Introduction to Formal Methods

Chapter 2. Temporal Logic

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr
2. Temporal Logic

• Motivation:
  – The elevator example includes two properties
    • “Any elevator request must ultimately be satisfied”
    • “The elevator never traverses a floor for which a request is pending without satisfying this request”
  – → Dynamic behavior of the system
  – In a first order logic,

  - \( \forall t, \forall n \ ( \text{app}(n, t) \Rightarrow \exists t' > t : \text{serv}(n, t') ) \)
  - \( \forall t, \forall t' > t, \forall n, \left[ ( \text{app}(n, t) \land H(t') \neq n \land \exists t_{\text{trav}} : \\
  t \leq t_{\text{trav}} \leq t' \leq H(t_{\text{trav}}) = n ) \Rightarrow ( \exists t_{\text{serv}} : t \leq t_{\text{serv}} \leq t' \land \text{serv}(n, t_{\text{serv}}) ) \right] \)
  – But, the above notation (mathematics) is quite cumbersome.

• Temporal Logic is a different formalism, better suited for our situation.
2. Temporal Logic

• Temporal Logic
  – A form of logic specifically tailored for
    • statements and reasoning
    • Involving the notion of order in time
  – Compared with the mathematical formulas
    • clearer and simpler
    • immediately ready for use (linguistic similarity of operators)
    • formal semantics (specification language tools)

• Organization of Chapter 2
  – The Language of Temporal Logic
  – The Formal Syntax of Temporal Logic
  – The Semantics of Temporal Logic
  – PLTL and CTL: Two Temporal Logics
  – The Expressivity of CTL*
2.1 The Language of Temporal Logic

- **CTL**
  - serves to formally state the properties concerned with the execution of a system
  - Variants (CTL, PLTL, LTL)
  - 6 characteristics

1. Atomic Propositions
   - *warm, ok, error*

2. Proposition Formula
   - using boolean combinators
   - true, false, \( \neg, \lor, \land, \Rightarrow \) (if then), \( \Leftrightarrow \) (if and only if)

   - *error \( \Rightarrow \neg \text{warm} \)  
     (if *error* then not *warm*)

\[ \sigma_1 : (q_0: \text{warm}, \text{ok}) \rightarrow (q_1: \text{ok}) \rightarrow (q_0: \text{warm}, \text{ok}) \rightarrow (q_1: \text{ok}) \rightarrow \ldots \]

\[ \sigma_2 : (q_0: \text{warm}, \text{ok}) \rightarrow (q_1: \text{ok}) \rightarrow (q_2: \text{error}) \rightarrow (q_0: \text{warm}, \text{ok}) \rightarrow (q_1: \text{ok}) \rightarrow \ldots \]

\[ \sigma_3 : (q_0: \text{warm}, \text{ok}) \rightarrow (q_1: \text{ok}) \rightarrow (q_2: \text{error}) \rightarrow (q_2: \text{error}) \rightarrow \ldots \]
3. Temporal combinators

- about the sequencing of states along an execution

- $X$: next state
- $F$: a future state
- $G$: all the future states

- $XP$: the next state satisfies $P$
- $FP$: a future state satisfies $P$ without specifying which state
  $\rightarrow P$ will hold some day (at least once)
- $GP$: all future states will satisfy $P$
  $\rightarrow P$ will always be

- $alert \Rightarrow F\ halt$ : if we are currently in a state of $alert$, then we will later be in
  a $halt$ state.
- $G (alert \Rightarrow F\ halt)$ : at any time, a state of $alert$ will necessarily be followed
  by a $halt$ state later.

- $G (warm \Rightarrow F\ \neg warm)$ : true
- $G (warm \Rightarrow X\ \neg warm)$ : true

- $G$ is the dual of $F$
  - $G \phi \equiv \neg F\neg \phi$
4. Arbitrary nesting of temporal combinators
   - give temporal logic its power and strength

   - \( GF \phi \): always there will some day be a state such that \( \phi \),
     \( \phi \) is satisfied infinitely often along the execution considered

   - \( FG \phi \): all the time from a certain time onward, at each time instant,
     possibly excluding a finite number of instants

   - \( GF \text{ warm} \lor FG \text{ error} \)

5. U combinator
   - for until
   - \( \phi_1 U \phi_2 : \phi_1 \) is verified until \( \phi_2 \) is verified
     \( \phi_2 \) will be verified some day, and \( \phi_1 \) will hold in the meantime

   - \( G (\text{alert} \Rightarrow (\text{alarm} U \text{halt})) \): starting from a state of \text{alert}, the \text{alarm} remains activated
     until the \text{halt} state is eventually and inexorably reached.

   - \( F \phi \equiv \text{true} \lor \phi \)
   - \( \phi_1 W \phi_2 \equiv (\phi_1 U \phi_2) \lor G \phi_1 \): weak until
6. Path quantifier

- A $\phi$: all the executions out of the current state satisfy property $\phi$
- E $\phi$: from the current state, there exists an execution satisfying $\phi$

- EF $P$: it is possible (by following a suitable execution) to have $P$ some day
- EG $P$: there exists an execution along which $P$ always holds

- AF $P$: we will necessarily have $P$ some day (regardless of the chosen execution)
- AG $P$: always true
2.2 Formal Syntax of Temporal Logic

- Abstract grammar
  - Needs parentheses, operator priority, specific set of atomic propositions, etc.
  - Most model checkers use a fragment of CTL* - CTL or LTL.

