ESE601: Hybrid Systems

Introduction to verification

Spring 2006
Suggested reading material

Papers (R14) - (R16) on the website.
The book “Model checking” by Clarke, Grumberg and Peled.
What is verification?

We need to make sure that the engineering systems we build are **safe, functioning correctly**, etc.

Systems can mean software, hardware, protocols, etc. Thus, not restricted to hybrid systems. In fact, verification originates in computer science, i.e. for discrete event systems.

How is verification done? The system is represented as **transition system**, the properties to be verified are represented as **temporal logic formulas**, whose truth values are to be determined/verified.
A transition system

\[ T = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle) \]

consists of

- A set of states \( Q \)
- A set of events \( \Sigma \)
- A set of observations \( O \)
- The transition relation \( q_1 \rightarrow q_2 \)
- The observation map \( \langle q_1 \rangle = o_0 \)

Initial may be incorporated

The sets \( Q, \Sigma, O \) may be infinite

Language of \( T \) is all sequences of observations
A painful example

The parking meter

States $Q = \{0, 1, 2, ..., 60\}$

Events $\{\text{tick}, 5p\}$

Observations $\{\text{exp}, \text{act}\}$

A possible string of observations $(\text{exp}, \text{act}, \text{act}, \text{act}, \text{act}, \text{act}, \text{exp}, ...)$
Temporal logic involves logical propositions whose truth values depend on time.

“Tomorrow is Thursday”

The time is related to the execution steps of the transition system.

The asserted property is related to the observation of the transition system.

“At the next state, the meter expires”
The basic verification problem

Given transition system $T$, and temporal logic formula $\varphi$

\[
T \models \varphi
\]

The transition system satisfies the formula if:
- All executions satisfy the formula (linear time)
- The initial states satisfy the formula (branching time)
Another verification problem

Given transition system $T$, and specification system $S$

Language inclusion problems. Recall supervisory control problem.

$L(T) \subseteq L(S)$
Linear temporal logic

Linear temporal logic syntax

The LTL formulas are defined inductively as follows

Atomic propositions
   All observation symbols p are formulas

Boolean operators
   If \( \varphi_1 \) and \( \varphi_2 \) are formulas then
   \[ \varphi_1 \lor \varphi_2 \quad \neg \varphi_1 \]

Temporal operators
   If \( \varphi_1 \) and \( \varphi_2 \) are formulas then
   \[ \varphi_1 U \varphi_2 \quad \bigcirc \varphi_1 \]
Linear temporal logic

LTL formulas are evaluated over (infinite) sequences of execution, which are called words.
Ex: \( w=(\text{exp},\text{act},\text{act},\text{act},\text{act},\text{act},\text{exp},...) \)

\[
(w, 0)|= \text{exp}, (w, 1)|= \text{act}, (w, 1)|= \neg \text{exp}, \ldots
\]

A word \( w \) satisfies a formula iff \((w,0)\) satisfies it.

\[
w|= \phi \iff (w, 0)|= \phi
\]

\[
w|= \bigcirc \phi \iff (w, 1)|= \phi
\]

\[
w|= \theta \bigcup \phi \iff (w, i)|= \theta, (w, N)|= \phi, 0 \leq i < N.
\]
### Linear temporal logic

Express temporal specifications along sequences

<table>
<thead>
<tr>
<th>Informally</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventually $p$</td>
<td>$\Diamond p$</td>
<td>$\ast \ast \ast \ast \ast \ast \ast \ast \ast \ast \ast \ast p$</td>
</tr>
<tr>
<td>Always $p$</td>
<td>$\Box p$</td>
<td>$pppppppppppppppppp$</td>
</tr>
<tr>
<td>If $p$ then next $q$</td>
<td>$p \Rightarrow \bigcirc q$</td>
<td>$\ast \ast \ast \ast \ast \ast \ast \ast \ast \ast \ast pq$</td>
</tr>
<tr>
<td>$p$ until $q$</td>
<td>$p U q$</td>
<td>$pppppppppppppqq \ast \ast \ast$</td>
</tr>
</tbody>
</table>
Linear temporal logic

