CSE 620: Selective Topics

Introduction to Formal Verification

Master Studies in CSE

Fall 2016

Lecture #7

Dr. Hazem Ibrahim Shehata

Assistant Professor

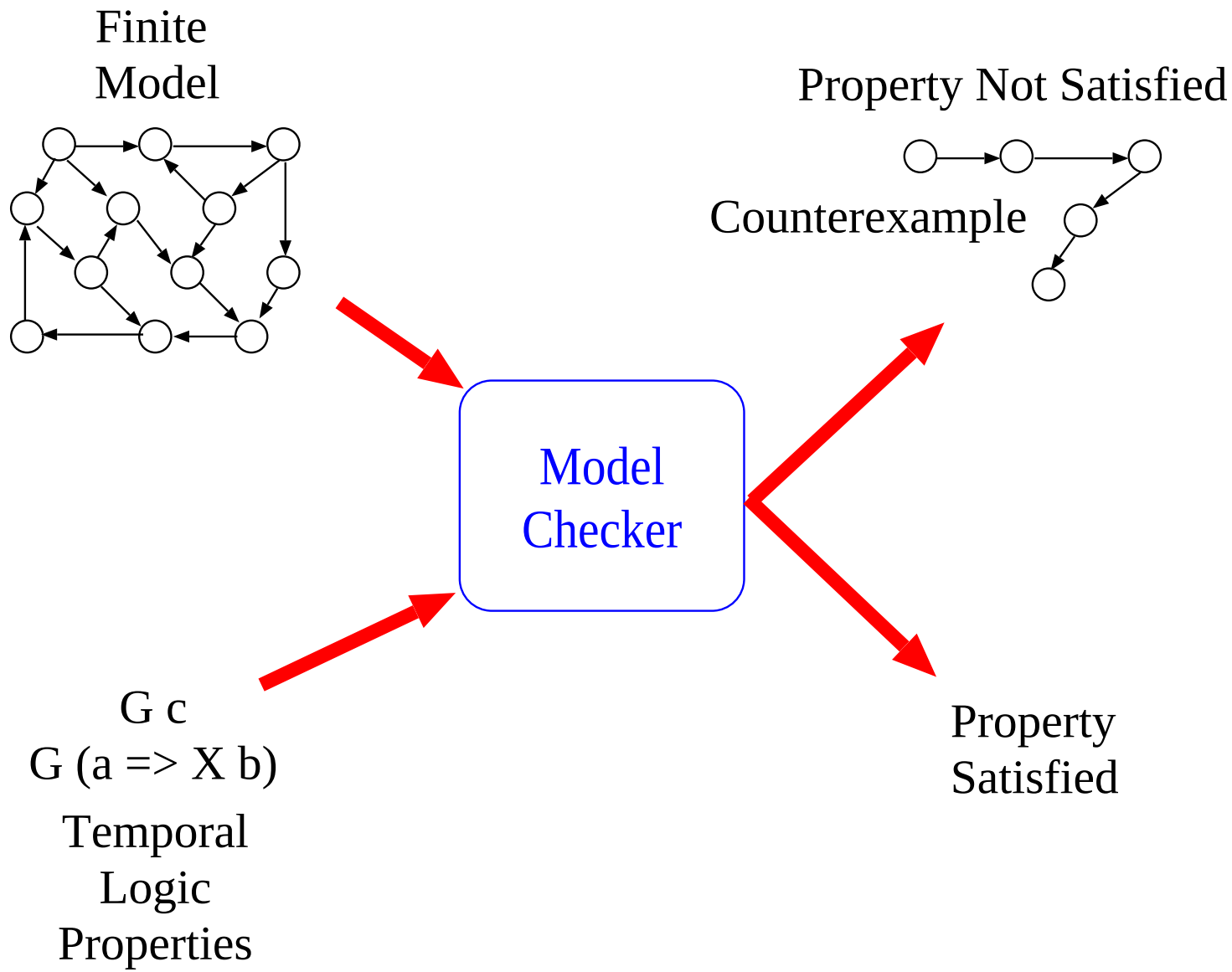
Dept. of Computer & Systems Engineering

Course Outline

- Computational Boolean Algebra
  - —Basics
    - – Shannon Expansion
    - – Boolean Difference
    - – Quantification Operators
      - + Application to Logic Network Repair
  - —Validity Checking (Tautology Checking)
  - —Binary Decision Diagrams (BDD's)
  - —Satisfiability Checking (SAT solving)
- Model Checking
  - —Temporal Logics ➔ LTL - CTL
  - —SMV: Symbolic Model Verifier
  - —Model Checking Algorithms ➔ Explicit CTL

