Recall the Mutex example from previous Lectures:

Figure 1: An Example Kripke Structure of a Mutex System

Keijo Heljanko
Some examples of $LTL$ properties in the Mutex example Kripke structure $M$ are:

- $M \models \Box \neg (CR0 \land CR1)$
  (both processes are never in their critical sections at the same time)

- $M \models \Box (TRY0 \Rightarrow \Diamond CR0)$
  (always when process 0 enters the trying section, it is eventually followed by process 0 entering the critical section)

- $M \models \Box (TRY0 \Rightarrow (TRY0 U CR0))$
  (always when process 0 enters the trying section, it stays in the trying section until it enters the critical section)

- $M \not\models \Diamond CR0$
  (the following does not hold: process 0 will eventually enter the critical section)

- $M \not\models \Box \Diamond CR0 \Rightarrow \Box \Diamond CR1$
  (the following does not hold: if process 0 is infinitely often in the critical section, then also process 1 is infinitely often in the critical section)
Some additional examples:

- \( M \models \Box \Diamond CR0 \Rightarrow \Box \Diamond TRY0 \)
  (if process 0 is infinitely often in the critical section, then it is also infinitely often in the trying section)

- \( M \models \Diamond \Box NC0 \Rightarrow \Diamond \Box \neg CR0 \)
  (if process 0 all the time from a certain point onwards is in the non-critical section, then process 0 will all the time from a certain point onwards be not in the critical section)

- \( M \models (\Diamond \Box TRY0 \land \Box \Diamond TRY1) \Rightarrow (\Box \Diamond CR0 \land \Box \Diamond CR1) \)
  (if both process 0 and 1 are infinitely often in the trying section, then both process 0 and 1 are infinitely often also in the critical section)
7.2 Temporal Logic $CTL^*$

The logic $CTL^*$ (called “full branching time logic $CTL^*$”) is a branching temporal logic, which includes all of $LTL$ (with one syntactical difference).

An additional feature over $LTL$ are the path quantifiers, $A$ (“for all paths”) and $E$ (“for some path”). They can be used to express whether the property should hold for all paths starting from a state, or whether the property should hold for at least one path.

Actually, as we will see, any $LTL$ formula $f$ is equivalent to the corresponding $CTL^*$ formula $Af$. 
There are two kinds of formulas in $CTL^*$: **state formulas** and **path formulas**.

A $CTL^*$ state formula $g$ is:

- true, false, or $p$ for $p \in AP$.
- $\neg g_1$ or $g_1 \lor g_2$, where $g_1$ and $g_2$ are $CTL^*$ state formulas.
- $Af_1$, where $f_1$ is a $CTL^*$ path formula.

A $CTL^*$ path formula $f$ is either:

- A $CTL^*$ state formula $g$.
- $\neg f_1, f_1 \lor f_2, Xf_1, f_1 U f_2$, where $f_1$ and $f_2$ are $CTL^*$ path formulas.

Moreover, the top-level formula $g$ is defined to always be a state formula in $CTL^*$. 
As before with $LTL$, we can define a non-minimal version of the syntax which includes $E, \land$, and $f_1 R f_2$:

A $CTL^*$ state formula $g$ is:

- **true, false, or** $p$ for $p \in AP$.
- $\neg g_1$ or $g_1 \lor g_2$, $g_1 \land g_2$, where $g_1$ and $g_2$ are $CTL^*$ state formulas.
- $Ef_1$, where $f_1$ is a $CTL^*$ path formula.
- $Af_1$, where $f_1$ is a $CTL^*$ path formula.

A $CTL^*$ path formula $f$ is either:

- A $CTL^*$ state formula $g$.
- $\neg f_1, f_1 \lor f_2, f_1 \land f_2, X f_1, f_1 U f_2, f_1 R f_2$, where $f_1$ and $f_2$ are $CTL^*$ path formulas.

All DeMorgan rules for $LTL$ are also valid for $CTL^*$, the only new DeMorgan rule is $E g_1 = \neg A \neg g_1$. 

Keijo Heljanko
7.2.1 Semantics of $CTL^*$

We give semantics to $CTL^*$ formulas as follows.

The notation $M, s \models g$ denotes that the $CTL^*$ state formula $g$ holds in state $s$ of the Kripke structure.

The notation $M, \pi \models f$ denotes that the $CTL^*$ path formula $f$ holds in the path $\pi$ (of the Kripke structure $M$).

Now (as also in the case of $LTL$), the notation $M \models g$ is a short hand for $M, s^0 \models g$. (A $CTL^*$ formula holds for a system $M$ if it holds in the initial state $s^0$ of the system.)

