land r)$ [*true*]
  $s \models \mathbf{G}\,\phi \iff \phi$ holds on all ***reachable*** states from s.
- $s_0 \models \mathbf{G}\, r$ [*false*]
  Witness Path: $\boxed{s_0} \longrightarrow s_2 \longrightarrow s_2 \cdots \not\models \mathbf{G}\, r$
- $s_2 \models \mathbf{G}\, r$ [*true*]

LTL Semantics: Model Satisfaction (2.4)

Consider the following system model:



- $s_0 \models \mathbf{F}\neg(p \land r)$ [*true*]
- $s_0 \models \mathbf{F}\, r$ [*true*]
- $s_0 \models \mathbf{F}(q \land r)$ [*false*]
  - Is is the case that $q \land r$ is eventually satisfied on every path?
  - No. Witness Path: $s_0 \longrightarrow s_2 \longrightarrow s_2 \longrightarrow \dots$
- $s_2 \models \mathbf{F}\, r$ [*true*]

LTL Semantics: Nested G and F (1)

Given a model $\mathbb{M} = (S, \longrightarrow, L)$ and a state $s \in S$:

$s \models \boldsymbol{F}\,\boldsymbol{G}\,\phi$ means that:

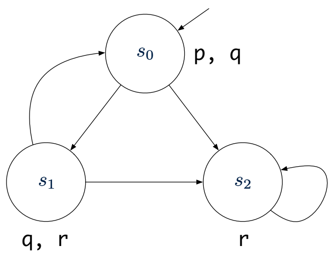
- **Each** path starting with s is such that **eventually**, ϕ holds **continuously**.
- For **all** paths π starting with s (i.e., $\pi = s \longrightarrow l \dots$):

$$\exists i \bullet i \geq 1 \land \left(\forall j \bullet j \geq i \Rightarrow \pi^i \models \phi\right)$$

- **Q.** How to ***prove*** and ***disprove*** the above formula pattern?
- **Hint.** Structure of pattern: $\forall \pi \bullet \dots \Rightarrow (\exists i \bullet \dots \land (\forall j \bullet \dots \Rightarrow \phi))$

LTL Semantics: Model Satisfaction (2.5.1)

Consider the following system model:



- $s_0 \models \mathbf{F}\,\mathbf{G}\, r$ [*false*]
  Witness: $s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow s_1 \longrightarrow \ldots$
- $s_0 \models \mathbf{F}\,\mathbf{G}(p \lor q)$ [*false*]
  Witness: $s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_2 \longrightarrow \ldots$
- $s_0 \models \mathbf{F}\,\mathbf{G}(p \lor r)$ [*true*]
  Justification: All possible paths from s_0 involve s_0, s_1, and s_2, all of which satisfying $p \lor r$.

LTL Semantics: Nested G and F (2)

Given a model $\mathbb{M} = (S, \longrightarrow, L)$ and a state $s \in S$:

$s \models \boldsymbol{F}\phi_1 \Rightarrow \boldsymbol{F}\boldsymbol{G}\phi_2$ means that:

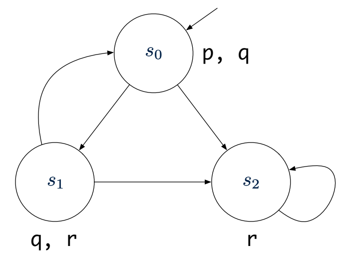
- **Each** path π starting with s is such that if ϕ_1 **eventually** holds on π, then ϕ_2 **eventually** holds **continuously** on the same π.

$$\forall \pi \bullet \pi = s \longrightarrow \ldots \Rightarrow \left(\begin{array}{l} (\exists i \bullet i \geq 1 \land \pi^i \models \phi_1) \\ \Rightarrow \\ (\exists i \bullet i \geq 1 \land (\forall j \bullet j \geq i \Rightarrow \pi^i \models \phi_2)) \end{array} \right)$$

- **Q.** How to ***disprove*** the above formula pattern?
- **A.** Find a witness path π which makes the "inner" implication ***false***.

LTL Semantics: Model Satisfaction (2.5.2)

Consider the following system model:



- $s_0 \models \mathbf{F}(\,\neg q \land r\,) \Rightarrow \mathbf{F}\,\mathbf{G}\, r$ [*true*]
  Justification:
  - $s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow \ldots$ never satisfies $\neg q \land r$.
  - $s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_2 \longrightarrow \ldots$ eventually satisfies $\neg q \land r$ continuously.
  - $s_0 \longrightarrow s_2 \longrightarrow s_2 \longrightarrow \ldots$ eventually satisfies $\neg q \land r$ continuously.
- $s_0 \models \mathbf{F}(\,\neg q \lor r\,) \Rightarrow \mathbf{F}\,\mathbf{G}\, r$ [*false*]
  Witness: $s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow \ldots$ eventually satisfies $\neg q \lor r$, but there is no point in this path where r holds continuously.

