Computation Tree Logic

Hao Zheng

Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
Email: zheng@cse.usf.edu
Phone: (813)974-4757
Fax: (813)974-5456
1 Introduction (Section 6.1)

2 Computation Tree Logic (Section 6.2)
   - CTL - Syntax
   - CTL - Semantics
   - CTL Semantics - Equivalences

3 CTL Model Checking (Section 6.4)

4 Comparing CTL and LTL (Section 6.3)
1 Introduction (Section 6.1)

2 Computation Tree Logic (Section 6.2)
   - CTL - Syntax
   - CTL - Semantics
   - CTL Semantics - Equivalences

3 CTL Model Checking (Section 6.4)

4 Comparing CTL and LTL (Section 6.3)
• **Linear** temporal logic:

  "Statements about (all) paths starting in a state."

  - \( s \models □(x ≤ 20) \) iff for all possible paths starting in \( s \) always \( x ≤ 20 \).
  - Quantifier \( ∀ \) is implicit: \( s \models □(x ≤ 20) \equiv s \models ∀□(x ≤ 20) \)
Intention (6.1)

- **Linear** temporal logic:
  
  "Statements about (all) paths starting in a state."

- 
  \[ s \models \Box (x \leq 20) \] iff for all possible paths starting in \( s \) always \( x \leq 20 \).

- Quantifier \( \forall \) is implicit: 
  \[ s \models \Box (x \leq 20) \equiv s \models \forall \Box (x \leq 20) \]

- **Branching** temporal logic:

  "Statements about all or some paths starting in a state."

- 
  \[ s \models \forall \Box (x \leq 20) \] iff for all paths starting in \( s \) always \( x \leq 20 \).

- \[ s \models \exists \Box (x \leq 20) \] iff for some path starting in \( s \) always \( x \leq 20 \).

- Nesting of path quantifiers is allowed.
• **Linear** temporal logic:

  “Statements about (all) paths starting in a state.”

  • $s \models \Box(x \leq 20)$ iff for all possible paths starting in $s$ always $x \leq 20$.
  • Quantifier $\forall$ is implicit: $s \models \Box(x \leq 20) \equiv s \models \forall\Box(x \leq 20)$

• **Branching** temporal logic:

  “Statements about all or some paths starting in a state.”

  • $s \models \forall\Box(x \leq 20)$ iff for all paths starting in $s$ always $x \leq 20$.
  • $s \models \exists\Box(x \leq 20)$ iff for some path starting in $s$ always $x \leq 20$.
  • Nesting of path quantifiers is allowed.

• Checking $\exists \varphi$ in LTL can be done using $\forall \neg \varphi$, but this does not work for nested formulas such as $\forall\Box\exists\Diamond a$.

  In any state of every computation ($\forall\Box$), it is possible ($\exists\Diamond$) to return to the initial state.

□◊$a$ vs $\forall\Box\exists\Diamond a$, difference?
Semantics is based on a branching notion of time.
- An infinite tree of states obtained by unfolding the transition system.
- One “time instant” may have several possible successor “time instants”.

\[
\begin{align*}
(s_0, 0) & \rightarrow (s_1, 1) \\
(s_2, 2) & \rightarrow (s_3, 2) \\
(s_3, 3) & \rightarrow (s_2, 3)
\end{align*}
\]
Incomparable expressiveness:
- There are properties that can be expressed in LTL, but not in CTL.
- There are also properties that can be expressed in CTL, but not in LTL.

Distinct model-checking algorithms with different time/space complexities.

Fairness assumptions require special treatment in CTL.
- A natural part of LTL.

Equivalences and preorders between transition systems based on simulation and bisimulation relations rather than traces.
Contents

1 Introduction (Section 6.1)

2 Computation Tree Logic (Section 6.2)
   - CTL - Syntax
   - CTL - Semantics
   - CTL Semantics - Equivalences

3 CTL Model Checking (Section 6.4)

4 Comparing CTL and LTL (Section 6.3)
Modal logic over infinite trees [Clarke & Emerson 1981].

- **Statements over states ($\Phi$):**
  - $a \in AP$
  - $\neg \Phi$ and $\Phi_1 \land \Phi_2$
  - $\exists \varphi$
  - $\forall \varphi$

- **Statements over paths ($\varphi$):**
  - $\circ \Phi$
  - $\Phi_1 \mathbin{U} \Phi_2$

  atomic proposition
  negation and conjunction
  there exists a path fulfilling $\varphi$
  all paths fulfill $\varphi$

  the next state fulfills $\Phi$
  $\Phi_1$ holds until a $\Phi_2$-state is reached
Modal logic over infinite trees [Clarke & Emerson 1981].