Review Material

Model Checking



Verification via Model Checking

$$\mathcal{M} \models \phi$$

- $\mathcal{M}$ is a model (or implementation)
- ϕ is a property (or specification)
- $\models$ is a relationship that should hold between $\mathcal{M}$ and ϕ

In model checking, the model is a Kripke structure (labelled state transition graph), and the properties are written in temporal logic. We rarely work with Kripke structures directly but rather specification notations that describe Kripke structures.

There are different temporal logics. For each temporal logic, we will define what $\models$ means.

Kripke Structures

Let AP be a set of atomic propositions. A Kripke structure $\mathcal{M}$ over AP is a four tuple $\mathcal{M} = (S, S_0, R, L)$ where

1. S is a finite set of states.
2. $S_0 \subseteq S$ is the set of initial states.
3. $R \subseteq S \times S$ is a transition relation that must be total, that is $\forall s \in S.\, \exists s'.\, R(s, s')$.
4. $L : S \to 2^{\mathrm{AP}}$ is a function that labels each state with the set of atomic propositions true in that state.

Note: there are no labels on the transition arrows.

Linear Temporal Logic (LTL)

Temporal logics use temporal operators. Linear temporal logic uses the following operators:

Symbol	Alternate Symbol	Informal Meaning
X	$\bigcirc$	Next
F	$\Diamond$	Eventually (in the future)
G	$\Box$	always (globally, henceforth)
U	$\bigcup$	strong until
W	**W**	weak until

LTL Syntax

If p is an atomic proposition, and f_1 and f_2 are LTL formulae, then the set of LTL formulae consists of:

1. p
2. $\neg f_1$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1 \Rightarrow f_2$
3. $\mathbf{X} f_1$
4. $\mathbf{G} f_1$
5. $\mathbf{F} f_1$
6. $f_1 \ \mathbf{U} \ f_2$

Brackets are used as necessary.

LTL Semantics

$\mathcal{M}, \pi \models g$ means the LTL formula g holds on path $\pi = s_0 s_1 s_2 \ldots$ in the Kripke structure $\mathcal{M} = (S, S_0, R, L)$. The relation $\models$ is defined inductively as follows:

$$
\begin{array}{lll}
\mathcal{M}, \pi \models p & \text{iff} & p \in L(s_0) \text{ (where } p \text{ is an atomic proposition)} \\
\mathcal{M}, \pi \models \neg g & \text{iff} & \mathcal{M}, \pi \not\models g \\
\mathcal{M}, \pi \models g_1 \vee g_2 & \text{iff} & \mathcal{M}, \pi \models g_1 \text{ or } \mathcal{M}, \pi \models g_2 \\
\mathcal{M}, \pi \models g_1 \wedge g_2 & \text{iff} & \mathcal{M}, \pi \models g_1 \text{ and } \mathcal{M}, \pi \models g_2 \\
\mathcal{M}, \pi \models \mathbf{X}\, g & \text{iff} & \mathcal{M}, \pi_1 \models g \\
\mathcal{M}, \pi \models \mathbf{G}\, g & \text{iff} & \forall i \geq 0,\ \mathcal{M}, \pi_i \models g \\
\mathcal{M}, \pi \models \mathbf{F}\, g & \text{iff} & \exists i \geq 0,\ \mathcal{M}, \pi_i \models g \\
\mathcal{M}, \pi \models g_1\ \mathbf{U}\ g_2 & \text{iff} & \exists i \geq 0,\ \mathcal{M}, \pi_i \models g_2 \\
 & & \quad \text{and } \forall j.\, 0 \leq j < i \Rightarrow \mathcal{M}, \pi_j \models g_1
\end{array}
$$

LTL Semantics

We just defined $\mathcal{M}, \pi \models g$, where π is a path.

An LTL formula g is satisfied in a state s of a model $\mathcal{M}$ if g is satisfied on every path starting at s.

