Verification by Model Checking

EECS4315 Z:
Mission-Critical Systems
Winter 2025

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 $\mathbb{M} \models \phi$ [*decidable*, *algorithmic*]
      i.e., Traversing $\mathbb{M}$'s ***state/reachability graph*** decides if ϕ is satisfied.
- **Domain of Application**
  - Hardware vs. Software
  - Sequential vs. Concurrent
  - Reactive (e.g., bridge controller) vs. Terminating (e.g., sorting alg.)
- **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*]
|	$\bot$	[*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`]

$\forall$ 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*]
|	$\bot$	[*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`]

- ***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*** (<u>Left</u>-Most Derivations)
  - Show using the ***RMD*** (<u>Right</u>-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 <u>subtrees</u> of ϕ's parse tree.

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

1. p [appearing twice in the parse tree]
2. r
3. $\mathbf{G}\,r$
4. $p \Rightarrow (\mathbf{G}\,r)$
5. $\mathbf{F}(p \Rightarrow (\mathbf{G}\,r))$
6. q
7. $\neg q$
8. p
9. $(\neg q)\,\mathbf{U}\,p$
10. $(\ \mathbf{F}(p \Rightarrow \mathbf{G}\,r) \lor ((\neg q)\,\mathbf{U}\,p)\ )$

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 <u>**finite**</u> set of ***states***
  - $\longrightarrow : S \leftrightarrow S$
    A ***transition relation*** on S
  - $L : S \to \mathbb{P}(P)$
    A ***labelling function*** mapping each <u>state</u> to its <u>satisfying atoms</u>

  Assumption. No state of the system can ***deadlock***:
  From any state, it's always possible to make progress (by taking a transition).

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

Background for Self-Study

- Topics of ***sets*** and ***relations*** were covered in EECS3342.
- Slide 18 to Slide 28 contain what you should recall.

Set of Tuples

Given n sets $S_1, S_2, \ldots, S_n$, a ***cross/Cartesian product*** of theses sets is a set of n-tuples.

Each ***n-tuple*** $(e_1, e_2, \ldots, e_n)$ contains n elements, each of which a member of the corresponding set.

$$S_1 \times S_2 \times \cdots \times S_n = \{(e_1, e_2, \ldots, e_n) \mid e_i \in S_i \land 1 \le i \le n\}$$

e.g., $\{a, b\} \times \{2, 4\} \times \{\$, \&\}$ is a set of triples:

$$\begin{array}{cl} & \{a, b\} \times \{2, 4\} \times \{\$, \&\} \\ = & \{\ (e_1, e_2, e_3) \mid e_1 \in \{a, b\} \land e_2 \in \{2, 4\} \land e_3 \in \{\$, \&\}\ \} \\ = & \left\{ \begin{array}{l} (a, 2, \$), (a, 2, \&), (a, 4, \$), (a, 4, \&), \\ (b, 2, \$), (b, 2, \&), (b, 4, \$), (b, 4, \&) \end{array} \right\} \end{array}$$

Relations (1): Constructing a Relation

A *relation* is a set of mappings, each being an ***ordered pair*** that maps a member of set S to a member of set T.

e.g., Say $S = \{1, 2, 3\}$ and $T = \{a, b\}$

- $\varnothing$ is the ***minimum*** relation (i.e., an empty relation).
- $S \times T$ is the ***maximum*** relation (say r_1) between S and T, mapping from each member of S to each member in T:

$$\{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$$

- $\{(x, y) \mid (x, y) \in S \times T \land x \ne 1\}$ is a relation (say r_2) that maps only some members in S to every member in T:

$$\{(2, a), (2, b), (3, a), (3, b)\}$$

Relations (2.1): Set of Possible Relations

- We use the ***power set*** operator to express the set of ***all possible relations*** on S and T:

$$\mathbb{P}(S \times T)$$

  Each member in $\mathbb{P}(S \times T)$ is a relation.
- To declare a relation variable r, we use the colon (`:`) symbol to mean ***set membership***:

$$r : \mathbb{P}(S \times T)$$

- Or alternatively, we write:

$$r : S \leftrightarrow T$$

  where the set $S \leftrightarrow T$ is synonymous to the set $\mathbb{P}(S \times T)$

Relations (2.2): Exercise

Enumerate $\{a,b\} \leftrightarrow \{1,2,3\}$.

- **Hints**:
  - You may enumerate all relations in $\mathbb{P}(\{a,b\} \times \{1,2,3\})$ via their *cardinalities*: 0, 1, ..., $|\{a,b\} \times \{1,2,3\}|$.
  - What's the *maximum* relation in $\mathbb{P}(\{a,b\} \times \{1,2,3\})$?

