The Philosophy of Formal Methods

Lee Pike

Formal Methods Group
NASA Langley Research Center
lee.s.pike@nasa.gov

September 21, 2004

(The contents herein are not necessarily endorsed by the United States Government.)

A Warning to Formal Methods Practitioners

Simplifying assumptions are made throughout to extract the central philosophical issues.

What are Formal Methods?

A formal method is a method applying formal mathematical techniques to prove (or disprove) a computer is correctly implemented.

$$\frac{\varepsilon_3}{\varepsilon_1} = \frac{A'}{A^2}\beta^2$$

$$\varepsilon_1 = \left(\frac{A}{A+1}\right)^2 E_1$$

$$\mu_3 = \mu$$

$$\frac{\varepsilon_4}{\varepsilon_1} = \frac{A'}{A+1-A'}\frac{\varepsilon_3}{\varepsilon_1}$$

$$\mu_4 = \mu$$

$$\longleftrightarrow$$

Why Formal Methods Matter

Return

Why Formal Methods Matter

Pentium FDIV Bug: It is estimated that a hardware bug in Intel's Pentium chip cost the company around 1/2 a billion dollars in the 1990's.

Return

Why Formal Methods Matter

Pentium FDIV Bug: It is estimated that a hardware bug in Intel's Pentium chip cost the company around 1/2 a billion dollars in the 1990's.

Therac-25: A radiation-therapy killed or maimed 6 people in the 1980's due to software bugs.

Return

Why Formal Methods Matter

Pentium FDIV Bug: It is estimated that a hardware bug in Intel's Pentium chip cost the company around 1/2 a billion dollars in the 1990's.

Therac-25: A radiation-therapy killed or maimed 6 people in the 1980's due to software bugs.

Missle Defense: A 1960's early warning system falsely asserted that a full-scale nuclear attack by the Soviets had occurred due to unanticipated radiation from the moon.

Return

Why Formal Methods Matter

Pentium FDIV Bug: It is estimated that a hardware bug in Intel's Pentium chip cost the company around 1/2 a billion dollars in the 1990's.

Therac-25: A radiation-therapy killed or maimed 6 people in the 1980's due to software bugs.

Missle Defense: A 1960's early warning system falsely asserted that a full-scale nuclear attack by the Soviets had occurred due to unanticipated radiation from the moon.

Testing alone did not uncover these errors.

Return

The Philosophical Challenge

"[Computers are] complex causal systems whose behavior, in principle, can only be known with the uncertainty that attends empirical knowledge as opposed to the certainty that attends specific kinds of mathematical demonstrations. For when the domain of entities that is thereby described consists of purely abstract entities, conclusive absolute verifications are possible; but when the domain of entities that is thereby described consists of non-abstract physical entities . . . only inconclusive relative verifications are possible."

James Fetzer: CACM, 1989

The Million Dollar Question (a.k.a. Intel's Half-Billion Dollar Question)

Can you *prove* a computer behaves correctly?

Abstract and Physical Computers

- Abstract Computers
  - E.g., Turing Machines, Rewrite-formalisms.
  - These are models that can be mathematically manipulated.
- Physical Computers
  - E.g., Digital wristwatches, laptops.
  - Can be pushed, prodded, and tested...
  - Only *models* of them can be mathematically manipulated.

Programs: Bridging the Great Divide

We want to prove that a program executed by a computer evokes the desired behavior.

Programs: Bridging the Great Divide

We want to prove that a program executed by a computer evokes the desired behavior.

- A program is a syntactic entity with causal powers.

Programs: Bridging the Great Divide

We want to prove that a program executed by a computer evokes the desired behavior.

- A program is a syntactic entity with causal powers.
- A program can be given a semantics via
  - An abstract computer.
  - A concrete computer.

Programs: Bridging the Great Divide

We want to prove that a program executed by a computer evokes the desired behavior.

- A program is a syntactic entity with causal powers.
- A program can be given a semantics via
  - An abstract computer.
  - A concrete computer.
- A program is the "interface" between the abstract and concrete.

Programs: Bridging the Great Divide

We want to prove that a program executed by a computer evokes the desired behavior.

- A program is a syntactic entity with causal powers.
- A program can be given a semantics via
  - An abstract computer.
  - A concrete computer.
- A program is the "interface" between the abstract and concrete.

From here on, "system" stands for a computer executing a program.

Specifications and Implementations

Specifications and Implementations

- A specification describes how a system should behave.

Specifications and Implementations

- A specification describes how a system should behave.
- An implementation is a system that should satisfy a fixed specification (e.g., it "adds detail").

