CS 515: Model-Based Testing vs. Model Checking

Stefan D. Bruda

Winter 2019

Algebraic vs. Logical Formal Verification

- Both algebraic methods (e.g., model-based testing) and logical methods (model checking) accomplish the same thing (verify a system against a specification)
- However, they use different models for both specification and system under test
  - The specification formalisms are dramatically different, but at least they accomplish the same thing (specify properties)
  - The system models are more similar in structure, but they specify dramatically different things
    - Evolving program states in Kripke structures vs. a behavioural model of actions that cause change of states in transition systems
    - They must however be equivalent on a more philosophical level: those reactive systems (transition systems) are computing systems and so ultimately implemented as a program (Kripke structure)
    - It is therefore expected that the two methods are equivalent (since they ultimately verify the same thing)
- Example of equivalence already established: CTL and stable failures/failure trace testing

https://arxiv.org/abs/1901.10925

Stable failures vs. CTL

Several steps necessary to establish equivalence between stable failures (failure trace tests actually) and CTL:

1. Need to define a way of converting transition systems into Kripke structures (or the other way around)
   - In turn this requires the definition of CTL semantics over transition systems (natural concept of outgoing actions as true propositions for each state)
   - At least two such conversions can be defined
   - They both require some minor variations of the CTL semantics
2. Need to construct equivalent failure trace tests for CTL formulae
   - Algorithmic construction by induction over number of operators in the formula
3. Need to construct equivalent CTL formulae for failure trace tests
   - Algorithmic construction by induction over the depth of the test
   - Naive conversion may result in infinite formulae, needs special handling of recursive tests (less elegant but still relatively straightforward)