Software Systems Verification and Validation

Lecture 10 - Model checking

Lect. dr. Andreea Vescan

Babeș-Bolyai University
Cluj-Napoca

2015-2016

System verification

- Information and Communication Technology (ICT)
- Correct ICT systems
  - It is all about money.
  - It is all about safety.
- The reliability of the ICT systems is a key issue in the system design process [KB08].
- System verification techniques

Schematic view of an posteriori system verification

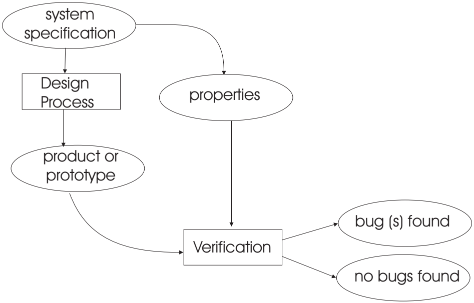
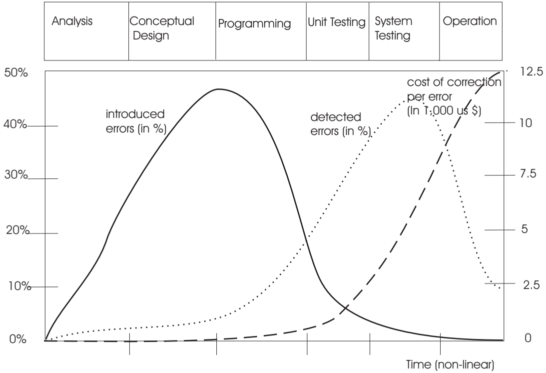


Figure: Schematic view of an posteriori system verification

Software verification techniques

- Non-formal verification techniques (used in practice): peer reviewing (SSVV Lecture 01) and testing (SSVV Lecture 2-3).
  - Peer review (static technique)
  - Software testing (Dynamic technique)
- Formal verification techniques.
  - Formal methods
  - Model-based simulation
  - Model checking(SSVV Lecture 10- today)
  - Model-based testing
  - Theorem proving (SSVV Lecture 06 and 07)

Catching software bugs: the sooner, the better [KB08]



Formal methods

- More time and effort spend on verification than on construction - in software/hardware design of complex systems.
- The role of formal methods:
  - To establish system correctness with mathematical rigor.
  - To facilitate the early detection of defects.
- Brands of verification technique:
  - **deductive methods** - the correctness of system is determined by properties in a mathematical theory, using tools as theorem provers and proof checkers
  - **model-based techniques**: model checking (exhaustive exploration), simulation (restrictive set of scenarios in the model), model-based testings, etc
- Any verification using model-based techniques is only as good as the model of the system.

Model checking

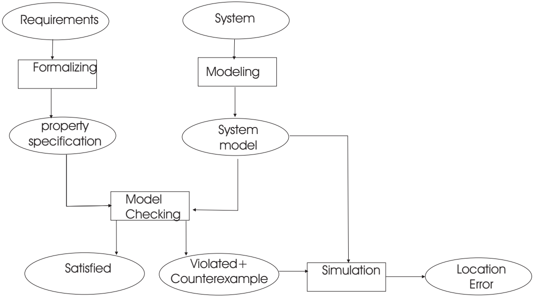


Figure: Schematic view of the model checking approach [KB08]

Characteristics of Model Checking

- Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model.
- The model checking process
  - Modeling phase
    - model the system under consideration
    - formalize the property to be checked.
  - Running phase
  - Analysis phase
    - property satisfied?
    - property violated?

Strengths and Weaknesses of model checking [KB08]

Strengths

- General verification approach
- Supports partial verification
- Provides diagnostic information
- Potential "push-button" technology
- Increasing interest by industry
- Easily integrated in existing development cycles

Weaknesses

- Appropriate to control-intensive applications
- Its applicability is subject to decidability issues
- It verifies a system model
- Checks only stated requirements
- Suffers from the state-space explosion problem
- Requires some expertise

Transition system

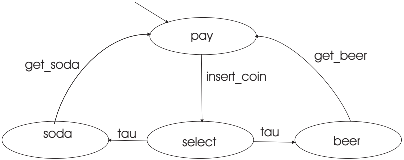
- Transition systems - used in computer science as models to describe the behavior of the systems.
- Transition systems - directed graphs:
  - Nodes - represent states;
  - Edges - model transitions, i. e. state changes.
- A Transition System (TS) is tuple (S, Act, $\rightarrow$, I, Ap, L), where
  - S is a set of states,
  - Act is a set of actions,
  - $\rightarrow \subseteq S \times Act \times S$ is a transition relation,
  - $I \subseteq S$ is a set of initial states,
  - AP is a set of atomic propositions, and
  - $L : S \rightarrow 2^{AP}$ is a labeling function.
- TS is called finite if S, Act and AP are finite.

Transition system - remarks

- Intuitive behavior of a transition system
  - Initial state $s_0 \in I$
  - Using the transition relation $\rightarrow$ the system evolves
  - Current state s, a transition $s \xrightarrow{\alpha} s'$ is selected *nondeterministically*
  - The selection procedure is repeated and finishes once a state is encountered that has no outgoing transitions.
- The labeling function L relates a set $L(s) \in 2^{AP}$ at atomic propositions to any state s. $L(s)$ intuitively stands for exactly those atomic propositions $a \in AP$ which are satisfied by state s.
- Given that ϕ is a propositional logic formula, then s satisfies the formula ϕ if the evaluation induced by $L(s)$ makes the formula ϕ true,

Beverage Vending Machine

- $S = \{pay, select, soda, beer\}$, $I = \{pay\}$
- $Act = \{insert\_coin, get\_soda, get\_bear, \tau\}$
- Example transitions: $pay \overset{insert\_coin}{\rightarrow} select$, $beer \overset{get\_beer}{\rightarrow} pay$
- Atomic propositions depends on the properties under consideration. A simple choice - to let the state names act as atomic propositions, i. e. $L(s) = \{s\}$. "The vending machine only delivers a drink after providing a coin," $AP = \{paid, drink\}$, $L(pay) = \emptyset$, $L(soda) = L(beer) =$



Linear-Time Properties

- Deadlock
- Safety properties = "nothing bad should happen".
  - The number of inserted coins is always at least the number of dispensed drinks.
- Liveness properties = " something good will happen in the future".

