Model Checking
Last week ...
Figure 5.1 sketches the intuitive meaning of temporal modalities for the simple case in event
ually until that future moment. The precedence order on the operators is as follows. The unary operators bind stronger
than the LTL formula at hand. We mostly abstain from explicitly indicating the set
of atomic proposition are formed according to the following
rules. Moreover, brackets will be used, e.g. in the LTL formula
\( \varphi \) consists of the following:
\[
\varphi ::= \text{true} \mid a \mid \varphi_1 \land \varphi_2 \mid \neg \varphi \mid \Box \varphi \mid \varphi_1 \mathcal{U} \varphi_2
\]
where \( a \in AP \).

The temporal operators are defined as follows:
\[
\Diamond \varphi \overset{\text{def}}{=} \text{true} \mathcal{U} \varphi \quad \quad \quad \quad \Box \varphi \overset{\text{def}}{=} \neg \Diamond \neg \varphi
\]
Before proceeding with the formal semantics of LTL, we present some examples.

Example 5.2. Properties for the Mutual Exclusion Problem

Consider the mutual exclusion problem for two concurrent processes $P_1$ and $P_2$, say. Process $P_i$ is modeled by three locations: (1) the noncritical section, (2) the waiting phase which is entered when the process intends to enter the critical section, and (3) the critical section. Let the propositions $\text{wait}_i$ and $\text{crit}_i$ denote that process $P_i$ is in its waiting phase and critical section, respectively.

The safety property stating that $P_1$ and $P_2$ never simultaneously have access to their critical sections can be described by the LTL-formula:

$$\Box (\neg \text{crit}_1 \lor \neg \text{crit}_2)$$

This formula expresses that always ($\Box$) at least one of the two processes is not in its critical section ($\neg \text{crit}_i$).

The liveness requirement stating that each process $P_i$ is infinitely often in its critical section can be described by the LTL-formula:

$$\Diamond \neg \text{wait}_i$$

This formula expresses that eventually ($\Diamond$) it is not true that the process is in its waiting phase ($\neg \text{wait}_i$).
This week ...
LTL Semantics

LTL semantics can be lifted from paths to states and transition systems.

\[ s \models \phi \iff \forall \pi : \pi[0] = s \implies \pi \models \phi \]

\[ TS \models \phi \iff \forall s \in I : s \models \phi \]
Example: $\Diamond (a \land b)$

- $s_1 \models \Diamond (a \land b)$
- $s_2 \not\models \Diamond (a \land b)$
- $TS \not\models \Diamond (a \land b)$
- $s_3 \not\models \Diamond (a \land b)$
To illustrate this effect, consider the transition system depicted in Figure 5.4. Let $TS$ be a transition system such that $TS \not\models \diamond a$ and $TS \not\models \neg \diamond a$. Figure 5.5: Transition system of semaphore-based mutual exclusion algorithm.

This is for paths, what about transition systems?

$\pi \models \phi \iff \pi \not\models \neg \phi$

Negation?
Equivalence of LTL formulae

### Definition

Two LTL formulas are equivalent if:

\[ \forall \pi : \pi \models \phi \iff \pi \models \psi \]

### Linear Temporal Logic

- **Duality Law**
  - \( \neg \Diamond \varphi \equiv \Box \neg \varphi \)
  - \( \neg \Box \varphi \equiv \Diamond \neg \varphi \)

- **Absorption Law**
  - \( \Diamond \Box \Diamond \varphi \equiv \Box \Diamond \varphi \)
  - \( \Box \Diamond \Box \varphi \equiv \Diamond \Box \varphi \)

- **Expansion Law**
  - \( \varphi \land \Box (\varphi \lor \psi) \equiv \varphi \land \Box \psi \)
  - \( \Diamond \varphi \equiv \varphi \land \Box \Diamond \psi \)
  - \( \Box \psi \equiv \varphi \land \Box \Diamond \psi \)

