Model Checking

Anastasia Semenova

Overview

- To introduce model checking and its purpose.
- Main approaches.

Systems need to be checked

- The impact of software systems on our lives is increasing.
  - banks,
  - phones,
  - microwaves,
  - hospitals,
  - air traffic control
  - etc.
- But how can you trust a system?

Or they fail

- We hear about so many failures of systems on the news, some of which have fatal outcomes (e.g. NASA).
- But there are ways to make sure systems are sound.

So we have procedures for checking

- System simulation and testing
  - If you have ever programmed it's not unlikely that you have used this technique.
  - Widely used, testing is preformed on the product, where as simulation is performed on an abstraction of the system.
  - Though this is cost effective and catches many errors, there is no guarantee that the system will not fail in some obscure condition

Another checking procedure

- Deductive Verification
  - The system correctness is checked by using axioms and proofs for verification.
  - This is lengthy though complete.
  - For that reason companies tend not to employ this procedure.

Model Checking

- "Model checking is an automatic technique for verifying finite-state reactive systems" Clarke, Grumberg&Long
- Developed in '80s independently by Clarke and Allen Emerson and Quiele and Sifakis.
- Main advantage over other methods is that it is highly automated

UCL ComputerScience

The basics

- Modelling and Specification: User provides a high level representation of the system and the specification to be checked.
- Verification: The model checker exhaustively explores the problem space.
- And will terminate with true if it is so, or give a counter example of execution if there is one. Very useful for finding subtle errors

UCL ComputerScience

Why model checking?

- Cost effective
- Automated
- Does not aim for being fully general
- Algorithmic and of low computational complexity
- Still detects the errors
- Being adopted as a standard of quality assurance procedure.

UCL ComputerScience

System Modeling

A system can be described as a tuple:
M = { S, I, A, _ }
(This is the minimal tupal, you can have larger ones.) where:

- S is the final set of states
- I is the initial set of states
- A, a subset of S*S, is the transition relation
- _ is a function that labels stages with the atomic propositions from a given language.

This tuple is more commonly know as a state transition graph or a Kirpike structure.

Temporal Logic

Using this to predicate over the state transition graphs we obtain a structure that is like an infinite tree that gives us all possible executions of the program.

Draw picture.

Temporal Logic cont.

- There are two ways of handling branching:
  - LTL(linear Temporal Logic): operators describe properties of all possible execution paths
  - CTL(Computational Tree Logic): quantifies over paths from a give state.

Remember Concurrency?

For those of you that did concurrency, or know it anyway LTSA was a model checking tool. And LTS was a very similar tuple structure to Transition Graphs.

Algorithms

The model checking problem:

Given a system M and a formula Q, does M hold for Q?

There are two approaches

The local LTL approach:

- Approach comes naturally when the properties to be checked are possible executions of the program
- Time complexity is exponential in term of formula, but linear in size of the transition system.

The other approach

Global CTL and other branching time logics

- Suitable for checking structure of the program
- Polynomial in both the size of the modal determined by the modal checker and in size of temporal logic specification
- (able to check graphs 10^4 to 10^5 states…not a lot)

CTL* and others

- CLT* is another type of branch-time logic, it combines branch-time and linear time approaches,
  - So it has the time complexity of the LTL algorithm
  - Most alternative techniques are based on the use of automata for the model specification and the implementation

Symbolic Model Checking

- The original model checking algorithm, together with the new representation for transition relations is symbolic model checking.
  - Used explicit representation of Kripke Structure as a labelled directed graph.
  - Practical for small systems, with few states.
  - As systems too complex it was realised that something new would have to be added to model checking, because of the "state explosion" problem.

Binary Decision Diagrams(OBDD)

- A major improvement was made on the regular techniques.
- This was the introduction of Ordered Binary Decision Diagrams.
- The behaviour of a reactive system could be determined by n boolean state variables, the transition relation can be determined by:

$$(v_1, v_2, \ldots, v_n, v'_1, v'_2, \ldots, v'_n)$$

Cont.

- Where v represents the current state and v' represents the next state.
- Now possible to model systems with 10^{30} states, systems with 10^{100} states have also been successfully checked.
- The boolean formula was much more compact, the model checking algorithm was now based on finding fixed points- no need for a graph.

SMV

- As a part of a Phd of Mc Millan he developed a a new way of checking:
- SMV extracts transition systems represented by a BDD from a program that uses a BDD-based search algorithm to determine weather a system satisfies it's specification…
  if not, it will produce the trace proving the invalidity of the specification.

Partial Order Reduction

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

UCL ComputerScience

Methods used for lager systems

- Partial order reduction is still only able to handle as many states as a BDD, so other approaches:
- Abstraction
- Compositional reasoning
- Symmetry Reduction
- Induction

UCL ComputerScience

Summary

- Automatic approach to checking systems
- Used by big companies Intel, Fujitsu, IBM…
- The task of model checking is to assert systems, this task is very delicate are lives depend on it.

UCL ComputerScience