- **Statements over states (Φ):**
  - $a \in AP$
  - $\neg \Phi$ and $\Phi_1 \land \Phi_2$
  - $\exists \varphi$
  - $\forall \varphi$

- **Statements over paths (φ):**
  - $\circ \Phi$
  - $\Phi_1 U \Phi_2$

⇒ Note that $\circ$ and $U$ *alternate* with $\forall$ and $\exists$:
  - $\forall \circ \circ \Phi$, $\forall \exists \circ \Phi \not\in$ CTL, but $\forall \circ \forall \circ \Phi$ and $\forall \circ \exists \circ \Phi \in$ CTL.
  - Four operators by the syntax rules:
    - $\forall \circ (AX)$, $\forall \Box (AG)$, $\forall U (AU)$, $\forall \Diamond (AF)$
    - $\exists \circ (EX)$, $\exists \Box (EG)$, $\exists U (EU)$, $\exists \Diamond (EF)$

- Check Example 6.2 in the book for some example formulas.
Derived Operators

potentially $\Phi$: $\exists \diamond \Phi = \exists (\text{true} U \Phi)$

inevitably $\Phi$: $\forall \diamond \Phi = \forall (\text{true} U \Phi)$

potentially always $\Phi$: $\exists \square \Phi = \neg \forall \diamond \neg \Phi$

invariantly $\Phi$: $\forall \square \Phi = \neg \exists \diamond \neg \Phi$

weak until:

$\exists (\Phi_1 U \Phi_2) = \neg \forall ((\Phi_1 \land \neg \Phi_2) U (\neg \Phi_1 \land \neg \Phi_2))$

$\forall (\Phi_1 U \Phi_2) = \neg \exists ((\Phi_1 \land \neg \Phi_2) U (\neg \Phi_1 \land \neg \Phi_2))$

The boolean connectives are derived as usual.
Example Properties in CTL

- Mutual exclusion:
  \[ \forall \Box (\neg crit_1 \lor \neg crit_2) \]

- Starvation freedom:
  \[ (\forall \Box \forall \Diamond crit_1) \land (\forall \Box \forall \Diamond crit_2) \]

- Each red light is preceded by a yellow light:
  \[ \forall \Box (yellow \lor \forall \bigcirc \neg red) \]

- Traffic light is infinitely often green:
  \[ \forall \Box \forall \Diamond green \]

- Every request is eventually granted:
  \[ \forall \Box (request \Rightarrow \forall \Diamond response) \]

- In every reachable state, it is possible to return to the start state:
  \[ \forall \Box \exists \Diamond start \]
CTL Semantics Visualization

\[ \exists \diamond \text{red} \]

\[ \exists \square \text{red} \]

\[ \exists (\text{yellow} \cup \text{red}) \]

\[ \forall \diamond \text{red} \]

\[ \forall \square \text{red} \]

\[ \forall (\text{yellow} \cup \text{red}) \]
**CTL Semantics - State Formulas**

Defined by a relation $|=\text{ such that}$

$s |= \Phi$ if and only if formula $\Phi$ holds in state $s$

- $s |= a$ iff $a \in L(s)$
- $s |= \neg \Phi$ iff $\neg (s |= \Phi)$
- $s |= \Phi \land \Psi$ iff $(s |= \Phi) \land (s |= \Psi)$
- $s |= \exists \varphi$ iff $\pi |= \varphi$ for *some* path $\pi$ that starts in $s$
- $s |= \forall \varphi$ iff $\pi |= \varphi$ for *all* paths $\pi$ that start in $s$
Define a relation $\models$ such that

\[ \pi \models \varphi \text{ if and only if path } \pi \text{ satisfies } \varphi \]

\[
\begin{align*}
\pi \models \bigcirc \Phi & \quad \text{iff } \pi[1] \models \Phi \\
\pi \models \Phi \cup \Psi & \quad \text{iff } (\exists \ j \geq 0. \ \pi[j] \models \Psi \ \land \ (\forall \ 0 \leq k < j. \ \pi[k] \models \Phi)) \\
\end{align*}
\]

where $\pi[i]$ denotes the state $s_i$ in the path $\pi$.
Let \( TS = (S, Act, \rightarrow, I, AP, L) \) be a transition system.

