Important note

Corrections have been made to the CTL part.
An important hypothesis was missing (see slide 15)

Crash course – Verification of Finite Automata
CTL model-checking

Exercise session - 08.12.2016
Romain Jacob
Reminders – Big picture

**Objective**
Verify properties over DES models
Formal method ⇒ Absolute guarantee!

**Problem**
Combinatorial explosion
→ Huge amount of states, computationally intractable

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

Sets

- A
- \( s \in A \)

\[ \sigma(s) = x_1 \bar{x}_0 = (1,0) \text{ and } \psi_A = x_1 + x_0 \]

\( \rightarrow s \models \psi_A \)

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

Boolean functions/Characteristic functions

- \( \psi_A \)
- \( \psi_A(\sigma(s)) = 1 \)
- \( \sigma(s) \models \psi_A \)
- or just \( s \models \psi_A \)

Example:

\[ f: x_1 + \bar{x}_1 \cdot x_2 + x_2 \cdot \bar{x}_3 \]

Fall \( x_1 = 0 \)

\( f_{|x_1=0} = x_2 + \bar{x}_3 \)

Fall \( x_2 = 0 \)

- \( f_{|x_1=0, x_2=0} = \bar{x}_3 \)

Fall \( x_2 = 1 \)

- \( f_{|x_1=0, x_2=1} = 1 \)

Fall \( x_1 = 1 \)

- \( f_{|x_1=1} = 1 \)

BBD representation of Boolean functions

Equivalence between sets and Boolean equations

\( \Rightarrow \)
Let see what you remember!
Discrete Event Systems

Roger P. Wattenhofer, Lothar Thiele, Laurent Vanbever

Learning objective
Over the past few decades the rapid evolution of computing, communication, and information technologies has brought about the proliferation of new dynamic systems. A significant part of activity in these systems is governed by operational rules designed by humans. The dynamics of these systems are characterized by asynchronous occurrences of discrete events, some controlled (e.g. hitting a keyboard key, sending a message), some not (e.g. spontaneous failure, packet loss).

The mathematical arsenal centered around differential equations that has been employed in systems engineering to model and study processes governed by the laws of nature is often inadequate or inappropriate for discrete event systems. The challenge is to develop new modeling frameworks, analysis techniques, design...
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),
   \[ \rightarrow \text{The successor states}, \]
3. Join the two sets,
4. Iterate from 2. until you reach a fix point.
5. Done!

*Is this guarantee to terminate?*
\[ \rightarrow \text{Only if you have a finite model!!} \]

*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

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

\[ q \rightarrow q' \]

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 \( Q \):

\[ q \in Q \quad \text{or equivalently} \quad \psi_Q(q) = 1 \]
Formalization of reachable states

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

$$q \mapsto q'$$

What is $$Q'$$?

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

$$\iff \exists q \in X, \psi_Q(q) \cdot \psi_\delta(q, q') = 1$$

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

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

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

\[ Q_R: \text{set of reachable states} \]
\[ Q_R = Q_0 \cup_{i \geq 0} Suc(Q_i, \delta) \]
\[ \Leftrightarrow \psi_{Q_R} = \psi_{Q_0} \sum_{i \geq 0} \psi_{Q_i} \cdot \psi_\delta \]

Again, finite union if finite model
Comparison of automata

Two automata are equivalent if the following term is true:

$$\exists y_1, y_2 : \psi_Y(y_1, y_2) \cdot (y_1 \neq y_2)$$

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.

Don’t compare states!

- Get rid of the input
- Compute $$Q_R$$
- Deduce reachable outputs
- Test for equivalence
Formulation of CTL properties

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

\[ A\phi \rightarrow \text{«All } \phi \text{», } \phi \text{ holds on all paths} \]
\[ E\phi \rightarrow \text{«Exists } \phi \text{», } \phi \text{ holds on at least one path} \]
\[ X\phi \rightarrow \text{«NeXt } \phi \text{», } \phi \text{ holds on the next state} \]
\[ F\phi \rightarrow \text{«Finally } \phi \text{», } \phi \text{ holds at some state along the path} \]
\[ G\phi \rightarrow \text{«Globally } \phi \text{», } \phi \text{ holds on all states along the path} \]
\[ \phi_1 U \phi_2 \rightarrow \text{«} \phi_1 \text{Until } \phi_2 \text{», } \phi_1 \text{ holds until } \phi_2 \text{ holds} \]
Formulation of CTL properties

Proper CTL formula: \( \{A,E\} \{X,F,G,U\} \phi \)

→ 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
Formulation of CTL properties

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

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

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

\( \text{AF } \phi : \text{“On all paths, at some state } \phi \text{ holds.”} \)

![Diagram showing the properties AF \( \phi \) with nodes q, r, and s and their implications.]

