SS 2017
Software Verification
Automated Verification

Prof. Dr. Holger Schlingloff\textsuperscript{1,2}
Dr. Esteban Pavese\textsuperscript{1}

\textsuperscript{(1)} Institut für Informatik der Humboldt Universität
\textsuperscript{(2)} Fraunhofer Institut für offene Kommunikationssysteme FOKUS
A Note about Laptop Usage

From: Carrie B. Fried, In-class laptop use and its effects on student learning; Elsevier Journal on Computers and Education, 2007:

- This research raises serious concerns about the use of laptops in the classroom. ... The use of laptops was negatively related to several measures of learning. The pattern of the correlations suggests that laptop use interfered with students’ abilities to pay attention to and understand the lecture material, which in turn resulted in lower test scores. The results of the regression analysis clearly show that success in the class was negatively related to the level of laptop use.

- Ultimately ... these results clearly demonstrate that the use of laptops can have serious negative consequences. ... Laptop use is negatively associated with student learning and it poses a distraction to fellow students.
Some Questions for Recap

- What are problems with material implication?
- Difference between \((p \rightarrow q)\) and \((p \rightarrow □q)\)?
- Explain the formula \((\neg □p \rightarrow \Box \neg p)\)!
- Syntax and semantics of modal logic?
- Which statements are valid (in \(K\)):  
  - \(\vdash (p \rightarrow □p)\)
  - \(\vdash (p \rightarrow \Box p)\)
  - \(\vdash (□p \rightarrow p)\)
  - \(p \vdash □p\)
  - \(p \vdash \Box p\)
  - \(□p \vdash p\)
- Relation between \(ML, PL, FOL\)?
Finite State Machines

A Finite State Machine (FSM) is a tuple $M = \langle S, s_0, A, \mathcal{AP}, R, V \rangle$ where

- $S$ is a non-empty, finite set of states
- $s_0 \in S$ is the initial state
- $A$ is a finite set of action labels
- $\mathcal{AP}$ is the set of propositional variables
- $R \subseteq S \times A \times S$ is the (labelled) transition relation
- $V: \mathcal{AP} \rightarrow 2^S$ is the valuation function

Essentially, an FSM is a Kripke Structure where the edges are labelled with an action
FSM: Semantics

- **Trace semantics:** Consider a (finite or infinite) sequence \( \sigma = (s_0, a_1, s_1, a_2, s_2, \ldots) \). We say that \( \sigma \) is *generated* by the FSM \( M \), or \( \sigma \) is a *trace* of \( M \), iff
  - \( s_0 \) is the initial state of \( M \), and
  - for all \( i \), \((s_i, a_{i+1}, s_{i+1}) \in R\)

- A trace is *maximal* if it is infinite, or finite with last state \( s_n \), and there is no \( a \) and \( s \) such that \((s_n, a, s) \in R\)

- **Tree semantics:** The set of all finite traces of \( M \) can be seen as the nodes of a (finite or infinite) tree \( T(M) \):
  - The root of \( T(M) \) is \((s_0)\)
  - The children of \((s_0, a_1, \ldots, s_n)\) are all traces \((s_0, a_1, \ldots, s_n, a_{n+1}, s_{n+1})\)

- Maximal traces are either leafs or infinite branches
FSM and ML

- Essentially, an FSM is a Kripke structure where edges are labelled with an action.
- Can we describe FSMs in ML?
- Add action labels to modalities: Multi-modal logic
- Syntax:

  \[
  \text{MML ::= P | } \bot | (\text{MML} \rightarrow \text{MML}) | \langle A \rangle \text{MML}
  \]

- As usual, \([a] \varphi = \neg \langle a \rangle \neg \varphi\)

- Semantics: Let \(\mathbf{M}\) be an FSM and \(s \in S\)
  - \((\mathbf{M},s) \models \langle a \rangle \varphi\) iff \(\exists s' \in S\) s.t. \((s,a,s') \in R\) and \((\mathbf{M},s') \models \varphi\)
  - \(\mathbf{M} \models \varphi\) iff \((\mathbf{M},s_0) \models \varphi\)
Some exercises on MML

\[
\begin{align*}
M &\vdash (\text{START} \rightarrow \Diamond \text{RED}) \, ? \\
M &\vdash (\text{RED} \rightarrow \Box \text{RED}) \, ? \\
M &\vdash (\text{START} \rightarrow \Diamond \Diamond \Box \text{RED}) \, ? \\
M &\vdash (\Box ( \text{RED} \lor \text{BLUE} \lor \text{GHOST}) \, ? \\
M &\vdash (\text{BLUE} \rightarrow \langle \text{collision} \rangle \text{GHOST}) \, ? \\
M &\vdash (\text{BLUE} \rightarrow [\text{collision}] \text{GHOST}) \, ? \\
M &\vdash (\text{BLUE} \land [\text{collision}] \downarrow \rightarrow \langle \text{update} \rangle \langle \text{update} \rangle \text{BLUE}) \, ? \\
\end{align*}
\]
FSM: Variants

- Labelled transition system LTS
- Finite transition system / Finite graph
- Kripke structure / Kripke model
- Nondeterministic finite automaton NFA
- Deterministic finite automaton DFA
- Finite-state program
Parallel FSMs

