3C05: Model Checking

By Dragana Cvijanovic

Model Checking

- Objectives
  - To introduce model checking and its role within the design and development process;
  - To explore some of the main model checking approaches.

Model Checking?

The Need for Formalism

- Nowadays, the involvement of hardware and software systems in our live is increasing. Where human lives and important investments are at risk, the failure is unacceptable.
- Just how many times have we heard about failures caused by an error in hardware or software system, that had fatal consequences?
- Ariane 5 rocket explosion

System Verification

- Reactive systems are systems that maintain an ongoing interaction with their environment.
- To ensure their correctness, most widely used techniques are:
  - Simulation and testing
  - Deductive verification
  - Model checking

Simulation and Testing

- Most widely used techniques.
- Simulation is usually performed on an abstraction of the system.
- Testing is usually performed on the actual product.
- These techniques can be a very cost effective way for finding many errors, but in case of complex, asynchronous systems, they can only cover a limited set of possible behaviours.

Deductive Verification

- Employment of axioms and proof rules for correctness verification;
- Process can be quite lengthy and time-consuming, requiring a skilled professional;
- Rarely used in practice.

7

What is Model Checking?

- Conceived as an automatic verification technique for finite state systems
  - developed independently by Clarke and Emmerson and by Queille and Siffakis in early 1980s;
  - performs an exhaustive search of the state space of the system in order to determine if some specification is true or not.

8

Why Model Checking?

- Model checking differs from traditional verification approaches in two crucial aspects:
  - it does not aim of being fully general;
  - it is fully algorithmic and of low computational complexity.
- It can detect errors that were missed by traditional verification techniques, and being cost-efficient, it is being adopted as a standard quality assurance procedure.

9

Model Checking Process

- Main building blocks of model checking are:
- Modelling
  - initial step is to convert a design into a suitable formal form accepted by a model-checking tool;
- Specification
  - ensure that all the properties that the design has to satisfy are stated, by using temporal logic;
- Verification
  - by exhaustively exploring the state space, determine whether some specification is true or not.

10

Model Checking Process (contd.)

- Verification
  - by exhaustively exploring the state space, determine whether some specification is true or not;
  - this process is guaranteed to finish by the finitness of the model;
  - if a specification is found not to hold, a counterexample (e.g. a proof of the offending behaviour of the system) is produced.

11

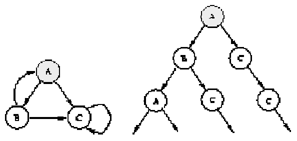
Systems and Properties

- A common framework for representing reactive systems is provided by the concept of transition systems.
- A finite state system can be described as a tuple $M = \{S, I, A, \delta\}$,
  - where S is a finite set of states;
  - $I \subseteq S$ is the set of initial states;
  - $A \subseteq S \times S$ is the transition relation, specifying the possible transitions from state to state;
  - δ is a function that labels states with the atomic propositions from a given language.
- Such a tuple is called state transition graph or Kripke structure (in honour of logician Saul A. Kripke).

12

Temporal Logics

- We use temporal logics to predicate over the behaviour defined by Kripke structures
  - the structure we obtain can be thought of as an infinite tree, representing all possible executions of the program.



13

Temporal Logics (contd.)

- According to how branching in the tree structure is handled, we distinguish Computation Tree Logic (CTL) and Linear Temporal Logic (LTL):
  - in LTL operators describe properties of all possible execution paths;
  - in CTL it is possible to quantify over paths from a given state.

14

Example: Labelled Transition Systems

- Just recall Labelled Transition systems from 3C03
  LTS:=(S,T,A, δ,c) where
  - S (a finite set of states)
  - T $\subseteq$ S $\times$S (a finite set of transitions)
  - A (an alphabet of atomic actions)
  - δ : T$\rightarrow$A (a transition labelling)
  - c $\in$ S (the current state)
  - In this way LTS determines all possible traces of a process

15

Algorithms For Model Checking

- Given a transition system M and a formula φ, the model checking problem is to decide whether $M \models \varphi$ holds or not:
  - *global* approach used by CTL and other branching-time logics;
  - *local* approach used by LTL.
- If the formula does not hold, provide the explanation why in form of counterexample.

