**CTL Model Checking**

**Goal** Method for proving $M \text{ sat } \sigma$, where $M$ is a Kripke structure and $\sigma$ is a CTL formula.

**Approach** Model checking!

- Mathematically, $M$ is a *model* of $\sigma$ if $s_I \models_M \sigma$.

- So determining if $M \text{ sat } \sigma$ amounts means checking whether $M$ is a model of $\sigma$. 

©2007 Rance Cleaveland. All rights reserved.
Recall the CTL Fragment of CTL*

... every path modality (i.e. F, G, U) must be preceded by a path quantifier A, E.

The syntax can also be given directly as follows.

\[ \sigma ::= a \]
\[ \left| \neg \sigma \right. \]
\[ \left| \sigma \lor \sigma \right. \]
\[ \left| \text{EX} \sigma \right. \]
\[ \left| E(\sigma \text{ U } \sigma) \right. \]
\[ \left| E(\sigma \text{ R } \sigma) \right. \]

Other operators (AX, AU, AR, EF, AF, EG, AG, etc.) can be defined in terms of these.
So What's the Big Deal About CTL?

- Formulas are “like” those in LTL, but more complex.

+ Model-checking problem easier to solve in CTL.
Properties in CTL

Expressiveness of CTL, LTL are incomparable.

One can reasonably argue that LTL is easier to understand.

However, one can turn LTL system specs into CTL formulas that are “at least as strong”, provided LTL formulas are in *positive normal form* (i.e. negations only applied to atomic propositions).

**E.g.**

<table>
<thead>
<tr>
<th>PNF:</th>
<th>GF executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not PNF:</td>
<td>¬GF executed</td>
</tr>
</tbody>
</table>
Any LTL formula can be put in PNF provided logic is extended with the necessary duals (i.e. $\land$, $R$).

\[
\neg (\phi_1 \lor \phi_2) \equiv (\neg \phi_1) \land (\neg \phi_2)
\]

\[
\neg (X\phi) \equiv X(\neg \phi)
\]

\[
\neg (\phi_1 \U \phi_2) \equiv (\neg \phi_1) R (\neg \phi_2)
\]
Generating CTL Approximations to LTL

So how do we generate CTL formulas “at least as strong” as LTL system specs?

1. Put LTL formula in PNF.

2. Insert A path quantifier in front of each path modality.

<table>
<thead>
<tr>
<th>LTL</th>
<th>CTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>G (send ⇒ F receive)</td>
<td>AG (send ⇒ AF receive)</td>
</tr>
<tr>
<td>(GF enabled) ⇒ (GF executed)</td>
<td>(AFAG¬enabled) ∨ (AGAF executed)</td>
</tr>
</tbody>
</table>
\[ \sigma ::= a \]
\[ \quad \quad \quad \quad \neg a \]
\[ \quad \quad \quad \quad \sigma \lor \sigma \]
\[ \quad \quad \quad \quad \sigma \land \sigma \]
\[ \quad \quad \quad \quad \text{EX} \sigma \]
\[ \quad \quad \quad \quad \text{AX} \sigma \]
\[ \quad \quad \quad \quad \text{E}(\sigma \cup \sigma) \]
\[ \quad \quad \quad \quad \text{A}(\sigma \cup \sigma) \]
\[ \quad \quad \quad \quad \text{E}(\sigma \text{ R } \sigma) \]
\[ \quad \quad \quad \quad \text{A}(\sigma \text{ R } \sigma) \]
The CTL Model-Checking Problem

Given

- Kripke structure $M = \langle S, A, R, \ell, s_I \rangle$
- CTL formula (in PNF) $\sigma$

Determine

Does $s_I \models_M \sigma$?

One approach

1. Define proof rules for CTL correctness assertions $s \models_M \sigma$.
2. Use rules to develop proofs.
Sample Proof Rules

Recall $M = \langle S, A, R, \ell, s_I \rangle$.

\begin{align*}
\mathcal{A} & \quad a \in \ell(s) \\
& \quad s \vdash_M a \\
\neg \mathcal{A} & \quad a \notin \ell(s) \\
& \quad s \vdash_M \neg a \\
\lor_1 & \quad s \vdash_M \sigma_1 \\
& \quad s \vdash_M \sigma_1 \lor \sigma_2 \\
\lor_2 & \quad s \vdash_M \sigma_2 \\
& \quad s \vdash_M \sigma_1 \lor \sigma_2 \\
\land & \quad s \vdash_M \sigma_1, s \vdash_M \sigma_2 \\
& \quad s \vdash_M \sigma_1 \land \sigma_2
\end{align*}

Are these rules sound? Complete?
Proof Rules for CTL Next-Step Modalities

How can we prove assertions of form $s \models_M \text{EX} \sigma$? $s \models_M \text{AX} \sigma$?

