Computer Aided Verification
計算機輔助驗證

Introduction to Model Checking
模型檢驗

Pao-Ann Hsiung
Department of Computer Science and Information Engineering
National Chung Cheng University, Taiwan
熊博安
國立中正大學 資訊工程研究所

These slide contents are adapted from the slides of Professor Edmund Clarke.
Contents

♦ What is Model Checking?
♦ Formal System Modeling
♦ Formal Specification
What is Model Checking?

Cindy Crawford

Unfortunately, not that kind of model!!
Temporal Logic Model Checking

♦ Model checking is an **automatic verification technique** for finite state concurrent systems.

♦ Developed independently by **Clarke and Emerson** and by **Queille and Sifakis** in early 1980’s.

♦ **Specifications** are written in **propositional temporal logic**.

♦ Verification procedure is an **exhaustive search of the state space** of the design.
Some Advantages of Model Checking

- No proofs!!!
- Fast
- Counterexamples
- No problem with partial specifications
- Logics can easily express many concurrency properties
Main Disadvantage

State Explosion Problem:
♦ Too many processes
♦ Data Paths

Much progress has been made on this problem recently!
Basic Temporal Operators

The symbol “p” is an atomic proposition, e.g. DeviceEnabled.

- **Fp** - p holds sometime in the future.
- **Gp** - p holds globally in the future.
- **Xp** - p holds next time.
- **pUq** - p holds until q holds.
Model of computation

Microwave Oven Example
Temporal Logic

- The oven doesn’t **heat up** until the **door is closed**.
- **Not heat_up** holds **until door_closed**
- $\neg\text{heat\_up} \mathbf{U} \text{door\_closed}$
Model Checking Problem

Let $M$ be a state-transition graph.

Let $f$ be the specification in temporal logic.

Find all states $s$ of $M$ such that $M, s \models f$.

Efficient Algorithms: CE81, CES83
The EMC System

Preprocessor

State Transition Graph
$10^4$ to $10^5$ states

Model Checker (EMC)

Specification

True or Counterexamples
Breakthrough!

Ken McMillan implemented our model checking algorithm using **Binary Decision Diagrams** in 1987.

**Now able to handle much larger examples!!**
An Alternative Approach to Model Checking

♦ Both the **system** and its **specification** are modeled as **automata**.

♦ These automata are **compared** to determine if the system behavior **conforms** to the specification.

♦ Different notions of conformance have been explored:
  – *Language Inclusion*
  – *Refinement orderings*
  – *Observational equivalence*
Implementation and Specification

- $M_{\text{imp}}$ corresponds to the implementation:

- $M_{\text{spec}}$ corresponds to the specification:
  “event C must happen at least once”:  

\[ a, b, c \]
The Behavior Conformance Problem

Given two automata $M_{\text{imp}}$ and $M_{\text{spec}}$, check if

$$L(M_{\text{imp}}) \subseteq L(M_{\text{spec}}).$$

♦ If a sequence is accepted by $M_{\text{imp}}$, then it is also accepted by $M_{\text{spec}}$.
♦ This can be determined algorithmically.)
Binary Decision Diagrams can be used to represent state transition systems more efficiently.

The partial order reduction can be used to reduce the number of states that must be enumerated.

Other techniques for alleviating state explosion include:
  – Abstraction.
  – Compositional reasoning.
  – Symmetry.
  – Cone of influence reduction.
  – Semantic minimization.
Model Checker Performance

- Model checkers today can routinely handle systems with between 100 and 300 state variables.
- Systems with $10^{120}$ reachable states have been checked.
- By using appropriate abstraction techniques, systems with an essentially unlimited number of states can be checked.
In 1992 Clarke and his students at CMU used SMV to verify the IEEE Future+ cache coherence protocol. They found a number of previously undetected errors in the design of the protocol. This was the first time that formal methods have been used to find errors in an IEEE standard. Although the development of the protocol began in 1988, all previous attempts to validate it were based entirely on informal techniques.
In 1992 Dill and his students at Stanford used Murphi to verify the cache coherence protocol of the IEEE Scalable Coherent Interface.

