Model Checking (MC) & Timed Automata (TA)

Libor Waszniowski
Czech Technical University in Prague
Department of Control Engineering
xwasznio@fel.cvut.cz
How to Improve SW reliability?

<table>
<thead>
<tr>
<th>Method</th>
<th>Model vs. Impl.</th>
<th>Exhaustive</th>
<th>Finite vs. Inf.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peer reviewing</td>
<td>Impl.</td>
<td>No</td>
<td>Inf.</td>
</tr>
<tr>
<td>Testing</td>
<td>Impl.</td>
<td>No</td>
<td>Inf.</td>
</tr>
<tr>
<td>Simulation</td>
<td>Model</td>
<td>No</td>
<td>Inf.</td>
</tr>
<tr>
<td>Model Checking</td>
<td>Model</td>
<td>Yes*</td>
<td>Finite</td>
</tr>
<tr>
<td>Theorem proving**</td>
<td>Model</td>
<td>Yes</td>
<td>Inf.</td>
</tr>
</tbody>
</table>

*MC Checks only modeled cases
**Theorem proving is sometime called formal verification or deductive verification
What the Verification is?

• Many different definitions of verification:
  – Process of obtaining the **formal proof** of a system correctness (*only theorem proving*)
  – Process of applying a technique that is supposed to be **able to establish system correctness** with respect its specification (*theorem proving and model checking*)
  – **Any process** attempt to find errors in a program (*also testing*)
What the Validation is?

• It is sometime distinguished verification and validation:
  – Verification = verifying the conformance between two descriptions of the SW (model and specification)
  – Validation = verifying whether the SW satisfy specification

• Each of these approaches plays an important role in a different phase of the development cycle
Formal Methods

- Collection of **notations and techniques** for describing and analyzing systems
- Based on **mathematical theories**
- Unambiguous interpretation
- Examples:
  - PN + algorithms for analysis of boundness, liveness, reachable markings,....
  - Automata + Model Checking (Promela + Spin)
  - Logic + Theorem proving
What are these?

- UML
- SART

= Formal Description languages or SW development methodologies
  - usually not unambiguous interpretation
  - no analysis algorithms
Concept of Model Checking

• An **automatic** method for verifying **finite** state model properties by **exhaustive** state-space search

```
State trans syst.:
- Finite automata
- Timed automata
- Petri nets
- SDL
- process algebra
- ....
```

```
Real system
modelling
Model of the system (possible behaviours)
Verifier
Counter example
```

```
Specification
formalising
Formal requirements (desired behaviours)
```

```
Counter example
```

```
OK
```

```
- Temporal logics (LTL, CTL...)
- Timed temporal logics (TCTL,....)
- Finite automata
- Timed automata
- ....
```
What the Modeled System is?

- Verified controller interact with its environment
Model of the System

Kripke Structure – oriented graph labeled by atomic propositions

\[ M = (S, \rho, \pi) \]

- \( S \) finite set of states
- \( \rho \subseteq S \times S \) transition function*
- \( \pi : S \rightarrow 2^{AP} \) labelling function
- \( AP \) set of atomic prop.

\*Reactive system \( \Rightarrow \) infinite behavior \( \Rightarrow \rho \) is total relation

inf. runs = \( s_1 (s_3 \ s_1)^* (s_2)^{\omega} \cup (s_1 \ s_3)^{\omega} \)

inf. traces = \( \{a\} (\{b\} \{a\})^* (\{a,b\})^{\omega} \cup (\{a\} \{b\})^{\omega} \)
Specification – Temporal Logic

• **Propositional logic** express static properties of **states**

• **Temporal logic** express dynamic properties of infinite **traces**
  – Linear time (describes inf. sequences)
    • Linear Temporal Logic (LTL)
  – Branching time (describes inf. trees)
    • Computation Tree Logic (CTL)
    • Extended Computation Tree Logic (CTL*)
LTL Syntax

$$\phi ::= a \mid \phi \land \phi \mid \neg \phi \mid \square \phi \mid \phi U \phi$$

$$a \in AP$$

$$\neg, \land$$ are basic propositional logic operators

$$\square, U$$ are basic temporal logic operators (Next, Until)

Derived temporal operators:

- Future (eventually): $$\Diamond \phi \equiv true U \phi$$
- Globally (always): $$\square \phi \equiv \neg \Diamond \neg \phi$$
LTL Semantic