\[
\begin{align*}
\phi, \psi &::= P_1 | P_2 | \ldots & \text{(atomic proposition)} \\
& \quad | \neg \phi | \phi \land \psi | \phi \rightarrow \psi | \ldots & \text{(boolean combinators)} \\
& \quad | X\phi | F\phi | G\phi | \phi \text{ U } \psi | \ldots & \text{(temporal combinators)} \\
& \quad | E\phi | A\phi & \text{(path quantifiers)}
\end{align*}
\]
2.3 The Semantics of Temporal Logic

- **Kripke structure**
  - Name of the models of temporal logic
  - Propositions labeling the states are important in CTL*
  - Transition labels ($E$) are neglected. $A = < Q, T, q_0, l >$, $T \subseteq Q \times Q$

- **Satisfaction**
  - $A, \sigma, i \models \phi$
    - “at time $i$ of the execution $\sigma$, $\phi$ is true.”
    - where $\sigma$ is an execution of $A$, which not required to start at the initial state
    - $A$ is often omitted.
  - $\sigma, i \not\models \phi$ : $\phi$ is satisfied at time $i$ of $\sigma$
  - $\sigma, i \not\models \phi$ : $\phi$ is not satisfied at time $i$ of $\sigma$

- $A \models \phi$ iff $\sigma, 0 \models \Phi$ for every execution of $\sigma$ of $A$
  - “the automaton $A$ satisfies $\phi$”
  - $A \not\models \phi \not\models \neg \phi$
  - $\sigma, i \not\models \phi = \sigma, i \models \neg \phi$
Semantics of CTL*

\[
\begin{align*}
\sigma, i &\models P \quad \text{iff } P \in l(\sigma(i)), \\
\sigma, i &\models \neg \phi \quad \text{iff it is not true that } \sigma, i \models \phi, \\
\sigma, i &\models \phi \land \psi \quad \text{iff } \sigma, i \models \phi \text{ and } \sigma, i \models \psi, \\
\sigma, i &\models X\phi \quad \text{iff } i < |\sigma| \text{ and } \sigma, i + 1 \models \phi, \\
\sigma, i &\models F\phi \quad \text{iff there exists } j \text{ such that } i \leq j \leq |\sigma| \text{ and } \sigma, j \models \phi, \\
\sigma, i &\models G\phi \quad \text{iff for all } j \text{ such that } i \leq j \leq |\sigma|, \text{ we have } \sigma, j \models \phi, \\
\sigma, i &\models \phi U \psi \quad \text{iff there exists } j, i \leq j \leq |\sigma| \text{ such that } \sigma, j \models \psi, \text{ and for all } k \text{ such that } i \leq k < j, \text{ we have } \sigma, k \models \phi, \\
\sigma, i &\models E\phi \quad \text{iff there exists a } \sigma' \text{ such that } \sigma(0) \ldots \sigma(i) = \sigma'(0) \ldots \sigma'(i) \text{ and } \sigma', i \models \phi, \\
\sigma, i &\models A\phi \quad \text{iff for all } \sigma' \text{ such that } \sigma(0) \ldots \sigma(i) = \sigma'(0) \ldots \sigma'(i), \text{ we have } \sigma', i \models \phi.
\end{align*}
\]

Semantics of CTL*

- Time is discrete.
- Nothing exists between \( i \) and \( i + 1 \).
- The instants are the points along the executions
2.4 PLTL and CTL: Two Temporal Logics

- Two most commonly used temporal logics in model checking tools
  - PLTL (Propositional Linear Temporal Logic)
  - CTL (Computational Tree Logic)
  - fragments of CTL*

- PLTL
  - No path quantifiers (A and E)
  - Linear time logic → Path formula
  - For example, PLTL cannot distinguish $A_1$ from $A_2$

Execution 1: $\{P, Q\} \cdot \{P\} \cdot \{-\}$
Execution 2: $\{P, Q\} \cdot \{P\} \cdot \{Q\}$
• **CTL**
  
  – Temporal combinators \((X, F, U)\) should be under the immediate scope of path quantifier \((A, E)\)
  – \(EX, AX, EU, AU, EF, EG, AG, AF, \ldots\)
  – State formulas
    – Truth only depends on the current state and the automaton regions made reachable by it
    – Not depend on a current execution.
    – \(q \models \phi : \phi\) is satisfied in state \(q\)

  – CTL can distinguish automata \(A_1\) and \(A_2\)

\[
A_1 : \\
\begin{array}{c}
P \ \rightarrow \ \ P \\
P \ \rightarrow \ \ Q \\
Q \ \rightarrow \ \end{array} \\
\begin{array}{c}
P \ \rightarrow \ \ P \\
P \ \rightarrow \ \ Q \\
Q \ \rightarrow \ \end{array}
\]

\[
A_2 : \\
\begin{array}{c}
P \ \rightarrow \ \ P \\
P \ \rightarrow \ \ P \\
Q \ \rightarrow \ \end{array}
\]

\[
A_{1,q_0} \models AX(\neg Q) \\
A_{2,q_0'} \not\models AX(\neg Q)
\]

\[\text{Potential reachability: } AG EF P\]

– Do not allow us to express very rich properties along the paths.
• **Which to choose CTL or PLTL?**
  - To state some properties
    → PLTL
  - To perform exhaustive verification of a system
    → CTL
  - For both purposes
    → CTL*
      • Less popular
      • More complicated than PLTL
  - CTL + Fairness properties → FCTL
  - If we use model checking tools, then we have no choice
    - SMV : CTL (CTL*)
    - SPIN : PLTL
    - VIS : CTL / PLTL
2.5 The Expressivity of CTL*

- No logic can express anything not taken into account by the modeling decision made

- CTL* is rather expressive enough, when
  - Properties concern the execution tree of our automata
  - CTL* combinators are sufficiently expressive
  - CTL* is almost always sufficient