Lecture 7: Computation Tree Logics

Expressive Power of Logics

CTL and LTL

Path Formulas and State Formulas

The Logic CTL*

Computation Tree Logics

Model of Computation
(Unwind State Graph to obtain Infinite Tree)

State Transition Graph or

Infinite Computation Tree

Knipeke Model

State Transition Graph or

Model of Computation
Model of Computation (Cont.)

Unles otherwise stated, all of our results apply only to finite Kripke structures.

We write $\pi'$ to denote the suffix of $\pi$ starting at $s'$.

A path in $M$ is an infinite sequence of states, $\pi = s_0, s_1, s_2, \ldots$ such that for $i \geq 0$, $(s_i, s_{i+1}) \in R$.

We assume that $R$ is total (i.e., for all states $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$).

Formally, a Kripke structure is a triple $(M, R, \mathcal{A})$ where

\[ \mathcal{A} = \{ \phi \mid \phi \text{ is an atomic proposition} \} \]

\[ \vdash \mathcal{T} \subseteq S \times S \subseteq R \]

\[ \mathcal{A} = \{ \phi \mid \phi \text{ is an atomic proposition} \} \]

\[ \vdash \mathcal{T} = \{ (s, s') \mid s, s' \in S \} \]

where

(\text{Model of Computation (Cont.)})
Computation Tree Logics

Temporal logics may differ according to how they handle branching in the underlying computation. In a linear temporal logic, operators are provided for describing events along a single computation path. In a branching-time logic, temporal operators quantify over the paths that are possible from a given state.
The computation tree logic CTL combines both branching-time and linear-time operators. In this logic a path quantifier can prefix an assertion composed of arbitrary combinations of the usual linear-time operators.

1. Path quantifiers:
   - \(\mathbf{E}\) — "there exists a path"
   - \(\mathbf{A}\) — "for every path"

2. Linear-time operators:
   - \(\mathbf{F}\) — holds sometime in the future
   - \(\mathbf{G}\) — holds globally in the future
   - \(\mathbf{X}\) — holds next time
   - \(\mathbf{b} \mathbf{U} \mathbf{d}\) — holds until holds

The computation tree logic CTL combines both branching-time and linear-time operators.
Two additional rules are needed to specify the syntax of path formulas:

If $p$ is a path formula, then $x(p)$ is a state formula.

If $p$ and $q$ are state formulas, then $\neg p \land q$ and $p \lor q$ are state formulas.

If $\exists d \in A_P$, then $d$ is a state formula.

The syntax of state formulas is given by the following rules:

Path Formulas and State Formulas
State Formulas (Cont.)

If $f$ is a state formula, the notation $M, s \models f$ means that $f$ holds at state $s$ in the Kripke structure $M$.

Assume $f_1$ and $f_2$ are state formulas and $g$ is a path formula. The relation $M, s \models f$ is defined inductively as follows:

1. $s \models p \iff p \in L(s)$.
2. $s \models \neg f_1 \iff s \not\models f_1$.
3. $s \models f_1 \lor f_2 \iff s \models f_1$ or $s \models f_2$.
4. $s \models \mathbf{E}(g) \iff$ there exists a path $\pi$ starting with $s$ such that $\pi \models g$. 
Path Formulas (Cont.)

If \( \varphi \) is a path formula, \( \varphi \) holds along path \( \psi \) in Kripke structure \( \mathcal{M} \).

Inductively as follows:

Assume \( \psi_1 \) and \( \psi_2 \) are path formulas and \( \psi \) is a state formula. The relation \( \mathcal{M} \), \( \not\models \psi \) is defined as:

\[
\not\models f = | \not\models s \text{ is the first state of } \not\models \text{ and } s.
\]

\[
\not\models f = | \not\models \exists \gamma (\psi_1 \cup \psi_2) \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]

\[
\not\models f = | \not\models \mu \iff \not\models \psi_1 \psi_2.
\]
In addition, we will use the following abbreviations in writing temporal operators:

\[ f \leftarrow P \leftarrow \equiv f \cap \bullet \]
\[ (f \sqcap \text{event}) \equiv f \bullet \]
\[ (f \leftarrow) \exists \equiv (f) \forall \bullet \]

The customary abbreviations will be used for the connectives of propositional logic.

---

**Standard Abbreviations**
LTL consists of formulas that have the form $\text{A}_f$ where $f$ is a path formula in which the only state

Example: $\text{A} (\text{FG})$

subformulas permitted are atomic propositions.

Example: $\text{A} (\text{GF})$

LTL is a restricted subset of CTL that permits only branching-time operators—each of the

Example: $\text{AG} (\text{EF})$

linear-time operators $\text{G}$, $\text{F}$, $\text{X}$, and $\text{U}$ must be immediately preceded by a path quantifier.

Example: $\text{AG} (\text{EF})$

CTL and LTL
It can be shown that the three logics discussed in this section have different expressive powers.

The disjunction $\forall (\text{EC}) \wedge (\text{EF})$ is a CTL formula that is not expressible in either CTL or LTL. Likewise, there is no LTL formula that is equivalent to the CTL formula $\forall (\text{AG}) (\text{EF})$.

For example, there is no CTL formula that is equivalent to the LTL formula $\forall (\text{AG}) (\text{EF})$.

It can be shown that the three logics discussed in this section have different expressive powers.
There are eight basic CTL operators:

- $A\Box$ and $E\Diamond$
- $A\Diamond$ and $E\Box$
- $A\forall$ and $E\exists$
- $A\exists$ and $E\forall$
- $A\mu$ and $E\nu$
- $A\nu$ and $E\mu$
- $A\mu$ and $E\nu$
- $A\nu$ and $E\mu$

Each of these can be expressed in terms of $E\Box$, $E\Diamond$, and $E\mu$:
The four most widely used CTL operators are illustrated below. Each computation tree has the state $s_0$ as its root.

### Basic CTL Operators
Typical CTL Formulas

• $\text{AG} (\text{deviceEnabled})$: DeviceEnabled holds infinitely often on every computation path.

• $\text{AG} (\text{AF} \text{DeviceEnabled})$: DeviceEnabled holds in every state.

• $\text{AG} (\text{Request} \Rightarrow \text{AF} \text{Acks})$: If a Request occurs, then it will be eventually acknowledged.

• $\text{AG} (\text{Acks} \Rightarrow \text{AF} \text{Ack})$: If an Ack occurs, then it will be eventually acknowledged.

• $\text{EF} (\text{Restart} \land \text{Shared})$: It is possible to get to a state where Shared holds but Ready does not hold.