Outline

1 Branching-time logic
   - Syntax of CTL
   - Semantics of computation tree logic
   - Practical patterns of specifications
   - Important equivalences between CTL formulae
   - Adequate sets of CTL connectives

2 CTL* and the expressive powers of LTL and CTL
In LTL, we specify properties about paths.

A model satisfies an LTL formula if all its paths from the initial state satisfy the formula.

Sometimes, we may wish to consider properties about states.
- Recall that non-blocking asks if a process can always enter the trying state when it is at the non-critical state.

Branching-time logic allows us to specify such properties.

We can ask if
- all paths from a state satisfy certain properties; and
- a path from a state satisfies certain properties.
Outline

1. Branching-time logic
   - Syntax of CTL
   - Semantics of computation tree logic
   - Practical patterns of specifications
   - Important equivalences between CTL formulae
   - Adequate sets of CTL connectives

2. CTL* and the expressive powers of LTL and CTL
Computation Tree Logic (CTL)

- Computation Tree Logic (CTL) is a branching-time logic.
- As usual, we fix a set \( \text{Atoms} \) of atomic formulae \( p, q, \ldots \).

**Definition**

Computation Tree Logic (CTL) has the following syntax:

\[
\phi ::= \bot | T | p | (\neg \phi) | (\phi \land \phi) | (\phi \lor \phi) | (\phi \Longrightarrow \phi) | \text{AX} \phi | \text{EX} \phi |
\text{AF} \phi | \text{EF} \phi | \text{AG} | \text{EG} | \text{A}[\phi \mathcal{U} \phi] | \text{E}[\phi \mathcal{U} \phi]
\]

where \( p \in \text{Atoms} \) is an atomic formula.

- Observe that CTL temporal connectives are either \( \text{A} \) (for all paths) or \( \text{E} \) (there exists) followed by an LTL temporal connectives \( X, F, G, U \).
By convention, binding powers of CTL connectives are:

<table>
<thead>
<tr>
<th>strongest</th>
<th>→</th>
<th>weakest</th>
</tr>
</thead>
<tbody>
<tr>
<td>¬, AX, EX, AF, EF, AG, EG</td>
<td>∧, ∨</td>
<td>⇒, AU, EU</td>
</tr>
</tbody>
</table>

Examples:

\[ \text{AG}(q \implies \text{EG} r) \quad \text{EFE}[r \mathcal{U} q] \quad \text{A}[p \mathcal{U} \text{EF} r] \]
\[ \text{EFEG} p \implies \text{AF} r \quad \text{A}[p_1 \mathcal{U} \text{A}[p_2 \mathcal{U} p_3]] \quad \text{E}[\text{A}[p_1 \mathcal{U} p_2] \mathcal{U} p_3] \]
\[ \text{AG}(p \implies \text{A}[p \mathcal{U} (\neg p \land \text{A}[\neg p \mathcal{U} q])]) \]

Non-examples:

\[ \text{EFG} r \quad \text{A} \neg \text{G} p \quad \text{F}[r \mathcal{U} q] \]
\[ \text{EF}(r \mathcal{U} q) \quad \text{AEF} r \quad \text{A}[(r \mathcal{U} q) \land (p \mathcal{U} r)] \]

A subformula of a CTL formula \( \phi \) is any formula \( \psi \) whose parse tree is a subtree of \( \phi \)'s parse tree.
1. **Branching-time logic**
   - Syntax of CTL
   - Semantics of computation tree logic
   - Practical patterns of specifications
   - Important equivalences between CTL formulae
   - Adequate sets of CTL connectives

2. **CTL* and the expressive powers of LTL and CTL**
Definition

Let $\mathcal{M} = (S, \rightarrow, L)$ be a transition system, $s \in S$, and $\phi$ a CTL formula. $\mathcal{M}, s \models \phi$ is defined as follows.

