Model Checking

Modeling Real-Time Systems

02.08.2018

Deepak Pal

WHAT IS A MODEL?

- A model is a description of a system's behavior.
- Behavior can be described in terms of input sequences, actions, conditions, output and flow of data from input to output.
- It should be practically understandable and can be reusable; shareable must have precise description of the system under test.

MODEL CHECKING (MC) PROBLEM: INTUITION

- Correct design means that the system under development must satisfy design requirements. The requirements are stated as correctness properties
- Correctness properties state what behaviours/features are correct and what are not in the system.
- To apply rigorous verification methods formalization is needed:
  - system description
  - correctness properties
- System is described formally with its model
- Properties are specified formally by assertions expressed in logic

MODEL CHECKING (FORMALLY)

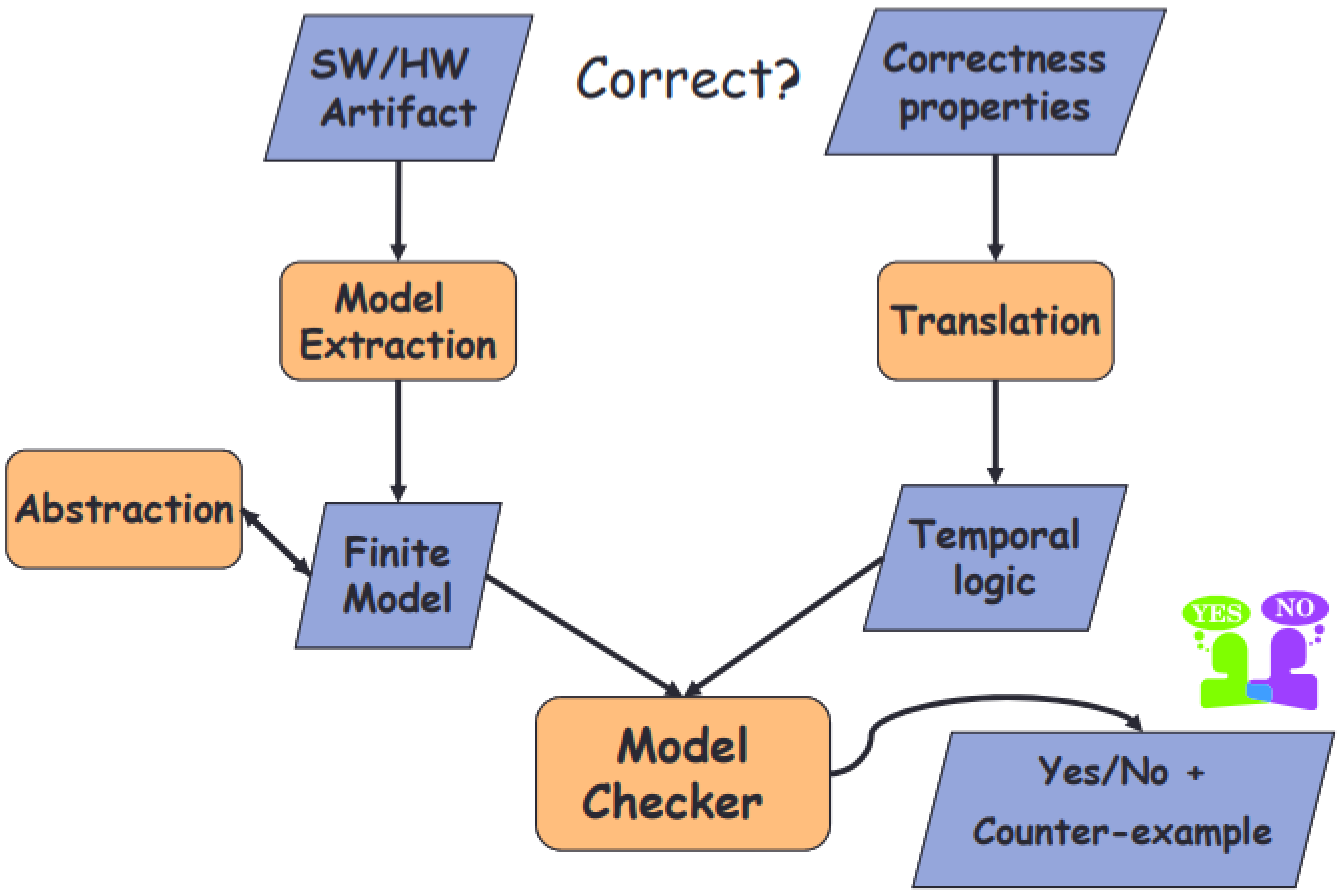
- Satisfaction relation (symbolically):