- $M$ contains information about transitions from states.
- Proof rules should use this information.

\[
\begin{align*}
\text{EX} & \quad s' \models_M \sigma, \langle s, s' \rangle \in R \\
& \Rightarrow s \models_M \text{EX} \sigma
\end{align*}
\]

\[
\begin{align*}
\text{AX} & \quad s_1 \models_M \sigma, \ldots, s_n \models_M \sigma, \{s_1, \ldots, s_n\} = \{s' \mid \langle s, s' \rangle \in R\} \\
& \Rightarrow s \models_M \text{AX} \sigma
\end{align*}
\]
Proof Rules for U, R Modalities

Idea  Use *recursive characterizations* of modalities.

Notation  $\sigma_1 \equiv \sigma_2$ means: for all $M, s$, $s \models_M \sigma_1$ iff $s \models_M \sigma_2$.

Then:  $\text{AF } \sigma \equiv \sigma \lor \text{AX}(\text{AF } \sigma)$.
Other Recursive Characterizations

\[ \text{EF} \sigma \equiv \]

\[ \text{AG} \sigma \equiv \]

\[ \text{E} \left( \sigma_1 \cup \sigma_2 \right) \equiv \]

\[ \text{E} \left( \sigma_1 \cap \sigma_2 \right) \equiv \]
Turning Recursion into Proof Rules: U

\[ \begin{align*}
\text{EU}_1 & : & s \vdash_M \sigma_2 & \quad \text{AU}_1 & : & s \vdash_M \sigma_2 \\
& & s \vdash_M E(\sigma_1 \cup \sigma_2) & & s \vdash_M A(\sigma_1 \cup \sigma_2) \\
\text{EU}_2 & : & s \vdash_M \sigma_1, s' \vdash_M E(\sigma_1 \cup \sigma_2), \langle s, s' \rangle \in R \\
& & s \vdash_M E(\sigma_1 \cup \sigma_2) \\
\text{AU}_2 & : & s \vdash_M \sigma_1, s_1 \vdash_M A(\sigma_1 \cup \sigma_2), \ldots, s_n \vdash_M A(\sigma_1 \cup \sigma_2), \\
& & \{s_1, \ldots, s_n\} = \{s' | \langle s, s' \rangle \in R\} \\
& & s \vdash_M A(\sigma_1 \cup \sigma_2)
\end{align*} \]
Turning Recursion into Proof Rules: R

\[
\begin{align*}
\text{ER}_1 & \quad s \vdash_M \sigma_1, \ s \vdash_M \sigma_2 \\
& \quad \frac{s \vdash_M E(\sigma_1 R \sigma_2)}{s \vdash_M E(\sigma_1 R \sigma_2)} \\
\text{AR}_1 & \quad s \vdash_M \sigma_1, \ s \vdash_M \sigma_2 \\
& \quad \frac{s \vdash_M A(\sigma_1 R \sigma_2)}{s \vdash_M A(\sigma_1 R \sigma_2)} \\
\text{ER}_2 & \quad s \vdash_M \sigma_2, \ s' \vdash_M E(\sigma_1 R \sigma_2), \ \langle s, s' \rangle \in R \\
& \quad \frac{s \vdash_M E(\sigma_1 R \sigma_2)}{s \vdash_M E(\sigma_1 R \sigma_2)} \\
\text{AR}_2 & \quad s \vdash_M \sigma_2, \\
& \quad s_1 \vdash_M A(\sigma_1 R \sigma_2), \ldots, s_n \vdash_M A(\sigma_1 R \sigma_2), \\
& \quad \{s_1, \ldots, s_n\} = \{s' \mid \langle s, s' \rangle \in R\} \\
& \quad \frac{s \vdash_M A(\sigma_1 R \sigma_2)}{s \vdash_M A(\sigma_1 R \sigma_2)} \\
\end{align*}
\]
But What about Circular Proofs?

Consider proof of $E(a \cup b)$ for Kripke structure below.

\[
\begin{align*}
    s_0 : \{a, c\} & \quad a \in \ell(s_1) \\
    & \quad s_1 \vdash_M a \\
    & \quad s_0 \vdash_M E(a \cup b) \\
    s_1 : \{a\} & \quad a \in \ell(s_0) \\
    & \quad s_0 \vdash_M a \\
    & \quad s_1 \vdash_M E(a \cup b) \\
    & \quad s_0 \vdash_M E(a \cup b)
\end{align*}
\]

Circularity is bad!
But What about Circular Proofs (cont.)?