Model of LTL formulas is an infinite sequence:
\[ \xi = (y_0, y_1, y_2, \ldots) \in (2^\text{AP})^\omega \]
\[ \{a\} \{\{b\} \{a\}\}^\omega \{a,b\}^\omega \]
\(\xi^i\) denotes the inf. subsequence of \(\xi\) beginning at the \(i^\text{th}\) successor of \(y_0\) e.g. \(\xi^1 = (y_1, y_2, \ldots)\)
\[ \{b\} \{a\}^\omega \{a,b\}^\omega \]
\(\xi \models \varphi\) states that the sequence \(\xi\) satisfies the formula \(\varphi\)

If \(M\) is a Kripke structure and \(s\) is state
\[ M,s \models \varphi \text{ iff } \xi^i \models \varphi \text{ for all traces } \xi^i \text{ starting at } s \]
Propositional Operators Semantic

\[ \xi \models a \iff a \in y_0 \text{ the first element of } \xi \]

\[ \xi \models \neg \phi \iff \neg (\xi \models \phi) \]

\[ \xi \models (\phi \land \psi) \iff (\xi \models \phi) \text{ and } (\xi \models \psi) \]

<table>
<thead>
<tr>
<th>( \varphi )</th>
<th>( \xi \models \varphi )</th>
<th>( \xi \models \neg \varphi )</th>
</tr>
</thead>
</table>
| \( a \) | \{a, b\} \{b\}.... | \{\} \{b\}...
| \( a \land b \) | \{a, b\} \{b\}... | \{a\} \{b\}
| \( \neg a \) | \{b\} \{a\}... | \{a,b\} \{b\}.... |
Operator “Next” Semantic

\[ \xi \models \circ \phi \quad \text{iff} \quad \xi^1 \models \phi \]

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \xi \models \phi )</th>
<th>( \xi \models \neg \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \circ a )</td>
<td>( {b} {a,b} {b} \ldots )</td>
<td>( {a} {} {a} \ldots )</td>
</tr>
<tr>
<td></td>
<td>( {a} {a} \ldots )</td>
<td>( {b} {b} \ldots )</td>
</tr>
</tbody>
</table>
Operator “Until” Semantic

\[ \xi \models (\varphi U \psi) \iff \exists i \geq 0, \xi^i \models \psi \land \forall j, 0 \leq j < i, \xi^j \models \varphi \]

<table>
<thead>
<tr>
<th>\varphi</th>
<th>\xi \models \varphi</th>
<th>\xi \models \neg \varphi</th>
</tr>
</thead>
<tbody>
<tr>
<td>( aU b )</td>
<td>{a}^* {a,c} {b}...</td>
<td>{a}^* {c} {b}...</td>
</tr>
<tr>
<td></td>
<td>{b}...</td>
<td>{a}^\omega</td>
</tr>
<tr>
<td>( (a \lor b)U(\bigcirc c) )</td>
<td>{a}^* {b} {} {c}...</td>
<td>{a}^* {} {b} {c}...</td>
</tr>
</tbody>
</table>
Operator “Future” Semantic

\( \xi \models \Diamond \varphi \) iff \( \exists i \geq 0, \xi^i \models \varphi \)

<table>
<thead>
<tr>
<th>\varphi</th>
<th>\xi \models \varphi</th>
<th>\xi \models \neg \varphi</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Diamond a</td>
<td>{b} \ast {c} \ast {a} ...</td>
<td>{b} \omega</td>
</tr>
<tr>
<td></td>
<td>{a} ...</td>
<td></td>
</tr>
</tbody>
</table>
Operator “Globally” Semantic

\[ \xi \models \Box \varphi \quad \text{iff} \quad \forall i \geq 0, \, \xi_i \models \varphi \]

<table>
<thead>
<tr>
<th>\varphi</th>
<th>\xi \models \varphi</th>
<th>\xi \models \neg \varphi</th>
</tr>
</thead>
<tbody>
<tr>
<td>\Box a</td>
<td>{a,b} {a}^\omega</td>
<td>{a} *{} {a}^\omega</td>
</tr>
</tbody>
</table>
LTL Formulas Examples

Invariance

□ ¬ (P1_in ∧ P2_in)

Response

□ (P1_req ⇒ ◇ P1_in)

No spontaneous response

□ ((◇ P1_req) ⇒ (¬ P1_in U P1_req))