    $\{\ (a,1),(a,2),(a,3),(b,1),(b,2),(b,3)\ \}$
- The answer is a set containing ***all*** of the following relations:
  - Relation with cardinality 0: $\varnothing$
  - How many relations with cardinality 1? $[\ \binom{|\{a,b\}\times\{1,2,3\}|}{1} = 6\ ]$
  - How many relations with cardinality 2? $[\ \binom{|\{a,b\}\times\{1,2,3\}|}{2} = \frac{6\times5}{2!} = 15\ ]$

    ...
  - Relation with cardinality $|\{a,b\} \times \{1,2,3\}|$:

    $\{\ (a,1),(a,2),(a,3),(b,1),(b,2),(b,3)\ \}$

Relations (3.1): Domain, Range, Inverse

Given a relation

$r = \{(a, 1), (b, 2), (c, 3), (a, 4), (b, 5), (c, 6), (d, 1), (e, 2), (f, 3)\}$

- ***domain*** **of** r: set of first-elements from r
  - Definition: $\text{dom}(r) = \{\ d \mid (d, r') \in r\ \}$
  - e.g., $\text{dom}(r) = \{a, b, c, d, e, f\}$
- ***range*** **of** r: set of second-elements from r
  - Definition: $\text{ran}(r) = \{\ r' \mid (d, r') \in r\ \}$
  - e.g., $\text{ran}(r) = \{1, 2, 3, 4, 5, 6\}$
- ***inverse*** **of** r: a relation like r with elements swapped
  - Definition: $r^{-1} = \{\ (r', d) \mid (d, r') \in r\ \}$
  - e.g., $r^{-1} = \{(1,a),(2,b),(3,c),(4,a),(5,b),(6,c),(1,d),(2,e),(3,f)\}$

Relations (3.2): Image

Given a relation

$r = \{(a, 1), (b, 2), (c, 3), (a, 4), (b, 5), (c, 6), (d, 1), (e, 2), (f, 3)\}$

relational image **of** r **over set** s: sub-range of r mapped by s.

- Definition: $r[s] = \{\ r' \mid (d, r') \in r \wedge d \in s\ \}$
- e.g., $r[\{a, b\}] = \{1, 2, 4, 5\}$

Relations (3.3): Restrictions

Given a relation

$r = \{(a, 1), (b, 2), (c, 3), (a, 4), (b, 5), (c, 6), (d, 1), (e, 2), (f, 3)\}$

- ***domain restriction*** **of** r **over set** ds: sub-relation of r with domain ds.
  - Definition: $ds \lhd r = \{\ (d, r') \mid (d, r') \in r \wedge d \in ds\ \}$
  - e.g., $\{a, b\} \lhd r = \{(\mathbf{a}, 1), (\mathbf{b}, 2), (\mathbf{a}, 4), (\mathbf{b}, 5)\}$
- ***range restriction*** **of** r **over set** rs: sub-relation of r with range rs.
  - Definition: $r \rhd rs = \{\ (d, r') \mid (d, r') \in r \wedge r' \in rs\ \}$
  - e.g., $r \rhd \{1, 2\} = \{(a, \mathbf{1}), (b, \mathbf{2}), (d, \mathbf{1}), (e, \mathbf{2})\}$

Relations (3.4): Subtractions

Given a relation

r = {(a, 1), (b, 2), (c, 3), (a, 4), (b, 5), (c, 6), (d, 1), (e, 2), (f, 3)}

- ***domain subtraction*** **of** r **over set** ds: sub-relation of r with domain not ds.
  - Definition: $ds ⩤ r = \{ (d, r') \mid (d, r') \in r \wedge d \notin ds \}$
  - e.g., $\{a, b\} ⩤ r = \{(\mathbf{c}, 3), (\mathbf{c}, 6), (\mathbf{d}, 1), (\mathbf{e}, 2), (\mathbf{f}, 3)\}$
- ***range subtraction*** **of** r **over set** rs: sub-relation of r with range not rs.
  - Definition: $r ⩥ rs = \{ (d, r') \mid (d, r') \in r \wedge r' \notin rs \}$
  - e.g., $r ⩥ \{1, 2\} = \{(c, \mathbf{3}), (a, \mathbf{4}), (b, \mathbf{5}), (c, \mathbf{6}), (f, \mathbf{3})\}$

Functions (1): Functional Property

- A ***relation*** r on sets S and T (i.e., $r \in S \leftrightarrow T$) is also a ***function*** if it satisfies the ***functional property***:

$$\texttt{isFunctional}(r) \iff \forall s, t_1, t_2 \bullet (s \in S \wedge t_1 \in T \wedge t_2 \in T) \Rightarrow ((s, t_1) \in r \wedge (s, t_2) \in r \Rightarrow t_1 = t_2)$$

