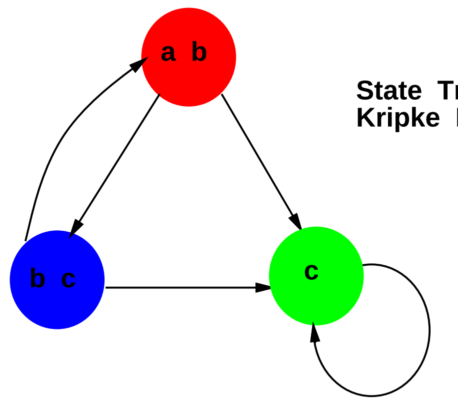
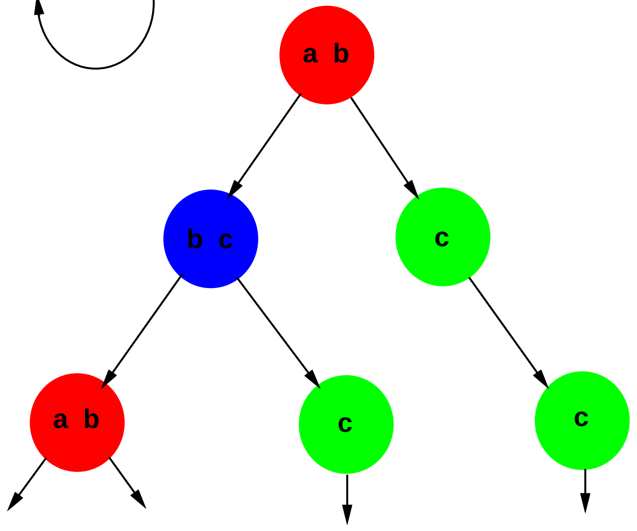
Lecture 0: Computation Tree Logics

- Model of Computation
- Computation Tree Logics
- The Logic CTL*
- Path Formulas and State Formulas
- CTL and LTL
- Expressive Power of Logics

Model of Computation



State Transition Graph or
Kripke Model



Infinite Computation Tree

(Unwind State Graph to obtain Infinite Tree)

Model of Computation (Cont.)

Formally, a Kripke structure is a triple $M = \langle S, R, L \rangle$, where

- S is the set of states,
- $R \subseteq S \times S$ is the transition relation, and
- $L : S \rightarrow \mathcal{P}(AP)$ gives the set of atomic propositions true in each state.

We assume that R is total (i.e., for all states $s \in S$ there exists a state $s' \in S$ such that $(s, s') \in R$).

A path in M is an infinite sequence of states, $\pi = s_0, s_1, \ldots$ such that for $i \geq 0$, $(s_i, s_{i+1}) \in R$.

We write π^i to denote the suffix of π starting at s_i.

Unless otherwise stated, all of our results apply only to finite Kripke structures.

Computation Tree Logics

Temporal logics may differ according to how they handle branching in the underlying computation tree.

In a linear temporal logic, operators are provided for describing events along a single computation path.

In a branching-time logic the temporal operators quantify over the paths that are possible from a given state.

The Logic CTL*

The computation tree logic CTL* combines both branching-time and linear-time operators.

In this logic a path quantifier can prefix an assertion composed of arbitrary combinations of the usual linear-time operators.

1. Path quantifier:
   - **A**—"for every path"
   - **E**—"there exists a path"
2. Linear-time operators:
   - $\mathbf{X}p$—p holds next time.
   - $\mathbf{F}p$—p holds sometime in the future
   - $\mathbf{G}p$—p holds globally in the future
   - $p\mathbf{U}q$—p holds until q holds

Path Formulas and State Formulas

The syntax of state formulas is given by the following rules:

- If $p \in AP$, then p is a state formula.
- If f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
- If f is a path formula, then $\mathbf{E}(f)$ is a state formula.

Two additional rules are needed to specify the syntax of path formulas:

- If f is a state formula, then f is also a path formula.
- If f and g are path formulas, then $\neg f$, $f \vee g$, $\mathbf{X}\, f$, and $(f \;\mathbf{U}\; g)$ are path formulas.

State Formulas (Cont.)

If f is a state formula, the notation $M, s \models f$ means that f holds at state s in the Kripke structure M.

Assume f_1 and f_2 are state formulas and g is a path formula. The relation $M, s \models f$ is defined inductively as follows:

1. $s \models p \quad \Leftrightarrow p \in L(s)$.
2. $s \models \neg f_1 \quad \Leftrightarrow s \not\models f_1$.
3. $s \models f_1 \vee f_2 \Leftrightarrow s \models f_1$ or $s \models f_2$.
4. $s \models \mathbf{E}(g) \quad \Leftrightarrow$ there exists a path π starting with s such that $\pi \models g$.

Path Formulas (Cont.)

If f is a path formula, $M, \pi \models f$ means that f holds along path π in Kripke structure M.