Specifications and Implementations

- A specification describes how a system should behave.
- An implementation is a system that should satisfy a fixed specification (e.g., it “adds detail”).
  - Abstract systems may be abstract implementations.
  - Physical systems may be concrete implementations.

Specifications and Implementations

- A specification describes how a system should behave.
- An implementation is a system that should satisfy a fixed specification (e.g., it “adds detail”).
  - Abstract systems may be abstract implementations.
  - Physical systems may be concrete implementations.
- An implementation is correct if it in fact satisfies its specification (?).

Specifications and Implementations

- A specification describes how a system should behave.
- An implementation is a system that should satisfy a fixed specification (e.g., it "adds detail").
  - Abstract systems may be abstract implementations.
  - Physical systems may be concrete implementations.
- An implementation is correct if it in fact satisfies its specification (?).
- An abstract implementation is also a formal specification.

Specifications and Implementations: An Example

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Abstract Implementation$_1$:
  Output $z = x + y$.

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Specification$_1$:
  Output $z = x + y$.

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Specification$_1$:
  Output $z = x + y$.
- Abstract Implementation$_2$:
  $plus(x, y) \stackrel{def}{=}$ if $x = 0$ then y else $plus(+1(x), +1(y))$

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Specification$_1$:
  Output $z = x + y$.
- Specification$_2$:
  $plus(x, y) \stackrel{def}{=}$ if $x = 0$ then y else $plus(+1(x), +1(y))$

Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Specification$_1$:
  Output $z = x + y$.
- Specification$_2$:
  $plus(x, y) \stackrel{def}{=}$ if $x = 0$ then y else $plus(+1(x), +1(y))$

$$\vdots$$

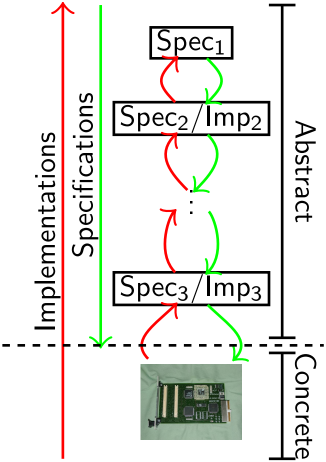
Specifications and Implementations: An Example

- Specification:
  For inputs $x, y \in \mathbb{N}$, output z where $z \geq x$ and $z \geq y$.
- Specification$_1$:
  Output $z = x + y$.
- Specification$_2$:
  $plus(x, y) \stackrel{def}{=}$ if $x = 0$ then y else $plus(+1(x), +1(y))$

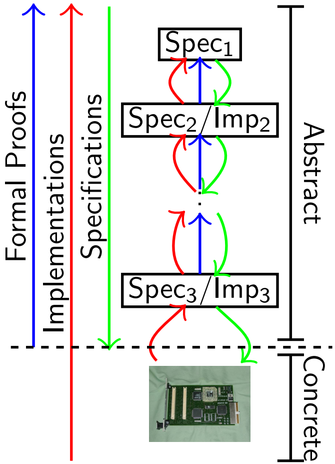
$$\vdots$$

- Concrete Implementation:
  A machine that accepts and emits electomagnetic pulses.

The Structure of Proofs in Formal Methods



The Structure of Proofs in Formal Methods



Formal Methods & Science

"... The semantic gap is sufficiently small to render Fetzer's objections inconsequential. To deny any relation ... is to deny that there can be any useful mathematical model of reality."

Bevier, Smith, Young: CACM, 1989.

Formal Methods & Science

". . . The semantic gap is sufficiently small to render Fetzer's objections inconsequential. To deny any relation . . . is to deny that there can be any useful mathematical model of reality."

Bevier, Smith, Young: CACM, 1989.

That is, if formal methods are not possible, than neither is applied mathematics in any scientific field.

Just Blame the Physicists

Just Blame the Physicists

- The reply seems to rest on the assumption that a chain of models is possible, all the way down to those of physics.

Just Blame the Physicists

- The reply seems to rest on the assumption that a chain of models is possible, all the way down to those of physics.
- In other words, if the concrete-abstract gap is small enough, it is based on the models of physics.

Just Blame the Physicists

- The reply seems to rest on the assumption that a chain of models is possible, all the way down to those of physics.
- In other words, if the concrete-abstract gap is small enough, it is based on the models of physics.
- If the physical implementation is incorrect, but the abstract implementations *down to the models of physics* are proved to meet their specifications, then physics is wrong.

