We now define CTL (Computation-Tree Logic) as a syntactic restriction of CTL*.

Operators are restricted to the following form:

- $\mathcal{E}$: there exists an execution
- $\mathcal{A}$: for all executions
- $\mathcal{Q}$: next
- $\mathcal{T}$: finally
- $\mathcal{X}$: globally
- $\mathcal{U}$: until

(and possibly others)
We define a minimal syntax first. Later we define additional operators with the help of the minimal syntax.

Let $AP$ be a set of atomic propositions: The set of CTL formulas over $AP$ is as follows:

- if $a \in AP$, then $a$ is a CTL formula;
- if $\phi_1, \phi_2$ are CTL formulas, then so are $\neg \phi_1$, $\phi_1 \lor \phi_2$, $EX \phi_1$, $EG \phi_1$, $\phi_1 EU \phi_2$
It is easy to see that every CTL formula is also a CTL* formula.

Previously, we defined the satisfaction relationship between valuation trees and CTL* formulae. Since each state of a Kripke structure has a clearly defined computation tree, we may just as well say that a state satisfies a CTL/CTL* formula, meaning that its computation tree does.

Let $\mathcal{K}$ be a Kripke structure, let $s$ one of its states, and let $\phi$ be a CTL formula. On the following slide, we define a set $\llbracket \phi \rrbracket_{\mathcal{K}}$ in such a way that $s \in \llbracket \phi \rrbracket_{\mathcal{K}}$ iff $T_{\mathcal{K}}(s) \models \phi$. 
Let $\mathcal{K} = (S, \rightarrow, r, AP, \nu)$ be a Kripke structure.

We define the semantic of every CTL formula $\phi$ over $AP$ w.r.t. $\mathcal{K}$ as a set of states $\llbracket \phi \rrbracket_\mathcal{K}$, as follows:

$$\llbracket a \rrbracket_\mathcal{K} = \{ s \mid a \in \nu(s) \} \quad \text{for } a \in AP$$

$$\llbracket \neg \phi_1 \rrbracket_\mathcal{K} = S \setminus \llbracket \phi_1 \rrbracket_\mathcal{K}$$

$$\llbracket \phi_1 \lor \phi_2 \rrbracket_\mathcal{K} = \llbracket \phi_1 \rrbracket_\mathcal{K} \cup \llbracket \phi_2 \rrbracket_\mathcal{K}$$

$$\llbracket EX \phi_1 \rrbracket_\mathcal{K} = \{ s \mid \text{there is a } t \text{ s.t. } s \rightarrow t \text{ and } t \in \llbracket \phi_1 \rrbracket_\mathcal{K} \}$$

$$\llbracket EG \phi_1 \rrbracket_\mathcal{K} = \{ s \mid \text{there is a run } \rho \text{ with } \rho(0) = s \text{ and } \rho(i) \in \llbracket \phi_1 \rrbracket_\mathcal{K} \text{ for all } i \geq 0 \}$$

$$\llbracket \phi_1 EU \phi_2 \rrbracket_\mathcal{K} = \{ s \mid \text{there is a run } \rho \text{ with } \rho(0) = s \text{ and } k \geq 0 \text{ s.t.} \rho(i) \in \llbracket \phi_1 \rrbracket_\mathcal{K} \text{ for all } i < k \text{ and } \rho(k) \in \llbracket \phi_2 \rrbracket_\mathcal{K} \}$$
We say that $\mathcal{K}$ satisfies $\phi$ (denoted $\mathcal{K} \models \phi$) iff $r \in \llbracket \phi \rrbracket_{\mathcal{K}}$.

The **local model-checking problem** is to check whether $\mathcal{K} \models \phi$.

The **global model-checking problem** is to compute $\llbracket \phi \rrbracket_{\mathcal{K}}$.

We declare two formulas equivalent (written $\phi_1 \equiv \phi_2$) iff for every Kripke structure $\mathcal{K}$ we have $\llbracket \phi_1 \rrbracket_{\mathcal{K}} = \llbracket \phi_2 \rrbracket_{\mathcal{K}}$.

