Model Checking Overview[1]

Paul Jackson
Paul.Jackson@ed.ac.uk

University of Edinburgh

Formal Verification
Autumn 2023

[1] Including contributions by Jacques Fleuriot and Bob Atkey

Goal of Model Checking

Concerned with automatically checking whether a formal model of some system has particular desired properties.

Systems

- Digital hardware
- Software, both sequential and concurrent
- Communication protocols
- Cyber-physical systems
- Biological systems
- ...

Properties of interest

- Functional - logical behaviour
- Dynamic - behaviour over time
- Security
- ...

Formal Models and Specifications

Formal models capture system behaviour of interest. Could involve

- Discrete or continuous time
- Non-determinism - handling input or hiding implementation details
- Probability
- Finite or infinite possible states
- Discrete and/or continuous state components
- ...

Formal property languages used for specifying properties of interest.

Alternatively,

- Desired properties can be captured in abstract formal models.
- Model checking then establishes whether all behaviours of the model of interest are consistent with the abstract model

Model Checking vs. Simulation & Testing

Testing is a standard approach for verifying software

Simulation is a standard approach for verifying digital hardware designs.

- Model checking considers all possible behaviours, starting from all possible initial states and considering all possible inputs
  - Simulation & testing are concerned with single runs or sampling of all possible behaviours
- Model checking provides results with logical certainty

Production of Counter-examples by Model Checking

When model checking fails, often counter-examples can be generated to help diagnose problems with model or properties.

Focus of Model-Checking Part of Course

Primarily will be concerned with

- Finite-state, discrete-time, non-deterministic models
  - Suprisingly-wide applicability.
  - Such models can be created as abstractions or approximations of more general classes of models (e.g. with large or infinite state, continuous state and continuous time)
- Properties expressed in temporal logics

Transition-System Models

A transition-system model of some system has

- A finite set of states
- A subset of states considered the initial states
- A transition relation which, given a current state, describes which next states a system can transition into.

Non-determinism

In general system descriptions are non-deterministic

- A system is non-deterministic when, from some state there are multiple alternative next states the system could transition to.
- Non-determinism good for
  - Modelling alternate inputs to the system from its environment (External non-determinism)
  - Allowing model to be under-specified, allowing it to capture many possible system implementations. (Internal non-determinism)

    Very common when modelling concurrency

Specifying Model Properties

- Interested in specifying behaviours of systems over time
- Elementary parts of specifications refer to properties of individual states at particular points in time
- Temporal specifications then relate such properties at different times
  - At **all times**, the read and write signals are never simultaneously asserted (at a logic '1')
  - If a request signal is asserted at **some time**, a corresponding grant signal will be asserted **within 10 time units**.

Linear & Branching Time

Linear Time

- Considers paths (sequences of states)
- If system non-deterministic, many paths for each initial state
- Questions of form
  - For all paths, does some path property hold?
  - Does there exist a path such that some path property holds?

Most basic linear-time logic is LTL (Linear Temporal Logic)

Branching Time

- Considers tree of possible future states from each initial state
- If system non-deterministic at some state, tree forks
- Questions more complex. E.g.
  - For all states reachable from an initial state, does there exist an onwards path to a state satisfying some property?

Most basic branching-time logic is CTL (Computation Tree Logic)

Temporal logic CTL* incorporates both CTL and LTL.