

FORMAL METHODS

LECTURE I: INTRODUCTION TO FORMAL METHODS

Alessandro Artale

Faculty of Computer Science – Free University of Bolzano

artale@inf.unibz.it

<http://www.inf.unibz.it/~artale/>

Reading List

- *Logic in Computer Science—Modelling and Reasoning about Systems*. Michael Huth and Mark Ryan. Publisher: Cambridge University Press, 2004.
- *Model Checking*. Edmund Clarke, Orna Grumberg and Doron Peled. Publisher: MIT Press, 1999.

Why Formal Methods

- Errors in safety-critical systems may cause catastrophic loss of money, time or even human life.
- In this course we are mainly interested in **Reactive Systems**: Systems that maintain an ongoing interaction with the environment (as opposed to computing a final value):
 - **Reactive systems Examples**: Operating systems, Nuclear reactor controllers, Air traffic control programs, Programs controlling ongoing processes such as mechanical or medical devices, etc.

Why Formal Methods: the INTEL Story

1. In 1994 INTEL lost US \$500 million by realizing that their Pentium chip had a fault in the division algorithm.
2. Using a formal method technique based on symbolic algebraic manipulation the error was detected automatically.
3. Since then INTEL uses Formal Methods techniques in the testing phase of its chips!!!

How Logic Helps Computer Scientists

- **Mathematical Logic has finally succeeded!!!**
 - The problem of **Design Validation**: Ensuring the correctness of a system *at design level* is a major challenge mainly for (very) large systems.
 - Specification Languages provide for a *Formal Specification* starting from the system design.
 - Theorem Provers and Model Checkers are the main techniques used to verify the correctness of a system.
- Formal Methods (FM) are nowadays used in industry. They help in replacing the old techniques of *Simulation* and *Testing*.

FM Vs. Simulation & Testing

- Simulation & Testing explore **some** of the possible behaviors and scenarios of a system:
 - They leave open the problem of possible bugs in the unexplored scenarios.
- FMs conduct an **exhaustive exploration** of all possible behaviors starting from a formal specification:
 - When a design is declared correct by a FM we can guarantee the design to be free of any bug.
- We call **Formal Verifications** those Logic-Based methods to check the correctness of a system.

Model Checking

Main Idea: Describe a system and its states by means of logical notions – Formal Specification. Apply an automatic logic checker to verify correctness.

Formal Specification. *Model* \mathcal{M} of a particular logical theory;

Property to Verify. *Formula* ϕ of the chosen logical language;

Verification Method. *Model Checking:* Check whether the model satisfies the property: $\mathcal{M} \models \phi$.

Model Checking (cont)

- **Model Checking over Temporal Logic (TL) formulas.**
- **Main Idea:** In TL a formula is no more true or false but can be true in a state and false in another state.
- In TL the truth of a formula becomes a *dynamic* notion.

Formal Specification. The *Model* is a *Transition System*;

Property to Verify. *Formula* ϕ in a Temporal Logic;

Verification Method. *Model Checking*: Check whether the model (i.e a Transition System) satisfies the TL formula: $\mathcal{M} \models \phi$.

Advantages of the Model Checking Method

In the context of currently practiced techniques, Model Checking is the ultimately superior simulation tool.

- It is fully automatic and it does not require user supervision;
- When the design fails to satisfy a property the method produces a *counterexample* that shows a behavior which falsifies the property.
- The advent of **Symbolic Model Checking** allows to describe implicitly an astronomic number of states.

Course Overview

- Introduction to the Notion of Formal Methods in Computer Science.
- Modeling concurrent systems as Kripke structures.
- Temporal Logics:
 - Linear-time Temporal Logic (LTL);
 - Computation Tree Logic (CTL and CTL*).
- Model Checking: An Algorithm for CTL.
- Binary Decision Diagrams (BDD) and Symbolic Model Checking.
- Model Checking Vs. Theorem Proving.

The Need for FM's

- There is a need for reliable hardware and software:
 - Air and Train traffic control systems;
 - Telephone switching networks;
 - Medical instruments;
 - etc.

Model Checking

- Model Checking over Temporal Logic is a technique for verifying *finite state concurrent systems*.
- Verification can be performed automatically.
- The procedure uses an exhaustive search of the state space of the system to determine if a specification is true.
- Can be implemented with efficient algorithms.
- Hardware controllers and communication protocols are usually finite state systems.
- For infinite state systems the research direction attempts to integrate both proof-based and model checking-based techniques.

Temporal Logic and Model Checking

- Temporal Logics (TL) can describe the system evolution over time.
- Pnueli (1977) was the first to use TL for reasoning about concurrent systems using a proof-based technique.
- Clarke and Emerson in the early 80s introduced the technique based on model checking with efficient algorithms:
 - Model checking in CTL is Linear in both the size of the model and the length of the specification;
 - Model checking in LTL is Linear in the size of the model and PSPACE in the length of the specification.

Symbolic Algorithms

- **Problem:** In systems with a large number of states the global state transition diagram is too large to handle.
- McMillian, a PhD student at CMU, in 1987 realized that by a **symbolic representation** of state transition graphs much larger systems could be verified.
- The symbolic representations are based on **Ordered Binary Decision Trees** (OBDD).
- With OBDD was possible to verify systems with 10^{20} states, currently we are able to verify systems with more than 10^{120} states.
- The model checking algorithm developed by McMillian is called SMV used to verify an IEEE protocol in 1992.

Disclaimer

Some of the material presented in these slides is courtesy of the following people, listed in alphabetical order:

- Massimo Benerecetti (`bene@na.infn.it`)
- Alessandro Cimatti (`cimatti@itc.it`)
- Michael Fisher (`m.fisher@csc.liv.ac.uk`)
- Fausto Giunchiglia (`fausto@dit.unitn.it`)
- Paritosh Pandya (`pandya@tifrr.res.in`)
- Marco Pistore (`pistore@dit.unitn.it`)
- Marco Roveri (`roveri@itc.it`)
- Roberto Sebastiani (`rseba@dit.unitn.it`)