In the following, we omit the index $\mathcal{K}$ from $\llbracket \cdot \rrbracket_{\mathcal{K}}$ if $\mathcal{K}$ is understood.
CTL: Extended syntax

\[ \phi_1 \land \phi_2 \equiv \neg (\neg \phi_1 \lor \neg \phi_2) \]
\[ \text{true} \equiv a \lor \neg a \]
\[ \text{false} \equiv \neg \text{true} \]
\[ \phi_1 \text{ EW } \phi_2 \equiv \text{ EG } \phi_1 \lor (\phi_1 \text{ EU } \phi_2) \]
\[ \text{EF } \phi \equiv \text{ true EU } \phi \]
\[ \text{AX } \phi \equiv \neg \text{ EX } \neg \phi \]
\[ \text{AG } \phi \equiv \neg \text{ EF } \neg \phi \]
\[ \text{AF } \phi \equiv \neg \text{ EG } \neg \phi \]
\[ \phi_1 \text{ AW } \phi_2 \equiv \neg (\neg \phi_2 \text{ EU } \neg (\phi_1 \lor \phi_2)) \]
\[ \phi_1 \text{ AU } \phi_2 \equiv \text{ AF } \phi_2 \land (\phi_1 \text{ AW } \phi_2) \]

Other logical and temporal operators (e.g. \( \rightarrow \), \( \text{ER} \), \( \text{AR} \)), \ldots may also be defined.
We use the following computation tree as a running example (with varying distributions of red and black states):

In the following slides, the topmost state satisfies the given formula if the black states satisfy $p$ and the red states satisfy $q$. 
AF p
AX p
EG p
EX p
p \text{ EU } q
Solving nested formulas: Is $s_0 \in \llbracket \text{AF AG } x \rrbracket$?

To compute the semantics of formulas with nested operators, we first compute the states satisfying the innermost formulas; then we use those results to solve progressively more complex formulas.

In this example, we compute $\llbracket x \rrbracket$, $\llbracket \text{AG } x \rrbracket$, and $\llbracket \text{AF AG } x \rrbracket$, in that order.
Bottom-up method (1): Compute $[[x]]$
Bottom-up method (2): Compute $[[AG \ x]]$
Bottom-up method (3): Compute $[[\text{AF AG } x]]$
Five philosophers are sitting around a table, taking turns at thinking and eating.

We shall express a couple of properties in CTL. Let us assume the following atomic propositions:

$$e_i \iff \text{philosopher } i \text{ is currently eating}$$
“Philosophers 1 and 4 will never eat at the same time.”
Properties of the Dining Philosophers

“Philosophers 1 and 4 will never eat at the same time.”

\[ AG \neg(e_1 \land e_4) \]

“It is possible that Philosopher 3 never eats.”
Properties of the Dining Philosophers

“Philosophers 1 and 4 will never eat at the same time.”

\[ \text{AG} \neg (e_1 \land e_4) \]

“It is possible that Philosopher 3 never eats.”

\[ \text{EG} \neg e_3 \]

“From every situation on the table it is possible to reach a state where only philosopher 2 is eating.”
Properties of the Dining Philosophers

“Philosophers 1 and 4 will never eat at the same time.”

\[ \text{AG} \neg(e_1 \land e_4) \]

“It is possible that Philosopher 3 never eats.”

\[ \text{EG} \neg e_3 \]

“From every situation on the table it is possible to reach a state where only philosopher 2 is eating.”

\[ \text{AG} \text{ EF}(\neg e_1 \land e_2 \land \neg e_3 \land \neg e_4) \]
Part 10: Algorithms for CTL
In the following, let $\mathcal{K} = (S, \rightarrow, r, AP, \nu)$ be a Kripke structure (where $S$ is finite) and $\phi$ a CTL formula over $AP$.

We shall solve the global model-checking problem for CTL, i.e. to compute $\llbracket \phi \rrbracket_\mathcal{K}$ (all states of $\mathcal{K}$ whose computation tree satisfies $\phi$).

Our solution works “bottom-up”, i.e. it considers simple subformulae first, and then successively more complex ones.

The solution shown here considers only the minimal syntax. For additional efficiency one could extend it by treating some cases of the extended syntax more directly.
The ‘bottom-up’ algorithm for CTL

The algorithm reduces $\phi$ step by step to a single atomic proposition. Reminder: $[p]_K = \{ s \mid p \in \nu(s) \}$ for $p \in AP$. In the following, we abbreviate this set as $\mu(p)$.

1. Check whether $\phi = p$, where $p \in AP$. If yes, output $\mu(p)$ and stop.

2. Otherwise, $\phi$ contains some subformula $\psi$ of the form $\neg p$, $p \lor q$, $\text{EX } p$, $\text{EG } p$, or $p \text{ EU } q$, where $p, q \in AP$. Compute $[\psi]_K$ using the algorithms on the following slides.

