Model Checking II
Temporal Logic Model Checking

Temporal Logic Model Checking

Specification Language: A propositional temporal logic called *CTL*.

Verification Procedure: Exhaustive search of the state space of the concurrent system to determine if the specification is true or not.

- E. M. Clarke and E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In *Logic of programs: workshop, Yorktown Heights, NY, May 1981*, volume 131 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.
- J.P. Quielle and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proceedings of the Fifth International Symposium in Programming*, volume 137 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.

Why Model Checking?

Advantages:

- No proofs!!!
- Fast
- Counter-examples
- No problem with partial specifications
- Logics can easily express many concurrency properties

Main Disadvantage: *State Explosion Problem*

- Too many processes
- Data Paths

Much progress has been made on this problem recently!!

Model of Computation

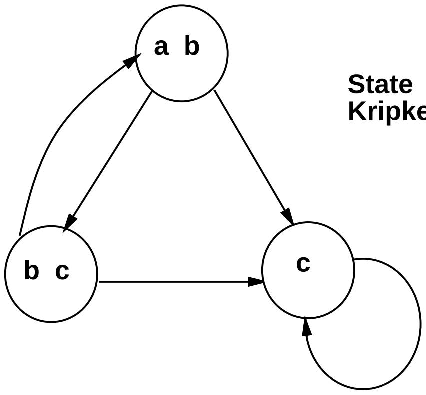
Finite-state systems are modeled by labeled state-transition graphs, called *Kripke Structures*.

If some state is designated as the *initial state*, the structure can be unwound into an infinite tree with that state as the root.

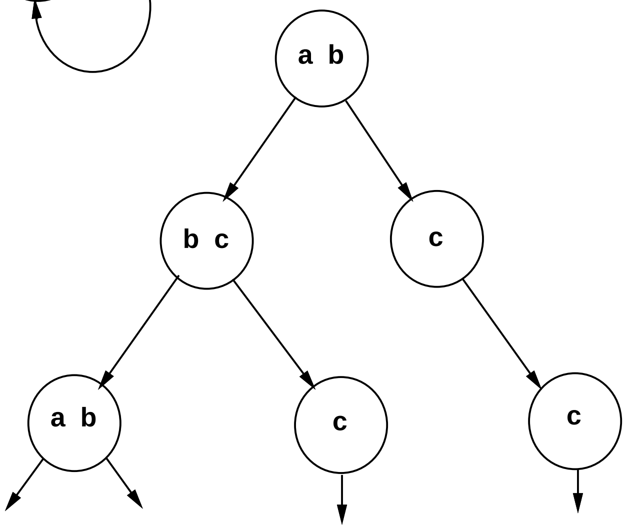
We will refer to the infinite tree obtained in this manner as the *computation tree* of the system.

Paths in the tree represent possible computations or behaviors of the program.

Model of Computation (Cont.)



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
- $L : S \to \mathcal{P}(AP)$ 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. $\pi \models g_1 \vee g_2 \Leftrightarrow \pi \models g_1$ 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:

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

The Logic CTL

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

More precisely, CTL is the subset of CTL* that is obtained if the following two rules are used to specify the syntax of path formulas.

- If f and g are state formulas, then $\mathbf{X}\, f$ and $f\,\mathbf{U}\, g$ are path formulas.
- If f is a path formula, then so is $\neg f$.

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

The Logic LTL

Linear temporal logic (LTL), on the other hand, will consist 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.

More precisely, a path formula is either:

- If $p \in AP$, then p is 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.

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

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 $\mathbf{A}(\mathbf{F}\mathbf{G}\, p)$.

