Overview

1. Expressiveness
2. Complexity Considerations
3. Trace and Bisimulation Equivalence
4. CTL* Model Checking
5. Summary

Linear Versus Branching Time

<table>
<thead>
<tr>
<th></th>
<th>Linear Time</th>
<th>Branching Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Behavior</strong></td>
<td>Path based traces</td>
<td>State based computation tree</td>
</tr>
<tr>
<td><strong>Temporal Logic</strong></td>
<td>LTL path formulas</td>
<td>CTL state formulas</td>
</tr>
<tr>
<td><strong>Model Checking</strong></td>
<td>$O(\text{size}(T) \cdot \exp(</td>
<td>\varphi</td>
</tr>
<tr>
<td><strong>Impl. Relation</strong></td>
<td>Trace inclusion, trace equivalence, PSPACE-complete</td>
<td>Simulation, bisimulation, PTIME</td>
</tr>
</tbody>
</table>
LTL and CTL are Incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box a$
  - $\Diamond (a \land \Diamond a)$
  There does not exist an equivalent CTL formula

- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall \Diamond \forall \Box a$
  - $\forall \Diamond (a \land \forall \Diamond a)$, and
  - $\forall \Box \exists \Diamond a$
  There does not exist an equivalent LTL formula

Relating LTL, CTL, and CTL*

![Diagram showing the relationships between LTL, CTL, and CTL*]

Overview
CTL vs. LTL Model Checking

LTL model checking is PSPACE-complete  
CTL model checking is PTIME-complete.

Take a property that can be expressed in both LTL and CTL

Is CTL model checking more efficient? No!  
LTL-formulae can be exponentially shorter than their CTL-equivalent

CTL Versus LTL

If $\Phi$ is equivalent to some LTL-formula $\varphi$ then:

$\Phi \equiv \varphi$ where $\varphi$ is obtained by removing all path quantifiers from $\Phi$.

In particular, $|\varphi| \leq |\Phi|$.

If $P \neq NP$, then there is a sequence $\varphi_n$, $n \geq 0$ of LTL formulas such that:

$\blacktriangleright |\varphi_n|$ is polynomial in $n$
$\blacktriangleright \varphi_n$ has an equivalent CTL formula
$\blacktriangleright$ no CTL formula of polynomial length is equivalent to $\varphi_n$

Proof.

$\varphi_n = \text{the absence of a Hamiltonian path in a digraph on } n \text{ vertices}$

LTL Encoding the Hamiltonian Path Problem

All $n!$ possibilities need to be explicitly enumerated

Suppose there is a CTL-formula of polynomial length equivalent to $\varphi_n$.

Then: as CTL model-checking is $\in P$,
the Hamiltonian path problem $\in P$, and $P = NP$.
Satisfiability Problem

The LTL satisfiability problem is PSPACE-complete.

The LTL satisfiability problem is equally hard as the LTL model checking problem.

The CTL satisfiability problem is EXPTIME-complete.

The CTL* satisfiability problem is 2EXPTIME-complete.

The CTL satisfiability problem is harder than the CTL model checking problem. This also applies to CTL* (and many more logics).

Trace Equivalence

Definition: trace equivalence

Transition systems $TS$ and $TS'$ (both over $AP$) are trace equivalent iff they exhibit the same traces:

$TS \equiv_{trace} TS'$ if and only if $Traces(TS) = Traces(TS')$.

Examples

Trace Equivalence and LT Properties

$TS \equiv_{trace} TS'$ if and only if $TS$ and $TS'$ satisfy the same LT properties:

$TS \equiv_{trace} TS'$ if and only if $(\forall E \subseteq (2^{AP})^\omega. TS \models E \iff TS' \models E)$.
Logical Equivalence

**Definition: logical equivalence**

For transition systems $TS$ and $TS'$ (both over $AP$):

- $TS \equiv_{LTL} TS'$ iff $(\forall \varphi \in LTL. TS \vDash \varphi \iff TS' \vDash \varphi)$
- $TS \equiv_{CTL} TS'$ iff $(\forall \Phi \in CTL. TS \vDash \Phi \iff TS' \vDash \Phi)$

In a similar way, $\equiv_L$ can be defined for logic $L$ (such as CTL$^*$ etc.).

$TS \equiv_{\text{trace}} TS'$ if and only if $TS \equiv_{LTL} TS'$

Bisimulation

**Definition: bisimulation relation**

Let $TS_i = (S_i, \text{Act}_i, \rightarrow_i, I_i, AP, L_i)$, $i=1, 2$, be transition systems. The symmetric relation $\mathcal{R} \subseteq S_1 \times S_2$ is a **bisimulation** for $(TS_1, TS_2)$ whenever:

1. for all initial states $s_1 \in I_1$, $(s_1, s_2) \in \mathcal{R}$ for some $s_2 \in I_2$
2. for all states $(s_1, s_2) \in \mathcal{R}$ it holds:
   2.1 $L_1(s_1) = L_2(s_2)$, and
   2.2 $s'_1 \in \text{Post}(s_1)$ implies $(s'_1, s'_2) \in \mathcal{R}$ for some $s'_2 \in \text{Post}(s_2)$.