- \( q \models \text{AF } \phi \)
- \( r \models \text{AF } \phi \)
- \( s \not\models \text{AF } \phi \)
Formulation of CTL properties

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

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

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

\( A\phi U \Psi \) : “On all paths, \( \phi \) holds until \( \Psi \) holds.”
Formulation of CTL properties

\( A\phi U\Psi \) : “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.”
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 $\phi$.”

$q \models EX\phi$

$r \models EX\phi$

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

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

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

$$q \vDash \text{AG EF } \phi$$

$$r \vDash \text{AG EF } \phi$$

$$s \vDash \text{AG EF } \phi$$

$$S \vDash \text{AG EF } \phi$$
Inverting properties is sometimes useful!

\[
\begin{align*}
AG \, \phi &\equiv \neg EF \neg \phi \\
AF \, \phi &\equiv \neg EG \neg \phi \\
EF \, \phi &\equiv \neg AG \neg \phi \\
EG \, \phi &\equiv \neg AF \neg \phi
\end{align*}
\]

"On all paths, for all states $\phi$ holds."

"There exists no path along which at some state $\phi$ doesn’t hold."

…

**Remark** There exists other temporal logics

→ LTL (Linear Tree Logic)

→ CTL* = {CTL,LTL}

→ ...
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/
Comparison of Finite Automata

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

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

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

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

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

$$\psi_f(x_A, x'_A, x_B, x'_B)$$

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

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

$$= \overline{x_A}x'_Ax_Bx'_B + \overline{x_A}x'_Ax_Bx'_B + x_Ax'_A\overline{x_B}x'_B + x_Ax'_Ax_Bx'_B +$$

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

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

\[
\psi_{X_0}(x_A, x_B) = \overline{x_A}x_B
\]

\[
i\psi_{X_1} = \overline{x_A}x_B + \overline{x_A}\overline{x_B} + x_Ax_B
\]

\[
\psi_{X_2} = \overline{x_A}x_B + \overline{x_A}\overline{x_B} + x_Ax_B
\]

\[
= \psi_{X_1}
\]

→ the fix-point is reached!

\[
\psi_X = \overline{x_A}x_B + \overline{x_A}\overline{x_B} + x_Ax_B
\]
Comparison of Finite Automata

d) Express the characteristic function of the reachable output, $\psi_Y(x_A, x_B)$.

\[
\psi_{gA} = \overline{x_A y_A} + x_A y_A \\
\psi_{gB} = \overline{x_B y_B} + x_B y_B \\
\text{and} \quad \psi_X = \overline{x_A x_B} + x_A \overline{x_B} + x_A x_B \\
\psi_Y(y_A, y_B) &= \left( \exists (x_A, x_B) : \psi_X \cdot \psi_{gA} \cdot \psi_{gB} \right) \\
&= y_A y_B + \overline{y_A y_B} + \overline{y_A y_B}
\]
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,

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

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. $\text{EF } a$

ii. $\text{EG } a$

iii. $\text{EX } \text{AX } a$

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

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

iii. $\text{EX AX } a$

iv. $\text{EF ( } a \text{ AND EX NOT}(a) \text{ )}$
Temporal Logic

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

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

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

iv. $\text{EF ( a AND EX NOT(a) )}$
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** \( \text{AF} Z \text{ \ not(EG not(Z))} \)

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

\[
\begin{align*}
\text{current} & = \text{NOT}(\psi_Z); \\
\text{next} & = \text{current AND } \psi_{\text{PRE}}(\text{current}, f); \\
\text{while next } & \neq \text{ current do} \\
& \text{ current} = \text{next}; \\
& \text{ next} = \text{current AND } \psi_{\text{PRE}}(\text{current}, f); \\
\text{end while} \\
\text{return } \psi_{\text{AF}} Z & = \text{NOT} (\text{current}); \\
\end{align*}
\]

\[
\begin{align*}
\triangleright \text{ Equivalence in term of sets:} \\
& \triangleright X_0 \\
& \triangleright X_1 = X_0 \cap \text{Pre}(X_0, f) \\
& \triangleright X_i \neq X_{i-1} \\
& \triangleright X_i = X_{i-1} \cap \text{Pre}(X_{i-1}, f) \\
& \triangleright X_f \models \text{EG NOT}(Z) \\
& \triangleright \overline{X_f} \models \text{AF} Z = \text{NOT}(\text{EG NOT}(Z))
\end{align*}
\]
Crash course – Verification of Finite Automata
CTL model-checking

See you next week!

Slides online on my webpage:
http://people.ee.ethz.ch/~jacobr/