Crash course – Verification of Finite Automata
CTL model-checking

Xiaoxi He
Reminders – Big picture

**Objective**
Verify properties over DES models
Formal method $\Rightarrow$ Absolute guarantee!

**Problem**
Combinatorial explosion
$\rightarrow$ Huge amount of states, computationally intractable

**Solution**
Work with sets of states
$\rightarrow$ Symbolic Model-Checking
$\rightarrow$ (O)BDDs
Reminders – First exercise session

Equivalence between sets and Boolean equations

Sets

- A
- \( s \in A \)

Boolean functions/Characteristic functions

- \( \psi_E = 1 \)
- \( \psi_A = f \)
- \( \psi_B = g \)
- \( \psi_{A \cap B} = f \cdot g \)

Example:
\( \sigma(s) = x_1 \overline{x_0} = (1,0) \) and \( \psi_A = x_1 + x_0 \)
\( \rightarrow s \vDash \psi_A? \)

BDD representation of Boolean functions

\( f: x_1 + \overline{x_1} x_2 + \overline{x_2} \overline{x_3} \)

Fall \( x_1 = 0 \)
- \( f_{|x_1=0}: x_2 + \overline{x_2} \overline{x_3} \)
  - Fall \( x_2 = 0 \)
    - \( f_{|x_1=0, x_2=0}: \overline{x_3} \)
  - Fall \( x_2 = 1 \)
    - \( f_{|x_1=0, x_2=1}: 1 \)

Fall \( x_1 = 1 \)
- \( f_{|x_1=1}: 1 \)
Today’s menu

1. Reachability of states
2. Comparison of automata
3. Formulation and verification of CTL properties

Can be formulated as reachability problems
Reachability of states

Fairly simple

1. Start from the initial set of states,
2. Compute all states you can transition to in one hop (one transition),
   → The successor states,
3. Join the two sets,
4. Iterate from 2. until you reach a fix point.
5. Done!

Is this guarantee to terminate?
Reachability of states

Fairly simple

1. Start from the initial set of states,
2. Compute all states you can transition to in one hop (one transition),
   → The successor states,
3. Join the two sets,
4. Iterate from 2. until you reach a fix point.
5. Done!

*Is this guarantee to terminate?*
→ Only if you have a finite model!!

*How can we formalize this problem?*
Formalization of reachable states

\[ \delta : X \subseteq E \rightarrow X' \subseteq E \]

\[ q \mapsto q' \]

\[ q \in X \iff \exists q' \in X', \delta(q, q') \text{ is defined} \]

\[ \psi_\delta(q, q') = 1 \]

\[ \overline{q} \notin X \iff \forall q' \in X', \delta(\overline{q}, q') \text{ is defined} \]

\[ \forall q' \in X, \psi_\delta(\overline{q}, q') = 0 \]
Formalization of reachable states

$$\delta : X \subseteq E \rightarrow X' \subseteq E$$

$$q \mapsto q'$$

What is Q’?

$$q' \in Q' \Rightarrow q' \in X' \Rightarrow \exists q \in X, \psi_\delta(q, q') = 1$$

Not sufficient!

We also need that q belongs to Q:

$$q \in Q$$

or equivalently

$$\psi_Q(q) = 1$$
Formalization of reachable states

\[ \delta : X \subseteq E \rightarrow X' \subseteq E \]
\[ q \mapsto q' \]

What is \( Q' \)?

\[ q' \in Q' \iff \exists q \in X, \psi_Q(q) = 1 \text{ and } \psi_\delta(q, q') = 1 \]
\[ \iff \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q, q') = 1 \]

\[ Q' = \text{Suc}(Q, \delta) = \{ q' \mid \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q, q') = 1 \} \]
Formalization of reachable states

\[ \delta : X \subseteq E \rightarrow X' \subseteq E \]

\[ q \xrightarrow{\delta} q' \]

\[ Q' = \text{Suc}(Q, \delta) = \{ q' \mid \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q, q') = 1 \} \]

\[ \Leftrightarrow \psi_{Q'} = \psi_Q \cdot \psi_\delta \]

\[ Q_R: \text{set of reachable states} \]

\[ Q_R = Q_0 \cup_{i \geq 0} \text{Suc}(Q_i, \delta) \]

\[ \Leftrightarrow \psi_{Q_R} = \psi_Q_0 \sum_{i \geq 0} \psi_{Q_i} \cdot \psi_\delta \]