How do we say there is some path where a formula is true?

For example, "from any state, there is always some way to get to a state where p is true".

In LTL we can't say this property. We turn now to another temporal logic: CTL.

LTL is conceptually simpler than CTL because we only have to think about one path at a time.

New Material

Linear and Branching Views

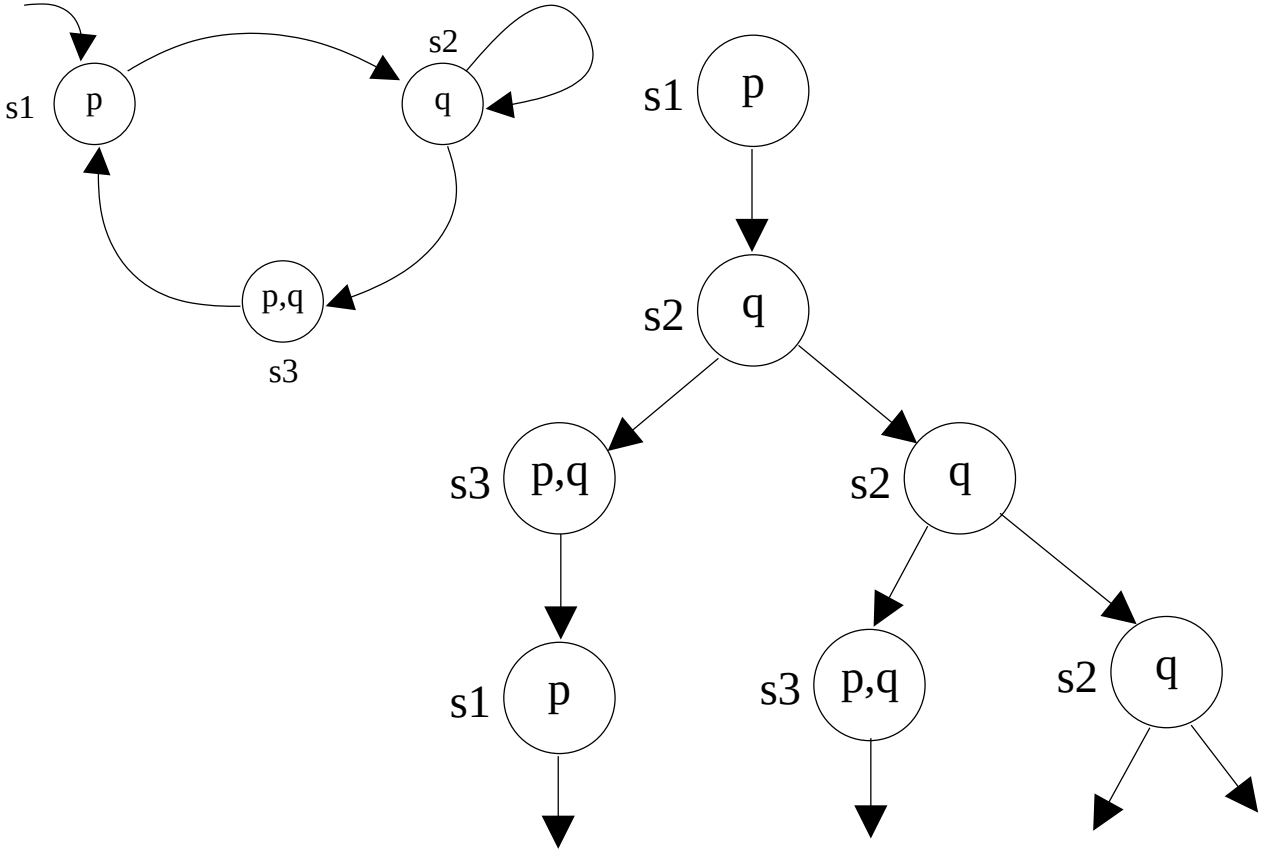
There are two ways to think about the computations of a system:

- linear-time: there is a single time line and a computation is the sequence of states that the system goes through in that time.
- branching-time: the possible computations of the system are described by a tree

LTL uses the linear-time view and CTL uses the branching-time view.

Computation Trees

A computation tree is the unwinding of a Kripke structure starting from some state at its root.