- For CTL-state-formula \( \Phi \), the satisfaction set \( Sat(\Phi) \) is defined by:
  \[
  Sat(\Phi) = \{ s \in S | s \models \Phi \}
  \]

- \( TS \) satisfies CTL-formula \( \Phi \) iff \( \Phi \) holds in all its initial states:
  \[
  TS \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I. s_0 \models \Phi
  \]
  This is equivalent to \( I \subseteq Sat(\Phi) \).
CTL Semantics - Examples

\[
\exists (a U (\neg a \land \forall (\neg a U b)))
\]

\[
\exists \Box a \quad \forall \Box a
\]

\[
\exists (\exists \Box a) \quad \forall (a U b)
\]

Figure 6.4: Interpretation of several CTL formulae.
Remark 6.10  The Semantics of Negation

$TS \not\models \Phi$ and $TS \not\models \neg \Phi$ is possible due to having multiple initial states, e.g., $s_0 \models \exists \Box \Phi$ and $s'_0 \not\models \exists \Box \Phi$.

$TS \not\models \exists \Box a$ and $TS \not\models \neg \exists \Box a$
6.2.3 CTL Equivalence

Definition 6.12

CTL-formulas $\Phi$ and $\Psi$ (over $AP$) are *equivalent*, denoted $\Phi \equiv \Psi$ if and only if $\text{Sat}(\Phi) = \text{Sat}(\Psi)$ for *all* transition systems $TS$ over $AP$.

$$\Phi \equiv \Psi \iff (TS \models \Phi \text{ if and only if } TS \models \Psi)$$
Duality Laws

\[ \forall \bigcirc \Phi \equiv \neg \exists \bigcirc \neg \Phi \]

\[ \exists \bigcirc \Phi \equiv \neg \forall \bigcirc \neg \Phi \]

\[ \forall \lozenge \Phi \equiv \neg \exists \square \neg \Phi \]

\[ \exists \lozenge \Phi \equiv \neg \forall \square \neg \Phi \]

\[ \forall (\Phi \cup \Psi) \equiv \neg \exists ((\Phi \land \neg \Psi) \cup (\neg \Phi \land \neg \Psi)) \]
Recall in LTL: $\varphi U \psi \equiv \psi \lor (\varphi \land \Box (\varphi U \psi))$

In CTL:

- $\forall(\Phi U \Psi) \equiv \Psi \lor (\Phi \land \forall \Box \forall(\Phi U \Psi))$
- $\forall \Diamond \Phi \equiv \Phi \lor \forall \Diamond \forall \Diamond \Phi$
- $\forall \Box \Phi \equiv \Phi \land \forall \Box \forall \Box \Phi$

- $\exists(\Phi U \Psi) \equiv \Psi \lor (\Phi \land \exists \Diamond \exists(\Phi U \Psi))$
- $\exists \Diamond \Phi \equiv \Phi \lor \exists \Diamond \exists \Diamond \Phi$
- $\exists \Box \Phi \equiv \Phi \land \exists \Diamond \exists \Box \Phi$
Distributive Laws (1)

Recall in LTL: \( \square(\varphi \land \psi) \equiv \square \varphi \land \square \psi \) and \( \Diamond(\varphi \lor \psi) \equiv \Diamond \varphi \lor \Diamond \psi \)

In CTL:

\[
\forall \square(\Phi \land \Psi) \equiv \forall \square \Phi \land \forall \square \Psi
\]

\[
\exists \Diamond(\Phi \lor \Psi) \equiv \exists \Diamond \Phi \lor \exists \Diamond \Psi
\]
Distributive Laws (2)

Note that $\exists \Box (\Phi \land \Psi) \not\equiv \exists \Box \Phi \land \exists \Box \Psi$ and $\forall \Diamond (\Phi \lor \Psi) \not\equiv \forall \Diamond \Phi \lor \forall \Diamond \Psi$.