Again, finite union if finite model
Comparison of automata

Two automata are equivalent ⇔ Same input produces same output

- Computation of the joint transition function,
  \[ \psi_\delta(q_1, q_2, q'_1, q'_2) = (\exists u : \psi_{\omega_1}(u, q_1, q'_1) \cdot \psi_{\omega_2}(u, q_2, q'_2)) \]

- Computation of the reachable states (method according to previous slides),
  \[ \psi_Q(q_1, q_2) \]

- Computation of the reachable output values,
  \[ \psi_Y(y_1, y_2) = (\exists q_1, q_2 : \psi_Q(q_1, q_2) \cdot \psi_{\omega_1}(q_1, y_1) \cdot \psi_{\omega_2}(q_2, y_2)) \]

- The automata are not equivalent if the following term is true,
  \[ \exists y_1, y_2 : \psi_Y(y_1, y_2) \cdot (y_1 \neq y_2) \]

Get rid of the input
Compute \( Q_R \)
Deduce reachable outputs
Test for equivalence
Don’t compare states!
Formulation of CTL properties

Based on atomic propositions ($\phi$) and quantifiers

$A\phi \rightarrow \langle \text{All } \phi \rangle$, $\phi$ holds on all paths
$E\phi \rightarrow \langle \text{Exists } \phi \rangle$, $\phi$ holds on at least one path
$X\phi \rightarrow \langle \text{Next } \phi \rangle$, $\phi$ holds on the next state
$F\phi \rightarrow \langle \text{Finally } \phi \rangle$, $\phi$ holds at some state along the path
$G\phi \rightarrow \langle \text{Globally } \phi \rangle$, $\phi$ holds on all states along the path
$\phi_1 U \phi_2 \rightarrow \langle \phi_1 \text{ Until } \phi_2 \rangle$, $\phi_1$ holds until $\phi_2$ holds

Quantifiers over paths
Path-specific quantifiers
Formulation of CTL properties

Proper CTL formula: \{A,E\} \{X,F,G,U\}φ

→ Quantifiers go by pairs, you need one of each.

Missing Hypothesis
Interpretation on CTL formula

→ Transition functions are fully defined (i.e. every state has at least one successor)

Simple “means” that we get rid of leaf nodes...
→ They transition to themselves

Automaton of interest

Automaton to work with
Formulation of CTL properties

$\text{EF } \phi : \text{ "There exists a path along which at some state } \phi \text{ holds."}$
Formulation of CTL properties

$\text{EF } \phi : \text{“There exists a path along which at some state } \phi \text{ holds.”} \]

$q \models \phi$

$q \models \text{EF } \phi$

$r \not\models \text{EF } \phi$

$s \not\models \text{EF } \phi$
Formulation of CTL properties

AF $\phi$ : “On all paths, at some state $\phi$ holds.”
Formulation of CTL properties

$\text{AF } \phi : \text{“On all paths, at some state } \phi \text{ holds.”}$
Formulation of CTL properties

\( \text{AG } \phi \): “On all paths, for all states \( \phi \) holds.”

\( q \models \phi \)
\( q \models \text{AG } \phi \)
\( r \models \) ?
\( s \models \) ?
Formulation of CTL properties

$AG\ \phi$ : “On all paths, for all states $\phi$ holds.”

$q \models AG\ \phi$

$r \models AG\ \phi$

$s \not\models AG\ \phi$
Formulation of CTL properties

\[ \text{EG } \phi : \text{“There exists a path along which for all states } \phi \text{ holds.”} \]
Formulation of CTL properties

EG $\phi$ : “There exists a path along which for all states $\phi$ holds.”

$q \models EG \phi$

$r \models EG \phi$

$s \not\models EG \phi$
Formulation of CTL properties

\(\phi \mathsf{EU} \Psi\) : “There exists a path along which \(\phi\) holds until \(\Psi\) holds.”
Formulation of CTL properties

$\phi \mathbf{E} \mathbf{U} \Psi$ : “There exists a path along which $\phi$ holds until $\Psi$ holds.”
Formulation of CTL properties

$\phi \text{AU} \Psi$ : “On all paths, $\phi$ holds until $\Psi$ holds.”
Formulation of CTL properties

\( \phi \mathsf{AU} \Psi \): “On all paths, \( \phi \) holds until \( \Psi \) holds.”
Formulation of CTL properties

**AXϕ** : “On all paths, the next state satisfies ϕ.”

**EXϕ** : “There exists a path along which the next state satisfies ϕ.”

![Diagram](image)

- Blue circles: ϕ
- White circles: ¬ϕ
- q ⊨ EXϕ
- r ⊨ ?
- s ⊨ ?
Formulation of CTL properties

$AX\phi$ : “On all paths, the next state satisfies $\phi$.”

