Temporal Logics

CS560: Reasoning About Programs

Roopsha Samanta

Based on slides by Georg Weissenbacher
Roadmap

Previously

- Propositional logic and SAT Solving
- First-order logic, first-order theories and SMT Solving

Today

- Temporal logic!
Specifying Correctness for Ongoing Systems

A software system controlling traffic lights

Each traffic light in the system can be in one of three states
Consider a crossing with two traffic lights $\mathcal{L}_1$ and $\mathcal{L}_2$

assert ($\neg \mathcal{L}_1 \lor \neg \mathcal{L}_2$)

Safety specification

Expresses *something bad should not happen*
A perfectly *safe* scenario
"not indefinitely (\text{\textcolor{red}{1}} \land \text{\textcolor{red}{2}})"

\textit{Liveness} specification

Expresses \textit{something good will eventually happen}
Temporal logics can express safety and liveness specifications for ongoing systems.
Modeling Ongoing Systems

Finite State Transition system \( \langle S, T, I \rangle \)

- A finite set of states \( S \)
- A set of initial states \( I \subseteq S \)
- A **total** transition relation \( T \subseteq S \times S \)

\[ \forall s \in S. \exists s' \in S. T(s, s') \]
Modeling Ongoing Systems

Kripke structure $\langle S, T, I, L \rangle$
- A finite set of states $S$
- A set of initial states $I \subseteq S$
- A total transition relation $T \subseteq S \times S$
- A labeling function $L : S \rightarrow 2^{AP}$

$AP$: set of atomic propositions
- Properties of states
- Abstracts values of variables

Set of subsets of $AP$
State Formulas

State Formula
Boolean combination of atomic propositions in $AP$

Given Kripke structure $M$, state $s$ and state formula $F$, we write $M, s \models F$ if $F$ holds in $s$

\[
\begin{align*}
M, s &\models p \quad \text{iff} \quad p \in L(s) \\
M, s &\not\models \neg F \quad \text{iff} \quad M, s \not\models F \\
M, s &\models F_1 \lor F_2 \quad \text{iff} \quad M, s \models F_1 \lor M, s \models F_2 \\
M, s &\models F_1 \land F_2 \quad \text{iff} \quad M, s \models F_1 \land M, s \models F_2
\end{align*}
\]
A path $\pi$ is a

- Sequence of states $s_0, s_1, ...$
- Such that $T(s_i, s_{i+1})$ (where $0 \leq i$)

We use $\pi^i$ to denote the suffix of $\pi$ starting at $s_i$
- In particular, $\pi = \pi^0$

Path Formulas

$\pi^0 = \pi$
$\pi' = s_1, s_2, s_3,$
$\pi^3 = s_3, s_4,$ .
Path Formulas

Given Kripke structure $M$, path $\pi$ and path formula $\phi$, we write $M, \pi \models \phi$ if $\phi$ holds for $\pi \in M$.

A state formula $F$ is also a path formula

\[ M, \pi \models F \quad \text{iff} \quad ?? \]
Path Formulas

Given Kripke structure $M$, path $\pi$ and path formula $\phi$, we write $M, \pi \models \phi$ if $\phi$ holds for $\pi \in M$

A state formula $F$ is also a path formula

$M, \pi \models F$  \iff  $F$ holds in the first state $s_0$ of $\pi$
From now on, we use
- $F$ to denote a state formula
- $\phi$ to denote a path formula

We introduce a number of temporal operators
- Allow us to specify what’s supposed to happen along a path
Temporal Operators: Next

\[ M, \pi \models X\phi \quad \text{iff} \quad M, \pi^1 \models \phi \]

For instance: \( M, \pi \models Xp \)

\[ s_i \models p \implies \pi^i \models p \implies \pi \not\models Xp \]

It *doesn’t matter* whether or not \( p \) holds in \( s_0 \) or \( s_2, s_3, \ldots \)
Temporal Operators: Next

\[ M, \pi \models X\phi \iff M, \pi^1 \models \phi \]

For instance: \( M, \pi \models Xp \)

\[ s_0 \xrightarrow{T} s_1 \xrightarrow{T} s_2 \xrightarrow{T} s_3 \xrightarrow{T} s_4 \]

\( X \) can be nested: \( M, \pi \models XXp \)

\[ s_0 \xrightarrow{T} s_1 \xrightarrow{T} s_2 \xrightarrow{T} s_3 \xrightarrow{T} s_4 \]
Temporal Operators: Eventually

\[ M, \pi \models F\phi \iff \exists k \geq 0. M, \pi^k \models \phi \]

- Basic liveness property
- For instance: \( M, \pi \models Fp \)
Temporal Operators: Eventually

\[ M, \pi \models F\phi \iff \exists k \geq 0. M, \pi^k \models \phi \]