Computation Tree Logic (CTL)

In CTL, there are the temporal operators of LTL, but there are also path quantifiers. These path quantifiers are used to describe the branching structure of a computation tree.

There are two path quantifiers:

- **A** means for all computation paths
- **E** means for some computation paths

These are used to describe the behaviour of the system from a particular state.

In CTL, we talk about a formula being true of a state rather than a path. (Then we check that it is true for all initial states of the system.)

CTL Syntax

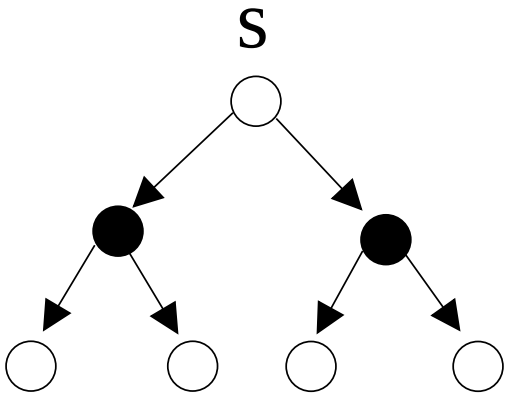
If p is an atomic proposition, and f_1 and f_2 are CTL formulae, then the set of CTL formulae consists of:

1. p
2. $\neg f_1$, $f_1 \wedge f_2$, $f_1 \vee f_2$, $f_1 \Rightarrow f_2$
3. $\mathbf{AX} f_1$, $\mathbf{EX} f_1$
4. $\mathbf{AG} f_1$, $\mathbf{EG} f_1$
5. $\mathbf{AF} f_1$, $\mathbf{EF} f_1$
6. $\mathbf{A}[f_1 \mathbf{U} f_2]$, $\mathbf{E}[f_1 \mathbf{U} f_2]$

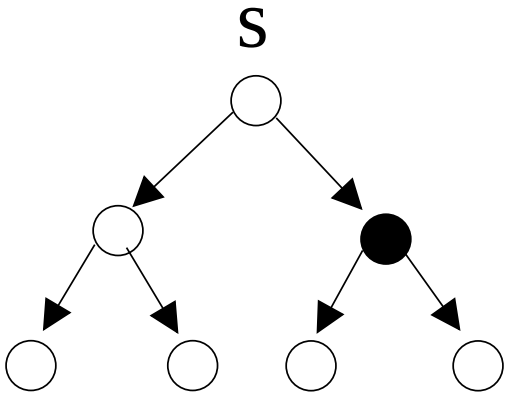
Note that the path quantifiers and temporal operators are always paired together.

Meaning of CTL Formulae

AXf if on all paths starting at state s, f holds in the next state.

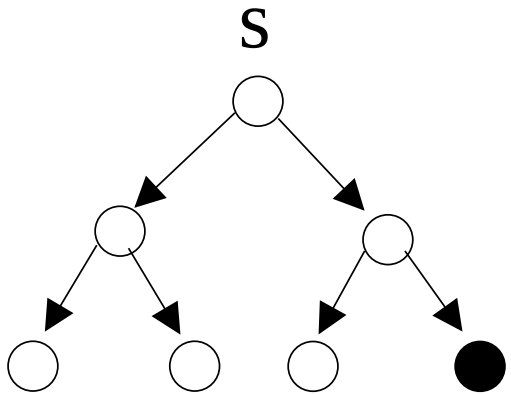


EXf if there exists a path starting at state s on which f holds at the next state.

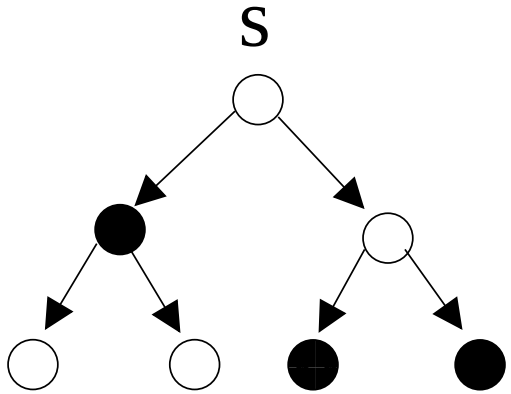


Meaning of CTL Formulae

EFf if f is reachable (i.e., if there exists a path starting at state s, on which f holds in some future state).