Some Problems

Next

Some Problems

- It is not a priori obvious that the models of physics and computer science are continuous, and no formal verification actually attempts this.

Next

Some Problems

- It is not a priori obvious that the models of physics and computer science are continuous, and no formal verification actually attempts this.
- It is not just computational models that are of concern (see the examples).

Next

Some Problems

- It is not a priori obvious that the models of physics and computer science are continuous, and no formal verification actually attempts this.
- It is not just computational models that are of concern (see the examples).
- Formal method practitioners do not experimentally verify their models. Indeed, formal methods are meant to *replace* experimental verification.

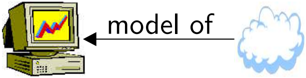
Next

Mind the Concrete-Abstract Gaps

Return

Mind the Concrete-Abstract Gaps

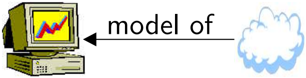
- Computers are formally modeled.

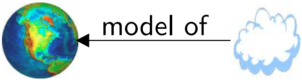


Return

Mind the Concrete-Abstract Gaps

- Computers are formally modeled.
- The world is formally modeled.

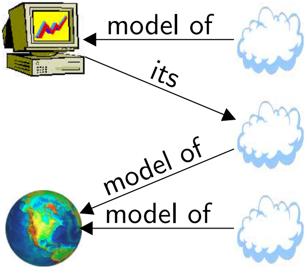




Return

Mind the Concrete-Abstract Gaps

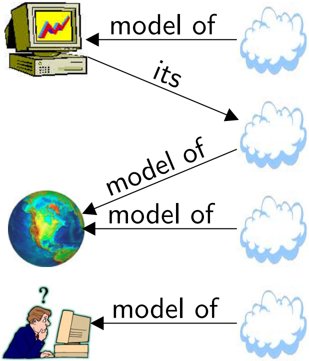
- Computers are formally modeled.
- The world is formally modeled.
- Computers' models of the world are formally modeled.



Return

Mind the Concrete-Abstract Gaps

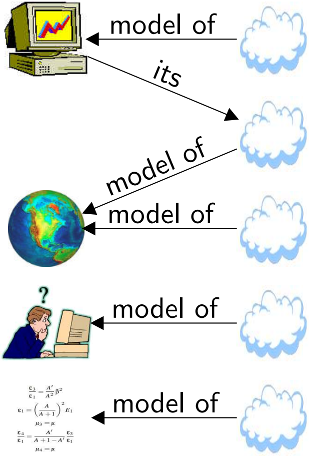
- Computers are formally modeled.
- The world is formally modeled.
- Computers' models of the world are formally modeled.
- The behavior we desire is formally modeled.



Return

Mind the Concrete-Abstract Gaps

- Computers are formally modeled.
- The world is formally modeled.
- Computers' models of the world are formally modeled.
- The behavior we desire is formally modeled.
- Proofs are formally modeled (in a logic).



Return

Where Are We Left?

Where Are We Left?

- The problem of mathematics in formal methods is not reducible to the problem of mathematics in the empirical sciences.

Where Are We Left?

- The problem of mathematics in formal methods is not reducible to the problem of mathematics in the empirical sciences.
- The possible salvation of formal methods: program semantics. . .

Concluding Remarks

Concluding Remarks

- Formal Methods is not an empirical science (is it an inchoate engineering discipline?), and its philosophical problems are not reducible to ones in science.

Concluding Remarks

- Formal Methods is not an empirical science (is it an inchoate engineering discipline?), and its philosophical problems are not reducible to ones in science.
- A better philosophical understanding of formal models and their interactions is needed.

Concluding Remarks

- Formal Methods is not an empirical science (is it an inchoate engineering discipline?), and its philosophical problems are not reducible to ones in science.
- A better philosophical understanding of formal models and their interactions is needed.
- Better philosophical understanding of the programs, algorithms, etc. is needed.

Concluding Remarks

- Formal Methods is not an empirical science (is it an inchoate engineering discipline?), and its philosophical problems are not reducible to ones in science.
- A better philosophical understanding of formal models and their interactions is needed.
- Better philosophical understanding of the programs, algorithms, etc. is needed.
- These considerations comprise the foundation of inevitable and important questions of ethics.

Some Web Resources

NASA Langley Research Center Formal Methods Group

`http://shemesh.larc.nasa.gov/fm/`

Google: nasa formal methods

A Good Online Bibliography

`http://www.cse.buffalo.edu/~rapaport/510/canprogsbeverified.html`

Google: rapaport programs verified

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.
- In the computer sciences, small changes in the world may mean huge changes in modeled behavior.

