SLR 204: Basics of verification of distributed systems

Vadim Malvone
vadim.malvone@telecom-paris.fr
Formal Specification of Properties in Branching-Time Temporal Logics
Branching-time Temporal Logics

- An LTL formula is satisfied over a system model $M$ if it is satisfied on every path starting at the initial states.

- Paths in $M$ represent all possible system computations.

- To check a formula on some paths of $M$, we need a logic that allows us to talk about branches of $M$.

- We study CTL and CTL*.
Computation Tree Logic (CTL)

- CTL uses the same temporal operators as LTL.

- Additionally, we use two path quantifiers:
  - E means ‘for some computations’
  - A means ‘for all computations’

  EX, EF, EG, EU
  AX, AF, AG, AU

- In CTL, formulas are investigated on states rather than paths.
Tree model unwinding

An infinite computation tree
EX red
“For at least a path, red will become true in the next step”
Operator AX

AX red

“For every path, red will become true in the next step”
EF red

“For at least a path, red will possibly become true”
Operator AF

AF red

“For every path, red will eventually become true”
EG red

“For at least a path, red always remains true”
AG red
“for every path, red is always true”
E yellow U red
“For at least a path, yellow is verified until red”
A yellow U red

“For every path, yellow is verified until red”
Formal Syntax of CTL (I)

\[ \phi := p \mid \neg \phi \mid \phi \lor \phi \mid \text{EX} \, \phi \mid \text{E} \, \phi \lor \phi \mid \text{A} \, \phi \lor \phi \]

where \( p \in \text{AP} \)

- CTL formulas are evaluated on infinite trees.
- A CTL formula describes a set of trees for which the formula holds.
Formal Syntax of CTL (II)

- Derived operators
  - $AX \varphi = \neg EX \neg \varphi$
  - $EF \varphi = E \text{true} U \varphi$
  - $AG \varphi = \neg EF \neg \varphi$
  - $AF \varphi = A \text{true} U \varphi$
  - $EG \varphi = \neg AF \neg \varphi$
Formal Semantics of CTL (I)

- Given a Kripke structure $M = (AP, S, S_0, R, \text{Lab})$.
- Given a state $s \in S$ of $M$.
- $s \models \phi$ means that the CTL formula $\phi$ holds on $s$.
- Let $\pi_i$ be the $i$-th element of $\pi$, with $i \geq 0$. 
The relation $\models$ is defined inductively as follows:

- $s \models p$ iff $p \in \text{Lab}(s)$
- $s \models \neg \varphi$ iff not $s \models \varphi$
- $s \models \varphi_1 \lor \varphi_2$ iff $s \models \varphi_1$ or $s \models \varphi_2$
- $s \models \text{EX} \varphi$ iff there exists a path $\pi$ starting at $s$ such that $\pi_1 \models \varphi$
- $s \models \text{E} \varphi_1 \lor \varphi_2$ iff there exists a path $\pi$ starting at $s$
  such that $\exists \ i \geq 0$, such that $\pi_i \models \varphi_2$ and
  $\forall \ 0 \leq j < i$, we have that $\pi_j \models \varphi_1$
- $s \models \text{A} \varphi_1 \lor \varphi_2$ iff for every path $\pi$ starting at $s$,
  $\exists \ i \geq 0$, such that $\pi_i \models \varphi_2$ and
  $\forall \ 0 \leq j < i$, we have that $\pi_j \models \varphi_1$
Formal Semantics of CTL (III)

- **Derived operators:**
  - $s \models AX \varphi$ iff for every path $\pi$ starting at $s$, $\pi_1 \models \varphi$
  - $s \models EF \varphi$ iff there exists a path $\pi$ starting at $s$ such that $\exists i \geq 0$, such that $\pi_i \models \varphi$
  - $s \models AF \varphi$ iff for every path $\pi$ starting at $s$, $\exists i \geq 0$, such that $\pi_i \models \varphi$
  - $s \models EG \varphi$ iff there exists a path $\pi$ starting at $s$ such that $\forall i \geq 0$, $\pi_i \models \varphi$
  - $s \models AG \varphi$ iff for every path $\pi$ starting at $s$, $\forall i \geq 0$, $\pi_i \models \varphi$
Example of Specification: Microwave

The microwave doesn’t **heat up** until the door is closed:

\[ A \rightarrow \text{Heat} U \text{Close} \]
CTL Model Checking

Given a Kripke structure $M = (AP, S, S_0, R, Lab)$ modelling the system and a CTL formula $\varphi$ over $AP$ representing the specification.