$$M \models \phi \,?$$

"Does model M satisfy logic assertion ϕ ?"

- Behavioural properties ϕ are stated often in temporal logic.
- M is a state-transition system that models the behavior of the implementation to be verified.

- <u>Procedural definition:</u>
  - Model checking is a state space exploration method to determine if the state space of model M satisfies the property ϕ.


SW/HW Artifact
Correct?
Correctness properties
Model Extraction
Translation
Abstraction
Finite Model
Temporal logic
YES
NO
Model Checker
Yes/No +
Counter-example

WHY MC?

- MC is fully automatic
- Good for bug-hunting because the "debugger" i.e. model checker that does not require full execution of your program
- Traceability – the diagnostic trace (counter example) generated by model checker helps in analyzing and detecting the sources of design bugs.

Witnesses and counterexamples

- Witnesses and counterexamples produced by model checkers provide a very useful source of diagnostic information.
- Witnesses that show why a formula is satisfied and (more often) counterexamples that show why it is not satisfied over a model.
- They are usually returned by model checkers in the form of a computation path.

- Uppaal tip: under the menu Options: Diagnostic trace select the option Shortest to let the verifier generates a counterexample in the simulator.

MODELLING

- Where the model M comes from?

  1. Formal modelling

  - It is a process of abstraction
  - It makes verification possible by retaining the part of the system that is relevant to modeling
  - It should not discard too much so that the result lacks certainty, or
  - discard too little so that the verification is not feasible

  2. Modelling techniques:

  - “manual“ composition by applying model patterns, abstractions, domain knowledge,…
  - automatic modelling by applying machine learning methods: • state and/or IO monitoring and automata learning from these logs
  - model extraction from code.

Choosing the modelling formalism?

- We focus on state-transition systems.
- They are
  - generally acceptable by model checkers;
  - represent finite set of states and transitions;
  - push-down automata/systems are possible;
  - also source programs can be used as models, e.g., Pathfinder for Java code;
  - abstract - symbolic encodings (logic formulae) specify abstract properties instead of explicit state behavior.

Modelling notions

- State
  - We want to express what is true in a particular state
  - A state is a "snapshot" of the system variables' valuation(s), e.g. • if a system is described by variables x, y, z then valuation x=2.4, y= 3.14, z=10 is one of its possible states.

- Transition represents relation between states.
  - It can be an abstraction of
    - C program statement, e.g. x++ transforming state where x=12 to a new state where x=13;
    - an electronic circuit;
    - or just an arrow, the source and destination states of which matter.

Atomicity of state transitions

- Execution of a transition is atomic, i.e. uninterruptable once started.

- Atomicity determines the abstraction level of the model
  - too big step may miss intermediate states that are important;
  - too small step may blow up the model unnecessarily.

- Atomicity of transitions must also consider concurrency
  - possible inter-leavings of transitions and interactions of parallel transitions systems must be explicit in the model.

Summary

- We touched the concept of MC at very high level:
  - MC is an automatic procedure that verifies temporal and state properties
  - Requires input:

    a state transition system

    a temporal property
- Model checking is an algorithmic framework tailored to perform verification task; on a high level, model checking can be viewed as an exhaustive search algorithm which exploits various optimization strategies to find a counterexample.
- The main practical problem in model checking is the combinatorial explosion of system states commonly known as the state explosion problem.