Formal Methods
Maturation Study

Software Engineering Research

17-939A, Fall 2001
Vahe Poladian

Outline of The Talk

• Trace the maturation of formal methods
• Focus on analysis tools and techniques
• Maturation study case(s):
  – Model Checking,
  – Other (?),
• Goal: argue that formal methods have become / are becoming an engineering discipline

Formal Methods: A Definition

• “mathematically-based languages, techniques, and tools for specifying and verifying software (and hardware) systems”

Wing, Clarke

The Ingredients of Formal Methods

• Specification:
  • Languages / notations for formally describing a system and its desired properties
  – Some methods focus on specification
• Analysis:
  • Techniques and tools that use formal specification for the purpose of reasoning about the system and deriving properties about its behavior
  – In this research, focus is on analysis tools, e.g. model checkers, theorem provers
  – Tracing maturation of the field from the conception of mathematical ideas to the commercialization of tools that implement those ideas
Examples of Formal Methods

- Specification:
  - Z, VDM, Larch, Temporal Logic variants
- Model Checkers:
  - SMV, Spin, Murphi, FormalCheck, SVE
- Theorem Provers:
  - PVS, Eves, Nqthm, ACL2
- Emerging:
  - Combination theorem provers / model checkers

Case-study: Model Checking

- Model checking explained
- Significant research results
- Redwine / Riddle framework applied
- How do we extend this study to the broader field

Model Checking: an Overview

- Express essential details of program as a finite state transition diagram (the “model” or the Kripke structure)
- Express the desired properties in (propositional, branching-time) temporal logic (the “specification”)
- Feed both of these to an efficient program (the “model checker”), which automatically verifies whether the former satisfies the latter by exhaustively searching all the states
- The result is yes or no:
  - “No” answer explains faults in design, uncovers bugs, etc

Model Checking Research Results

- Clarke, Emerson: CTL, Model Checking 1977-81
- 1978-80
- 1981
- 1987
- 1986
- 1986
- Vardi, Wolper: Automata-Theoretic Approach to Model Checking, LTL
- Bryant: BDDs
- McMillan: SMV
- 1977-81
- 1981
- 1986
- 1987
Model Checking Research Results

- Clarke, Emerson: CTL, Model Checking
  1977-81
- Vardi, Wolper: Automata-Theoretic Approach to Model Checking, LTL
  1986
- McMillan: SMV
  1987
- Bryant: BDDs
  1986
- Redwine / Riddle Applied to Model Checking Tools

Redwine / Riddle Applied to Model Checking Tools

Basic Research $\Rightarrow$ concept formation
  - 1977-81: synthesis of code for data abstraction / temporal logic
Concept formation $\Rightarrow$ development and extension
  - 1981: model checker algorithm, seminal paper
Development and extension $\Rightarrow$ internal exploration
  - 1986-87: BDDs, SMV: a usable model checker
Internal Exploration $\Rightarrow$ external exploration
  - early 1990s: tools widely used outside of research
External Exploration $\Rightarrow$ popularization
  - mid 1990s: commercial support for tools, market penetration
  - Q: can we come up with some empirical evidence here?

How well does the framework apply?

- Applies well to the first 4 phases of maturation
- Identifying the turning points beyond internal exploration requires hard-to-get data from industrial practice
  - How do we know 40 (70) per cent use it?
  - 40 per cent of what group? People with compilers on their computer? Safety system programmers? Business programmers?
- Instead, ask the following questions:
  - Is the tool used outside of (research) community that created it?
  - Is the tool applied in real projects?
  - Is there commercial support / license for a tool, or its variants?
  - Are there specialized companies that use / apply the tool routinely?