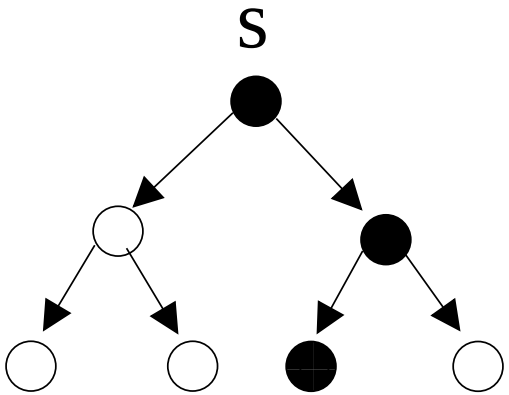


AFf if f is inevitable (i.e., if on all paths that start at state s, f holds in some future state).

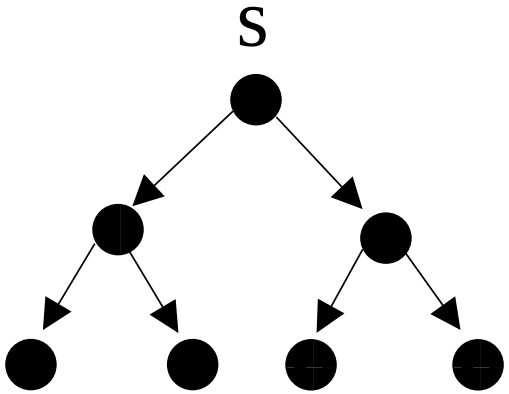


Meaning of CTL Formulae

EGf if there exists a path starting at state s, on which f holds globally.

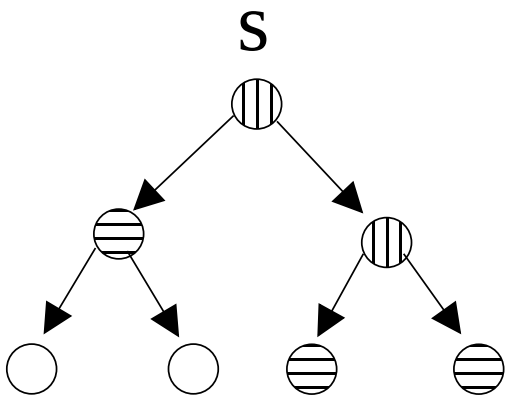


AGf if f is invariant (i.e., if on all paths that start at state s, f holds globally).

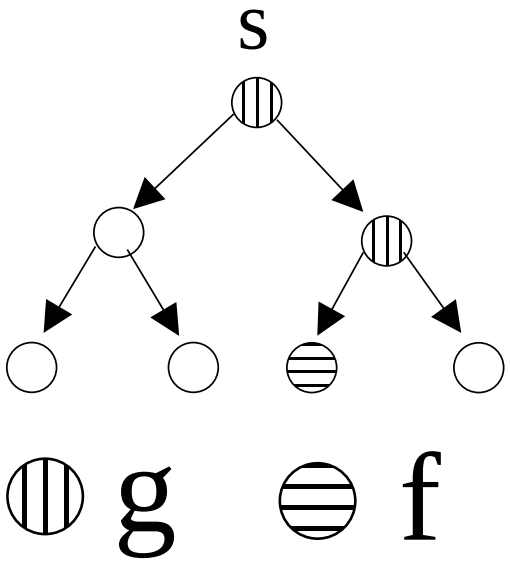


Meaning of CTL Formulae

E[g **U** f] if there exists a path starting at state s, on which g holds until f eventually holds.



A[g **U** f] if on all paths that start at state s, g holds until f eventually holds.



Examples of CTL Formulae

From Huth and Ryan [R17], p. 165:

"It is possible to get to a state where started holds, but ready does not hold."

"For any state, if a request occurs, then it will eventually be acknowledged."

"It is always the case that enabled is true infinitely often on every computation path."

Examples of CTL Formulae

"Whatever happens, the system will eventually be permanently deadlocked."

From any state it is possible to get to a state where restart is true.