- **Distributive Law**
  - \( \Diamond (\varphi \lor \psi) \equiv \Diamond \varphi \lor \Diamond \psi \)
  - \( \Box (\varphi \land \psi) \equiv \Box \varphi \land \Box \psi \)
  - \( \Diamond (\varphi \land \psi) \equiv \Diamond \varphi \land \Diamond \psi \)

- **Idempotency Law**
  - \( \Diamond \Diamond \varphi \equiv \Diamond \varphi \)
  - \( \Box \Box \varphi \equiv \Box \varphi \)
  - \( \varphi \lor \Box \psi \equiv \varphi \lor \Box \psi \)
  - \( \Diamond \varphi \equiv \Diamond \varphi \)
  - \( \Box \varphi \equiv \Box \varphi \)
Categorization of LTL properties

- **Safety**: nothing bad should happen.
  \[\Phi = \neg\text{crit}_1 \lor \neg\text{crit}_2\]

- **Invariants**: a condition \(\Phi\) holds for all reachable states.

- **Safety Properties**: any infinite path does not have a bad finite prefix.
  \[\square (\text{red} \rightarrow \neg\Diamond \text{green})\]

- **Liveness**: something good will happen in future.
  \[\square (\text{request} \rightarrow \Diamond \text{response})\]

Any finite prefix can be extended to a trace that satisfies the property.
Limitations of LTL

Let $TS$ and $TS'$ be transition systems without terminal states and with the same set of atomic propositions. Then:

$\text{Traces}(TS) = \text{Traces}(TS') \iff TS$ and $TS'$ satisfy the same LT properties.

There thus does not exist an LT property that can distinguish between trace-equivalent transition systems. Stated differently, in order to establish that the transition systems $TS$ and $TS'$ are not trace-equivalent it suffices to find one LT property that holds for one but not for the other.

**Example 3.19. Two Beverage Vending Machines**

Consider the two transition systems in Figure 3.8 that both model a beverage vending machine. For simplicity, the observable action labels of transitions have been omitted. Both machines are able to offer soda and beer. The left transition system models a beverage machine that after insertion of a coin nondeterministically chooses to either provide soda or beer. The right one, however, has two selection buttons (one for each beverage), and after insertion of a coin, nondeterministically blocks one of the buttons. In either case, the user has no control over the beverage obtained—the choice of beverage is under full control of the vending machine.

Let $AP = \{\text{pay}, \text{soda}, \text{beer}\}$. Although the two vending machines behaved differently, it is not difficult to see that they exhibit the same traces when considering $AP$, as for both machines traces are alternating sequences of $\text{pay}$ and either $\text{soda}$ or $\text{beer}$. The vending machines are thus trace-equivalent. By Corollary 3.18 both vending machines satisfy exactly the same LT properties. Stated differently, it means that there does not exist an LT property that distinguishes between the two vending machines.

These two transition systems satisfy the same set of LTL formulas. But they function in different ways.

They are **trace equivalent**.
Computational Tree Logic (CTL)
Computational Tree Logic (CTL)

\[ \begin{align*}
\textbf{s0} & \xrightarrow{x \neq 0} \textbf{s1} \xleftarrow{x = 0} \\
& \xrightarrow{x = 0} \textbf{s2} \\
& \xrightarrow{x = 1, x \neq 0} \textbf{s3}
\end{align*} \]

\[ 
\begin{array}{c}
(s0, 0) \\
\downarrow \\
(s1, 1) \\
\downarrow \\
(s2, 2) & (s3, 2) \\
\downarrow & \downarrow \\
(s3, 3) & (s2, 3) \\
\downarrow & \downarrow \\
(s2, 4) & (s3, 4)(s2, 4) \\
\downarrow & \downarrow \\
\vdots & \vdots \\
\end{array}
\]
# Computational Tree Logic (CTL)