The semantics can now be inductively defined as follows.
The semantics of $CTL^*$ state formulas is given by:

- $M, s \models \text{true}$
- $M, s \not\models \text{false}$
- $M, s \models p$ iff $p \in L(s)$ ($p$ is an atomic proposition)
- $M, s \models \neg g$ iff not $M, s \models g$
- $M, s \models g_1 \lor g_2$ iff $M, s \models g_1$ or $M, s \models g_2$
- $M, s \models g_1 \land g_2$ iff $M, s \models g_1$ and $M, s \models g_2$
- $M, s \models Ef_1$ iff there exists a path $\pi$ starting from $s$ such that $M, \pi \models f_1$
- $M, s \models Af_1$ iff for all paths $\pi$ starting from $s$ it holds that $M, \pi \models f_1$
The semantics of $CTL^*$ path formulas follows closely the definition for $LTL$ formulas, only the first item is new.

- $M, \pi \models g_1$ iff $s$ is the first state of $\pi$ and $M, s \models g_1$ ($g_1$ is a state formula)
- $M, \pi \models \neg f_1$ iff not $M, \pi \models f_1$
- $M, \pi \models f_1 \lor f_2$ iff $M, \pi \models f_1$ or $M, \pi \models f_2$
- $M, \pi \models f_1 \land f_2$ iff $M, \pi \models f_1$ and $M, \pi \models f_2$
- $M, \pi \models X f_1$ iff $M, \pi^1 \models f_1$
- $M, \pi \models f_1 U f_2$ iff there exists $j \geq 0$, such that $M, \pi^j \models f_2$ and for all $0 \leq i < j$, $M, \pi^i \models f_1$
- $M, \pi \models f_1 R f_2$ iff for all $j \geq 0$, if for every $0 \leq i < j$ $M, \pi^i \not\models f_1$ then $M, \pi^j \models f_2$
Some examples of properties which cannot be expressed in $LTL$, but can be expressed in $CTL^*$ are:

- $AG(\neg Restart)$
  (From all the states of the system it is possible to reach the restart state.)

- $AG(\neg (EG \neg Restart))$
  (From all the states of the system it is possible to execute an infinite execution without going through the restart state.)

- $AF(AXa)$
  (In all executions of the system from the initial state eventually a state will be reached, such that in all its possible successors $a$ holds.)
7.3 Temporal Logic $CTL$

The temporal logic $CTL$ is the subset of $CTL^*$, where the temporal operators $X$ and $U$ are always immediately preceded by a path quantifier.

An example $CTL^*$ formula not expressible in $CTL$ is $A(FGp) \lor AG(EFp)$.

Often $CTL$ formulas are actually written in the following syntax: $EX(f_1), AX(f_1), EF(f_1), AF(f_1), EG(f_1), AG(f_1), EU(f_1, f_2)$, and $AU(f_1, f_2)$, where the last two are different in the usual syntax, namely $E(f_1 U f_2)$ and $A(f_1 U f_2)$.

The temporal logic $CTL$ is interesting, because it has a very efficient model checking algorithm.

Some examples of properies expressable in $CTL$ follow. In the figures black color denotes the fact that $f_2$ holds, and gray color the fact that $f_1$ holds. In white nodes neither of the two subformulas hold.
Figure 2: $M, s_0 \models A(f_1 U f_2)$. 
Figure 3: $M, s_0 \models E(f_1 U f_2)$. 
Figure 4: $M, s_0 \models EG(f_1)$
7.4 Complexity of Model Checking

Let $AP$ be fixed, $|M|$ be the size of the Kripke structure ($|S| + |R|$), and $|f|$ the size of the formula (number of subformulas of $f$).

The following time complexity model checking algorithms are used in practise:

- $CTL$ model checking can be done in time $O(|M| \cdot |f|)$
- $LTL$ and $CTL^*$ model checking can be done in time $O(|M| \cdot 2^{O(|f|)})$
The claim used by LTL proponents is that often formulas to be considered are small, and thus it makes sense to talk about the case when the formula $f$ is fixed:

- Model checking a fixed formula $f$ of CTL, LTL, or CTL* is NLOGSPACE-complete in $|M|$.
- Alternatively, it can be done in time $O(|M|)$ (by using $|M|$ space).

Suppose our system is composed out of automata synchronization, i.e., $M$ is actually the set of reachable states of an automaton $A = A_1 \times A_2 \times \cdots \times A_n$. Now define the program size as $|A| = |A_1| + |A_2| + \cdots + |A_n|$.

The following bound is also known:

- Model checking a fixed formula $f$ of CTL, LTL, or CTL* is PSPACE-complete in $|A|$.
- Alternatively, it can be done in time $2^{O(|A|)}$ (by using $2^{O(|A|)} = O(|M|)$ space).