Order preservation

□(P1_req ∧ ¬P2_req ∧ ◇P2_req ⇒
◇(P1_in ∧ ¬P2_in ∧ ◇P2_in))

Infinitely often

□ ◇ P1_measure
CTL Syntax

\( \varphi ::= a \mid \neg \varphi \mid \varphi \land \varphi \mid \exists \bigcirc \varphi \mid \exists[\varphi U \varphi] \mid \forall[\varphi U \varphi] \)

\( a \in AP \)

\( \neg, \land \) are basic propositional logic operators

\( \bigcirc, U \) are basic temporal logic operators (Next, Until)

\( \exists, \forall \) are quantifiers (existential and universal)

Derived operators:

\( \forall \bigcirc \varphi \equiv \neg \exists \bigcirc \neg \varphi \)

\( \exists \diamond \varphi \equiv \exists[true U \varphi] \)

\( \forall \diamond \varphi \equiv \forall[true U \varphi] \)

\( \exists \square \varphi \equiv \neg \forall \diamond \neg \varphi \)

\( \forall \square \varphi \equiv \neg \exists \diamond \neg \varphi \)
CTL Semantic

Model of CTL is Kripke structure:
\[ M = (S, \rho, \pi) \]

\[ P_M(s) = \{(s_0, s_1, s_2, \ldots) \in S^\omega \mid s_0 = s \text{ and } (s_n, s_{n+1}) \in \rho \text{ for all } n \geq 0\} \]

is set of all runs in \( M \), starting in \( s \)

\[ P_M(s_1) = (s_1 s_3)^\omega \cup s_1 (s_3 s_1)^* (s_2)^\omega \]

If \( \sigma \in P_M(s) \), \( \sigma[i] \) denotes the \( i \)-th element of the run \( \sigma \), therefore \( \sigma[0]=s \)

\[ M, s \models \varphi \] states that state \( s \) of Kripke structure \( M \) satisfies the formula \( \varphi \)
CTL Operators Semantic

\[ M, s \models a \iff a \in \pi(s) \]
\[ M, s \models \neg \varphi \iff \neg (M, s \models \varphi) \]
\[ M, s \models (\varphi \land \psi) \iff (M, s \models \varphi) \text{ and } (M, s \models \psi) \]
\[ M, s \models \exists \bigcirc \varphi \iff \exists \sigma \in P_M(s), \sigma[1] \models \varphi \]
\[ M, s \models \exists[\varphi U \psi] \iff \exists \sigma \in P_M(s), \]
\[ (\exists i \geq 0 \sigma[i] \models \psi \land (\forall j, 0 \leq j < i, \sigma[j] \models \varphi)) \]
\[ M, s \models \forall[\varphi U \psi] \iff \forall \sigma \in P_M(s), \]
\[ (\exists i \geq 0 \sigma[i] \models \psi \land (\forall j, 0 \leq j < i, \sigma[j] \models \varphi)) \]
Examples of CTL Formulas

∀[aUb]

∀ □ a

∀ □ a

∃ □ a

∃ □ a

∀ [aU∀[¬aUb]]
LTL vs. CTL - Syntax
CTL vs. LTL - Semantic

Same traces \((\{a\}{a}(\{b\})^\omega \cup \{a\}{a}(\{c\})^\omega)\)
⇒ LTL can not distinguish models – LTL express path properties

Different runs
⇒ CTL formula \(\forall \bigcirc (\exists \bigcirc b \land \exists \bigcirc \neg b)\) distinguish models
CTL express states properties ⇒ it can not express rich properties along the paths.
Properties

- Safety
- Liveness
- State Reachability
- Deadlock freeness
- Fairness
MC Algorithms

• Differ for different properties and formalisms
  – CTL – marking model states by sub-formulas satisfied in the state
  – LTL – translating $\neg \phi$ to Buchy automaton $B_{\neg \phi}$ and than checking emptiness of language accepted by $M \otimes B_{\neg \phi}$

• State-space explosion is prevented by symbolic state-space representation based on Binary Decision Diagrams (BDD)
And What About the Real Time?