Therefore, $s \models \forall \Diamond (a \lor b)$ since
\[
\begin{align*}
  s' \models a \quad &\implies \quad s' \models a \lor b \\
  s'' \models a \quad &\implies \quad s'' \models a \lor b
\end{align*}
\]

However, $s \not\models \forall \Diamond a$ and $s \not\models \forall \Diamond b$. 
1 Introduction (Section 6.1)

2 Computation Tree Logic (Section 6.2)
   - CTL - Syntax
   - CTL - Semantics
   - CTL Semantics - Equivalences

3 CTL Model Checking (Section 6.4)

4 Comparing CTL and LTL (Section 6.3)
The set of CTL formulas in *existential normal form* (ENF) is given by:

\[
\Phi ::= \text{true} \quad | \quad a \quad | \quad \Phi_1 \land \Phi_2 \quad | \quad \neg \Phi \quad | \quad \text{EX} \Phi \quad | \quad \exists(\Phi_1 \cup \Phi_2) \quad | \quad \text{EG} \Phi
\]

- For each CTL formula, there exists an equivalent CTL formula in ENF.

\[
\text{AX} \Phi \equiv \neg \text{EX} \neg \Phi
\]

\[
\forall(\Phi \cup \Psi) \equiv \neg \exists(\neg \Psi \cup (\neg \Phi \land \neg \Psi)) \land \neg \text{EG} \neg \Psi
\]

- Handle only EX \Phi, EG \Phi, and \exists(\Phi_1 \cup \Phi_2).
CTL Model Checking

- How to check whether $TS$ satisfies CTL formula $\hat{\Phi}$?
  - Convert the formula $\hat{\Phi}$ into the equivalent $\Phi$ in ENF.
  - Compute recursively the set $Sat(\Phi) = \{ s \in S | s \models \Phi \}$.
  - $TS \models \Phi$ if and only if $I \subseteq Sat(\Phi)$.
- Recursive bottom-up computation of $Sat(\Phi)$:
  - Consider the parse-tree of $\Phi$.
  - Start to compute $Sat(\Psi_i)$, for all leafs, then go one level up in the tree and determine $Sat(\cdot)$ for these nodes, repeat until the root is computed.

  e.g., $Sat(\underbrace{\Psi_1 \land \Psi_2}_\text{node at level } i) = Sat(\underbrace{\Psi_1}_{\text{node at level } i+1}) \cap Sat(\underbrace{\Psi_2}_{\text{node at level } i+1})$
CTL Model Checking: An Example

\[ \Phi = \mathbf{EX}_a \land \exists (b \mathbf{U} \mathbf{EG} \neg c) \]
Theorem 6.23  Characterization of $Sat$ (1)

For all CTL formulas $\Phi, \Psi$ over $AP$ it holds:

\[
Sat(\text{true}) = S \\
Sat(a) = \{ s \in S \mid a \in L(s) \} , \text{ for any } a \in AP \\
Sat(\Phi \land \Psi) = Sat(\Phi) \cap Sat(\Psi) \\
Sat(\neg \Phi) = S \setminus Sat(\Phi) \\
Sat(\text{EX } \Phi) = \{ s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset \}
\]

where $TS = (S, Act, \rightarrow, I, AP, L)$ is a transition system without terminal states.
Theorem 6.23  Characterization of \( Sat (2) \)

- \( Sat(\exists (\Phi U \Psi)) \) is the smallest subset \( T \) of \( S \), such that:
  1. \( Sat(\Psi) \subseteq T \) and
  2. \((s \in Sat(\Phi) \text{ and } Post(s) \cap T \neq \emptyset) \) implies \( s \in T \)

- \( Sat(\text{EG} \Phi) \) is the largest subset \( T \) of \( S \), such that:
  1. \( T \subseteq Sat(\Phi) \) and
  2. \( s \in T \) implies \( Post(s) \cap T \neq \emptyset \)

where \( TS = (S, Act, \rightarrow, I, AP, L) \) is a transition system without terminal states.
Algorithm 14  Computation of $\text{Sat}$

switch($\Phi$):

\begin{align*}
\text{a} &: \quad \text{return } \{ s \in S \mid a \in L(s) \}; \\
\text{...} &: \quad \text{....}
\end{align*}

$\text{EX } \Psi$ : return $\{ s \in S \mid \text{Post}(s) \cap \text{Sat}(\Psi) \neq \emptyset \}$;