1. $\mathcal{M}, s \models \top$; $\mathcal{M}, s \not\models \bot$; $\mathcal{M}, s \models p$ if $p \in L(s)$; $\mathcal{M}, s \models \neg \phi$ if $\mathcal{M}, s \not\models \phi$;
2. $\mathcal{M}, s \models \phi \land \psi$ if $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$; $\mathcal{M}, s \models \phi \lor \psi$ if $\mathcal{M}, s \models \phi$ or $\mathcal{M}, s \models \psi$;
3. $\mathcal{M}, s \models \phi \implies \psi$ if $\mathcal{M}, s \models \psi$ whenever $\mathcal{M}, s \models \phi$;
4. $\mathcal{M}, s \models \text{AX} \phi$ if for all $s'$ that $s \rightarrow s'$ we have $\mathcal{M}, s' \models \phi$;
5. $\mathcal{M}, s \models \text{EX} \phi$ if for some $s'$ that $s \rightarrow s'$ we have $\mathcal{M}, s' \models \phi$;
6. $\mathcal{M}, s \models \text{AG} \phi$ if for all paths $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ we have $\mathcal{M}, s_i \models \phi$ for every $i \geq 0$;
7. $\mathcal{M}, s \models \text{EG} \phi$ if for some path $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ we have $\mathcal{M}, s_i \models \phi$ for every $i \geq 0$;
8. $\mathcal{M}, s \models \text{AF} \phi$ if for all paths $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ we have $\mathcal{M}, s_i \models \phi$ for some $i \geq 0$;
9. $\mathcal{M}, s \models \text{EF} \phi$ if for some path $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ we have $\mathcal{M}, s_i \models \phi$ for some $i \geq 0$;
10. $\mathcal{M}, s \models \text{A}[\phi \mathcal{U} \psi]$ if for all paths $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ there is $i \geq 0$ that $\mathcal{M}, s_i \models \psi$ and for every $0 \leq j < i$ we have $\mathcal{M}, s_j \models \phi$;
11. $\mathcal{M}, s \models \text{E}[\phi \mathcal{U} \psi]$ if for some path $s_0(=s) \rightarrow s_1 \rightarrow \cdots$ there is $i \geq 0$ that $\mathcal{M}, s_i \models \psi$ and for every $0 \leq j < i$ we have $\mathcal{M}, s_j \models \phi$. 
Visualization $- \mathcal{M}, s \models p$
Visualization – $\mathcal{M}, s \models AXp$
Visualization – $M, s \models \mathbf{EX}p$
Visualization – $\mathcal{M}, s \models AGp$
Visualization – $M, s \models EGp$
Visualization \( \mathcal{M}, s \models \text{AF} \, p \)
Visualization – $\mathcal{M}, s \models EFp$
Visualization – $\mathcal{M}, s \models A[p U q]$
Visualization – $\mathcal{M}, s \models E[p U q]$
Recall the transition system $\mathcal{M}$ above.

We will give examples of $\mathcal{M}, s \models \phi$.

To do so, we first “unroll” $\mathcal{M}$ from its initial state $s_0$. 
Example II

- $\mathcal{M}, s_0 \models p \land q$; $\mathcal{M}, s_0 \models \neg r$; $\mathcal{M}, s_0 \models T$;
- $\mathcal{M}, s_0 \models \text{EX}(q \land r)$; $\mathcal{M}, s_0 \models \neg \text{AX}(q \land r)$;
- $\mathcal{M}, s_0 \models \neg \text{EF}(p \land r)$ since a state with $p, r$ is not reachable from $s_0$;
- $\mathcal{M}, s_1 \models \neg \text{EF}(p \land r)$ since a state with $p, r$ is not reachable from $s_0$;
- $\mathcal{M}, s_2 \models \text{AF}r$ since $r$ is always reachable from $s_0$;
- $\mathcal{M}, s_0 \models \text{E}[(p \land q) \text{ U r}]$ since $s_0 \rightarrow s_2 \rightarrow s_2 \rightarrow \cdots$;
- $\mathcal{M}, s_0 \models \text{A}[p \text{ U r}]$ since $s_0 \rightarrow s_1 \rightarrow \cdots$ and $s_0 \rightarrow s_2 \rightarrow \cdots$;
- $\mathcal{M}, s_0 \models \text{AG}(p \lor q \lor r \implies \text{EFEG}r)$. 
1 Branching-time logic
- Syntax of CTL
- Semantics of computation tree logic
- Practical patterns of specifications
- Important equivalences between CTL formulae
- Adequate sets of CTL connectives