Semantics of CTL

CTL formulae are evaluated with respect to a Kripke structure $\mathcal{M} = (S, S_0, R, L)$ and a state s (recall $\pi = s_0 s_1 s_2 \ldots$).

$$
\begin{aligned}
\mathcal{M}, s &\models p && \text{iff} && p \in L(s) \text{ where } p \text{ is an atomic proposition} \\
\mathcal{M}, s &\models \neg g && \text{iff} && \mathcal{M}, s \not\models g \\
\mathcal{M}, s &\models f_1 \vee f_2 && \text{iff} && \mathcal{M}, s \models f_1 \text{ or } \mathcal{M}, s \models f_2 \\
\mathcal{M}, s &\models f_1 \wedge f_2 && \text{iff} && \mathcal{M}, s \models f_1 \text{ and } \mathcal{M}, s \models f_2 \\
\mathcal{M}, \pi &\models f && \text{iff} && s \text{ is the first state of } \pi \text{ and } \mathcal{M}, s \models f \\
\mathcal{M}, s &\models \mathbf{EX} f && \text{iff} && \exists\, \pi \text{ from } s \text{ such that } \mathcal{M}, \pi_1 \models f \\
\mathcal{M}, s &\models \mathbf{AX} f && \text{iff} && \forall\, \pi \text{ from } s \text{ such that } \mathcal{M}, \pi_1 \models f
\end{aligned}
$$

Semantics of CTL

$$\mathcal{M}, s \models \mathbf{EF} f \quad \text{iff} \quad \exists\, \pi \text{ from } s, \exists k \geq 0 \text{ such that } \mathcal{M}, \pi_k \models f$$

$$\mathcal{M}, s \models \mathbf{AF}\, f \quad \text{iff} \quad \forall\, \pi \text{ from } s, \exists k \geq 0 \text{ such that } \mathcal{M}, \pi_k \models f$$

$$\mathcal{M}, s \models \mathbf{EG}\, f \quad \text{iff} \quad \exists\, \pi \text{ from } s \text{ such that } \forall k \geq 0,\ \mathcal{M}, \pi_k \models f$$

$$\mathcal{M}, s \models \mathbf{AG}\, f \quad \text{iff} \quad \forall\, \pi \text{ from } s \text{ such that } \forall k \geq 0,\ \mathcal{M}, \pi_k \models f$$

Semantics of CTL

$$\mathcal{M}, s \models \mathbf{E}[f_1 \ \mathbf{U} \ f_2] \quad \text{iff} \quad \exists\, \pi \text{ from } s, \exists k \geq 0 \text{ such that}$$
$$\mathcal{M}, \pi_k \models f_2 \ \text{ and } \ \forall j \,.\, 0 \leq j < k \ \ \mathcal{M}, \pi_j \models f_1$$

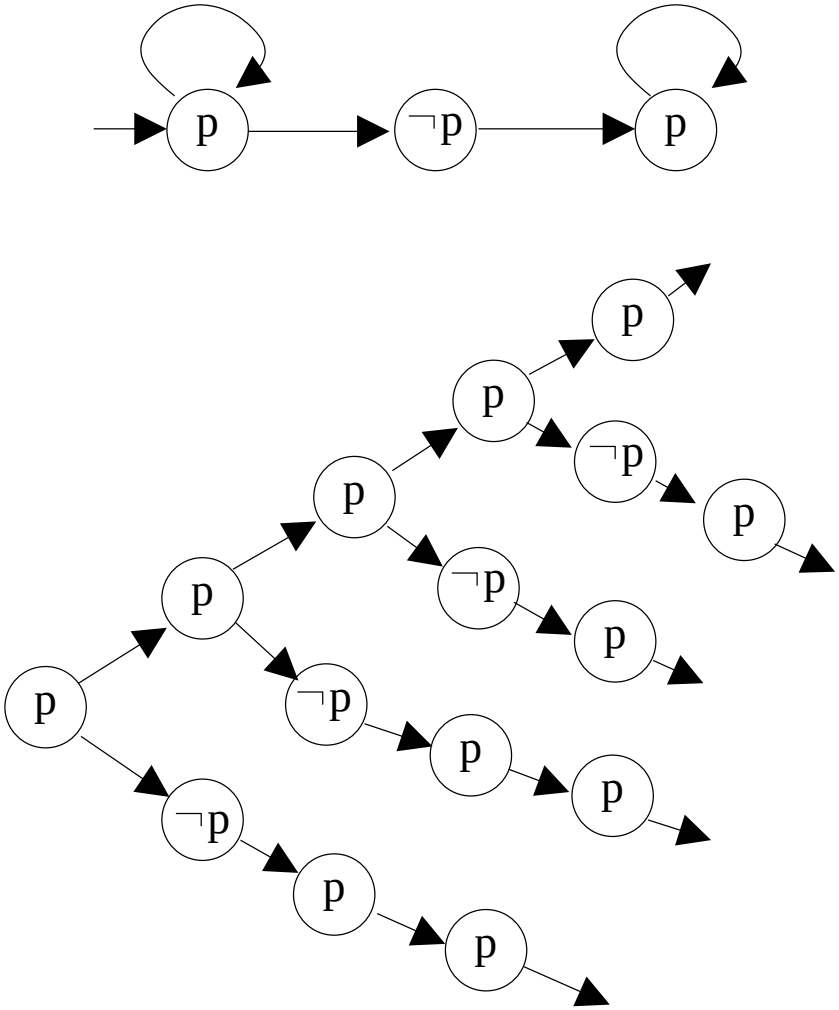
$$\mathcal{M}, s \models \mathbf{A}[f_1 \ \mathbf{U} \ f_2] \quad \text{iff} \quad \forall\, \pi \text{ from } s, \exists k \geq 0 \text{ such that}$$
$$\mathcal{M}, \pi_k \models f_2 \ \text{ and } \ \forall j \,.\, 0 \leq j < k \ \ \mathcal{M}, \pi_j \models f_2$$