Example

Visually

- $s_1 \rightarrow s'_1$
- $s_2 \rightarrow s'_2$

$\mathcal{R}$ can be completed to $\mathcal{R}$ and by symmetry
### Bisimulation on Paths

Whenever we have:

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots \]

\( R \)

\( t_0 \)

this can be completed to

\[ s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \ldots \]

\( R \quad R \quad R \quad R \quad R \)

\( t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \ldots \)

**Proof.**

By induction on index \( i \) of state \( s_i \). □

### Bisimilarity And Trace Equivalence

1. \( TS_1 \sim TS_2 \) implies \( TS_1 \equiv_{\text{trace}} TS_2 \).

2. For \( AP \)-deterministic\(^1\) \( TS_1, TS_2 \): \( TS_1 \sim TS_2 \) iff \( TS_1 \equiv_{\text{trace}} TS_2 \).

**Proof.**

1. Follows from the fact that bisimulation carries over to infinite paths.

2. Left as exercise.

\( TS_1 \sim TS_2 \) implies \( TS_1 \equiv_{\text{LTL}} TS_2 \). The converse also holds for \( AP \)-deterministic transition systems

---

\(^1\)Transition system \( TS \) is \( AP \)-deterministic whenever it has at most one initial state and \( |\text{Post}(s) \cap \{ s' \in S \mid L(s') = A \}| \leq 1 \).
Trace and Bisimulation Equivalence

Bisimulation on States

Definition: bisimulation/bisimilarity on states

Symmetric relation $R \subseteq S \times S$ is a bisimulation on $TS$ (with state space $S$) if for any $(s_1, s_2) \in R$:
1. $L(s_1) = L(s_2)$
2. $s'_1 \in \text{Post}(s_1)$ then $(s'_1, s'_2) \in R$ for some $s'_2 \in \text{Post}(s_2)$.

The states $s_1$ and $s_2$ are bisimilar, denoted $s_1 \sim_{TS} s_2$, if $(s_1, s_2) \in R$ for some bisimulation $R$ for $TS$.

$$s_1 \sim_{TS} s_2 \text{ if and only if } TS_{s_1} \sim TS_{s_2} \text{ where } TS_{s_i} \text{ denotes the transition system } TS \text{ in which } s_i \text{ is the only initial state.}$$

Proof (1)

Bisimilarity And CTL

Theorem: Bisimilarity, CTL and CTL$^*$

Let $TS$ be a finitely branching transition system and $s, s'$ states in $TS$. The following statements are equivalent:
1. $s \sim_{TS} s'$
2. $s$ and $s'$ are CTL-equivalent, i.e., $s \equiv_{CTL} s'$
3. $s$ and $s'$ are CTL$^*$-equivalent, i.e., $s \equiv_{CTL^*} s'$.

Proof.

This is proven in three steps: $\equiv_{CTL} \subseteq \sim \subseteq \equiv_{CTL^*} \subseteq \equiv_{CTL}$. The last step is trivial, since CTL$^*$ is more expressive than CTL.

---

$^2$This means that every state has only finitely many direct successors. This theorem does not hold for arbitrary infinite-state transition systems.

Proof (2)
Trace and Bisimulation Equivalence

Example

For any transition systems $TS$ and $TS'$ (over $AP$):

$TS \sim TS' \iff TS \equiv_{CTL} TS' \iff TS \equiv_{CTL^*} TS'$.

Infinite-Branching

Definition: $CTL^-$

$CTL^-$ state-formulas with $a \in AP$ obey the grammar:

$\Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid \exists \diamond \Phi \mid \forall \diamond \Phi$

No until-modalities, so no $\square$ and no $\Diamond$.

1. $CTL^-$ is strictly less expressive than $CTL$ (and than $CTL^*$).
2. $CTL^-$ equivalence coincides with $CTL$ (and $CTL^*$) equivalence.

Proof.

Follows from the fact that in the proof of equivalence of $\sim$, $\equiv_{CTL}$ and $\equiv_{CTL^*}$ only $CTL^-$-formulas are used. In particular, no until-modalities are used.
The Importance of These Results

- CTL⁻, CTL⁻ and CTL⁺-equivalence **coincide**
  - despite the fact that these logics have different expressivity

- Bisimilar transition systems preserve the same CTL⁺ formulas
  - and thus the same LTL formulas (and LT properties)

- Non-bisimilarity can be shown by a single CTL⁻ formula Φ
  - TS₁ ⊧ Φ and TS₂ ⊭ Φ implies TS₁ ∼ TS₂

- One does not even need to use an until-modality

On Complexity

The decision problem whether two finite transition systems are trace equivalent is PSPACE-complete.