2 CTL* and the expressive powers of LTL and CTL
Patterns of Specifications

- It is possible to get to a state where \textit{started} holds but \textit{ready} doesn’t:  
  \[ \text{EF}(\text{started} \land \neg \text{ready}) \];

- For any state, if a \textit{requested} occurs, it will be \textit{acknowledged} eventually:  
  \[ \text{AG}(\text{requested} \implies \text{AF acknowledged}) \];

- A process is \textit{enabled} infinitely often on all paths:  
  \[ \text{AG}(\text{AF enabled}) \];

- A process will eventually be \textit{stable} permanently:  
  \[ \text{AF}(\text{AG stable}) \];

- From any state it is possible to get to a \textit{restart} state:  
  \[ \text{AG}(\text{EF restart}) \];

- Process \( P_1 \) can always request to enter its critical section:  
  \[ \text{AG}(n_1 \implies \text{EX} t_1) \).
  
  - Recall that non-blocking is not expressible in LTL.

- Consider the following property: “if a process is \textit{enabled} infinitely often, it is \textit{running} infinitely often. This is \textbf{not} expressed by  
  \[ \text{AGAF enabled} \implies \text{AGAF running} \].
  
  - In fact, this property is not expressible in CTL.
  
  - We can express the property by  
    \[ \text{GF enabled} \implies \text{GF running} \text{ in LTL.} \]
Outline

1 Branching-time logic
   - Syntax of CTL
   - Semantics of computation tree logic
   - Practical patterns of specifications
   - Important equivalences between CTL formulae
   - Adequate sets of CTL connectives

2 CTL* and the expressive powers of LTL and CTL
Semantic Equivalences

**Definition**

Let $\phi$ and $\psi$ be CTL formulae. $\phi$ and $\psi$ are semantically equivalent (written $\phi \equiv \psi$) if for all models $M$ and all state $s$ in $M$, $M, s \models \phi$ iff $M, s \not\models \psi$.

- Since $A$ is a universal quantifier and $E$ an existential quantifier, it is not hard to see the following semantically equivalent formulae:
  
  $\neg AF\phi \equiv EG\neg \phi \quad \neg EF\phi \equiv AG\neg \phi \quad \neg AX\phi \equiv EX\neg \phi$.

- Moreover, by similar arguments in LTL, we have
  
  $AF\phi \equiv A[\top U \phi] \quad EF\phi \equiv E[\top U \phi]$.

- Finally, we have
  
  $AG\phi \equiv \phi \land AXAG\phi \quad EG\phi \equiv \phi \land EXEG\phi$
  $AF\phi \equiv \phi \lor AXAF\phi \quad EF\phi \equiv \phi \lor EXEF\phi$
  $A[\phi U \psi] \equiv \psi \lor (\phi \land AXA[\phi U \psi])$
  $E[\phi U \psi] \equiv \psi \lor (\phi \land EXE[\phi U \psi])$. 
Outline

1 Branching-time logic
   - Syntax of CTL
   - Semantics of computation tree logic
   - Practical patterns of specifications
   - Important equivalences between CTL formulae
   - Adequate sets of CTL connectives

2 CTL* and the expressive powers of LTL and CTL
Adequate Sets of CTL Connectives

1. \{\text{AU, EU, AX}\} is an adequate set of CTL connectives.
   - Observe that
     \[
     \begin{align*}
     \text{AX}\phi & \equiv \neg\text{EX}\neg\phi \\
     \text{AG}\phi & \equiv \neg\text{EF}\neg\phi \\
     \text{AF}\phi & \equiv \text{A}[\top\ U\ \phi] \\
     \text{EG}\phi & \equiv \neg\text{AF}\neg\phi \\
     \text{EF}\phi & \equiv \text{E}[\top\ U\ \phi].
     \end{align*}
     \]