The Evolution and Current Status of a Subset of Tools

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Math model</th>
<th>Origins (date, creator)</th>
<th>Int / Ext</th>
<th>Commercial support</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMV</td>
<td>Symbolic CTL</td>
<td>1987, McMillan, CMU</td>
<td>External</td>
<td>Cadence (research only)</td>
<td>Both</td>
</tr>
<tr>
<td>SPIN</td>
<td>On-the-fly (LTL)</td>
<td>early 1980s, Bell Labs</td>
<td>External</td>
<td>Cadence</td>
<td>Software</td>
</tr>
<tr>
<td>FormalCheck</td>
<td>On-the-fly (LTL)</td>
<td>Lucent (Bell Labs)</td>
<td>External</td>
<td>Cadence</td>
<td>Hardware</td>
</tr>
<tr>
<td>RuleBase</td>
<td>Symbolic CTL</td>
<td>IBM (SMV-based)</td>
<td>Internal</td>
<td>IBM FM group</td>
<td>Both</td>
</tr>
<tr>
<td>SVE</td>
<td>?</td>
<td>Siemens</td>
<td>External</td>
<td>Abstract Hardware, Versys</td>
<td>Hardware</td>
</tr>
<tr>
<td>Murphi</td>
<td>?</td>
<td>Stanford</td>
<td>External</td>
<td>Abstract (research only)</td>
<td>Hardware</td>
</tr>
</tbody>
</table>
Model Checking Success Stories

- Pentium FDIV bug (post facto)
- AFS and Coda cache coherence protocol (SMV)
- AT&T ISDN protocol
- Security protocols, e.g. SSL
- Mach Kernel
- Solaris memory model (Murphi)
- IBM Gigahertz Processor and various product lines, from mainframes to desktop PCs (RuleBase)

Redwine / Riddle Contd.

- It seems that Redwine / Riddle applies crisply to a specific group of methods
- Can we refine the framework to understand the maturation of formal methods more broadly?
- Can we use the results of the application of framework for groups of formal methods, e.g. model checkers, theorem provers, and then say something about the broader field?

Formal Methods As An Engineering Discipline

- Are Formal Methods Cost Effective?
- Do they solve practical problems?
- ... of increasing scale?
- ... and by application of advanced scientific knowledge?

- Attempt to analyze FMs in the framework suggested by Shaw in 1990 paper

Cost Effectiveness

- Applying formal methods in software development is NOT effortless:
  - Application of FMs requires training in techniques and tools
  - Requires thinking hard about the problem
  - Demands additional time and effort investment during the design phase
- So where is the pay-back?
  - Early fault detection, resulting in cost reduction (Pentium Bug)
  - Conformance to desired properties, resulting in increased quality, utility, value
  - Empirical results?
Practicality and Scale

- Examples of model checking success already seen
- Specialized companies have emerged that:
  - extend and support tools
  - design and develop new tools
  - consult in both tools and expertise
- Major software (and hardware) development companies have teams that specialize in the practice of formal methods

Clarke, et al, 1986

- Pro forma abstract (enhanced method)
  - Studies of the original logic system (CTL) for model checking proved to be deficient in expressing “fairness” of execution. Authors introduce an improved version of logic which captures fairness, thereby eliminating the possibility of infinite states. By considering only fair execution sequences, authors demonstrate the correctness of the alternating bit communication protocol
- Question: feasibility (can also argue improvement)
- Result: method / tool
- Validation: analysis, proof


- Enhanced method pro forma abstract – could also argue that this is an existing method applied to a new area
  - Existing verification methods of hardware circuit design: automatic theorem proving, human-assisted theorem proving, proof-checking are all deficient in dealing with sequential behavior. Furthermore, existing methods for dealing with sequential behavior such as model checking require exhaustive exploration of the state space of the circuit face the problem of state space explosion. Authors propose a new method of model checking, which uses Ordered Binary Decision Diagrams to significantly reduce the search space of the problem. Authors report empirical results supporting their claims.
- Question: improvement
- Result: method / tool
- Validation: formal analysis, empirical analysis

STOP

- Paper Summaries
- Discussion