Disastrous software bugs

Mariner 1 probe, 1962
See http://en.wikipedia.org/wiki/Mariner_1
- Destroyed 293 seconds after launch
- Missing hyphen in the data or program? No!
- Overbar missing in the mathematical specification:
  \[ \dot{R}_n \] \( n \)th smoothed value of the time derivative of a radius.
Without the smoothing function indicated by the bar, the program treated normal minor variations of velocity as if they were serious, causing spurious corrections that sent the rocket off course.

Disastrous software bugs

Spirit Rover (Mars Exploration), 2004
- Ceased communicating on January 21.
- Flash memory management anomaly: too many files on the file system
- Resumed to working condition on February 6.

Disastrous software bugs

Other well-known bugs
- Therac-25, at least 3 death by massive overdoses of radiation.
  Race condition in accessing shared resources.
- Electricity blackout, USA and Canada, 2003, 55 millions people.
  Race condition in accessing shared resources.
  Flaw in the division algorithm, discovered by Thomas Nicely.
  See http://en.wikipedia.org/wiki/Pentium_FDIV_bug
- Needham-Schroeder, authentication protocol based on symmetric encryption.
  Published in 1978 by Needham and Schroeder
  Proved correct by Burrows, Abadi and Needham in 1989
  Flaw found by Lowe in 1995 (man in the middle)
  Automatically proved incorrect in 1996.
  See http://en.wikipedia.org/wiki/Needham-Schroeder_protocol

Need for formal verification methods

Critical systems
- Transport
- Energy
- Medicine
- Communication
- Finance
- Embedded systems
- ...

Formal verifications methods

Based on
- a formal model of the system
- a formal semantics

Complementary approaches
- Theorem prover
- Model checking
- Static analysis
- Test

Transition system or Kripke structure

Definition: TS
\[ M = (S, \Sigma, T, I, A \Sigma, \ell) \]

- \( S \): set of states (finite or infinite)
- \( \Sigma \): set of actions
- \( T \subseteq S \times \Sigma \times S \): set of transitions
- \( I \subseteq S \): set of initial states
- \( A \Sigma \): set of atomic propositions
- \( \ell : S \to 2^{A \Sigma} \): labelling function.

Every discrete system may be described with a TS.

Example: Digicode ABA
All non trivial problems about Turing machines are undecidable

Rice, 1953

→

→formal verification methods must rely on less expressive formalisms

Undecidable problems

Halting problem for Turing machines [Turing, 1936]

- Assume \( P : (Q, x) \mapsto \{\text{true}, \text{false}\} \) such that \( P(Q, x) \) halts if \( P(Q) \) does not halt.
- Build \( R \) such that \( R(Q) \) halts if \( Q \) does not halt.
- Then \( R(R) \) does not halt.

Rice, 1953

All non trivial problems about Turing machines are undecidable

→formal verification methods must rely on less expressive formalisms

Model Checking

3 steps

- Constructing the model \( M \) (transition systems)
- Formalizing the specification \( \varphi \) (temporal logics)
- Checking whether \( M \models \varphi \) (algorithmics)

Main difficulties

- Size of models (combinatorial explosion)
- Expressivity of models or logics
- Decidability and complexity of the model-checking problem
- Efficiency of tools

Challenges

- Extend models and algorithms to cope with more systems.

Example: Man, Wolf, Goat, Cabbage

- Infinite systems, parameterized systems, probabilistic systems, concurrent systems, timed systems, hybrid systems, . . . See Modules 2.8 & 2.9
- Scale current tools to cope with real-size systems.
- Needs for modularity, abstractions, symmetries, . . .

Description Languages

Pb: How can we easily describe big systems?