**Proof.**

Reduction from language equivalence of finite-state automata.

The decision problem whether two finite transition systems are bisimilar is PTIME-complete.

**Proof.**

A polynomial-time algorithm will be dealt with in an upcoming lecture. PTIME-hardness is outside the scope of this lecture.

Overview

- Expressiveness
- Complexity Considerations
- Trace and Bisimulation Equivalence
- CTL⁺ Model Checking
- Summary

Syntax of CTL⁺

**Definition: Syntax CTL⁺**

- CTL⁺ **state**-formulas with a ∈ AP obey the grammar:

  \[ Φ ::= \text{true} \mid a \mid Φ_1 \land Φ_2 \mid \neg Φ \mid \exists ϕ \]

- and ϕ is a CTL⁺ **path**-formula formed by the grammar:

  \[ ϕ ::= Φ \mid ϕ_1 \land ϕ_2 \mid \neg ϕ \mid \diamond ϕ \mid ϕ_1 U ϕ_2 \]

where Φ is a CTL⁺ state-formula, and ϕ, ϕ₁ and ϕ₂ are path-formulas.

**in CTL⁺**: ∀ϕ = ¬∃¬ϕ. This does not hold in CTL.
Example

\[
\begin{array}{c}
\mathcal{T}:
\begin{array}{c}
\emptyset \\
\{a\}
\end{array}
\end{array}
\]

\[\mathcal{T} \not\models \Box \Box a\]

\[\mathcal{T} \models \Box \Box a\]

note: \(Sat(\Box a) = \{\bullet, \bar{\bullet}\}\)

hence: \(\cdots \models \Box \Box a\)

computation tree:

Embedding LTL

For LTL formula \(\varphi\) and TS without terminal states (both over \(AP\)) and for each \(s \in S\):

\[s \models \varphi \quad \text{if and only if} \quad s \models \forall \varphi\]

LTL semantics

CTL\(^*\) semantics

In particular:

\[TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL^*} \forall \varphi\]

CTL\(^*\) Model Checking

[Emerson & Lei, 1985]

 Adopt a recursive descent over the parse tree of \(\Phi\) (as for CTL)

 Replace maximal proper state sub-formula \(\Psi\) by new proposition \(a\_\Psi\)

 adjust labeling such that \(a\_\Psi \in L(s)\) if and only if \(s \in Sat(\Psi)\)

 In the end, this yields an LTL formula

 Most interesting case: formulas of the form \(\exists \varphi\)

\[s \not\models_{CTL^*} \exists \varphi \quad \text{iff} \quad s \not\models_{CTL^*} \forall \neg \varphi \quad \text{iff} \quad s \not\models_{LTL} \neg \varphi\]

CTL\(^*\) semantics

LTL semantics

\[Sat_{CTL^*}(\exists \varphi) = S \setminus Sat_{LTL}(\neg \varphi) = S \setminus \{s \in S \mid s \models_{LTL} \neg \varphi\}\]
### Complexity

The CTL* model-checking algorithm for finite transition system $TS$ and CTL* -formula $\Phi$ has a time complexity in $O(|TS| \cdot 2^{|\Phi|})$.

**Proof.**
The recursive descent is linear in $|\Phi|$. The most expensive procedure for a node, i.e., sub-formula $\Psi = \exists \phi$ of $\Phi$, in the parse tree is in $O(|TS| \cdot 2^{|\Psi|})$.

The CTL* model-checking problem is PSPACE-complete.

**Proof.**
Outside the scope of this lecture series.

### Summary: Equivalences

![Equivalences Diagram]

- Finite trace equivalence
- Trace equivalence
- CTL* equivalence
- LTL equivalence
- LTL safety properties
- Trace equivalence w.r.t.

### Complexity Overview

<table>
<thead>
<tr>
<th></th>
<th>CTL</th>
<th>LTL</th>
<th>CTL*</th>
</tr>
</thead>
<tbody>
<tr>
<td>model checking</td>
<td>PTIME</td>
<td>PSPACE</td>
<td>PSPACE</td>
</tr>
<tr>
<td>algorithmic complexity</td>
<td>$</td>
<td>TS</td>
<td>\cdot</td>
</tr>
<tr>
<td>satisfiability</td>
<td>EXPTIME</td>
<td>PSPACE</td>
<td>2EXPTIME</td>
</tr>
<tr>
<td>equivalence checking</td>
<td>PTIME</td>
<td>PSPACE</td>
<td>PTIME</td>
</tr>
<tr>
<td>equivalence</td>
<td>bisimilarity</td>
<td>trace equivalence</td>
<td>bisimilarity</td>
</tr>
</tbody>
</table>

All theoretical complexity indications are complete.
Thursday December 5, 10:30