$EX\phi$ : “There exists a path along which the next state satisfies $\phi$.”
Formulation of CTL properties

\( \text{AG EF } \phi \) : “On all paths and for all states, there exists a path along which at some state \( \phi \) holds.”

\[ q \models \text{AG EF } \phi \]
\[ r \models ? \]
\[ s \models ? \]
Formulation of CTL properties

\[\text{AG EF } \phi : \text{“On all paths and for all states, there exists a path along which at some state } \phi \text{ holds.”}\]

Diagram:

- \(q \models \phi\)
- \(q \models \text{AG EF} \phi\)
- \(r \models \text{AG EF} \phi\)
- \(s \models \text{AG EF} \phi\)
Inverting properties is sometimes useful!

\begin{align*}
\text{AG } \phi & \equiv \neg \text{EF } \neg \phi \\
\text{AF } \phi & \equiv \neg \text{EG } \neg \phi \\
\text{EF } \phi & \equiv \neg \text{AG } \neg \phi \\
\text{EG } \phi & \equiv \neg \text{AF } \neg \phi
\end{align*}

“On all paths, for all states \( \phi \) holds.”

“There exists no path along which at some state \( \phi \) doesn’t hold.”

\textbf{Remark} \quad \text{There exists other temporal logics}

\begin{align*}
\rightarrow & \quad \text{LTL (Linear Tree Logic)} \\
\rightarrow & \quad \text{CTL* = \{CTL,LTL\}} \\
\rightarrow & \quad \ldots
\end{align*}
How to verify CTL properties?

*Convert the property verification into a reachability problem*

1. Start from states in which the property holds;
2. Compute all predecessor states for which the property still holds true; (same as for computing successor, with the inverse the transition function)
3. If initial states set is a subset, the property is satisfied by the model.

*Computation specifics are described in the lecture slides.*
So... what is Model-Checking exactly?

An algorithm

**Input**
- A DES model, $M$
  - Finite automata,
  - Petri nets,
  - Kripke machine, ...
- A logic property, $\phi$
  - CTL,
  - LTL, ...

**Output**
- $M \models \phi$?
- A trace for which the property does not hold!
Crash course – Verification of Finite Automata

CTL model-checking

Your turn to work!

Slides online on my webpage:

http://people.ee.ethz.ch/~jacobr/
Computer Engineering and Networks

Technische Informatik und Kommunikationsnetze
Comparison of Finite Automata

a) Express the characteristic function of the transition relation for both automaton, $\psi_r(x, x', u)$.

$$\psi_A(x_A, x'_A, u) = \overline{x_A} \overline{x'_A} u + \overline{x_A} x'_A u + x_A x'_A \overline{u}$$

$$\psi_B(x_B, x'_B, u) = \overline{x_B} \overline{x'_B} u + \overline{x_B} x'_B u + x_B x'_B \overline{u}$$
Comparison of Finite Automata

b) Express the joint transition function, $\psi_f$.

$$
\psi_f(x_A, x'_A, x_B, x'_B) = (\exists u : \psi_A(x_A, x'_A, u) \cdot \psi_B(x_B, x'_B, u))
$$

$$
\psi_f(x_A, x'_A, x_B, x'_B) = (\overline{x_A}x'_A + x_Ax'_A) \cdot (\overline{x_B}x'_B + x_Bx'_B) + \\
(\overline{x_A}x'_A + x_Ax'_A) \cdot (\overline{x_B}x'_B + x_Bx'_B)
$$

$$
= \overline{x_A}x'_A \overline{x_B}x'_B + \overline{x_A}x'_A x_Bx'_B + x_Ax'_A \overline{x_B}x'_B + x_Ax'_A x_Bx'_B + \\
\overline{x_A}x'_A \overline{x_B}x'_B + \overline{x_A}x'_A x_Bx'_B + x_Ax'_A \overline{x_B}x'_B + x_Ax'_A x_Bx'_B
$$
Comparison of Finite Automata

c) Express the characteristic function of the reachable states, $\psi_X(x_A, x_B)$.

