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 α, β are state formulas and φ is a path formula, then the following are state formulas:

$\neg\alpha, \quad \alpha \vee \beta, \quad \mathbf{E}\varphi, \quad \mathbf{A}\varphi$

Syntax of Path Formulas:

If α is a state formula and φ and ψ are path formulas, then the following are path formulas:

$\alpha, \quad \neg\varphi, \quad (\varphi \vee \psi), \quad \mathbf{X}\varphi, \quad \mathbf{F}\varphi, \quad \mathbf{G}\varphi, \quad (\varphi\mathbf{U}\psi)$

Semantics of State Formulas:

$(\mathcal{T}, s) \models p \quad \Leftrightarrow \quad p \in L(s)$

$(\mathcal{T}, s) \models \neg\alpha \quad \Leftrightarrow \quad (\mathcal{T}, s) \not\models \alpha$

$(\mathcal{T}, s) \models (\alpha \vee \beta) \quad \Leftrightarrow \quad (\mathcal{T}, s) \models \alpha$ or $(\mathcal{T}, s) \models \beta$

$(\mathcal{T}, s) \models \mathbf{E}\varphi \quad \Leftrightarrow$ there exists path π, $\pi[0] = s, \quad (\mathcal{T}, \pi) \models \varphi$

$(\mathcal{T}, s) \vDash \mathbf{A}\varphi \quad \Leftrightarrow$ for all π such that $\pi[0] = s, \quad (\mathcal{T}, \pi) \vDash \varphi$

For α a state formula, $\quad (\mathcal{T}, \pi) \vDash \alpha \quad \Leftrightarrow \quad (\mathcal{T}, \pi[0]) \vDash \alpha$

Semantics of Path Formula : (same as in LTL)

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

$(\mathcal{T}, \pi) \models (\alpha \vee \beta)$ iff $(\mathcal{T}, \pi) \models \alpha$ or $(\mathcal{T}, \pi) \models \beta$

$(\mathcal{T}, \pi) \models \mathbf{X}\alpha$ iff $\pi^1 \models \alpha$

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

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

$(\mathcal{T}, \pi) \models (\alpha\mathbf{U}\beta)$ iff $\exists i \geq 0\ ((\mathcal{T}, \pi^i) \models \beta\ \wedge\ \forall j < i\ (\mathcal{T}, \pi^j) \models \alpha)$

Some Temporal Logic Equivalence:

$$\begin{aligned}
\mathbf{F}\varphi &\equiv \neg\mathbf{G}\neg\varphi \\
\mathbf{F}\varphi &\equiv \top\mathbf{U}\varphi \\
\mathbf{A}\varphi &\equiv \neg\mathbf{E}\neg\varphi \\
\mathbf{E}\varphi &\equiv \neg\mathbf{A}\neg\varphi \\
\mathbf{AX}\varphi &\equiv \neg\mathbf{EX}\neg\varphi \\
\mathbf{AG}\varphi &\equiv \neg\mathbf{EF}\neg\varphi
\end{aligned}$$

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 α, β are CTL formulas, then so are:

$\neg\alpha, \quad \alpha \vee \beta, \quad \mathbf{EX}\alpha, \quad \mathbf{EF}\alpha, \quad \mathbf{EG}\alpha, \quad \mathbf{E}(\alpha\mathbf{U}\beta), \quad \mathbf{AX}\alpha, \quad \mathbf{AF}\alpha, \quad \mathbf{AG}\alpha, \quad \mathbf{A}(\alpha\mathbf{U}\beta)$

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

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

Proof: $\mathcal{T}$ is a graph with $n = |S|$ vertices and $m = |R|$ edges. The number of subformulas of φ is less than $|\varphi|$. We now show that for each subformula γ of φ, we can recursively label all the states that satisfy γ, 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 α. Time: $O(n)$.

$\alpha \vee \beta$: Label a state $\alpha \vee \beta$ if it is labeled α, or β. Time: $O(n)$.

$\mathbf{EX}\alpha$: For each state, s, go through its adjacency list and if any of s's succesors is labeled α, then label s, $\mathbf{EX}\alpha$.

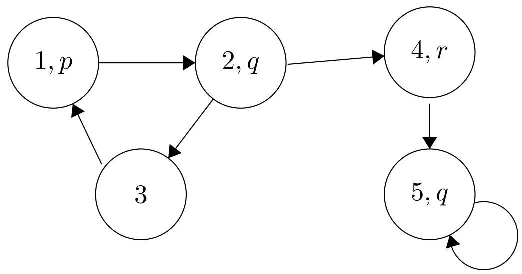
$\mathbf{E}(\alpha\mathbf{U}\beta)$: Make a copy of the graph and delete all edges that satisfy neither α nor β. Now label each remaining state $\mathbf{E}(\alpha\mathbf{U}\beta)$ if it is reachable backwards from a state marked β. We can compute this by reversing the direction of the edges and doing a DFS, starting from all vertices labeled β. Time: $O(n + m)$.

$\mathbf{EG}\alpha$: We want to label all states that have an infinite path all of whose states are labled α. First make a copy, A, of the graph in which we have deleted all the vertices not labelled α. 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 a 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 $\mathbf{EG}\alpha$ 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 $(\mathcal{T}, 2) \models \mathbf{AF}q$ and $(\mathcal{T}, 2) \models \mathbf{AGF}q$.



$(\mathcal{T}, s) \models \mathbf{EF}p \quad \Leftrightarrow \quad$ there is some path from s to a state which satisfies p.
$(\mathcal{T}, s) \models \mathbf{EG}p \quad \Leftrightarrow \quad$ there is some path from s along which p always holds.
$(\mathcal{T}, s) \models \mathbf{AG}(p \to \mathbf{EX}q) \quad \Leftrightarrow \quad$ Whenever p holds along a path from s, q holds at some next state.

$\mathbf{AG}(\mathbf{G}r \to \mathbf{F}c) =$ weak fairness (expressible in CTL), "Always trying implies eventually succeeding."
$\mathbf{A}(\mathbf{GF}r \to \mathbf{GF}c)$ = strong fairness (not expressible in CTL, expressible in CTL^*), "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 φ that they satisfy.