Consider proof of $\text{EG} \, a$ for Kripke structure below.

\[ s_0 : \{a, c\} \]
\[ s_1 : \{a\} \]

\[
\frac{
\begin{align*}
\text{a} \in \ell(s_0) \\
n_0 \vdash_M a
\end{align*}
}{n_0 \vdash_M \text{EG} \, a}
\]

\[
\frac{
\begin{align*}
\text{a} \in \ell(n_1) \\
n_1 \vdash_M a
\end{align*}
}{n_1 \vdash_M \text{EG} \, a}
\]

Circularity is good!
Is Circularity Bad Or Good?

It depends on the modality ... but how? And why?

Precise answers depend on understanding *fixpoint* characterizations of the CTL operators.

These characterizations will also lead to model-checking algorithms for finite-state Kripke structures.
CTL Formulas and Fixpoints

Recall:

\[ AF \sigma \equiv \sigma \lor AX(AF \sigma) \]

Equivalently, \( AF \sigma \) may be seen as:

- a solution to the equation
  \[ w \equiv \sigma \lor AX w, \]
- a fixpoint of the function
  \[ f(w) = \sigma \lor AX w \]

Is the solution to the above equation unique? No! Consider the formula \( tt \):

\[ tt \equiv \sigma \lor AX tt \] (why?).
So What?

AF $\sigma$ is a solution to an equation, but not a unique solution.

How does this help us with circularity?

Answer Tarski!

... Polish emigré mathematician

... Active in early to mid 1900's

... Well-known for work in logic, algebra, lattice theory

In 1950's, Tarski and Knaster proved:

**Theorem (Tarski-Knaster Fixpoint Theorem)** Every monotonic function over a complete lattice has a complete lattice of fixpoints.
A complete lattice consists of:

- a set $E$ of elements
- a partial ordering ("less than or equal to") $\sqsubseteq \subseteq E \times E$
- a least upper-bound operator $\bigcup \in 2^E \to E$
- a greatest lower-bound operator $\bigcap \in 2^E \to E$

**Example**

Let $S$ be a set. Then take:

- $E = 2^S$
- $\subseteq = \subseteq$
- $\bigcup = \bigcup$
- $\bigcap = \bigcap$

This is a complete lattice!
Facts about Lattices

**Theorem** Let \( \langle E, \sqsubseteq, \sqcup, \sqcap \rangle \) be a complete lattice. Then:

1. \( E \) has a greatest element \( \top = \sqcap \emptyset \).
2. \( E \) has a least element \( \bot = \sqcup \emptyset \).
3. (Tarski-Knaster). Let \( f \in E \rightarrow E \) be monotonic, i.e. if \( e_1 \sqsubseteq e_2 \) then \( f(e_1) \sqsubseteq f(e_2) \). Then the structure \( \langle \{ e \mid e = f(e) \}, \sqsubseteq, \sqcup, \sqcap \rangle \) is also a complete lattice (“complete lattice of fixpoints”).
How Can This Possibly Help?

All the equations describing CTL operators have unique least and greatest solutions! Let $M = \langle S, A, \ell, R, s_I \rangle$ be a Kripke structure.

- $\langle 2^S, \subseteq, \cup, \cap \rangle$ forms a complete lattice.

- Each equation has equivalent form $w = f(w)$ where $f$ maps sets of states (meanings of formulas) to sets of states.

- Each of the $f$ turns out to be monotonic over the lattice.

- Any complete lattice has a unique greatest and least element.
Example

AF $\sigma$ is the unique least solution to $w \equiv f(w)$, where $f(w) \equiv \sigma \lor AXx$.

(More precisely, the set of states satisfying AF$\sigma$ is the smallest set satisfying the equation.)

That is, any other solution is implied by AF $\sigma$.

What is the largest?
Another Example

Consider:

$$f(w) = \sigma \land \text{EX } w$$

What is the least fixpoint? Greatest fixpoint?
One More Example

Consider:

\[ f(w) = \sigma_1 \lor (\sigma_2 \land \text{AX } w) \]

What is the least fixpoint? Greatest fixpoint?
Recall Motivation: Circular Reasoning

Least fixpoint CTL operator: Circularity bad!