The CTL model checking problem is to check whether

$$s \models \varphi$$

for each $s \in S_0$
Branching vs Linear: LTL but not CTL

- In LTL, we can write: FG p, which means “on all paths, there is a state from which the atomic proposition p will hold forever”.
- There is no equivalent CTL formula.
- In particular, FG p \neq AF AG p.
- The model M satisfies FG p, but not AF AG p.
- Indeed, it has a path in which \neg p is always a possible successor.
Branching vs Linear: CTL but not LTL

- AG EF Start
- E.g.: “for every computation, it is always possible to return to the initial state”.
- Consider $\phi = \text{EX} (\text{Start} \land \text{AX} \text{Stop})$.
- Consider $\psi = \text{X} (\text{Start} \land \text{X} \text{Stop})$.
- $M$ and $M^*$ have equivalent paths:
  - $\pi^a = s_0 s_1 (s_3)^\omega$ and $\pi^b = t_0 t_1 (t_2)^\omega$
  - $\pi^c = s_0 s_2 (s_4)^\omega$ and $\pi^d = t_0 t_1 (t_3)^\omega$
- The CTL formula $\phi$ holds on $M$ but not on $M^*$.
- The LTL formula $\psi$ does not hold on both models.
Branching vs Linear

- LTL makes it possible to state the properties more simply.
- CTL is preferable to avoid combinatorial explosion problems during exhaustive verification.
- LTL and CTL can be seen as fragments of the same logic: CTL*.
- CTL* allows complex nesting such as: AXX, EFG, AGX, AFG, EXFG, ...
A full Branching-time Temporal Logic: CTL* (I)

- Syntax of CTL* is given by means of path and state formulas.

- State formulas:
  - Every atomic proposition $p$ is a state formula.
  - If $\varphi_1$ and $\varphi_2$ are state formulas then so are $\neg \varphi_1$, $\varphi_1 \land \varphi_2$, and $\varphi_1 \lor \varphi_2$.
  - If $\psi$ is a path formula, then $A \psi$ and $E \psi$ are state formulas.

- State formulas can be true or false at each state of the model.
A full Branching-time Temporal Logic: CTL* (II)

- Path formulas:
  - Every state formula is a path formula.
  - If $\psi_1$ and $\psi_2$ are path formulas then so are $\neg \psi_1$, $\psi_1 \land \psi_2$, and $\psi_1 \lor \psi_2$.
  - If $\psi_1$ and $\psi_2$ are path formulas then so are $X\psi_1$, $F\psi_1$, $G\psi_1$, and $\psi_1 U \psi_2$.

- Path formulas are interpreted over sequences of states of the model.

- As for LTL and CTL, abbreviations, equivalences, and step unfolding hold in CTL*.
Formal Syntax of CTL*

- CTL* state-formulas are formed according to the grammar:

\[ \phi := p \mid \neg \phi \mid \phi \lor \phi \mid E \psi \]

where \( p \in AP \) and \( \psi \) is a path-formula

- CTL* path-formulas are formed according to the grammar:

\[ \psi := \phi \mid \neg \psi \mid \psi \lor \psi \mid X \psi \mid \psi U \psi \]

where \( \phi \) is a state-formula
Formal Semantics of CTL*

Let M a Kripke structure and s a state of M:

- \( s \models p \iff p \in \text{Lab}(s) \)
- \( s \models \neg \phi \iff \neg s \models \phi \)
- \( s \models \phi_1 \lor \phi_2 \iff s \models \phi_1 \text{ or } s \models \phi_2 \)
- \( s \models E \psi \iff \text{there exists a path } \pi \text{ starting at } s \text{ in } M \text{ such that } \pi \models \psi \)
- \( \pi \models \phi \iff \pi_0 \models \phi \)
- \( \pi \models \neg \psi \iff \neg \pi \models \psi \)
- \( \pi \models \psi_1 \lor \psi_2 \iff \pi \models \psi_1 \text{ or } \pi \models \psi_2 \)
- \( \pi \models X \psi \iff \pi_{\geq 1} \models \psi \)
- \( \pi \models \psi_1 \lor \psi_2 \iff \exists i \geq \text{ such that } \pi_{\geq 2i} \models \psi_2 \text{ and } \forall 0 \leq j < i, \text{ we have that } \pi_{\geq 2j} \models \psi_1 \)
Syntax of LTL in terms of state and path formulas

- LTL state-formulas are formed according to the grammar:

\[ \varphi := A \psi \]

where \( p \in AP \) and \( \psi \) is a path-formula

- LTL path-formulas are formed according to the grammar:

\[ \psi := p \mid \neg \psi \mid \psi \lor \psi \mid X \psi \mid \psi U \psi \]
Syntax of CTL in terms of state and path formulas

- CTL state-formulas are formed according to the grammar:

\[ \varphi := p | \neg \varphi | \varphi \vee \varphi | E \psi | A \psi \]

where \( p \in AP \) and \( \psi \) is a path-formula

- CTL path-formulas are formed according to the grammar:

\[ \psi := X \varphi | \varphi U \varphi \]

where \( \varphi \) is a state-formula