$\exists (\Phi_1 \cup \Phi_2)$ : $T := \text{Sat}(\Phi_2)$;  \textit{compute the smallest fixed point}

while $\{ s \in \text{Sat}(\Phi_1) \setminus T \mid \text{Post}(s) \cap T \neq \emptyset \} \neq \emptyset$ do

\begin{align*}
\text{let } s &\in \{ s \in \text{Sat}(\Phi_1) \setminus T \mid \text{Post}(s) \cap T \neq \emptyset \}; \\
T &= T \cup \{ s \};
\end{align*}

od;

return $T$;

$\text{EG } \Phi$ : $T := \text{Sat}(\Phi)$;  \textit{compute the greatest fixed point}

while $\{ s \in T \mid \text{Post}(s) \cap T = \emptyset \} \neq \emptyset$ do

\begin{align*}
\text{let } s &\in \{ s \in T \mid \text{Post}(s) \cap T = \emptyset \}; \\
T &= T \setminus \{ s \};
\end{align*}

od;

return $T$;

end switch
Computing $\text{Sat}(\exists(\Phi \cup \Psi))$ – An Example

Check $\text{EF}((p = r) \land (p \neq q)) \equiv \exists(true \cup ((p = r) \land (p \neq q)))$
Computing $\text{Sat}(\exists (\Phi \cup \Psi))$—Summary

- $\text{Sat}(\exists (\Phi \cup \Psi))$ is the smallest set $T \subseteq S$ such that
  
  $\begin{align*}
  (1) & \quad \text{Sat}(\Psi) \subseteq T \quad \text{and} \\
  (2) & \quad \{ s \in T \mid s \models \Phi \land \text{Post}(s) \cap T \neq \emptyset \} 
  \end{align*}$

- Initially, $T_0 = \{ \text{Sat}(\Psi) \}$.
- Iteratively compute

  $T_{i+1} = T_i \cup \{ s \in \text{Sat}(\Phi) \mid \text{Post}(s) \cap T_i \neq \emptyset \}$ for $i \geq 0$.

- In other words, computing $\text{Sat}(\exists (\Phi \cup \Psi))$ results in

  $T_0 \subseteq T_1 \subseteq \ldots \subseteq T_j \subseteq T_{j+1} \subseteq \ldots$

- Since we assume $TS$ to be finite, there exists a $j \geq 0$ such that

  $T_j = T_{j+1} = \ldots = \text{Sat}(\exists (\Phi \cup \Psi))$
Computing $\text{Sat}(\text{EG } \Phi)$ – An Example

Check $\text{EG} \, q$

Let's check the CTL-formula $\exists \, \lozenge \, ((p = r) \land (p \neq q))$.
Computing $Sat(EG \Phi)$— Summary

- $Sat(EG \Phi)$ is the largest set $T \subseteq S$ such that
  
  \begin{align*}
    (1) \quad & T \subseteq Sat(\Phi) \quad \text{and} \\
    (2) \quad & \{ s \in T \mid Post(s) \cap T \neq \emptyset \}
  \end{align*}

- Initially, $T_0 = Sat(\Phi)$.

- Then, iteratively compute
  
  $$T_{i+1} = T_i \cap \{ s \in Sat(\Phi) \mid Post(s) \cap T_i \neq \emptyset \}$$

- Thus, computing $Sat(EG \Phi)$ results in
  
  $$T_0 \supseteq T_1 \supseteq \ldots$$

- Since we assume $TS$ to be finite, there exists a $j \geq 0$ such that
  
  $$T_j = T_{j+1} = \ldots = Sat(EG \Phi)$$
Alternative Algorithm for Computing $\text{Sat}(\text{EG } \Phi)$

Check $\text{EG } q$

\begin{align*}
\{ p, q, r \} & \rightarrow \{ q, r \} \\
\{ q, r \} & \rightarrow \{ q \} \\
\{ q \} & \rightarrow \{ p \} \\
\{ p \} & \rightarrow \{ p, q \} \\
\{ p, q \} & \rightarrow \{ p, r \} \\
\{ p, r \} & \rightarrow \{ p \} \\
\{ p \} & \rightarrow \{ q \} \\
\{ q \} & \rightarrow \{ r \} \\
\{ r \} & \rightarrow \emptyset \\
\emptyset & \rightarrow \mathcal{K}[q]
\end{align*}
For transition system $TS$ with $N$ states and $M$ transitions, and CTL formula $\Phi$, the CTL model-checking problem $TS \models \Phi$ can be determined in time $O(|\Phi| \cdot (N + M))$.