Greatest fixpoint CTL operator: Circularity good!
Circularity Example #1

- Circularity involves least-fixpoint operator (EU)
- This proof is therefore invalid.
Circularity Example #2

- Circularity involves greatest-fixpoint operator (EG)
- This proof is therefore valid.
Constructing Proofs for $s \vdash_M \sigma$

- Use proof rules
  $A, \neg A, \lor_1, \lor_2, \land, EX, AX, EU_1, EU_2, AU_1, AU_2, ER_1, ER_2, AR_1, AR_2$

- Proofs are valid if they end in leaves or circularities only involve maximum fixpoint formulas.
Example (Invalid) Proof: AFAG\textsubscript{a}

\[
\begin{array}{c}
\dfrac{s_0 \vdash \mathcal{M} \text{ AFAG} \ a}{s_0 \vdash \mathcal{M} \text{ AFAG} \ a} \\
\dfrac{s_1 \vdash \mathcal{M} \text{ AFAG} \ a}{\ldots} \\
\dfrac{s_0 \vdash \mathcal{M} \text{ AFAG} \ a}{s_0 \vdash \mathcal{M} \text{ AFAG} \ a}
\end{array}
\]
Soundness and Completeness

- Proof construction is sound.

- Proof construction is complete for finite-state Kripke structures.

- Rules can be modified to be complete for arbitrary Kripke structures (sets of states rather than single states on the left of $\vdash_M$).
Algorithmic Model Checking

- We have talked about model-checking in terms of proof.

- For certain kinds of Kripke structures (i.e. finite-state), model-checking can be performed automatically.

- Model-checking algorithms may be seen as conducting proof search.
The Finite-State Model-Checking Problem for CTL

Given

- Kripke structure $M = \langle S, A, R, \ell, s_I \rangle$ with $|S| < \infty$
- CTL formula $\sigma$

Compute

- Does $s_I \models_M \sigma$?
Traditional CTL Model-Checking Algorithms

- Compute all states in $S$ satisfying $\sigma$.
- See if $s_I$ is in this set.

Why is the calculation of these sets of states be possible?

Because of Kleene and the recursive characterizations of operators!
Continuous Functions on Lattices

**Definition**  Let \( \langle E, \sqsubseteq, \bigvee, \bigsqcap \rangle \) be a lattice. Then \( f \in E \to E \) is **continuous** if for every chain \( e_0 \sqsubseteq e_1 \sqsubseteq \cdots \),

\[
f(\bigvee_{i=0}^{\infty} e_i) = \bigvee_{i=0}^{\infty} f(e_i)
\]

**Lemma**

1. Every continuous function is monotonic.
2. If \( |E| < \infty \) then every monotonic function is continuous.
Kleene’s Fixpoint Theorem

Let \( \langle E, \sqsubseteq, \sqcup, \sqcap \rangle \) be a complete lattice, and let \( f \in E \rightarrow E \) be continuous. Then \( \mu f \in E \) and \( \nu f \in E \), the least and greatest fixpoints of \( f \), respectively, can be given as follows.

\[
\mu f = \bigcup_{i=0}^{\infty} f_i, \text{ where } \\
f_0 = \bot \\
f_{i+1} = f(f_i)
\]

\[
\nu f = \bigcap_{i=0}^{\infty} \widehat{f}_i, \text{ where } \\
\widehat{f}_0 = \top \\
\widehat{f}_{i+1} = f(\widehat{f}_i)
\]
How Does This Help?

- For a finite-state Kripke structure \( \langle S, A, R, \ell, s_I \rangle \), complete lattice \( \langle 2^S, \subseteq, \cup, \cap \rangle \) is finite.

- CTL operators are least / greatest fixpoints of functions \( f(w) \) over this lattice.

- All functions for PNF CTL are monotonic, hence continuous over this lattice.
Calculating the Least Solution to an Equation

Assume equation is $x \equiv f(x)$.

- Set $x = \emptyset$ (i.e. ff).
- Compute $f(x)$.
- If $x = f(x)$ we’re done; otherwise set $x$ to $f(x)$ and repeat.

E.g. $x \equiv a \lor AXx$. (In other words, which states satisfy $AF a$?)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
<th>$\models a \lor AX\emptyset$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>${1}$</td>
<td>$\models a \lor AX\emptyset$</td>
</tr>
<tr>
<td>${1}$</td>
<td>${1, 2}$</td>
<td>$\models a \lor AX{1}$</td>
</tr>
<tr>
<td>${1, 2}$</td>
<td>${1, 2}$</td>
<td>$\models a \lor AX{1, 2}$</td>
</tr>
</tbody>
</table>
Another Example: \( E(a \ U \ b) \)

We need to calculate the least solution to \( x \equiv b \lor (a \land EXx) \).