3. Let $p' \notin AP$ be a “fresh” atomic proposition. Add $p'$ to $AP$ and set $\mu(p') := [\psi]_K$. Replace all occurrences of $\psi$ in $\phi$ by $p'$ and continue at step 1.
Computation of $\llbracket \psi \rrbracket_K$: simple cases

Case 1: $\psi \equiv \neg p$, $p \in AP$

By definition, $\llbracket \psi \rrbracket_K = S \setminus \mu(p)$.

Case 2: $\psi \equiv p \lor q$, $p, q \in AP$

Then $\llbracket \psi \rrbracket_K = \mu(p) \cup \mu(q)$.

Case 3: $\psi \equiv \text{EX} p$, $p \in AP$

In the following, let $\text{pre}(X)$, for $X \subseteq S$, denote the set

$$\text{pre}(X) := \{ s \mid \exists t \in X: s \rightarrow t \}.$$ 

Then by definition $\llbracket \psi \rrbracket_K = \text{pre}(\mu(p))$. 
We shall first define \textbf{EU} and \textbf{EG} in terms of fixed points.

\textbf{EU} is characterized by a \textit{smallest} fixed point: We first assume that no state satisfies the EU formula and then, one by one, identify those that do satisfy it after all.

By contrast, \textbf{EG} can be characterized by a \textit{largest} fixed point: We first assume that all states satisfy a given EG formula and then, one by one, eliminate those that do not.

Based on this, we then derive algorithms for EG and EU.
Computation of $[[\psi]]_\mathcal{K}$: EG

Case 4: $\psi \equiv \text{EG } p, \quad p \in AP$

Lemma 1: $[[\text{EG } p]]_\mathcal{K}$ is the largest solution (w.r.t. $\subseteq$) of the equation

$$X = \mu(p) \cap \text{pre}(X).$$

Proof: We proceed in two steps:

1. We show that $[[\text{EG } p]]_\mathcal{K}$ is indeed a solution of the equation, i.e.

$$[[\text{EG } p]]_\mathcal{K} = \mu(p) \cap \text{pre}([[\text{EG } p]]_\mathcal{K}).$$

Reminder: $[[\text{EG } p]]_\mathcal{K} = \{ s \mid \exists \rho: \rho(0) = s \land \forall i \geq 0: \rho(i) \in \mu(p) \}$.

$\Rightarrow$ Let $s \in [[\text{EG } p]]_\mathcal{K}$ and $\rho$ a “witness” path. Then obviously $s \in \mu(p)$. Moreover, $\rho(1) \in [[\text{EG } p]]_\mathcal{K}$ (because of $\rho^1$), hence $s \in \text{pre}([[\text{EG } p]]_\mathcal{K})$. 

317
Continuation of the proof of Lemma 1:

1. “⇐” Let $s \in \mu(p) \cap \text{pre}([\text{EG } p]_\mathcal{K})$. Then $s$ has a direct successor $t$, where a path $\rho$ starts proving that $t \in [\text{EG } p]_\mathcal{K}$. Thus, $s \rho$ is a path witnessing that $s \in [\text{EG } p]_\mathcal{K}$.

2. We show that $[\text{EG } p]_\mathcal{K}$ is indeed the largest solution, i.e., if $M$ is a solution of the equation, then $M \subseteq [\text{EG } p]_\mathcal{K}$.

Let $M \subseteq S$ be a solution of the equation, i.e. $M = \mu(p) \cap \text{pre}(M)$, and let $s \in M$. We shall show $s \in [\text{EG } p]_\mathcal{K}$.

– Since $s \in M$, we have $s \in \mu(p)$ and $s \in \text{pre}(M)$.
– Since $s \in \text{pre}(M)$, there exist $s_1 \in M$ with $s \to s_1$.
– Repeating this argument, we can construct an infinite path $\rho = ss_1 \cdots$ in which all states are contained in $\mu(p)$. Therefore, $s \in [\text{EG } p]_\mathcal{K}$. 
Lemma 2: Consider the sequence $S, \pi(S), \pi(\pi(S)), \ldots$, i.e. $(\pi^i(S))_{i \geq 0}$,
where $\pi(X) := \mu(p) \cap \text{pre}(X)$.
For all $i \geq 0$ we have $\pi^i(S) \supseteq \boxed{[\text{EG } p]}_K$.

We state the following two facts:

(1) $\pi$ is monotone: if $X \supseteq X'$, then $\pi(X) \supseteq \pi(X')$.
(2) The sequence is descending: $S \supseteq \pi(S) \supseteq \pi(\pi(S)) \ldots$ (follows from (1)).