Temporal Logic

- Propositional temporal logics [KB08], [Fre10]- extensions of propositional logic by temporal modalities.
- The elementary temporal modalities that are present in most temporal logics include the operators
  - "eventually" (eventually in the future) - $\lozenge$
  - "always" (now and forever in the future - $\square$
- The nature of time in temporal logics can be either linear or branching.
- The adjective "temporal"
  - specification of the relative order of events;
  - does not support any means to refer to the precise timing of events.

Linear-Time Logic

- Construction of LTL formulae in LTL - ingredients:
  - atomic propositions $a \in AP$, (stands for the state label a in a transition system)
  - boolean connectors like conjunction $\wedge$ and negation $\neg$,
  - basic temporal modalities "next" $\bigcirc$ and "until" $\bigcup$.
- LTL formulae over the set AP of atomic proposition are formed according to the following grammar:
  $\varphi ::= true | a | \varphi_1 \wedge \varphi_2 | \neg\varphi | \bigcirc \varphi | \varphi_1 \bigcup \varphi_2$, where $a \in AP$.

LTL temporal modalities

- The until operator allows to derive the temporal modalities $\Diamond$ ("eventually", sometimes in the future) and $\Box$ ("always", from now on forever) as follows:
  - $\Diamond\varphi = \text{true} \bigcup \varphi$.
  - $\Box\varphi = \neg\Diamond\neg\varphi$.
- By combining the temporal modalities $\Diamond$ and $\Box$, new temporal modalities are obtained:
  - $\Box\Diamond\varphi$ - "infinitely often φ."
    at any moment j there is a moment i $i \geq j$ at which an a state is visited
  - $\Diamond\Box\varphi$ - "eventually forever φ."
    from some moment j on, only a-states are visited.

Intuitive semantics of temporal modalities

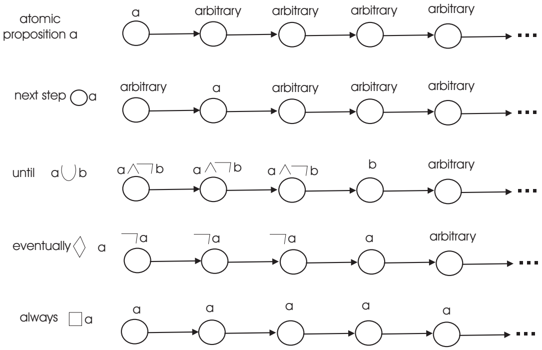
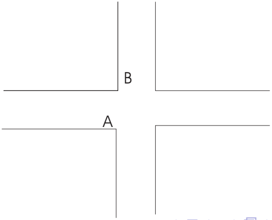


Figure: Intuitive semantics of temporal modalities

Semaphore example

- $\Box(\neg(A = green \wedge B = green))$
  - A and B can not be simultaneously green.
- $\Box(A = yellow \rightarrow A = red)$
  - If A is yellow eventually will become red.
- $\Box(A = yellow \rightarrow \bigcirc(A = red))$
  - If A is yellow then it will be red into the next state.
- $\Box(\neg(B = green) \bigcup (A = red))$
  - B will not be green until A changes in red.



CTL

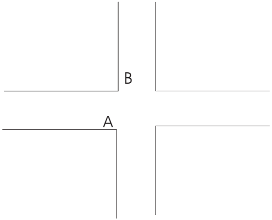
- Construction of CTL formulae:
  - as in LTL by the next-step and until operators,
  - must be not combined with boolean connectives
  - no nesting of temporal modalities is allowed.
- CTL formulae over the set AP of atomic proposition are formed according to the following grammar:
  $\phi ::=$ true $|a|\phi_1 \wedge \phi_2|\neg\phi|\exists\phi|\forall\phi$, where $a \in$ *AP and* φ is a path formula.
- CTL path formulae are formed according to the following grammar:
  $\varphi ::= \bigcirc\phi|\phi_1 \bigcup \phi_2$, where $\phi, \phi_1 and \phi_2$ are state fromulae.

CTL

- CTL distinguishes between state formulae and path formulae:
  - State formulae express a property of a state.
  - Path formulae express a property of a path, i.e. an infinite sequence of states.
- Temporal PATH operators $\bigcirc$ and $\bigcup$
  - $\bigcirc\phi$ holds for a path if ϕ holds in the next state of the path;
  - $\phi \bigcup \psi$ holds for a path if there is some state along the path for which ψ holds, and ϕ holds in all states prior to that state.
- Path formulae $\Rightarrow$ state formulae by prefixing them with
  - path quantifier $\exists$ (pronounced "for some path");
    $\exists\phi$ - holds in a state if there exists some path satisfying ϕ that starts in that state.
  - path quantifier $\forall$ (pronounced "for all paths".)

Semaphore example

- $\forall\Box(B = yellow \rightarrow \forall\bigcirc(B = red))$.
  - If B is yellow, it will become (sometime in the future) red.



Next lecture

- CMM