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
</table>

\[
\begin{array}{c}
0 \\
1 \\
2 \\
3
\end{array}
\]
Calculating the Largest Solution to an Equation

- Set $x = S$ (i.e. tt)
- Compute $f(x)$.
- If $x = f(x)$, we’re done; otherwise, set $x$ to $f(x)$ and repeat.

E.g. $x \equiv a \land EXx$. (In other words, which states satisfy $EG a$?)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${0, 1}$</td>
<td>${0}$</td>
</tr>
<tr>
<td>${0}$</td>
<td>${0}$</td>
</tr>
</tbody>
</table>
Least vs. Greatest: An Example

Consider the equation \( x \equiv b \lor (a \land AXx) \).

Least solution:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
</table>

Greatest solution:

| \( x \) | \( f(x) \) |
Classical CTL Model Checking

Recall that traditional CTL model checkers:

- Calculate *all* states that satisfy a given formula ...

- Then ask if the start state is in this set.

So how is the set of states calculated? By processing the formula from the inside out!
Example: AFAGa

First: For AG a, calculate largest solution to $x = a \land AX x$.

Second: For AF $x$, compute least solution to $y = x \lor AX y$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y$</th>
<th>$g(y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let $M = \langle S, A, R, \ell, s_I \rangle$ be a Kripke structure.

Define: $|M| = |S| + |R|$. 

Claim: Equations defining CTL operators can be solved in $|M|$ time.

How? *Counters*
Huh?

E.g. to get least solution to $x = \sigma \lor AX x$ (assuming $\sigma$ already known):

- Associate counter to each state.

- Counter reflects number of transitions leading to states not in current approximation to solution.

- When a state moves into solution, counters of states with transitions leading to state must be updated.
Example: AF $a$

Must calculate smallest solution to $x = a \lor AX \ x$

$$x = \{\}$$
Complexity

Best classical algorithms process each state/transition once per subformula.

How many subformulas are there in formula $\sigma$? $|\sigma|!$ ($|\sigma|$ is number of operators in $\sigma$).

So for Kripke structure $M$, CTL formula $\sigma$, model checking takes: $O(|M| \cdot |\sigma|)$ time.
Pragmatics II: “Short-circuiting”

... stop computation once status of start state is known.

E.g. If least solution is being calculated, and start state added to intermediate approximation, can stop.

... does not affect complexity, but can improve “practical performance.”
... short-circuiting taken to the extreme.

... takes a “top-down” view (“what is the minimal information I need to compute to check if $s_I \models_M \sigma$”).

Approaches can be formulated in terms of proof search involving proof rules like the ones we have studied.

Subtlety: circularity.
Pragmatics IV: Efficient Data Structures

Classical algorithms require manipulations of sets of states:

- Unions, intersections
- Equality checking
- Transitions from/to sets of states

The right data structure can yield dramatic time/space improvement!
Example: (Ordered) Binary Decision Diagrams

In some applications states are fixed-width bit vectors (e.g. $\text{M}ur_{\varphi}$ with only boolean variables).

OBDDs are data structures for representing sets of bit vectors compactly.

Union, intersection, equality all supported efficiently.

In hardware community, most successful model checkers use OBDDs.
OBDDs and Sets of Fixed-Width Bit Vectors

An OBDD is...

- a directed acyclic graph, with
- a leaf labeled 0 and a leaf labeled 1, and
- each internal node labeled by a variable, and
- each node having two edges, one labeled 0 and one 1.

In addition, OBDDs satisfy:

1. No isomorphic subgraphs
2. No “don’t cares”
Example

An OBDD for the set \(\{000, 001, 011\}\) with ordering \(v_1v_2v_3\).
Example

An OBDD for the set \( \{000, 001, 011\} \) with ordering \( v_3v_1v_2 \).
Facts about OBDDs

1. Variable ordering influences size of OBDDs.

2. Given a fixed variable ordering, set representation is *canonical* (equal sets have isomorphic OBDDs).

3. Efficient implementations exist for union, intersection, complementation, projection, ....
How Are OBDDs Used in Model Checking?

... To represent Kripke structures

- States represented as bit-vectors of length $n$
- Transitions represented as bit-vectors of length $2n$

... To represent approximate solutions during equation solving

- If $x = f(x)$ is an equation, process of “applying” $f$ to get new approximations can be given as function on OBDDs!