Assume g_1 and g_2 are path formulas and f is a state formula. The relation $M, \pi \models f$ is defined inductively as follows:

1. $\pi \models f \quad \Leftrightarrow$ s is the first state of π and $s \models f$.
2. $\pi \models \neg g_1 \quad \Leftrightarrow \pi \not\models g_1$.
3. $\pi \models g_1 \vee g_2 \quad \Leftrightarrow \pi \models g_1$ or $\pi \models g_2$.
4. $\pi \models \mathbf{X}\, g_1 \quad \Leftrightarrow \pi^1 \models g_1$.
5. $\pi \models (g_1 \mathbf{U}\, g_2) \Leftrightarrow$ there exists a $k \geq 0$ such that $\pi^k \models g_2$ and for $0 \leq j < k$, $\pi^j \models g_1$.

Standard Abbreviations

The customary abbreviations will be used for the connectives of propositional logic.

In addition, we will use the following abbreviations in writing temporal operators:

- $\mathbf{A}(f) \equiv \neg\,\mathbf{E}(\neg f)$
- $f \equiv (true\;\mathbf{U}\;f)$
- $\mathbf{G}\,f \equiv \neg\,\mathbf{F}\,\neg f$

CTL and LTL

CTL is a restricted subset of CTL* that permits only branching-time operators—each of the linear-time operators **G**, **F**, **X**, and **U** must be immediately preceded by a path quantifier.

Example: $\mathbf{AG}(\mathbf{EF}\, p)$

LTL consists of formulas that have the form $\mathbf{A}\, f$ where f is a path formula in which the only state subformulas permitted are atomic propositions.

Example: $\mathbf{A}(\mathbf{FG}\, p)$

Expressive Power

It can be shown that the three logics discussed in this section have different expressive powers.

For example, there is no CTL formula that is equivalent to the LTL formula $\mathbf{A}(\mathbf{FG}\, p)$.

Likewise, there is no LTL formula that is equivalent to the CTL formula $\mathbf{AG}(\mathbf{EF}\, p)$.

The disjunction $\mathbf{A}(\mathbf{FG}\, p) \vee \mathbf{AG}(\mathbf{EF}\, p)$ is a CTL* formula that is not expressible in either CTL or LTL.

Basic CTL Operators

There are eight basic CTL operators:

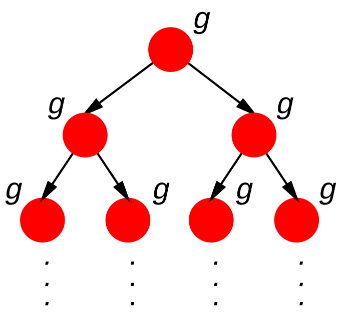
- **AX** and **EX**,
- **AG** and **EG**,
- **AF** and **EF**,
- **AU** and **EU**

Each of these can be expressed in terms of **EX**, **EG**, and **EU**:

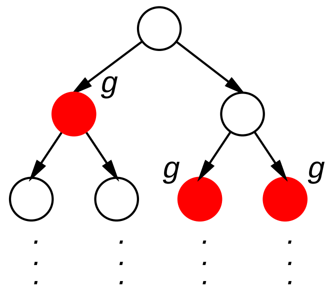
- $\mathbf{AX}\, f = \neg\, \mathbf{EX}(\neg f)$
- $\mathbf{AG}\, f = \neg\, \mathbf{EF}(\neg f)$
- $\mathbf{AF}\, f = \neg\, \mathbf{EG}(\neg f)$
- $\mathbf{EF}\, f = \mathbf{E}[true\ \mathbf{U}\ f]$
- $\mathbf{A}[f\ \mathbf{U}\ g] \equiv \neg\, \mathbf{E}[\neg g\ \mathbf{U}\ \neg f \wedge \neg g] \wedge \neg\, \mathbf{EG}\, \neg g$

Basic CTL Operators

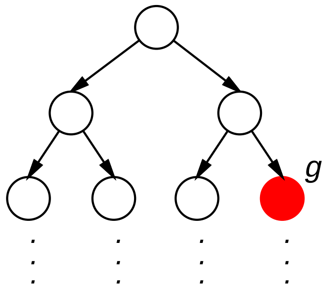
The four most widely used CTL operators are illustrated below. Each computation tree has the state s_0 as its root.



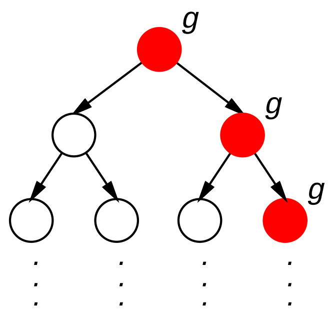
$M, s_0 \models \textbf{AG}\, g$



$M, s_0 \models \textbf{AF}\, g$



$M, s_0 \models \textbf{EF}\, g$



$M, s_0 \models \textbf{EG}\, g$

Typical CTL Formulas

- $\mathbf{EF}(Started \wedge \neg Ready)$: it is possible to get to a state where *Started* holds but *Ready* does not hold.
- $\mathbf{AG}(Req \Rightarrow \mathbf{AF} Ack)$: if a *Request* occurs, then it will be eventually *Acknowledged*.
- $\mathbf{AG}(\mathbf{AF}\ DeviceEnabled)$: *DeviceEnabled* holds infinitely often on every computation path.
- $\mathbf{AG}(\mathbf{EF}\ Restart)$: from any state it is possible to get to the *Restart* state.