EECS 219C: Computer-Aided Verification
Models and Properties: Temporal Logic

Sanjit A. Seshia
EECS, UC Berkeley

Announcements

- Project topics due by e-mail to me next Monday
  - Include a short 1 paragraph description of the project
Finite-State Model Checking

\[ G(p \rightarrow X q) \]

Temporal logic

FSM

\[ \text{Model Checker} \]

Yes, property satisfied

System description
(RTL, source code, gates, etc.)

Recap

- We’re verifying closed systems
- Modeled as Kripke structures \((S, S_0, R, L)\)
  - Represents the product of the “system” with its “environment”
System Behavior

- A sequence of states, starting with an initial state
  \[- s_0 \ s_1 \ s_2 \ \ldots \ \text{such that } R(s_i, s_{i+1}) \text{ is true} \]
- Also called “run”, or “(computation) path”
- Trace: sequence of observable parts of states
  \[- \text{Sequence of state labels} \]

Safety vs. Liveness

- Safety property
  \[- \text{Error trace is finite} \]
- Liveness property
  \[- \text{Error trace is infinite} \]
Temporal Logic

- A logic for specifying properties over time
  - E.g., Behavior of a finite-state system

- We will study *propositional* temporal logic
  - Other temporal logics exist:
    - E.g., real-time temporal logic

Atomic State Property (Label)

A Boolean formula over state variables

We will denote each unique Boolean formula by
- a distinct color
- a name such as p, q, ...

req
req & !ack
Globally (Always) p: $G \ p$

$G \ p$ is true for a computation path if $p$ holds at all states (points of time) along the path.

Suppose $G \ p$ holds along the path below:

$$p = \bullet$$

Eventually p: $F \ p$

- $F \ p$ is true for a path if $p$ holds at some state along that path.

Does $F \ p$ hold for the following examples?

$$p = \bullet$$
Next \( p: X \ p \)

- \( X \ p \) is true along a path starting in state \( s_i \) (suffix of the main path) if \( p \) holds in the next state \( s_{i+1} \)

\[ p = \bullet \]

Suppose \( X \ p \) holds along the path starting at state \( s_2 \)

Nesting of Formulas

- \( p \) need not be just a Boolean formula.
- It can be a temporal logic formula itself!

\[ p = \bullet \]

"\( X \ p \) holds for all suffixes of a path"

How do we draw this?

How can we write this in temporal logic?

Write down formal definitions of \( Gp, Fp, Xp \)
Notation

• Sometimes you’ll see alternative notation in the literature:
  \( G \square \)
  \( F \lozenge \)
  \( X \circ \)

Examples: What do they mean?

• \( G F p \)
• \( F G p \)
• \( G( p \rightarrow F q ) \)
• \( F( p \rightarrow (X X q) ) \)
**p Until q: p U q**

- **p U q** is true along a path starting at s if
  - q is true in some state reachable from s
  - p is true in all states from s until q holds

  \[ p = \text{red} \quad q = \text{cyan} \]

Suppose p U q holds for the path below

\[ 0 \quad 1 \quad 2 \quad \ldots \]

---

**Temporal Operators & Relationships**

- G, F, X, U: All express properties along paths

- Can you express G p purely in terms of F, p, and Boolean operators?

- How about G and F in terms of U and Boolean operators?

- What about X in terms of G, F, U, and Boolean operators?
Examples in Temporal Logic

1. “No more than one processor (in a 2-processor system) should have a cache line in write mode”
   - $\text{wr}_1 / \text{wr}_2$ are respectively true if processor 1 / 2 has the line in write mode

2. “The grant signal must be asserted at some time after the request signal is asserted”
   - Signals: grant, req

3. “A request signal must receive an acknowledge and the request should stay asserted until the acknowledge signal is received”
   - Signals: req, ack

Examples in Temporal Logic

4. “From any state, it is possible to return to the reset state along some execution”
   - Signal indicating reset state: reset

5. “The grant signal must always be asserted 3 cycles after the request signal is asserted”
   - Signals: grant, req
Linear Temporal Logic

• What we’ve seen so far are properties expressed over a single computation path or run
  – LTL

Temporal Logic Flavors

• Linear Temporal Logic

• Computation Tree Logic
  – Properties expressed over a tree of all possible executions
  – Where does this “tree” come from?
Temporal Logic Flavors

- Linear Temporal Logic (LTL)

- Computation Tree Logic (CTL, CTL*)
  - Properties expressed over a tree of all possible executions
  - CTL* gives more expressiveness than LTL
  - CTL is a subset of CTL* that is easier to verify than arbitrary CTL*
**Computation Tree Logic (CTL*)**

- Introduce two new operators $\mathbf{A}$ and $\mathbf{E}$ called “Path quantifiers”
  - Corresponding properties hold in states (not paths)
  - $\mathbf{A} p$: Property $p$ holds along all computation paths starting from the state where $\mathbf{A} p$ holds
  - $\mathbf{E} p$: Property $p$ holds along at least one path starting from the state where $\mathbf{E} p$ holds
- Example:
  “The grant signal must always be asserted some time after the request signal is asserted”
  $\mathbf{A} G (\text{req} \rightarrow \mathbf{A} F \text{grant})$

- Notation: $\mathbf{A}$ sometimes written as $\forall$, $\mathbf{E}$ as $\exists$

---

**CTL**

- Every $\mathbf{F}$, $\mathbf{G}$, $\mathbf{X}$, $\mathbf{U}$ must be immediately preceded by either an $\mathbf{A}$ or a $\mathbf{E}$
  - E.g., Can’t write $\mathbf{A}$ ($\mathbf{F} \mathbf{G} p$)

- LTL is just like having an “$\mathbf{A}$” on the outside
Why CTL?

- Verifying LTL properties turns out to be computationally harder than CTL
- But LTL is more intuitive to write
- Complexity of model checking
  - Exponential in the size of the LTL expression
  - Linear for CTL
- For both, model checking is linear in the size of the state graph

CTL as a way to approximate LTL

- \( \text{AG EF } p \) is weaker than \( \text{GF } p \)  
  \[ \text{Good for finding bugs...} \]
- \( \text{AF AG } p \) is stronger than \( \text{FG } p \)  
  \[ \text{Good for verifying correctness...} \]

Why? And what good is this approximation?
More CTL

• “From any state, it is possible to get to the reset state along some path”

\[ A \forall G ( E F \text{ reset} ) \]

CTL vs. LTL Summary

• Have different expressive powers

• Overall: LTL is easier for people to understand, hence more commonly used in property specification languages
From Temporal Logic to Monitors

• A monitor for a temporal logic formula
  – is a finite state machine (automaton)
  – Accepts exactly those behaviors that satisfy the temporal logic formula
    • “Accepts” means that the accepting state is visited infinitely often
• Properties are often specified as automata

Monitor for $\text{G } p$, $p$ a Boolean formula
Monitor for $F\ p$, $p$ a Boolean formula?

Monitor for $GF\ p$, $p$ a Boolean formula?
Summary

• What we did today: Properties in Temporal Logic, LTL, CTL, CTL*
• Next: Start model checking algorithms