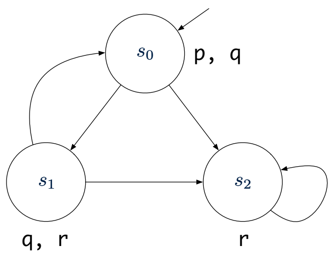
LTL Semantics: Nested G and F (3)

Given a model $\mathbb{M} = (S, \longrightarrow, L)$ and a state $s \in S$:

- $s \models \boldsymbol{G}\boldsymbol{F}\phi$ means that:
  - **Each** path starting with s is such that **continuously**, ϕ holds **eventually**.
    $\Rightarrow \phi$ holds ***infinitely often***!
  - For **all** paths π starting with s (i.e., $\pi = s \longrightarrow l \ldots$):
    $$\forall i \bullet i \geq 1 \Rightarrow (\exists j \bullet j \geq i \land \pi^i \models \phi)$$
  - **Q.** How to ***prove*** and ***disprove*** the above formula pattern?
  - **Hint.** Structure of pattern: $\forall \pi \bullet \ldots \Rightarrow (\forall i \bullet \ldots \Rightarrow (\exists j \bullet \cdots \land \phi))$

LTL Semantics: Model Satisfaction (2.6)

Consider the following system model:



- $s_0 \models \mathbf{G}\,\mathbf{F}\,p$ [*false*]
  Witness: In $s_0 \longrightarrow s_2 \longrightarrow \dots$, p is not satisfied ***infinitely often***.
- $s_0 \models \mathbf{G}\,\mathbf{F}(\,p \vee r\,)$ [*true*]
- $s_0 \models \mathbf{G}\,\mathbf{F}\,p \Rightarrow \mathbf{G}\,\mathbf{F}\,r$ [*true*]
  Hint: Consider paths making the antecedent $\mathbf{G}\,\mathbf{F}\,p$ ***true***.
- $s_0 \models \mathbf{G}\,\mathbf{F}\,r \Rightarrow \mathbf{G}\,\mathbf{F}\,p$ [*false*]
  Witness: $s_0 \longrightarrow s_2 \longrightarrow \dots$ [Why?]

LTL Semantics: Path Satisfaction (2.2)

Definition. Given a ***model*** $\mathbb{M} = (S, \longrightarrow, L)$ and a ***path*** $\pi = s_1 \longrightarrow \dots$ in $\mathbb{M}$, whether or not path π satisfies an ***LTL formula*** is defined by the *satisfaction relation* $\models$ as follows:

$$\pi \models \phi_1\,\mathbf{U}\,\phi_2 \iff \left(\exists i \bullet i \geq 1 \wedge \left(\begin{array}{l} \pi^i \models \phi_2 \\ \wedge \\ (\forall j \bullet 1 \leq j \leq i-1 \Rightarrow \pi^j \models \phi_1) \end{array} \right) \right)$$

$$\pi \models \phi_1\,\mathbf{W}\,\phi_2 \iff \left(\begin{array}{l} \phi_1\,\mathbf{U}\,\phi_2 \\ \vee \quad (\forall k \bullet k \geq 1 \Rightarrow \pi^k \models \phi_1) \end{array} \right)$$

$$\pi \models \phi_1\,\mathbf{R}\,\phi_2 \iff \left(\begin{array}{l} \left(\exists i \bullet i \geq 1 \wedge \left(\begin{array}{l} \pi^i \models \phi_1 \\ \wedge \\ (\forall j \bullet 1 \leq j \leq i \Rightarrow \pi^j \models \phi_2) \end{array} \right) \right) \\ \vee \quad (\forall k \bullet k \geq 1 \Rightarrow \pi^k \models \phi_2) \end{array} \right)$$

LTL Semantics: Recall Model Satisfaction

- **Definition**. Given:
  - a model $\mathbb{M} = (S, \longrightarrow, L)$
  - a state $s \in S$
  - an LTL formula ϕ

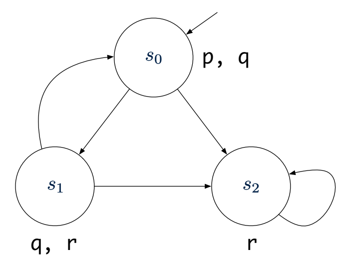
  $\mathbb{M}, s \models \phi$ if and only if for **every** path π of $\mathbb{M}$ starting at s, $\pi \models \phi$.

$$\mathbb{M}, s \models \phi \iff (\,\forall \pi \bullet (\,\pi = s \longrightarrow \dots\,) \Rightarrow \pi \models \phi\,)$$

- When the model $\mathbb{M}$ is clear from the context, we write: $s \models \phi$.