They found several errors, ranging from uninitialized variables to subtle logical errors.

The errors also existed in the complete protocol, although it had been extensively discussed, simulated, and even implemented.
Notable Examples-PowerScale

- In 1995 researchers from Bull and Verimag used LOTOS to describe the processors, memory controller, and bus arbiter of the PowerScale multiprocessor architecture.

- They identified four correctness requirements for proper functioning of the arbiter.

- The properties were formalized using bisimulation relations between finite labeled transition systems.

- Correctness was established automatically in a few minutes using the CAESAR/ ALDEBARAN toolbox.
Notable Examples - HDLC

♦ A High-level Data Link Controller was being designed at AT&T in Madrid in 1996.

♦ Researchers at Bell Labs offered to check some properties of the design using the FormalCheck verifier.

♦ Within five hours, six properties were specified and five were verified.

♦ The sixth property failed, uncovering a bug that would have reduced throughput or caused lost transmissions!
Notable Examples
PowerPC 620 Microprocessor

♦ Richard Raimi used Motorola’s Verdict model checker to debug a hardware laboratory failure.

♦ Initial silicon of the PowerPC 620 microprocessor crashed during boot of an operating system.

♦ In a matter of seconds, Verdict found a BIU deadlock causing the failure.
Notable Examples—Analog Circuits

- In 1994, Bosscher, Polak, and Vaandrager won a best-paper award for proving manually the correctness of a control protocol used in Philips stereo components.

- In 1995, Ho and Wong-Toi verified an abstraction of this protocol automatically using HyTech.

- Later in 1995, Daws and Yovine used Kronos to check all the properties stated and hand proved by Bosscher, et al.
Notable Examples - ISDN/ISUP

♦ The NewCoRe Project (89-92) was the first application of formal verification in a software project within AT&T.

♦ A special purpose model checker was used in the development of the CCITT ISDN User Part Protocol.

♦ Five “verification engineers” analyzed 145 requirements.

♦ A total of 7,500 lines of SDL source code was verified.

♦ 112 errors were found; about 55% of the original design requirements were logically inconsistent.
Notable Examples - Building

♦ In 1995 the Concurrency Workbench was used to analyze an active structural control system to make buildings more resistant to earthquakes.

♦ The control system sampled the forces being applied to the structure and used hydraulic actuators to exert countervailing forces.

♦ A timing error was discovered that could have caused the controller to worsen, rather than dampen, the vibration experienced during earthquakes.
Model Checking Systems

♦ There are many other successful examples of the use of model checking in hardware and protocol verification.

♦ The fact that industry (INTEL, IBM, MOTOROLA) is starting to use model checking is encouraging.

♦ Below are some well-known model checkers, categorized by whether the specification is a formula or an automaton.
Temporal Logic Model Checkers

♦ The first two model checkers were EMC and Caesar.

♦ SMV is the first model checker to use BDDs.

♦ Spin uses the partial order reduction to reduce the state explosion problem for software systems.

♦ Verus, Kronos, and UPPAAL check properties of real-time systems.

♦ HyTech is designed for reasoning about hybrid systems.
Behavior Conformance Checkers

- The **Cospan/FormatCheck** system is based on showing inclusion between w-automata.

- **FDR** checks refinement between CSP programs; recently, used to debug security protocols.

- The **Concurrency Workbench** can be used to determine if two systems are observationally equivalent.
Combination Checkers

♦ Berkeley’s HSIS combines model checking with language inclusion.

♦ Stanford’s STeP system combines model checking with deductive methods.

♦ VIS integrates model checking with logic synthesis and simulation.

♦ The PVS theorem prover has a model checker for model mu-calculus.
Directions for Future Research

♦ Investigate the use of abstraction, compositional reasoning, and symmetry to reduce the state explosion problem.

♦ Develop methods for verifying parameterized designs.

♦ Develop practical tools for real-time and hybrid systems.

♦ Combine with deductive verification.

♦ Develop tool interfaces suitable for system designers.