$$
\psi_{X_0}(x_A, x_B) = \overline{x_A} x_B \\
\psi_{X_1}(x'_A, x'_B) = \psi_{X_0}(x'_A, x'_B) + (\exists (x_A, x_B) : \psi_{X_0}(x_A, x_B) \cdot \psi_f(x_A, x'_A, x_B, x'_B)) \\
= x'_A x'_B + x'_A x'_B \\
\psi_{X_2}(x'_A, x'_B) = x'_A x'_B + x'_A x'_B + x'_A x'_B + x'_A x'_B \\
\psi_{X_3}(x'_A, x'_B) = x'_A x'_B + x'_A x'_B + x'_A x'_B + x'_A x'_B \\
= \psi_{X_2} \rightarrow \text{the fix-point is reached!}
$$

$$
\psi_X = \overline{x_A} x_B + x_A \overline{x_B} + x_A x_B + \overline{x_A} x_B
$$
d) Express the characteristic function of the reachable output, $\psi_Y(x_A, x_B)$.

\[
\psi_{g_A} = \overline{x_A}y_A + x_Ay_A \\
\psi_{g_B} = \overline{x_B}y_B + x_B\overline{y_B}
\]

and
\[
\psi_X = \overline{x_A}x_B + x_A\overline{x_B} + x_Ax_B + \overline{x_A}x_B
\]

\[
\psi_Y(y_A, y_B) = (\exists (x_A, x_B) : \psi_X \cdot \psi_{g_A} \cdot \psi_{g_B}) \\
= y_Ay_B + \overline{y_A}y_B + \overline{y_A}y_B + y_A\overline{y_B}
\]
Comparison of Finite Automata

e) Are the automata equivalent? **Hint:** Evaluate, for example, $\psi_Y(0,1)$.

$$\psi_Y((y_A, y_B) = (0, 1)) = 1$$

Or, in a more general way,

$$\psi_Y(y_A, y_B) = y_A y_B + \overline{y_A} y_B + \overline{y_A} y_B + y_A \overline{y_B}$$

and $(y_A \neq y_B) = \overline{y_A} y_B + y_A \overline{y_B}$

implies $\psi_Y \cdot (y_A \neq y_B) \neq 0$

→ Automata are not equivalent.
Temporal Logic

i. EF a

ii. EG a

iii. EX AX a

iv. EF ( a AND EX NOT(a) )
Temporal Logic

i. EF a
   \[ Q = \{0, 1, 2, 3\} \]

ii. EG a

iii. EX AX a

iv. EF ( a AND EX NOT(a) )
Temporal Logic

i. EF a
   \[ Q = \{0, 1, 2, 3\} \]

ii. EG a
    \[ Q = \{0, 3\} \]

iii. EX AX a

iv. EF ( a AND EX NOT(a) )
Temporal Logic

i. EF $a$
   $$Q = \{0, 1, 2, 3\}$$

ii. EG $a$
    $$Q = \{0, 3\}$$

iii. EX AX $a$
     $$Q = \{1, 2\}$$

iv. EF ( $a$ AND EX NOT($a$) )
Temporal Logic

i. EF a
   \[ Q = \{0, 1, 2, 3\} \]

ii. EG a
    \[ Q = \{0, 3\} \]

iii. EX AX a
     \[ Q = \{1, 2\} \]

iv. EF ( a AND EX NOT(a) )
    \[ Q = \{0, 1, 2, 3\} \]
Temporal Logic

**Trick**  \( AF \neg Z \not= (EG \neg (Z)) \)

**Require:** \( \psi_Z, \psi_f \)

- current = NOT(\( \psi_Z \));
- next = current AND \( \psi_{PRE}(current,f) \);
- while next != current do
  - current = next;
  - next = current AND \( \psi_{PRE}(current,f) \);
- end while
- return \( \psi_{AF Z} = \not= NOT(current) \);

\[ \triangleright \text{Equivalence in term of sets:} \]

\[ \triangleright X_0 \]

\[ \triangleright X_1 = X_0 \cap Pre(X_0, f) \]

\[ \triangleright X_i \not= X_{i-1} \]

\[ \triangleright X_i = X_{i-1} \cap Pre(X_{i-1}, f) \]

\[ \triangleright X_f |\not= EG \neg (Z) \]

\[ \triangleright \overline{X_f} |\not= AF \neg Z = NOT(EG \neg (Z)) \]
Crash course – Verification of Finite Automata
CTL model-checking

See you next week!