  - That is, in a ***function***, it is forbidden for a member of S to map to more than one members of T.
  - Equivalently, in a ***function***, two distinct members of T cannot be mapped by the same member of S.
- e.g., Say $S = \{1, 2, 3\}$ and $T = \{a, b\}$, which of the following ***relations*** satisfy the above ***functional property***?
  - $S \times T$ [No]
    Witness 1: $(1, a), (1, b)$; ***Witness*** 2: $(2, a), (2, b)$; ***Witness*** 3: $(3, a), (3, b)$.
  - $(S \times T) \setminus \{(x, y) \mid (x, y) \in S \times T \wedge x = 1\}$ [No]
    Witness 1: $(2, a), (2, b)$; ***Witness*** 2: $(3, a), (3, b)$
  - $\{(1, a), (2, b), (3, a)\}$ [Yes]
  - $\{(1, a), (2, b)\}$ [Yes]

Functions (2.1): Total vs. Partial

Given a **relation** $r \in S \leftrightarrow T$

- r is a ***partial function*** if it satisfies the ***functional property***:

$$r \in S \mathrel{+\!\!\!\to} T \iff (\texttt{isFunctional}(r) \wedge \mathrm{dom}(r) \subseteq S)$$

  Remark. $r \in S \mathrel{+\!\!\!\to} T$ means there **may (or may not) be** $s \in S$ s.t. $r(s)$ is ***undefined*** (i.e., $r[\{s\}] = \varnothing$).
  - e.g., $\{ \{(\mathbf{2}, a), (\mathbf{1}, b)\}, \{(\mathbf{2}, a), (\mathbf{3}, a), (\mathbf{1}, b)\} \} \subseteq \{1, 2, 3\} \mathrel{+\!\!\!\to} \{a, b\}$
- r is a ***total function*** if there is a mapping for each $s \in S$:

$$r \in S \to T \iff (\texttt{isFunctional}(r) \wedge \mathrm{dom}(r) = S)$$

  Remark. $r \in S \to T$ implies $r \in S \mathrel{+\!\!\!\to} T$, but not vice versa. Why?
  - e.g., $\{(\mathbf{2}, a), (\mathbf{3}, a), (\mathbf{1}, b)\} \in \{1, 2, 3\} \to \{a, b\}$
  - e.g., $\{(\mathbf{2}, a), (\mathbf{1}, b)\} \notin \{1, 2, 3\} \to \{a, b\}$

Functions (2.2): Relation Image vs. Function Application

- Recall: A ***function*** is a ***relation***, but a ***relation*** is not necessarily a ***function***.
- Say we have a ***partial function*** $f \in \{1, 2, 3\} \mathrel{+\!\!\!\to} \{a, b\}$:

$$f = \{(\mathbf{3}, a), (\mathbf{1}, b)\}$$

  - With f wearing the ***relation*** hat, we can invoke *relational images*:

$$\begin{array}{lcl} f[\{3\}] & = & \{a\} \\ f[\{1\}] & = & \{b\} \\ f[\{2\}] & = & \varnothing \end{array}$$

  Remark. $\Rightarrow |f[\{v\}]| \le 1$ ∵
    - each member in $\mathrm{dom}(f)$ is mapped to at most one member in $\mathrm{ran}(f)$
    - each input set $\{v\}$ is a **singleton** set
  - With f wearing the ***function*** hat, we can invoke *functional applications*:

$$\begin{array}{lcl} f(3) & = & a \\ f(1) & = & b \\ f(2) & \texttt{is} & \textbf{\texttt{undefined}} \end{array}$$

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:

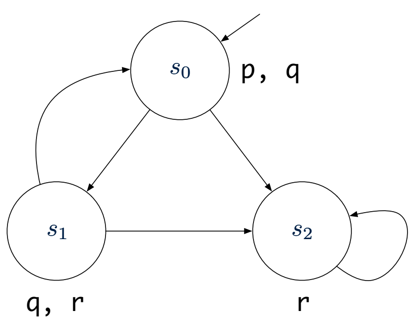
$$0 \leq c \leq 3$$

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

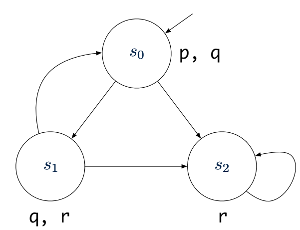
$$s_1 \longrightarrow s_2 \longrightarrow \ldots$$