- Basic liveness property
- For instance: \( M, \pi \models Fp \)
  - \( p \) holds after a finite number of steps

![Diagram of a transition system with states and transitions]
Temporal Operators: Globally

\[ M, \pi \models G\phi \iff \forall i \geq 0. M, \pi^i \models \phi \]

- Basic safety property
- For instance: \( M, \pi \models Gp \)

\[ G(Xp) \land G(Fp) \]
Temporal Operators: Globally

\[ M, \pi \models G\phi \iff \forall i \geq 0. M, \pi^i \models \phi \]

- Basic safety property
- For instance: \( M, \pi \models Gp \)
  - \( p \) holds after any number of steps

![Diagram showing the transition from states](image)
**Temporal Operators: Until**

\[ M, \pi \models \phi_1 U \phi_2 \iff \exists k \geq 0. M, \pi^k \models \phi_2 \]
\[ \forall j \in \{0..k-1\}. M, \pi^j \models \phi_1 \]

- \( \phi_1 \) holds until \( \phi_2 \) holds
- Also: \( \phi_2 \) has to hold eventually!
- For instance: \( M, \pi \models qU\rho \)

![Diagram of temporal operators until]

- \( q, \bar{p} \) to \( s_0 \)
- \( q, \bar{p} \) to \( s_1 \)
- \( q, \bar{p} \) to \( s_2 \)
- \( \bar{q}, p \) to \( s_3 \)
- \( \bar{q}, \bar{p} \) to \( s_4 \)
Temporal Operators: Until

\[ M, \pi \models \phi_1 U \phi_2 \iff \exists k \geq 0. M, \pi^k \models \phi_2 \]
\[ \forall j \in \{0..k - 1\}. M, \pi^j \models \phi_1 \]

- \(\phi_1\) holds until \(\phi_2\) holds
- Also: \(\phi_2\) has to hold eventually!
- For instance: \(M, \pi \models qU p\)

Note: \(q\) doesn’t have to hold anymore once discharged by \(p\)
Temporal Operators: More Examples

\[ M, \pi \models p \ U \ (Gq) \]
Temporal Operators: More Examples

\[ M, \pi \models p \mathbf{U} (Gq) \]

\[
\begin{array}{c}
p, \overline{q} \\
\downarrow \text{T} \\
s_0 \\
p, \overline{q} \\
\downarrow \text{T} \\
s_1 \\
p, \overline{q} \\
\downarrow \text{T} \\
s_2 \\
\overline{p}, q \\
\downarrow \text{T} \\
s_3 \\
p, q \\
\end{array}
\]
Temporal Operators: More Examples

\[ M, \pi \models p \mathbf{U} (Gq) \]

\[ M, \pi \models F(Gp) \]
Temporal Operators: More Examples

\[ M, \pi \models p \mathbf{U} (Gq) \]

\[
\begin{array}{cccccc}
  p, \bar{q} & \rightarrow & p, \bar{q} & \rightarrow & p, \bar{q} & \rightarrow \bar{p}, q & \rightarrow p, q \\
  s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow s_3 & \rightarrow s_4 \\
  \end{array}
\]

\[ M, \pi \models F(Gp) \]

\[
\begin{array}{cccccc}
  \bar{p} & \rightarrow & \bar{p} & \rightarrow & \bar{p} & \rightarrow p & \rightarrow p \\
  s_0 & \rightarrow & s_1 & \rightarrow & s_2 & \rightarrow s_3 & \rightarrow s_4 \\
  \end{array}
\]
Temporal Operators: More Examples

“not indefinitely ($Q_1 \land Q_2$)"

$$M, \pi \models F (\neg \quad) \quad \text{or} \quad M, \pi \models \neg G (\quad)$$
Temporal Operators: More Examples

“not indefinitely ($\neg (t_1 \land t_2)$)

$M, \pi \models F\ (\neg t)$
or

$M, \pi \models \neg G\ (t)$
Temporal Operators: Redundancies

\[ M, \pi \models \phi_1 U \phi_2 \]

- Last example shows:
  - Some temporal operators can be expressed in terms of others

\[ G \phi \equiv \neg F(\neg \phi) \]
\[ \neg G \phi \equiv F(\neg \phi) \]
\[ F \phi \equiv true \ U \phi \]
Temporal Operators: Redundancies

\[ M, \pi \models \phi_1 U \phi_2 \]

- Last example shows:
  - Some temporal operators can be expressed in terms of others

\[ G \phi \equiv \neg F(\neg \phi) \]

\[ F \phi \equiv true U \phi \]

- \( \neg, X, U \) are sufficient to express \( G \) and \( F \)
- (c.f. “basis” (\( \neg, V \)) in propositional logic)
Temporal Operators: Path Quantifiers

\[ M, \pi \models \phi_1 U \phi_2 \]

- So far, we can only talk about individual paths
- To amend this, we introduce *path quantifiers*

- \[ M, s \models E \phi \iff \exists \pi \text{ starting at } s \text{ such that } M, \pi \models \phi \]
- \[ M, s \models A \phi \iff \forall \pi \text{ starting at } s \text{ it holds that } M, \pi \models \phi \]
Remember:
- Unwinding transition function results in infinite tree
Accordingly, our logic is appropriately called **Computation Tree Logic**.