This result applies to both algorithms for $\text{EG} \Phi$. 
∃◊ (∃□a)

∃(a U (¬a ∧ ∀(¬a U b)))
### Definition 6.17

CTL-formula $\Phi$ and LTL-formula $\varphi$ (both over $AP$) are equivalent, denoted $\Phi \equiv \varphi$, if for any transition system $TS$ (over $AP$):

$$TS \models \Phi \text{ if and only if } TS \models \varphi$$

### Theorem 6.18

Let $\Phi$ be a CTL-formula, and $\varphi$ the LTL-formula obtained by eliminating all path quantifiers in $\Phi$. Then:

$[\text{Clarke & Draghicescu}]$

$$\Phi \equiv \varphi$$

or

there does not exist any LTL-formula that is equivalent to $\Phi$. 
LTL and CTL are Incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box a$
  - $\Diamond (a \land \lozenge a)$

- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall \Diamond \forall \Box a$
  - $\forall \Diamond (a \land \forall \lozenge a)$
  - $\forall \Box \exists \Diamond a$

$\Rightarrow$ Cannot be expressed $=$ there does not exist an equivalent formula.
\(\forall \Diamond \forall \Box a \neq \Diamond \Box a.\)
Comparing LTL and CTL (Lemma 6.19)

∀ ◇ ∀ □ a ̸≡ ◇ □ a.

s_0 \models ◇ □ a \quad \text{but} \quad s_0 \not\models ∀ ◇ ∀ □ a

path s_0^\omega \text{ violates it}
Comparing LTL and CTL (Lemma 6.20)

\[ \forall \Diamond (a \land \forall \bigcirc a) \neq \Diamond (a \land \bigcirc a). \]
Comparing LTL and CTL (Lemma 6.20)

\[ \forall \lozenge (a \land \forall \bigcirc a) \not\equiv \lozenge (a \land \bigcirc a). \]

\[ s_0 \models \lozenge (a \land \bigcirc a) \quad \text{but} \quad s_0 \not\models \forall \lozenge (a \land \forall \bigcirc a) \]

path \( s_0 s_1 (s_2)^\omega \) violates it
Comparing LTL and CTL (3)

The CTL-formula $\forall \Box \exists \Diamond a$ cannot be expressed in LTL

This is shown by contradiction: assume $j \sqsupset \Box \exists \Diamond a$; let:

$TS_0 = \{ a \}$

$s_0 \not\in 0 \sqsupset \Box \exists \Diamond a$, thus

$TS_0 \sqsupset \Box \exists \Diamond a = j$.

But $TS_0 \not\sqsupset \Box \exists \Diamond a$ as path $s \in 0 \sqsupset \Diamond a$.

Chris J. Myers (Lecture 6: CTL) Verification of Cyber-Physical Systems 28 / 155
## Linear-Time vs. Branching-Time Summary

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Linear Time</th>
<th>Branching Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“behavior”</strong> in a state $s$</td>
<td>path-based:  ( \text{trace}(s) )</td>
<td>state-based: computation tree of $s$</td>
</tr>
<tr>
<td><strong>temporal logic</strong></td>
<td>LTL: path formulas $\varphi$ [ s \models \varphi \iff \forall \pi \in \rho(s). \pi \models \varphi ]</td>
<td>CTL: state formulas existential path quantification ( \exists \varphi ) universal path quantification: ( \forall \varphi )</td>
</tr>
<tr>
<td><strong>complexity of the model checking problems</strong></td>
<td>$\mathcal{O} (</td>
<td>TS</td>
</tr>
<tr>
<td><strong>implementation-relation</strong></td>
<td>trace inclusion and the like (proof is PSPACE-complete)</td>
<td>simulation and bisimulation (proof in polynomial time)</td>
</tr>
<tr>
<td><strong>fairness</strong></td>
<td>no special techniques</td>
<td>special techniques needed</td>
</tr>
</tbody>
</table>
Conclusion

- Branching time semantics of computation
- CTL for expressing branching time properties
- CTL model checking algorithms
- CTL and LTL are NOT comparable