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.
- In the computer sciences, small changes in the world may mean huge changes in modeled behavior.

Example: Flipping a bit.

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.
- In the computer sciences, small changes in the world may mean huge changes in modeled behavior.

Example: Flipping a bit.

- $100010_2 = 34$.

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.
- In the computer sciences, small changes in the world may mean huge changes in modeled behavior.

Example: Flipping a bit.

- $100010_2 = 34$.
- $100010_2 \longrightarrow 000010_2$.

Computational Models are Discrete

- In the physical sciences, small changes in the world mean small changes in modeled behavior.
- In the computer sciences, small changes in the world may mean huge changes in modeled behavior.

Example: Flipping a bit.

- $100010_2 = 34$.
- $100010_2 \longrightarrow 000010_2$.
- $000010_2 = 2$.

Computational Models are Discontinous

- Computational Fluid Dynamics can be used simulate *continuous* airfoil behavior.

Computational Models are Discontinous

- Computational Fluid Dynamics can be used simulate *continuous* airfoil behavior.
- Relatively simple programs can have billions of *discontinuous* states.

The Difference Between Formal Methods and Science

In practice, the model-world gap is wider in formal methods than in the sciences (e.g., physics):

The Difference Between Formal Methods and Science

In practice, the model-world gap is wider in formal methods than in the sciences (e.g., physics):

- Formal verification requires a multitude of models; most other science requires just one.

The Difference Between Formal Methods and Science

In practice, the model-world gap is wider in formal methods than in the sciences (e.g., physics):

- Formal verification requires a multitude of models; most other science requires just one.
- Computer science is fledgling: new discoveries lead to new models.

The Difference Between Formal Methods and Science

In practice, the model-world gap is wider in formal methods than in the sciences (e.g., physics):

- Formal verification requires a multitude of models; most other science requires just one.
- Computer science is fledgling: new discoveries lead to new models.
- The concrete objects are of enormous complexity (e.g., Windows XP has approx. 40 million lines of code), and so are their models.

The Difference Between Formal Methods and Science

In practice, the model-world gap is wider in formal methods than in the sciences (e.g., physics):

- Formal verification requires a multitude of models; most other science requires just one.
- Computer science is fledgling: new discoveries lead to new models.
- The concrete objects are of enormous complexity (e.g., Windows XP has approx. 40 million lines of code), and so are their models.

But these are differences of degree, not of kind.

Next

Reasoning About Computers

- The mathematical domain used to model computers is logic and discrete mathematics.

Reasoning About Computers

- The mathematical domain used to model computers is logic and discrete mathematics.
- The mathematical domain used to model most other physical objects is The Calculus. Behavior is simulated by solving (differential) equations.

Mathematics in the Sciences

In science...

In Formal Methods...

Mathematics in the Sciences

In science...

- Theories about the behavior of the world are formulated.

In Formal Methods...

- Theories about the behavior of the world (and computers, and their interactions) are formulated.

Mathematics in the Sciences

In science...

- Theories about the behavior of the world are formulated.
- Then these theories are tested by experimentation.

In Formal Methods...

- Theories about the behavior of the world (and computers, and their interactions) are formulated.
- Formal methods does not test these theories!

Formal Methods as an Engineering Discipline

- Formal methods practitioners do not attempt to develop and test new theories.

Formal Methods as an Engineering Discipline

- Formal methods practitioners do not attempt to develop and test new theories.
- Rather, established theories are used to develop and validate new designs.

Formal Methods as an Engineering Discipline

- Formal methods practitioners do not attempt to develop and test new theories.
- Rather, established theories are used to develop and validate new designs.

The bane of formal methods: The engineering practice is being developed concurrently with the science of computation.

Shrinking the Gap

- The behavior of a program executed on an abstract computer can be verified.

Shrinking the Gap

- The behavior of a program executed on an abstract computer can be verified.
- If the semantics we give to programs match those computers give to them, we're home free.

Shrinking the Gap

- The behavior of a program executed on an abstract computer can be verified.
- If the semantics we give to programs match those computers give to them, we're home free.
- How to do this? Compile to a small, simple instruction set that we can check relatively easily.

Shrinking the Gap

- The behavior of a program executed on an abstract computer can be verified.
- If the semantics we give to programs match those computers give to them, we're home free.
- How to do this? Compile to a small, simple instruction set that we can check relatively easily.
- Programs are the complex, changing part of a system. We might gather enough empirical evidence that computers give the right semantics to trust our formal verification of the program.