More specifically: CTL*
Computation Tree Logic CTL*: Examples

\[ M, s_0 \models AF (\text{green light}) \]
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF (\text{green}) \checkmark$
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF (\text{green}) \checkmark$
- $M, s_0 \models AX(EG(\text{green}))$
Computation Tree Logic CTL*: Examples

- $M, s_0 \vDash AF \left(\text{\textcolor{red}{\text{red}} \text{\textcolor{green}{green}}}\right) \checkmark$

- $M, s_0 \vDash AX(EG(\text{\textcolor{red}{\text{red}} \text{\textcolor{green}{green}}))) \checkmark$
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF (\text{red}) \checkmark$
- $M, s_0 \models AX(EG( a )) \checkmark$
- $M, s_0 \models EGX (\text{green})$

Diagram:

- $s_0$ 
- $s_1$
- $s_2$
- $s_3$
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF (\text{green light}) \checkmark$
- $M, s_0 \models EGX (\text{green light}) \checkmark$
- $M, s_0 \models AX(EG(\text{green light})) \checkmark$
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF (✓) \checkmark$
- $M, s_0 \models EG (✓) \checkmark$
- $M, s_0 \models AX(EG(✓)) \checkmark$
- $M, s_0 \models EGX (✓) \checkmark$
- $M, s_0 \models AGX (✓) \checkmark$
Computation Tree Logic CTL*: Examples

- $M, s_0 \models AF(\text{G}) \checkmark$
- $M, s_0 \not\models EG(\text{G})$
- $M, s_0 \models EG(\text{G}) \checkmark$
- $M, s_0 \models AX(EG(\text{G})) \checkmark$
- $M, s_0 \models AGX(\text{G}) \times$
Branching Time and Linear Time Logic

- Commonly used subsets of CTL*:
  - *branching-time logic*
    - quantifies over paths possible from a given state
  - *linear-time logic*
    - for events along a single computation path only
Clarke & Emerson, *Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic*, 1981

Algorithmic framework for exhaustive exploration of finite-state transition systems to check temporal properties
Branching Time Logic: Computation Tree Logic

- Computation Tree Logic CTL
-CTL $\subset$ CTL*
-Restriction:
  $X, F, G,$ and $U$, must be immediately preceded by $A$ or $E$
Branching Time Logic: Computation Tree Logic

- Computation Tree Logic CTL
- CTL ⊂ CTL*
- Restriction: X, F, G, and U, must be immediately preceded by A or E
- Examples:
  - EF(start ∧ ¬ready) There’s a path on which we start at some point despite not being ready
  - AG(req A ⇒ AF ack) Each request eventually acknowledged
  - AG EX progress No deadlocks
Branching Time Logic: Computation Tree Logic

- What are the restrictions?
  - Some properties can’t be expressed!
  - $A(\mathbf{FG} \ p)$ can’t be expressed in CTL!

- And the advantages?
  - More efficient to check than cull CTL*
    - Checking CTL-formula $\phi$ for $\langle S, T, I, L \rangle$ is $O(|\phi| \cdot (|S| + |T|))$
    - Checking CTL* lies in PSPACE
  - Can be checked using fixed points!
Linear Temporal Logic

- Linear Temporal Logic: Another subset of CTL* for events along a single computation path only

- Formulas have the form $A\phi$
  - State formulas can only be atomic propositions
  - In particular, $\phi$ doesn't contain $A$, $E$, conjunctions, or disjunctions of path formulas
Linear Temporal Logic

- Linear Temporal Logic: Another subset of CTL* for events along a single computation path only

- Formulas have the form $A\phi$
  - State formulas can only be atomic propositions
  - $\phi$ doesn’t contain $A$ or $E$

- Intuitively, $\phi$ is always interpreted over all paths
Examples for LTL formulas:

- $A(FG \ p)$ “all paths eventually stabilize with property $p$”
  - This can’t be expressed in CTL
- $A(GF \ p)$ “$p$ is visited infinitely often”
- $AG(\text{try} \Rightarrow F \ \text{succeed})$ “every attempt eventually succeeds”

We can’t express

- $AG(\text{EF} \ p)$
  - This can be expressed in CTL
Summary

Today
- Temporal logic as a specification language
- Branching time logic CTL
- Linear time logic LTL

Next
- Odds and Ends