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

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

Basic CTL Operators

There are eight basic CTL operators:

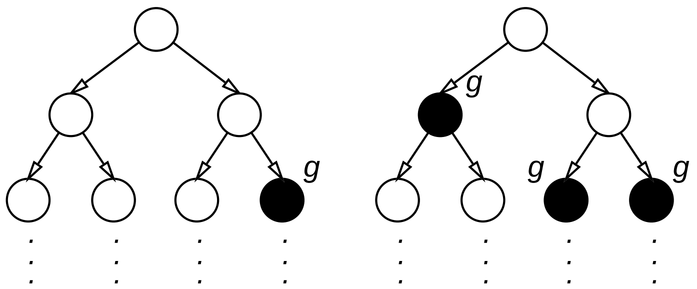
- $\mathbf{AX}$ and $\mathbf{EX}$,
- $\mathbf{AG}$ and $\mathbf{EG}$,
- $\mathbf{AF}$ and $\mathbf{EF}$,
- $\mathbf{AU}$ and $\mathbf{EU}$

Each of these can be expressed in terms of $\mathbf{EX}$, $\mathbf{EG}$, and $\mathbf{EU}$:

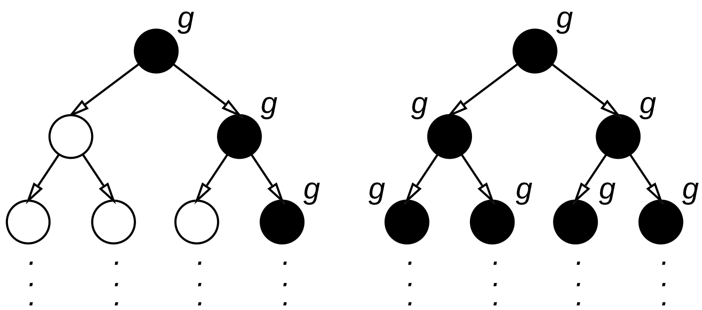
- $\mathbf{AX}\, f = \neg\, \mathbf{EX}(\neg f)$
- $\mathbf{AG}\, f = \neg\, \mathbf{EF}(\neg f)$
- $\mathbf{AF}\, f = \neg\, \mathbf{EG}(\neg f)$
- $\mathbf{EF}\, f = \mathbf{E}[true\, \mathbf{U}\ f]$
- $\mathbf{A}[f\ \mathbf{U}\ g] \equiv \neg\, \mathbf{E}[\neg g\, \mathbf{U}\, \neg f \wedge \neg g]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

Basic CTL Operators (Cont.)

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



$M, s_0 \models \mathbf{EF}\, g$ $\quad$ $M, s_0 \models \mathbf{AF}\, g$



$M, s_0 \models \mathbf{EG}\, g$ $\quad$ $M, s_0 \models \mathbf{AG}\, g$

Typical CTL* formulas

- $\mathbf{EF}(\mathit{Started} \wedge \neg \mathit{Ready})$: It is possible to get to a state where $\mathit{Started}$ holds but Ready does not hold.
- $\mathbf{AG}(\mathit{Req} \rightarrow \mathbf{AF}\, \mathit{Ack})$: If a request occurs, then it will be eventually acknowledged.
- $\mathbf{AG}(\mathbf{AF}\, \mathit{DeviceEnabled})$: The proposition $\mathit{DeviceEnabled}$ holds infinitely often on every computation path.
- $\mathbf{AG}(\mathbf{EF}\, \mathit{Restart})$: From any state it is possible to get to the $\mathit{Restart}$ state.
- $\mathbf{A}(\mathbf{GF}\, \mathit{enabled} \Rightarrow \mathbf{GF}\, \mathit{executed})$: if a process is infinitely-often $\mathit{enabled}$, then it is infinitely-often $\mathit{executed}$.

Note that the first four formulas are CTL formulas.

Model Checking Problem

Let M be the state–transition graph obtained from the concurrent system.

Let f be the specification expressed in temporal logic.

Find all states s of M such that

$$M, s \models f.$$

There exist very efficient model checking algorithms for the logic CTL.

- E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Programming Languages and Systems*, 8(2):pages 244–263, 1986.

The EMC Verification System