16

LTL Model Checking algorithm

- Approach comes natural when the properties to be checked are expressed in terms of all possible executions of the program.
- Time complexity of LTL algorithm is exponential in size of the formula, but linear in size of the transition system (Pnueli and Lichtenstein).

17

CTL Model Checking Algorithm

- Introduced in early 1980s by Clarke and Emmerson:
  - suitable for checking properties in terms of structure of the program;
  - proved to be polynomial in both size of the model determined by the model checking program and in size of its temporal logic specification;
  - introduced notion of fairness without increasing the complexity of algorithm.
- These early model checking systems were able to check state transition graphs with between 10^4 and 10^5 states at a rate of about 100 states per second for typical formulas.

18

CTL* and Alternative Techniques

- Another type of branching-time logic is CTL*, combining both branching-time and linear time approaches:
  - introduced by Clarke, Emmeson and Sistla;
  - with time complexity as the LTL model checking algorithm.
- Most of the alternative techniques are based on use of automata for the model specification as well as for the implementation.

19

Symbolic Model Checking (SMC)

- The initial implementation
  - Used explicit representation for the Kripke structure as a labelled, directed graph;
  - This was practical for systems with small number of states;
  - Faced with complex concurrent systems, this method failed to handle extensive number of states – "State explosion problem".

20

Binary Decision Diagrams (BDDs)

- Major improvement was achieved with introduction of Binary Decision Diagrams by McMillan in 1987
  - it is now possible to model systems with between 10^{20} and 10^{30} states.
- A state of the system is symbolically represented by an assignment of boolean values to the set of state variables.

21

Binary Decision Diagrams (contd.)

- A boolean formula (and thus its BDD) is a compact representation of the set of the states represented by the assignments, which make the formula true.
- The transition relation can be expressed as a boolean formula in two sets of variables, one encoding the current state and the other encoding the next state.
- The model checking algorithm is based on computing fixpoints.
- Now, there is now need for the construction of the entire state graph for the system.

22

SMV

- Later on, McMillan as part of his PhD thesis developed a model checking system called SMV.
- SMV extracts a transition system represented as a BDD from a program and uses a BDD-based search algorithm to determine whether the system satisfies its specification
  - if not, it will produce an execution trace proving the invalidity of the specification.

23

Partial Order Reduction

- Algorithms based on the explicit state enumeration could be improved if only a fraction of the reachable pairs needs to be explored.
- Used in asynchronous systems composed of concurrent processes with relatively little interaction.
- The *interleaved model*, has all the actions of the individual processes arranged in a linear order called an *interleaving sequence*.
- The full transition system considers all the possible interleavings of these sequences, resulting in an enormously large state space.
- Partial order reduction algorithms employed: Stubborn sets, Persistent sets, Ample sets, Unfolding technique, Sleep sets.

24

Alternative Approaches

- Symbolic representations and partial-order reduction have increased the size of the systems that can be verified to 10^{20} - 10^{30} states.
- Still, there are a vast number of systems that are too complex to handle.
- Several techniques that can be used in conjunction with existing approaches have been introduced
  - Abstraction
  - Compositional reasoning
  - Symmetry reduction
  - Induction.

25

Fields of Application

- In 1992 Clarke and his students used SMV to verify the IEEE Future+ cache coherence protocol. They found a number of previously undetected errors in the design.
- 1992 Dill and his students found several errors, ranging from uninitialised variables to subtle logical errors during the verification of the Cache Coherence Protocol of the IEEE Scalable Coherent Interface.

26

Fields of Application (contd.)

- R. Anderson et al. verified part of preliminary version of the system requirement specifications of TCAS II (Traffic Alert and Collision Avoidance System II). TCAS II is an aircraft collision avoidance system required on most commercial aircrafts in United States.
- A High-level Data Link Controller was being designed at AT&T in Madrid in 1996. Using FormalChecker verifier, 6 properties were specified and five verified. The sixth property failed, uncovering a bug that would have reduced throughput or caused lost transmissions.

27

Key Points

- Model checking is a completely automatic and fast verification technique for finite state systems. It is now being widely used by large companies such as AT&T, Fujitsu, Intel, IBM, and Motorola etc. as part of their development process.
- The central task of model checking is to verify the correctness of systems, providing their quality assurance. The importance of this task is immense, since in certain cases, failures cannot be tolerated.

28