2. \{\text{EG, EU, EX}\} is an adequate set of CTL connectives.
   - Recall \(\phi\ U\ \psi \equiv \neg(\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi))\ \land\ \text{F}\psi\) in LTL.
   - Observe that \(\text{A}[\phi\ U\ \psi] \equiv \neg(\text{E}[\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi)]\ \lor\ \text{EG}\neg\psi).\)
     \[
     \begin{align*}
     \text{A}[\phi\ U\ \psi] & \equiv \text{A}[\neg(\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi))\ \land\ \text{F}\psi] \\
     & \equiv \neg\text{E}[\neg(\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi))\ \land\ \text{F}\psi] \\
     & \equiv \neg\text{E}[(\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi)]\ \lor\ \text{G}\neg\psi] \\
     & \equiv \neg(\text{E}[\neg\psi\ U\ (\neg\phi\ \land\ \neg\psi)]\ \lor\ \text{EG}\neg\psi).\)
     \]
More generally, we have

**Theorem**

A set of CTL temporal connectives is adequate iff it contains at least one of \(\{\text{AX, EX}\}\), at least one of \(\{\text{EG, AF, AU}\}\), and \(\text{EU}\).

We can also define \(\text{AR, ER, AW, and EW}\) by \(\text{EU}\):

\[
\begin{align*}
\text{AR} & : \phi R \psi \\
\text{AW} & : \phi W \psi \\
\text{ER} & : \phi U \psi \\
\text{EW} & : \phi U \psi
\end{align*}
\]
1 Branching-time logic

2 CTL* and the expressive powers of LTL and CTL
In LTL, we have temporal connectives $X, F, G, U$.

In CTL, we have temporal connectives $AX, EX, AF, EF, AG, EG, AU, EU$.

Observe that CTL temporal connectives are LTL connectives prefixed by path quantifiers $A$ or $E$.

Consider the property: for some path, if $p$ occurs eventually, then $q$ occurs eventually.

Using LTL, one would write $Fp \implies Fq$.

- But LTL considers all paths from a specified state.

Using CTL, one would write $EFp \implies EFq$.

- But this is not what we want either.

How about $E(Fp \implies Fq)$?

- But this is not in LTL nor CTL!

We need a more expressive logic.
Definition

CTL* has the following syntax:

- **state formulae**

\[ \phi ::= T \mid p \mid (\neg \phi) \mid (\phi \land \phi) \mid A[\alpha] \mid E[\alpha] \]

where \( p \in \text{Atoms} \) is an atomic formula

- **path formulae**

\[ \alpha ::= \phi \mid (\neg \alpha) \mid (\alpha \land \alpha) \mid (\alpha U \alpha) \mid G\alpha \mid F\alpha \mid X\alpha \]
An LTL formula is in fact a CTL* path formula.

Since we consider all paths in LTL, an LTL formula $\alpha$ is in fact the CTL* state formula $\mathbf{A}[\alpha]$.

We obtain CTL by restricting path formulae to

$$\alpha ::= (\phi \mathbf{U} \phi) \mid (\mathbf{G}\phi) \mid (\mathbf{F}\phi) \mid (\mathbf{X}\phi)$$

Hence both LTL and CTL are subclasses of CTL*.
Expressive Powers of LTL, CTL, and CTL*

- $\psi_1 = \text{AGEF} p$.

- $\psi_2 = \text{AG}(p \implies \text{AF} q)(\text{CTL}) = \text{G}(p \implies \text{F} q)(\text{LTL})$.

- $\psi_3 = \text{GF} p \implies \text{F} q$.

- $\psi_4 = \text{E}[\text{GF} p]$.
More Examples

- $FGp$ and $AFAGp$ are not equivalent.
- $XFp \equiv FXp$ (LTL) and $AXAFp$ (CTL) are equivalent. But $AFAXp \not\equiv AXAFp$.
- So, between LTL and CTL, which one is better?
  - In 1980’s, many debates (and papers) were fought for the question.
  - Now, people generally believe they are just different: one is not better than the other.
  - Try to compare LTL and CTL.