Proof of Lemma 2: (induction over $i$)
Base: $i = 0$: obvious.
Step: $i \rightarrow i + 1$:

$$\pi^{i+1}(S) = \mu(p) \cap \text{pre}(\pi^i(S))$$
$$\supseteq \mu(p) \cap \text{pre}(\boxed{[\text{EG } \varphi]}_K) \quad \text{(i.h. and monotonicity)}$$
$$= \boxed{[\text{EG } p]}_K$$
Lemma 3: There exists an index $i$ such that $\pi^i(S) = \pi^{i+1}(S)$, and $[\text{EG } p]_K^C = \pi^i(S)$.

Proof: Since $S$ is finite, the descending sequence must reach a fixed point, say after $i$ steps. Then we have $\pi^i(S) = \pi(\pi^i(S)) = \mu(p) \cap \text{pre}(\pi^i(S))$. Therefore, $\pi^i(S)$ is a solution of the equation from Lemma (1).

Because of Lemma 1, we have $\pi^i(S) \subseteq [\text{EG } p]_K^C$. Because of Lemma 2, we have $\pi^i(S) \supseteq [\text{EG } p]_K^C$. 
An algorithm for $\text{EG}$

Lemma 3 gives us a strategy for computing $[\text{EG } p]_\mathcal{K}$: compute the sequence $S, \pi(S), \cdots$ until a fixed point is reached.

For practicality, one would start immediately with $X := \mu(p)$. Then, in each round, one eliminates those states having no successors in $X$.

This can be efficiently implemented in $O(|\mathcal{K}|)$ time (“reference counting”).
Example: Computation of $\lbrack E G y \rbrack_K$ (1/4)
Example: Computation of $[E_G y]_K (2/4)$

$\pi^1(S) = \mu(y) \cap pre(S)$
Example: Computation of $\lceil E G y \rceil_\mathcal{K}$ (3/4)

\[ \pi^2(S) = \mu(y) \cap pre(\pi^1(S)) \]
Example: Computation of $\mathbb{K}^{[\text{EG} \ y]}$ (4/4)

$$\pi^3(S) = \mu(y) \cap \text{pre}(\pi^2(S)) = \pi^2(S): \quad \mathbb{K}^{[\text{EG} \ y]} = \{s_0, s_2, s_4\}$$
Computation of EU

Case 5: $\psi \equiv p \text{EU} q$, $p, q \in AP$

Analogous to EG (proofs omitted):

Lemma 4: $[p \text{EU} q]_K$ is the smallest solution (w.r.t. $\subseteq$) of the equation

$$X = \mu(q) \cup (\mu(p) \cap \text{pre}(X)).$$

Lemma 5: $[p \text{EU} q]_K$ is the fixed point of the sequence

$$\emptyset, \xi(\emptyset), \xi(\xi(\emptyset)), \ldots$$

where

$$\xi(X) := \mu(q) \cup (\mu(p) \cap \text{pre}(X)).$$
An algorithm for \( \text{EU} \)

Lemma 5 proposes a strategy: Compute the sequence \( \emptyset, \xi(\emptyset), \cdots \) until a fixed point is reached.

In practice one would start with \( X := \mu(q) \). Then, in each step, one can add those direct predecessors that are in \( \mu(p) \).

Can be done efficiently in \( \mathcal{O}(|\mathcal{K}|) \) time (multiple backwards DFS).
Example: Computation of $[[z \textbf{EU} y]]_{\mathcal{K}}$ (1/4)

$\xi^0(\emptyset) = \emptyset$
Example: Computation of $[[z \mathbf{EU} y]]_{\mathcal{K}}$ (2/4)

\[
\xi^1(\emptyset) = \mu(y) \cup (\mu(z) \cap \text{pre}(\xi^0(\emptyset)))
\]
Example: Computation of $\left[ z \mathbf{E} U y \right]_{\mathcal{K}} (3/4)$

\[
\xi^2(\emptyset) = \mu(y) \cup (\mu(z) \cap \text{pre}(\xi^1(\emptyset)))
\]
Example: Computation of $[[z \text{ EU } y]]_{\mathcal{K}}$ (4/4)

$$\xi^3(\emptyset) = \mu(y) \cup (\mu(z) \cap \text{pre}(\xi^2(\emptyset))) = \xi^2(\emptyset)$$

$$[[z \text{ EU } y]]_{\mathcal{K}} = \{s_0, s_1, s_2, s_4, s_5, s_6\}$$