Syntactic boolean abbreviations

Conjunction: \( \varphi_1 \land \varphi_2 = \neg(\neg \varphi_1 \lor \neg \varphi_2) \)

Implication: \( \varphi_1 \Rightarrow \varphi_2 = \neg \varphi_1 \lor \varphi_2 \)

Equivalence: \( \varphi_1 \Leftrightarrow \varphi_2 = (\varphi_1 \Rightarrow \varphi_2) \land (\varphi_2 \Rightarrow \varphi_1) \)

Syntactic temporal abbreviations

Eventually: \( \Diamond \varphi = \top U \varphi \)

Always: \( \Box \varphi = \neg \Diamond \neg \varphi \)

In 3 steps: \( \bigcirc_3 \varphi = \bigcirc \bigcirc \bigcirc \bigcirc \bigcirc \varphi \)
Linear temporal logic semantics

The LTL formulas are interpreted over infinite (omega) words

\[ w = p_0 \ p_1 \ p_2 \ p_3 \ p_4 \ldots \]

\[(w, i) \models p \iff p_i = p\]
\[(w, i) \models \varphi_1 \lor \varphi_2 \iff (w, i) \models \varphi_1 \text{ or } (w, i) \models \varphi_2\]
\[(w, i) \models \neg \varphi_1 \iff (w, i) \not\models \varphi_1\]
\[(w, i) \models \bigcirc \varphi_1 \iff (w, i + 1) \models \varphi_1\]
\[(w, i) \models \varphi_1 \cup \varphi_2\]
\[\exists j \geq i \ (w, j) \models \varphi_2 \text{ and } \forall i \leq k \leq j \ (w, k) \models \varphi_2\]

\[w \models \varphi \iff (w, 0) \models \varphi\]
\[T \models \varphi \iff \forall w \in L(T) \ w \models \varphi\]
LTL examples

Two processors want to access a critical section. Each processor can have three observable states:

\[ p_1 = \{\text{inCS, outCS, reqCS}\} \]
\[ p_2 = \{\text{inCS, outCS, reqCS}\} \]

**Mutual exclusion**
Both processors are not in the critical section at the same time.

\[ \square \neg (p_1 = \text{inCS} \land p_2 = \text{inCS}) \]

**Starvation freedom**
If process 1 requests entry, then it eventually enters the critical section.

\[ \square p_1 = \text{reqCS} \Rightarrow \lozenge p_1 = \text{inCS} \]
LTL Model Checking

Given transition system and LTL formula we have

**LTL model checking**

Determine if $T \models \varphi$

- System verified
- Counterexample

The transition system satisfies the formula if all executions satisfy it.

LTL model checking is decidable for finite $T$

**Complexity:** $O(n + m)2^{O(k)}$

- states
- transitions
- formula length
Computational tree logic (CTL)

CTL is based on branching time. Its formulas are evaluated over the tree of trajectories generated from a given state of the transition system.
**Computation tree logic**

**CTL syntax**

The CTL formulas are defined inductively as follows:

<table>
<thead>
<tr>
<th>Atomic propositions</th>
<th>All observation symbols $p$ are formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean operators</td>
<td>If $\varphi_1$ and $\varphi_2$ are formulas then $\varphi_1 \lor \varphi_2$ and $\neg \varphi_1$</td>
</tr>
<tr>
<td>Temporal operators</td>
<td>If $\varphi_1$ and $\varphi_2$ are formulas then $\exists U \varphi_2$, $\exists \bigcirc \varphi_1$ and $\exists \bigboxdot \varphi_1$</td>
</tr>
</tbody>
</table>
Express specifications in computation trees (branching time)

<table>
<thead>
<tr>
<th>Informally</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inevitably next p</td>
<td>$\forall \bigcirc p$</td>
<td><img src="image" alt="Inevitably next p tree" /></td>
</tr>
<tr>
<td>Possibly always p</td>
<td>$\exists \square p$</td>
<td><img src="image" alt="Possibly always p tree" /></td>
</tr>
</tbody>
</table>
CTL Model Checking

Given transition system and CTL formula we have

**CTL model checking**

Determine if $T \models \varphi$

- System verified
- Counterexample

The transition system satisfies the formula if all initial states satisfy it.