Kripke Structure and temporal logics do not express quantitative time

\[ \square (P1\_req \Rightarrow \Diamond P1\_in) \]

- RT systems correctness depends on the quantitative time
- \( \Rightarrow \) Timed Automata (TA) and TCTL etc.
Notion of the Time

• Discrete time
  – time is represented by positive integer number (discrete steps)
  – appropriate only for synchronous systems (no events between successive steps)

• Continuous (dense) time
  – time is represented by positive real number
  – nature of real (non-artificial) systems
Timed Automata

- Finite state automaton extended by real valued clocks
- Clocks are assumed to proceed at the same rate
- Clocks can be tested and reset
- Clock Constraints
  
  For set $C$ of clocks, with $x, y \in C$, the set of clock constraints over $C$, $\mathcal{P}(C)$, consist of formulas defined by
  
  $\alpha ::= x \sim c \mid x-y \sim c \mid \neg \alpha \mid (\alpha \land \alpha)$

  $c \in \mathbb{N}$ (natural number)

  relation $\sim$ is one of $\{\leq, \geq, =, <, >\}$. 
Timed Automata Syntax

\[ A = (L, l_0, E, I, V) \]

- \( L \) is a finite set of locations
- \( l_0 \) is the initial location
- \( E \subseteq L \times \Psi(C) \times \text{Act} \times 2^C \times L \) is the set of edges
  - \((l, g, a, r, l') \in E\) represents an edge from the location \( l \) to the location \( l' \) with clock constraint \( g \), action \( a \) and the set of clocks \( r \) to be reset
- \( I: L \rightarrow \Psi(C) \) is the invariant condition of the location
- \( V: L \rightarrow 2AP \) is an observation function
Example of Timed Automata

Light control system

- **Idle**
  - \( y \leq 50 \)
  - \( y := 0 \)
  - press!

- **Off**
  - \( x := 0 \)
  - press?

- **Light**
  - \( x \leq 3 \)
  - press?

- **Bright**
  - press?
Semantic of Timed Automata

infinite timed transition system

\[ S_A = (S, s_0, \rightarrow, V) \]

- **S** is the set of states - pairs \((l, u)\) where \(l \in L\) and \(u\) are clock assignment for \(C\).
- **s_0** is the initial state of \(A\) \((l_0, u_0)\), where \(u_0\) mapping all clocks to \(0\).
- **\(\rightarrow\)** is the transition relation:
  - **action transition**: \((l, u) \xrightarrow{a} (l', u')\)
  - **delay transition**: \((l, u) \xrightarrow{d} (l, u+d)\)

**V**: \(S \rightarrow 2^{AP}\) is an observation function
Example of the Timed Automata Semantic

(Off, 0) \xrightarrow{\text{press}} (Light, 0) \xrightarrow{5.3} (Light, 5.3) \xrightarrow{\text{press}} (Off, 5.3)

(Off, 5.3) \xrightarrow{47.4} (Off, 52.7) \xrightarrow{\text{press}} (Light, 0) 

..............
Timed Automata Model Checking

**Infinite state-space** of TA

\[ \Downarrow \]

**Finite representation** by Region Graph

Clock valuation is abstracted by clock regions
States in clock region are equivalent from the verification point of view
- satisfy the same constraints
- have the same future
Region Equivalence

Two clock assignments $u, v$ are region-equivalent iff:

1. for all $x \in C$, either $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$ or both $u(x) > \text{Ceil}_x$ and $v(x) > \text{Ceil}_x$,

2. for all $x \in C$, if $u(x) \leq \text{Ceil}_x$ then $\{u(x)\} = 0$ iff $\{v(x)\} = 0$,

3. for all $x, y \in C$, if $u(x) \leq \text{Ceil}_x$ and $u(y) \leq \text{Ceil}_y$ then $\{u(x)\} \leq \{u(y)\}$ iff $\{v(x)\} \leq \{v(y)\}$

- $\text{Ceil}_x$ is the largest constant that clock $x$ is compared with.
- $\{d\}$ denote the fractional part of $d$
- $\lfloor d \rfloor$ denote the integer part of $d$
Clock Regions

TA with one clock $x$

$x=0 \quad 0<x<1 \quad x=1 \quad 1<x<2 \quad x=2 \quad 2<x<3 \quad x=3 \quad 3<x$

TA with two clocks $x$ and $y$
Region Graph $R_A$

State of TA ($l,u$)

$\Downarrow$

State of $R_A$ ($l,[u]$)

$[u]$ is region containing clock assignment $u$

Transitions of $R_A$ corresponds to transitions of TA
Example of Region Graph
Size of a Region Graph

• Number of regions is exponential on number of clocks and ceilings of clocks

• In MC tools is used more efficient symbolic representation based on clock zones
Clock Zones Based Symbolic Representation

Clock Zone = polyhedron bounded by linear inequalities on clocks in the form:

\[ \bigwedge_{0 \leq i \neq j \leq n} x_i - x_j \sim c_{i,j} \]

– where special clock \( x_0 = 0 \)
– \( n \) is the number of clocks
– relation \( \sim \) is one of \( \{ \leq, < \} \).

