16.1 CTL*

CTL* stands for Computation Tree Logic. This is sometimes called “branching-time logic” as opposed to LTL which considers all possible linear paths from some initial state.

We will see that LTL and CTL are proper subsets of CTL*.

In CTL*, we have both path formulas and state formulas.

16.2 Syntax and Semantics of CTL*

Syntax of State Formulas:

**base case:** If $p \in AP$, then $p$ is a state formula.

**inductive cases:** if $\alpha, \beta$ are state formulas and $\varphi$ is a path formula, then the following are state formulas:
- $\neg \alpha$,
- $\alpha \lor \beta$,
- $E \varphi$,
- $A \varphi$

Syntax of Path Formulas:

If $\alpha$ is a state formula and $\varphi$ and $\psi$ are path formulas, then the following are path formulas:
- $\alpha$,
- $\neg \varphi$,
- $(\varphi \lor \psi)$,
- $X \varphi$,
- $F \varphi$,
- $G \varphi$,
- $(\varphi U \psi)$

Semantics of State Formulas:

$$(T, s) \models p \iff p \in L(s)$$

$$(T, s) \models \neg \alpha \iff (T, s) \not\models \alpha$$

$$(T, s) \models (\alpha \lor \beta) \iff (T, s) \models \alpha \text{ or } (T, s) \models \beta$$

$$(T, s) \models E \varphi \iff \text{there exists path } \pi, \pi[0] = s, \ (T, \pi) \models \varphi$$

$$(T, s) \models A \varphi \iff \text{for all } \pi \text{ such that } \pi[0] = s, \ (T, \pi) \models \varphi$$

For $\alpha$ a state formula, $$(T, \pi) \models \alpha \iff (T, \pi[0]) \models \alpha$$

Semantics of Path Formula : (same as in LTL)

$$(T, \pi) \models \neg \alpha \iff (T, \pi) \not\models \alpha$$

$$(T, \pi) \models (\alpha \lor \beta) \iff (T, \pi) \models \alpha \text{ or } (T, \pi) \models \beta$$

$$(T, \pi) \models X \alpha \iff \pi^1 \models \alpha$$

$$(T, \pi) \models G \alpha \iff \forall i \geq 0 \ (T, \pi^i) \models \alpha$$

$$(T, \pi) \models F \alpha \iff \exists i \geq 0 \ (T, \pi^i) \models \alpha$$

$$(T, \pi) \models (\alpha U \beta) \iff \exists i \geq 0 \ ((T, \pi^i) \models \beta \land \forall j \leq i \ (T, \pi^j) \models \alpha)$$
Some Temporal Logic Equivalence:

\[
\begin{align*}
F \varphi & \equiv \neg G \neg \varphi \\
F \varphi & \equiv T U \varphi \\
A \varphi & \equiv \neg E \neg \varphi \\
E \varphi & \equiv \neg A \neg \varphi \\
A X \varphi & \equiv \neg E X \neg \varphi \\
A G \varphi & \equiv \neg E F \neg \varphi
\end{align*}
\]

16.3 CTL

Emerson and Clarke defined CTL as the following subset of the state formulas of CTL*:

**Syntax of CTL:**

**base case:** If \( p \in AP \), then \( p \) is a CTL formula.

**inductive cases:** if \( \alpha, \beta \) are CTL formulas, then so are:

- \( \neg \alpha \)
- \( \alpha \lor \beta \)
- \( E X \alpha \)
- \( E F \alpha \)
- \( E (\alpha U \beta) \)
- \( A X \alpha \)
- \( A F \alpha \)
- \( A G \alpha \)
- \( A (\alpha U \beta) \)

Thus, CTL formulas are formed by pairing path quantifiers: \( E, A \), with temporal operators: \( X, F, G, U \) in all possible ways.

**Theorem 16.1** (Emerson & Clarke) There is an algorithm which given a transition system \( T = (S, R, L) \) and a CTL formula \( \varphi \) marks the states \( s \in S \) such that \( (T, s) \models \varphi \) and takes time \( O(|T| \cdot |\varphi|) \)

**Proof:** \( T \) is a graph with \( n = |S| \) vertices and \( m = |R| \) edges. The number of subformulas of \( \varphi \) is less than \( |\varphi| \). We now show that for each subformula \( \gamma \) of \( \varphi \), we can recursively label all the states that satisfy \( \gamma \), in time \( O(n + m) \).

**base case:** \( \gamma \in AP \): \( L \) already gives the labeling.

- \( \neg \alpha \): Label a state \( \neg \alpha \) if it is not labeled \( \alpha \). Time: \( O(n) \).
- \( \alpha \lor \beta \): Label a state \( \alpha \lor \beta \) if it is labeled \( \alpha \), or \( \beta \). Time: \( O(n) \).
- \( E X \alpha \): For each state, \( s \), go through its adjacency list and if any of \( s \)'s successors is labeled \( \alpha \), then label \( s \), \( E X \alpha \).
- \( E (\alpha U \beta) \): Make a copy of the graph and delete all edges that satisfy neither \( \alpha \) nor \( \beta \). Now label each remaining state \( E (\alpha U \beta) \) if it is reachable backwards from a state marked \( \beta \). We can compute this by reversing the direction of the edges and doing a DFS, starting from all vertices labeled \( \beta \). Time: \( O(n + m) \).
- \( E G \alpha \): We want to label all states that have an infinite path all of whose states are labeled \( \alpha \). First make a copy, \( A \), of the graph in which we have deleted all the vertices not labelled \( \alpha \). A subgraph, \( C \), of a graph is called a strongly connected component (SCC) if for every two vertices \( a, b \in C \), there is a path from \( a \) to \( b \). An SCC is called non-trivial, if it has at least one edge. (Trivial SCC’s consist of single vertices without self-loops.) You should know from your Algorithms Course, that using DFS, we can compute all the SCC’s in time \( O(n + m) \).

So, compute all the non-trivial SCC’s in \( A \). Now we should label a vertex \( E G \alpha \) if it is reachable in the reverse graph from a non-trivial SCC. We can compute this in time \( O(n + m) \) by doing a DFS of the reverse graph of \( A \), starting at all vertices in a non-trivial SCC. \( \square \)
Some examples:

In the graph, $\mathcal{T}$, below we have $(\mathcal{T}, 2) \models \text{AF}_q$ and $(\mathcal{T}, 2) \models \text{AG}_q$.

\begin{align*}
(\mathcal{T}, s) \models \text{EF}_p & \iff \text{there is some path from } s \text{ to a state which satisfies } p. \\
(\mathcal{T}, s) \models \text{EG}_p & \iff \text{there is some path from } s \text{ along which } p \text{ always holds.} \\
(\mathcal{T}, s) \models \text{AG}(p \rightarrow \text{EX}_q) & \iff \text{Whenever } p \text{ holds along a path from } s, \text{ } q \text{ holds at some next state.}
\end{align*}

$\text{AG}(G_r \rightarrow F_c) = \text{weak fairness (expressible in } CTL\text{), } \text{“Always trying implies eventually succeeding.”}$  

$\text{A(GF}_r \rightarrow GF_c) = \text{strong fairness (not expressible in } CTL\text{, expressible in } CTL^*\text{), } \text{“Infinitely often trying implies infinitely often succeeding.”}$

The running time for model checking $LTL$ is $O(|\mathcal{T}|^2|\varphi|)$. We are not going to do this proof, but the intuitive idea is that we can represent paths via the subset of the subformulas of $\varphi$ that they satisfy.