LTL Semantics: Model Satisfaction (3.1)

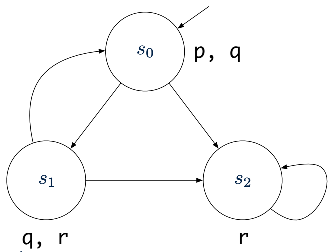
Consider the following system model:



- $s_0 \models p\,\mathbf{U}\,r$ [*true*]
  s_0 (satisfying p) branches out to s_1 or s_2 (both both satisfying r).
- $s_0 \models p\,\mathbf{W}\,r$ [*true*]
  $\phi_1\,\mathbf{U}\,\phi_2 \Rightarrow \phi_1\,\mathbf{W}\,\phi_2$
- $s_0 \models r\,\mathbf{R}\,p$ [*false*]
  Witness: Say $\pi = s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow s_1 \dots$: $\pi \not\models p \wedge r$ and $\pi \not\models \mathbf{G}\,p$.

LTL Semantics: Model Satisfaction (3.2)

Consider the following system model:



- $s_0 \models (p \lor r)\,\mathbf{U}(p \land r)$ [*false*]
  Witness: In $s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow s_1 \ldots$, $p \land r$ never holds.
- $s_0 \models (p \lor r)\,\mathbf{W}(p \land r)$ [*true*]
  It is the case that: $s_0 \models \mathbf{G}(p \lor r)$.
- $s_0 \models (p \land r)\,\mathbf{R}(p \lor r)$ [*true*]
  It is the case that: $s_0 \models \mathbf{G}(p \lor r)$.

Clarification on the "Until" Connective

- $\phi_1\,\mathbf{U}\,\phi_2$ requires that:
  - ϕ_2 must eventually become ***true***.
  - Before ϕ_2 becomes ***true***, ϕ_1 must hold.
- **Exercise**. Say:
  - Atom t: I was 22.
  - Atom s: I smoke.

  Formulate "I had smoked until I was 22" using LTL.
  - $s\,\mathbf{U}\,t$ [***inaccurate***]
  - $\phi_1\,\mathbf{U}\,\phi_2$ does not insist $\boxed{\neg\phi_1}$ after $\boxed{\phi_2}$ eventually becomes ***true***.
  - "I smoked both before and after I was 22" satisfies $s\,\mathbf{U}\,t$.
  - Solution? [$s\ \mathbf{U}\ (\ t \land (\mathbf{G}\neg s)\ )$]

Formulating English as LTL Formulas (1)

- Assume the following atomic propositions:
  busy, *requested*, *acknowledged*, *enabled*, *floor2*, *floor5*, *directionUp*, *buttonPresssed5*.
- It is impossible to reach a state where the system is started but not ready.
  - $\mathbf{G}\,\neg(\mathit{started} \land \neg \mathit{ready})$ [$\neg(\ \mathbf{F}(\mathit{started} \land \neg\mathit{ready})\ )$]
- Whenever a request is made, it will be eventually be acknowledged.
  - $\mathbf{G}(\ \mathit{requested} \Rightarrow \mathbf{F}\,\mathit{acknowledged}\ )$
- A certain process will always be enabled.
  - $\mathbf{G}\,\mathit{enabled}$
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor.
  - $$\mathbf{G}\left(\begin{array}{c} \mathit{floor2} \land \mathit{directionUp} \land \mathit{buttonPresssed5} \\ \Rightarrow (\ \mathit{directionUp}\,\mathbf{U}\,\mathit{floor5}\ ) \end{array}\right)$$
  - Is it ok to change from **U** to **W**?

Formulating English as LTL Formulas (2)

Assume the following atomic propositions:

$$\mathit{requested}, \mathit{waiting}, \mathit{granted}, \mathit{noOneInCS}$$

Whenever a process makes a request, it starts waiting. As soon as no other process is in the critical section, the process is granted access to the critical section.

$$\mathbf{G}\ (\mathit{requested} \Rightarrow (\mathit{noOneInCS}\ \ \mathbf{R}\ \ \mathit{waiting}))$$

Q. Does the above formulation guarantee ***no starvation***?

Hint. Check the formal definition of **R**.

Formulating English as LTL Formulas (3)

Assume the following atomic propositions:

$$\textit{degReqFullfilled}, \textit{allowedForGraduation}$$

Until a student fullfils all their degree requirements, their academic staus remains "not allowed for graduation". The change of status, when qualified, may not be instantaneous to account for human/manual processing.

$$\neg \textit{allowedForGraduation}\ \mathbf{W}\ (\textit{degReqFulfilled} \land \mathbf{G}\ \textit{allowedForGraduation})$$

Q. Does the above formulation account for situations where a student never fulfills their degree requirements?

Hint. Check the formal definition of **W**.

Index (1)

Index (2)

Index (3)

Index (4)