• The smallest possible clock zones are clock regions.
Operations over clock zones

- Intersection ($\varphi \land \psi$)
  - conjunction of clock constraints of $\varphi$ and $\psi$

- Reset of set of clocks $r$, ($\varphi[r:=0]$)
  - projection to the $r=0$

- Elapsing of time ($\varphi\uparrow$)
  - set of all clock assignments $u + d$, where $u \in \varphi$ and $d \in \mathbb{R}_{\geq 0}$
Zone Graph

Nodes of a zone graph $Z_A$ are zones $(l, \varphi)$, The initial node is $(l_0, [C:=0])$

The transition relation in $Z_A$ is defined as follow:

• $(l, \varphi) \xrightarrow{a} (l', (\varphi \land g)[r:=0] \land l(l'))$ in $Z_A$ for each edge $e=(l, g, a, r, l')$ of $A$.

• $(l, \varphi) \xrightarrow{\uparrow} (l, \varphi \uparrow \land l(l))$ in $Z_A$ for each location $l$ of $A$.

During $Z_A$ construction zones normalization must be applied
Example of Zone Graph

- **Off**
  - press?
  - $x := 0$
  - press?
  - $x > 3$

- **Light**
  - press?
  - $x \leq 3$

- **Bright**

- **Off**
  - $3 < x$

- **Off**
  - $0 \leq x$

- **Off**
  - $x = 0$

- **Light**
  - press?
  - $0 \leq x$

- **Light**
  - $x = 0$

- **Bright**
  - $x \leq 3$

- **Bright**
  - $x = 0$
Reachability Analysis Algorithm

Passed:=\{\}, Wait:={((l_0, [C:=0]))}

while Wait \neq \{\} do
    take \((l, \varphi)\) from Wait
    if \(l=l_f\) and \(\varphi \land \varphi_f \neq \{\}\) then return “YES”
    if not \(\varphi \subseteq \varphi’\) for all \((l, \varphi’)\) \in Passed then
        add \((l, \varphi)\) to Passed
        for all \((l’, \varphi’)\) such that \((l, \varphi) \rightarrow (l’, \varphi’)\) do
            add \((l’, \varphi’)\) to Wait
        end for
    end if
end while
return “NO”
## Difference Bound Matrices

- An efficient way to represent a clock zone
- Each entry $D_{i,j}$ has the form $(d_{i,j}, \sim_{i,j})$
  - represents the bound of difference of clocks $i$ and $j$
  - defined by inequality $x_i - x_j \sim_{i,j} d_{i,j}$, where the relation $\sim_{i,j}$ is one of \{≤, <\}.
- $D_{i,j}$ is $(\infty, <)$, if no such bound is known

<table>
<thead>
<tr>
<th>Expression</th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 - x_2 &lt; 2 \land 0 &lt; x_2 \leq 2 \land 1 \leq x_1$</td>
<td>$(0, \leq)$</td>
<td>$(-1, \leq)$</td>
<td>$(0, &lt;)$</td>
</tr>
<tr>
<td>$x_0$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1$</td>
<td></td>
<td>$(\infty, &lt;)$</td>
<td>$(0, \leq)$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$(2, \leq)$</td>
<td>$(\infty, &lt;)$</td>
<td>$(0, \leq)$</td>
</tr>
</tbody>
</table>
Timed Temporal Logic

CTL: $\exists \diamond a$ no explicit time

TCTL: $\exists \diamond_{<5} a$ explicit time limits of temporal operators

$\phi ::= a \mid \phi \land \phi \mid \neg \phi \mid \exists \Diamond \phi \mid \exists[\phi U_{\sim c} \phi] \mid \forall[\phi U_{\sim c} \phi]$

~ is one of binary relations $<, \leq, =, \geq, >$

$a \in AP$

$c \in N$
Further Studying