LTL vs CTL

In LTL, we could write: **FG** p, which means "on all paths, there is some state from which p will forever hold".

There is no equivalent of this formula in CTL.

For example, in this model, **FG** p holds, but **AF**(**AG** p) does not.



LTL vs CTL

Note that all top-level LTL formulae can be thought of as having the path quantifier **A** in front of them.

CTL is a branching time logic. LTL is a linear time logic. In a branching time logic, there are path quantifiers. In CTL, all the temporal operators must be immediately preceded by a path quantifiers. In LTL, we can have arbitrarily nested boolean connectives and temporal operators.

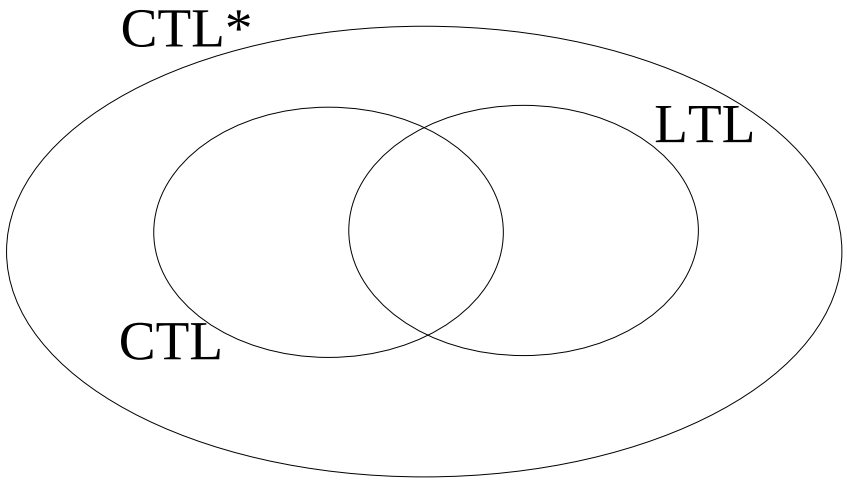
An LTL formula is true/false relative to a path.

A CTL formula is true/false relative to a state.

CTL*

CTL* is another more expressive temporal logic, that is a superset of both LTL and CTL. In CTL*, we can nest temporal operators and Boolean connectives before applying the path quantifiers.

In CTL*, we can write properties such as: **E**(**GF** p) which means "there is a path along which p is infinitely often true". We can't write this in LTL or CTL.



See Huth and Ryan [R17] or Clarke et al. [R11] for further discussion of CTL*.

Specification Patterns

Further examples demonstrating how natural language phrases are matched to temporal logic formulae (or other logics for expressing the ordering of events) can be found in Dwyer, Avrunin, and Corbett [R13].