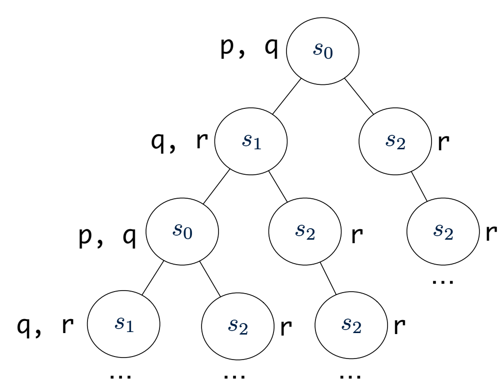
- **Note.** s_1 in the above path pattern denotes the first, initial state of the path, but in general, the actual name of the initial state may cause confusion, e.g., s_0.
- A ***path*** $\pi = s_1 \longrightarrow s_2 \longrightarrow \ldots$ represents a ***possible future*** of $\mathbb{M}$.
- We write 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* $\vDash$ as follows:

$$
\begin{array}{lllll}
\pi & \vDash & p & \iff & p \in L(s_1) \\
\pi & \vDash & \top & & \\
\pi & \nvDash & \bot & & \\
\pi & \vDash & \neg\phi & \iff & \neg(\pi \vDash \phi) \\
\pi & \vDash & \phi_1 \wedge \phi_2 & \iff & \pi \vDash \phi_1 \wedge \pi \vDash \phi_2 \\
\pi & \vDash & \phi_1 \vee \phi_2 & \iff & \pi \vDash \phi_1 \vee \pi \vDash \phi_2 \\
\pi & \vDash & \phi_1 \Rightarrow \phi_2 & \iff & \pi \vDash \phi_1 \Rightarrow \pi \vDash \phi_2
\end{array}
$$

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

- If ϕ_1 and ϕ_2 are sophisticated, decompose it to $\pi \vDash \phi_1$ and $\pi \vDash \phi_2$.
- 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* $\vDash$ as follows:

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

LTL Semantics: Model Satisfaction (1)

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

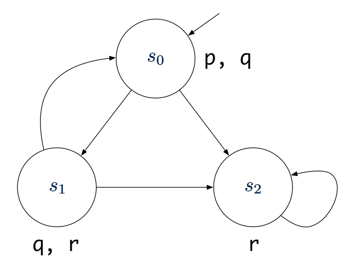
  $\mathbb{M}, s \vDash \phi$ if and only if for **every** path π of $\mathbb{M}$ starting at s, $\pi \vDash \phi$.

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

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

LTL Semantics: Model Satisfaction (2.1)

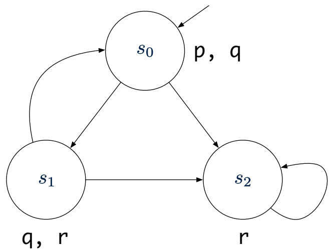
Consider the following system model:



- $s_0 \vDash \top$ [*true*]
- $s_0 \nvDash \bot$ [*true*]
- $s_0 \vDash p \wedge q$ [*true*]
- $s_0 \vDash r$ [*false*]

LTL Semantics: Model Satisfaction (2.2)

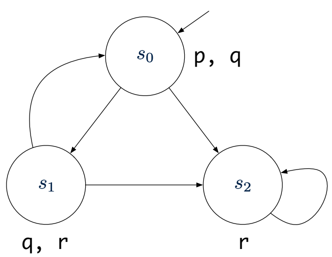
Consider the following system model:



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

LTL Semantics: Model Satisfaction (2.3)

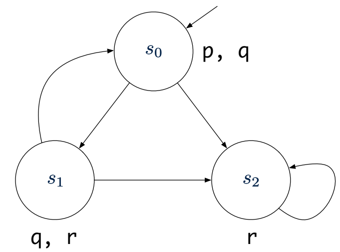
Consider the following system model:



- $s_0 \models \mathbf{G} \neg(p \wedge r)$ [*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:



- $s_0 \models \mathbf{F} \neg(p \wedge r)$ [*true*]
- $s_0 \models \mathbf{F}\, r$ [*true*]
- $s_0 \models \mathbf{F}(q \wedge r)$ [*false*]
  - Is is the case that $q \wedge r$ 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:

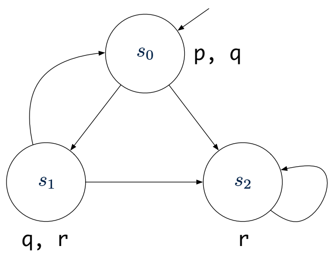
- **Each** path starting with s is such that **eventually**, ϕ holds **continuously**.
- For **all** paths π starting with s (i.e., $\pi = s \longrightarrow l \dots$):

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

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

LTL Semantics: Model Satisfaction (2.5.1)

Consider the following system model:



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

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:

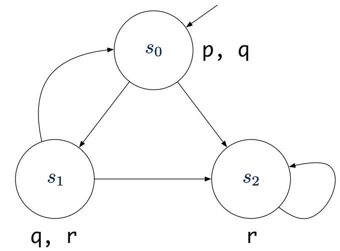
- **Each** path π starting with s is such that if ϕ_1 **eventually** holds on π, then ϕ_2 **eventually** holds **continuously** on the same π.