**CTL model checking is decidable for finite $T$**

**Complexity:** $O((n + m)k)$

- states transitions
- formula length
Comparing logics

CTL

LTL

CTL*
Dealing with complexity

- Bisimulation
- Simulation
- Language Inclusion
Language Equivalence

Consider two transition systems $T_1$ and $T_2$ over same $\Sigma$ and $O$

$L(T_1) = L(T_2)$
Consider two transition systems $T_1$ and $T_2$ and an LTL formula.

**Language equivalence**

If $L(T_1) = L(T_2)$ then $T_1 \models \varphi \iff T_2 \models \varphi$

**Language inclusion**

If $L(T_1) \subseteq L(T_2)$ then $T_2 \models \varphi \Rightarrow T_1 \models \varphi$

Language equivalence and inclusion are difficult to check.
Language Equivalence $\iff$ CTL equivalence

$T_1$

$T_2$

false $\forall \bigcirc \exists \diamondsuit o_1$ true
Simulation Relations

Consider two transition systems

\[ T_1 = (Q_1, \Sigma, \rightarrow_1, O, \langle \cdot \rangle_1) \]
\[ T_2 = (Q_2, \Sigma, \rightarrow_2, O, \langle \cdot \rangle_2) \]

over the same set of labels and observations. A relation \( S \subseteq Q_1 \times Q_2 \) is called a simulation relation (of \( T_1 \) by \( T_2 \)) if it

1. Respects observations
   
   If \((q, p) \in S\) then \(\langle q \rangle_1 = \langle p \rangle_2\)

2. Respects transitions
   
   If \((q, p) \in S\) and \(q \xrightarrow{\sigma} q'\), then \(p \xrightarrow{\sigma} p'\) for some \((q', p') \in S\)

If a simulation relation exists, then \( T_1 \preceq T_2 \)
Game theoretic semantics

Simulation is a matching game between the systems

\[ T_1 \]

\[ q_0 \]

\[ q_1 \]

\[ q_2 \]

\[ q_3 \]

\[ q_4 \]

\[ o_0 \]

\[ \sigma \]

\[ o_1 \]

\[ o_2 \]

\[ T_2 \]

\[ p_0 \]

\[ p_1 \]

\[ p_3 \]

\[ p_4 \]

\[ o_0 \]

\[ \sigma \]

\[ o_1 \]

\[ o_2 \]

Check that \( T_1 \leq T_2 \) but it is not true that \( T_2 \leq T_1 \)
The parking example

The parking meter

A coarser model

\[ S = \{(0,0), (1, \text{many}), \ldots, (60, \text{many})\} \]
Simulation relations

Consider two transition systems $T_1$ and $T_2$

**Simulation implies language inclusion**

If $T_1 \leq T_2$ then $L(T_1) \subseteq L(T_2)$

Complexity of $L(T_1) \subseteq L(T_2)$ $O((n_1 + m_1)2^{n_2})$

Complexity of $T_1 \leq T_2$ $O((n_1 + m_1)(n_2 + m_2))$
Two important cases

Abstraction

\[ T_1 \leq T_2 \]

\[ T_1 \]

\[ T_2 \]

Refinement

\[ T_1 \leq T_2 \]

\[ T_1 \]

\[ T_2 \]
Consider two transition systems

\[ T_1 = (Q_1, \Sigma, \rightarrow_1, O, \langle \cdot \rangle_1) \]
\[ T_2 = (Q_2, \Sigma, \rightarrow_2, O, \langle \cdot \rangle_2) \]

over the same set of labels and observations. A relation \( S \subseteq Q_1 \times Q_2 \) is called a bisimulation relation if it