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Linear time</th>
<th>Branching time</th>
</tr>
</thead>
<tbody>
<tr>
<td>“behavior” in a state $s$</td>
<td>path-based: $\text{trace}(s)$</td>
<td>state-based: computation tree of $s$</td>
</tr>
<tr>
<td>temporal logic</td>
<td>LTL: path formulae $\phi$ $s \models \phi$ iff $\forall \pi \in \text{Paths}(s). \pi \models \phi$</td>
<td>CTL: state formulae $\exists \phi$ existential path quantification $\exists \phi$</td>
</tr>
</tbody>
</table>

Table 6.1 summarizes the main differences between the linear-time and branching-time perspective in a succinct way.
Computation Tree Logic

The Boolean operators true, false, "eventually", "always", and "weak until" can be derived—similarly to LTL formulae, path formulae in CTL are simpler: as in LTL they are built by the branching structure, while path formulae express temporal properties of paths. Compared to algorithm for CTL, and introduce some extensions of CTL.

6.2 Computation Tree Logic

CTL

Some path”) or the path quantifier which following grammar:

\[ Ψ : = \text{true} \mid a \mid Φ_1 \land Φ_2 \mid ¬Φ \mid ∃ϕ \mid ∀ϕ \]

\[ Φ : = \Box Φ \mid Φ_1 ∨ Φ_2 \]

\[ ϕ : = □ Φ \]

Examples:  
\[ ∃\Box (x = 1) \]
\[ ∀\Box (x = 1) \]

But not:  
\[ ∃(x = 1 \land ∀\Box (x ≥ 3)) \]
\[ ∃\Box (\text{true} \lor (x = 1)) \]
Eventually and Always

eventually: \( \exists \diamond \Phi = \exists (true \cup \Phi) \)

\( \forall \diamond \Phi = \forall (true \cup \Phi) \)

always: \( \exists \square \Phi = \neg \forall \diamond \neg \Phi \)

\( \forall \square \Phi = \neg \exists \diamond \neg \Phi \)
More Examples

Computation Tree Logic

∃ □ black
∀ □ black

∃ ♦ black
∀ ♦ black

∃ (gray U black)
∀ (gray U black)

Figure 6.2: Visualization of semantics of some basic CTL formulae.
CTL Semantics

\[
\begin{align*}
    s |\models a & \quad \text{iff} \quad a \in L(s) \\
    s |\models \neg \Phi & \quad \text{iff} \quad \text{not } s |\models \Phi \\
    s |\models \Phi \land \Psi & \quad \text{iff} \quad (s |\models \Phi) \text{ and } (s |\models \Psi) \\
    s |\models \exists \varphi & \quad \text{iff} \quad \pi |\models \varphi \text{ for some } \pi \in \text{Paths}(s) \\
    s |\models \forall \varphi & \quad \text{iff} \quad \pi |\models \varphi \text{ for all } \pi \in \text{Paths}(s)
\end{align*}
\]

\[
\begin{align*}
    \pi |\models \mathcal{O} \Phi & \quad \text{iff} \quad \pi[1] |\models \Phi \\
    \pi |\models \Phi \mathcal{U} \Psi & \quad \text{iff} \quad \exists j \geq 0. \ (\pi[j] |\models \Psi \land (\forall 0 \leq k < j. \pi[k] |\models \Phi))
\end{align*}
\]
Writing CTL Properties

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

Each red light phase is preceded by a yellow light phase.
\[ \forall \Box (\text{yellow} \lor \forall \Diamond \neg \text{red}) \]

In every reachable state of the system, it is possible to return to a start state of the system.
\[ \forall \Box \exists \Diamond \text{start} \]

Each process has access to the critical section infinitely often.
\[ (\forall \Box \forall \Diamond crit_1) \land (\forall \Box \forall \Diamond crit_2) \]
CTL for LTSs

Figure 6.4: Interpretation of several CTL formulae.