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

- **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:



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

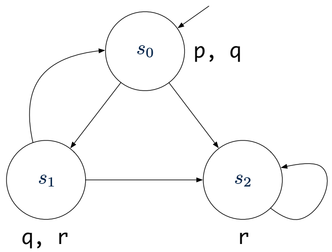
LTL Semantics: Nested G and F (3)

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., $\pi = s \longrightarrow l \dots$):
    $$\forall i \bullet i \geq 1 \Rightarrow (\exists j \bullet j \geq i \wedge \pi^i \models \phi)$$
  - **Q.** How to ***prove*** and ***disprove*** the above formula pattern?
  - **Hint.** Structure of pattern: $\forall \pi \bullet \dots \Rightarrow (\forall i \bullet \dots \Rightarrow (\exists j \bullet \dots \wedge \phi))$

LTL Semantics: Model Satisfaction (2.6)

Consider the following system model:



- $s_0 \models \mathbf{G}\,\mathbf{F}\, p$ [*false*]
  Witness: In $s_0 \longrightarrow s_2 \longrightarrow \ldots$, p is not satisfied ***infinitely often***.
- $s_0 \models \mathbf{G}\,\mathbf{F}(\ p \lor r\ )$ [*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: $s_0 \longrightarrow s_2 \longrightarrow \ldots$ [Why?]

LTL Semantics: Path Satisfaction (2.2)

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

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

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

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

LTL Semantics: Recall Model Satisfaction

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

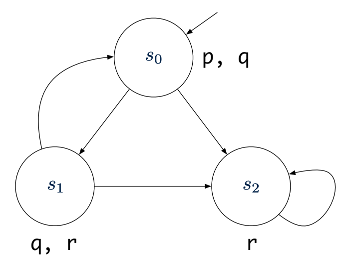
  $\mathbb{M}, s \models \phi$ if and only if for **every** path π of $\mathbb{M}$ starting at s, $\pi \models \phi$.

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

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

LTL Semantics: Model Satisfaction (3.1)

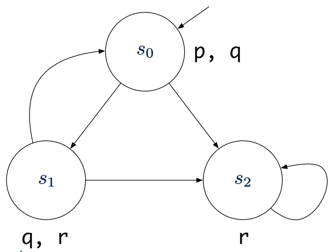
Consider the following system model:



- $s_0 \models p \,\mathbf{U}\, r$ [*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 $\pi = s_0 \longrightarrow s_1 \longrightarrow s_0 \longrightarrow s_1 \ldots$: $\pi \not\models p \land r$ and $\pi \not\models \mathbf{G}\, p$.

LTL Semantics: Model Satisfaction (3.2)

Consider the following system model:



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

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? [$s\ \mathbf{U}\ (\ t \wedge (\mathbf{G}\neg s)\ )$]

Formulating English as LTL Formulas (1)

- Assume the following atomic propositions:
  busy, *requested*, *acknowledged*, *enabled*, *floor*2, *floor*5, *directionUp*, *buttonPresssed*5.
- It is impossible to reach a state where the system is started but not ready.
  - $\mathbf{G}\,\neg(started \wedge \neg ready)$ [$\neg(\ \mathbf{F}(started \wedge \neg ready)\ )$]
- Whenever a request is made, it will be eventually be acknowledged.
  - $\mathbf{G}(\ requested \Rightarrow \mathbf{F}\,acknowledged\ )$
- A certain process will always be enabled.
  - $\mathbf{G}\,enabled$
- An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor.
  - $$\mathbf{G}\left(\begin{array}{c} floor2 \wedge directionUp \wedge buttonPresssed5 \\ \Rightarrow (\ directionUp\,\mathbf{U}\,floor5\ ) \end{array}\right)$$
  - Is it ok to change from **U** to **W**?

Formulating English as LTL Formulas (2)

Assume the following atomic propositions:

requested, *waiting*, *granted*, *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.

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

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:

$$degReqFullfilled, allowedForGraduation$$

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.

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

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)

Index (5)