1. **Respects observations**
   
   if \((q, p) \in S\) then \(\langle q \rangle_1 = \langle p \rangle_2\)

2. **Respects transitions**
   
   if \((q, p) \in S\) and \(q \xrightarrow{\sigma} q'\), then \(p \xrightarrow{\sigma} p'\) for some \((q', p') \in S\)

   if \((q, p) \in S\) and \(p \xrightarrow{\sigma} p'\), then \(q \xrightarrow{\sigma} q'\) for some \((q', p') \in S\)

If a simulation relation exists, then \(T_1 \equiv T_2\)

---

**Penn**
Bisimulation

Consider two transition systems $T_1$ and $T_2$

Bisimulation

$$T_1 \equiv T_2 \quad \text{if} \quad T_1 \leq T_2 \land T_2 \leq T_1$$

Bisimulation is a symmetric simulation
Strong notion of equivalence for transition systems

CTL* (and LTL) equivalence

If $T_1 \equiv T_2$ then $T_1 \models \varphi \iff T_2 \models \varphi$

If $T_1 \equiv T_2$ then $L(T_1) = L(T_2)$
Special quotients

When is the quotient language equivalent or bisimilar to $T$?

Abstraction

$T/\simeq$

$T \leq T/\simeq$

$T$
Quotient Transition Systems

Given a transition system

\[ T = (Q, \Sigma, \rightarrow, O, \langle \cdot \rangle) \]

and an observation preserving partition \( \approx \subseteq Q \times Q \), define

\[ T/ \approx = (Q/\approx, \Sigma, \rightarrow_{\approx}, O, \langle \cdot \rangle_{\approx}) \]

naturally using

1. Observation Map

\[ \langle P \rangle_{\approx} = o \text{ iff there exists } p \in P \text{ with } \langle p \rangle = o \]

2. Transition Relation

\[ P \overset{\sigma}{\rightarrow}_{\approx} P' \text{ iff there exists } p \in P, p' \in P' \text{ with } p \overset{\sigma}{\rightarrow} p' \]
Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system $T$

When does original system $T$ simulate the quotient system $T/\approx$?
Bisimulation Algorithm

Quotient system $T/\approx$ always simulates the original system $T$

When does original system $T$ simulate the quotient system $T/\approx$?
Transition Systems

A region is a subset of states $P \subseteq Q$

We define the following operators

$$\text{Pre}_\sigma(P) = \{q \in Q | \exists p \in P \; q \xrightarrow{\sigma} p\}$$

$$\text{Pre}(P) = \{q \in Q | \exists \sigma \in \Sigma \; \exists p \in P \; q \xrightarrow{\sigma} p\}$$

$$\text{Post}_\sigma(P) = \{q \in Q | \exists p \in P \; p \xrightarrow{\sigma} q\}$$

$$\text{Post}(P) = \{q \in Q | \exists \sigma \in \Sigma \; \exists p \in P \; p \xrightarrow{\sigma} q\}$$
Bisimulation algorithm

Bisimulation Algorithm

initialize $Q/\sim = \{p \sim q \text{ iff } <q>=<p>\}$
while $\exists P, P' \in Q/\sim$ such that $\emptyset \not\subseteq P \cap Pre(P') \not\subseteq P$

$P_1 := P \cap Pre(P')$
$P_2 := P \setminus Pre(P')$
$Q/\sim := (Q/\sim \setminus \{P\}) \cup \{P_1, P_2\}$
end while

If $T$ is finite, then algorithm computes coarsest quotient.
If $T$ is infinite, there is no guarantee of termination.
Relationships

**Bisimulation**
Strongest, more properties, easiest to check

**Simulation**
Weaker, less properties, easy to check

**Language Inclusion**
Weakest, less properties, difficult to check
Complexity comparisons

- **Bisimulation**
  \[ O(m \cdot \log(n)) \]

- **Simulation**
  \[ O(m \cdot n) \]

- **Language Equivalence**
  \[ O(m \cdot 2^n) \]