- Programming languages
- Boolean circuits
- Modular description, e.g., parallel compositions
- problems: concurrency, synchronization, communication, atomicity, fairness, . . .
- Petri nets (intermediate level)
- Transition systems (intermediate level)
- with variables, stacks, channels, . . . synchronized products
- Logical formulae (low level)

Operational semantics

High level descriptions are translated (compiled) to low level (infinite) TS.
### Transition systems with variables

**Definition:** TSV

\[ M = \langle S, \Sigma, \nu, (D_v)_{v \in \nu}, T, I, AP, \ell \rangle \]

- \( V \) set of (typed) variables, e.g., boolean, \([0,4]\), \(N\), ..
- Each variable \( v \in V \) has a domain \( D_v \) (finite or infinite). Let \( D = \prod_{v \in V} D_v \).
- Guard or Condition \( g \) with semantics \( [g] \subseteq D \) (unary predicate)
- Instruction or Update \( f \) with semantics \( [f] : D \rightarrow D \)
- Symbolic descriptions: \( x < 5 \), \( x + y = 10 \), ..
- Transition system: \( T \subseteq (S \times \Sigma \times \tau \times S) \)
- Symbolic descriptions: \( s \times \epsilon \cup \{ (x, y) \mid x = y + \text{coin} \} \rightarrow s' \)
- \( I \subseteq S \times \tau \times S \)
- Symbolic descriptions: \( (s_0, x = 0) \)

**Example:** Vending machine

- coffee: 50 cents, orange juice: 1 euro, ..
- possible coins: 10, 20, 50 cents
- we may shuffle coin insertions and drink selection

---

### Symbolic representation

**Example:** Logical representation

\( s = 1 \land \text{cpt} < n \land s' = 1 \land \text{cpt}' = \text{cpt} + 1 \)

\( \lor \ s = 1 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 \)

\( \lor \ s = 2 \land s' = 3 \land \text{cpt}' = \text{cpt} \)

\( \lor \ s = 3 \land \text{cpt} < n \land s' = \text{cpt}' = \text{cpt} + 1 \)

\( \lor \ s = 3 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 \)

### Synchronized products

**Definition:** General product

- Components: \( M_i = \langle S_i, \Sigma_i, T_i, I_i, AP_i, \ell_i \rangle \)
- Product: \( M = \prod_{i} M_i \)
- Various semantics for the parallel composition \( \parallel \)
- Various communication mechanisms between components:
  - Shared variables, FIFO channels, Rendez-vous, ..
- Various restrictions

**Atomic propositions** are inherited from the local systems.

**Example:** Elevator with 1 cabin, 3 doors, 3 calling devices

- Cabin: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Door for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Call for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)

The actual system is a synchronized product of all these automata. It consists of \((at most) 3 \times 2^3 \times 2^3 = 192 \) states.

---

### Table: Transition systems with variables

<table>
<thead>
<tr>
<th>States</th>
<th>Symbols</th>
<th>Guard</th>
<th>Transition</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_0 )</td>
<td>( \text{cpt} &lt; n )</td>
<td>( \text{cpt} &lt; n )</td>
<td>++</td>
<td>( s_1 )</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>( \text{cpt} &lt; n )</td>
<td>( \text{cpt} &lt; n )</td>
<td>+</td>
<td>( s_2 )</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>( \text{cpt} &lt; n )</td>
<td>( \text{cpt} &lt; n )</td>
<td>+</td>
<td>( s_3 )</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>( \text{cpt} &lt; n )</td>
<td>( \text{cpt} &lt; n )</td>
<td>+</td>
<td>( s_4 )</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>( \text{cpt} &lt; n )</td>
<td>( \text{cpt} &lt; n )</td>
<td>+</td>
<td>( s_5 )</td>
</tr>
<tr>
<td><strong>ERROR</strong></td>
<td><strong>ERROR</strong></td>
<td><strong>ERROR</strong></td>
<td><strong>ERROR</strong></td>
<td><strong>ERROR</strong></td>
</tr>
</tbody>
</table>

---

### Example: Digicode

- Opened state: \( 1, 0 2, 0 3, 0 4, 0 \)
- Closed state: \( 1, 1 2, 1 3, 1 4, 1 \)
- Error state: \( 5, 3 \)