- Consider $M_1 = \langle S_1, s_{1,0}, A_1, P_1, R_1, V_1 \rangle$ and $M_2 = \langle S_2, s_{2,0}, A_2, P_2, R_2, V_2 \rangle$, where $P_1 \cap P_2 = \emptyset$. Define $M = M_1 || M_2$ by
  - $S = S_1 \times S_2$
  - $s_0 = \langle s_{1,0}, s_{2,0} \rangle$
  - $A = A_1 \cup A_2$
  - $P = P_1 \cup P_2$
  - $V(p) = \{ \langle s_1, s_2 \rangle \mid p \in P_1 \text{ and } s_1 \in V_1(p) \text{ or } p \in P_2 \text{ and } s_2 \in V_2(p) \}$
  - $\langle \langle s_1, s_2 \rangle, a, \langle s'_1, s'_2 \rangle \rangle \in R$ iff
    - $a \in A_1 \setminus A_2$, $\langle s_1, a, s'_1 \rangle \in R_1$, and $s_2 = s'_2$, or
    - $a \in A_2 \setminus A_1$, $\langle s_2, a, s'_2 \rangle \in R_2$, and $s_1 = s'_1$, or
    - $a \in A_1 \cap A_2$, $\langle s_1, a, s'_1 \rangle \in R_1$, and $\langle s_2, a, s'_2 \rangle \in R_2$

- Intuitively, $M_1$ and $M_2$ “synchronize on shared actions”
An embedded-systems example

A video camcorder

- owner's manual almost incomprehensible
- can be found in the internet
- typical for such devices

- Multifunctional on-off switch:
  - **up**: off
  - **down**: cycles through "tape", "memory" and "play/edit" mode

- Intuitively, tape mode is for video, memory mode for pictures and play mode for viewing recorded material
How can we formally describe the behaviour of this switch?
(Natural language description is ambivalent: What does "cycle" mean? What if I push dn-dn-up-dn?)

Modelling by finite state machine:

- **States:** \(\{\text{off, tape, memory, play}\}\)
- **Transition relations:** \(\{\text{up, dn}\}\)

![Finite State Machine for VCR Switch](image)
Parallel FSM for VCR Switch

- Our FSM models the behaviour of the camera in reaction to the signals it receives from the switch
- Must be put in parallel with model of switch
- Every model abstracts from details
  (e.g., there is a little green button within the switch which arrests it in the "off" position)
Hierarchical Parallel FSM

- Structuring the model can be done by grouping all operating modes in one “on”-state
- An “up”-transition brings the camera to state “off” from all sub-states of “on”
- Additionally modelled here: Power failure
Extended FSMs

- Allow variables on arbitrary finite domains in states
  - $\mathcal{V} = (x_1, \ldots, x_n)$, $\mathcal{D} = (D_1, \ldots, D_n)$
  - Valuation $\mathbf{V}: S \times \mathcal{V} \rightarrow \mathcal{D}$
  - Propositional variables can be seen as special case ($\mathcal{D} = \{true, false\}$)
- Allow transitions to *update* variables
  - $\langle s, a/x:=t, s' \rangle \in \mathcal{R}$ implies that $\mathbf{V}(s', x) = \mathbf{I}(t)$
- Allow transitions to *depend on* variables
  - let $\varphi$ be a (propositional) formula
  - $\langle s, a[\varphi], s' \rangle \in \mathcal{R}$ implies that $s \models \varphi$
- Build *equivalence classes* of states
  - a *mode* is a set of states only differing in the valuation of one or more variables
  - usually, a bubble in a graph represents a mode, not a state
What is a „State“

- Wikipedia:
  - **State** (**computer science**), a unique configuration of information in a program or machine
  - **Program state**, in computer science, a snapshot of the measure of various conditions in the system
  - a program is described as stateful if it is designed to remember preceding events or user interactions; the remembered information is called the state of the system

- Propositional viewpoint: A state is a complete description of the current value of variable properties of a system
  - properties might be physical measurables such as weight, temperature, shape, ...
  - or they might be the current value of the bits in memory (variable value assignment)
  - properties may change over time (thus they are variable)
  - properties may be observable or unobservable / only indirectly observable
UML State Machines – States

- UML states are really modes (sets of states)
- A UML state models a situation during which (usually implicit) invariant conditions hold
  - e.g. waiting for an event to occur
  - e.g. performing some behavior
- Associated with each state may be
  - entry, do and exit actions
  - constraints (=state invariants)
- Pseudostates
  - initial, history, fork, join, junction, choice, entry, exit, terminate, (final)
UML State Machines – Transitions

- A transition is a directed relationship between a source vertex and a target vertex
- Labels consist of
  
  Trigger [Guard] / Action

  where
  - trigger is a transition label (i/o-event)
  - guard is a logical formula on internal variables
  - action is an update of the variables

- “Run-to-completion” semantics
  - if an action is also a trigger, it will be processed before the next external trigger is taken into consideration
  - maintaining an “event pool” during execution
State Machine Meta Model

H. Schlingloff, E. Pavese SS2017: Software-Verifikation