---

### Example: Logical representation

- Transition system: \( T \subseteq (S \times \Sigma \times \tau \times S) \)
- Symbolic descriptions: \( s \times \epsilon \cup \{ (x, y) \mid x = y + \text{coin} \} \rightarrow s' \)
- \( I \subseteq S \times \tau \times S \)
- Symbolic descriptions: \( (s_0, x = 0) \)

---

### Programs = Kripke structures with variables

- Program counter = states
- Instructions = transitions
- Variables = variables

---

### Modular description of concurrent systems

\[ M = M_1 \parallel M_2 \parallel \cdots \parallel M_n \]

- Various semantics for the parallel composition \( \parallel \)
- Various communication mechanisms between components:
  - Shared variables, FIFO channels, Rendez-vous, ..
- Various restrictions

**Atomic propositions** are inherited from the local systems.

**Example:** Elevator with 1 cabin, 3 doors, 3 calling devices

- Cabin: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Door for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Call for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)

The actual system is a synchronized product of all these automata. It consists of \((at most) 3 \times 2^3 \times 2^3 = 192 \) states.

---

### Transition systems with variables

**Semantics:** low level TS

- \( S' = S \times D \)
- \( I' = \{ (s, \nu) \mid \exists (s, g) \in I \text{ with } \nu \models g \} \)
- Transitions: \( T' \subseteq (S \times D) \times \Sigma \times (S \times D) \)

** SOS:** Structural Operational Semantics

- \( AP^+: \) we may use atomic propositions in \( AP \) or guards such as \( x > 0 \).

---

### Example: Logical representation

\[ \delta_B = \begin{cases} s = 1 \land \text{cpt} < n \land s' = 1 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 1 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 2 \land s' = 3 \land \text{cpt}' = \text{cpt} & \lor \ s = 3 \land \text{cpt} < n \land s' = 1 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 3 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 \end{cases} \]

---

### Symbolic representation

**Example:** Logical representation

\[ \delta_B = \begin{cases} s = 1 \land \text{cpt} < n \land s' = 1 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 1 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 2 \land s' = 3 \land \text{cpt}' = \text{cpt} & \lor \ s = 3 \land \text{cpt} < n \land s' = 1 \land \text{cpt}' = \text{cpt} + 1 & \lor \ s = 3 \land \text{cpt} = n \land s' = 5 \land \text{cpt}' = \text{cpt} + 1 \end{cases} \]

---

### Synchronized products

**Definition:** General product

- Components: \( M_i = \langle S_i, \Sigma_i, T_i, I_i, AP_i, \ell_i \rangle \)
- Product: \( M = \prod_{i} M_i \)
- Various semantics for the parallel composition \( \parallel \)
- Various communication mechanisms between components:
  - Shared variables, FIFO channels, Rendez-vous, ..
- Various restrictions

**Atomic propositions** are inherited from the local systems.

**Example:** Elevator with 1 cabin, 3 doors, 3 calling devices

- Cabin: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Door for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)
- Call for level: \( \begin{bmatrix} \hline 0 & 1 & 2 \hline \end{bmatrix} \)

The actual system is a synchronized product of all these automata. It consists of \((at most) 3 \times 2^3 \times 2^3 = 192 \) states.
Example: Printer manager

Example: Asynchronous product
Restriction on states: \((P, P)\) is forbidden

\[
\begin{array}{c}
\text{Idle} \quad \text{Wait} \quad \text{Print} \\
\text{Idle} \quad \text{Wait} \quad \text{Print} \\
\text{Idle} \quad \text{Wait} \quad \text{Print} \\
\text{Idle} \quad \text{Wait} \quad \text{Print}
\end{array}
\]

Example: Digicode

Example: Synchronous product
Restriction on transitions

\[
\begin{array}{c}
\text{OPEN} \quad \text{A} \quad \text{B} \\
\text{OPEN} \quad \text{A} \quad \text{B} \\
\text{OPEN} \quad \text{A} \quad \text{B} \\
\text{OPEN} \quad \text{A} \quad \text{B}
\end{array}
\]