Formal Methods

Program Verification

Program Verification

Devi Prasad
School of Information Sciences, Manipal
devi.prasad@manipal.edu

Formal Methods

Program Verification

Intent

Introduce the basic ideas of formal methods.

Show the direct application of rigorous techniques.

Hopefully inspire some of you.

Content

broad brush and inclusive.

Appeals to your intuition.

Shows key techniques and methods throughout.

Interaction

informal. Lively. Spontaneous. Conversational. Interactive.

You ask questions in the flow.

“There is no such thing as a dumb question!”

Let’s Start!

The Specification

“Develop a procedure in the C programming language to implement the linear searching algorithm. Assume the inputs include an array of integers and a key to search.”

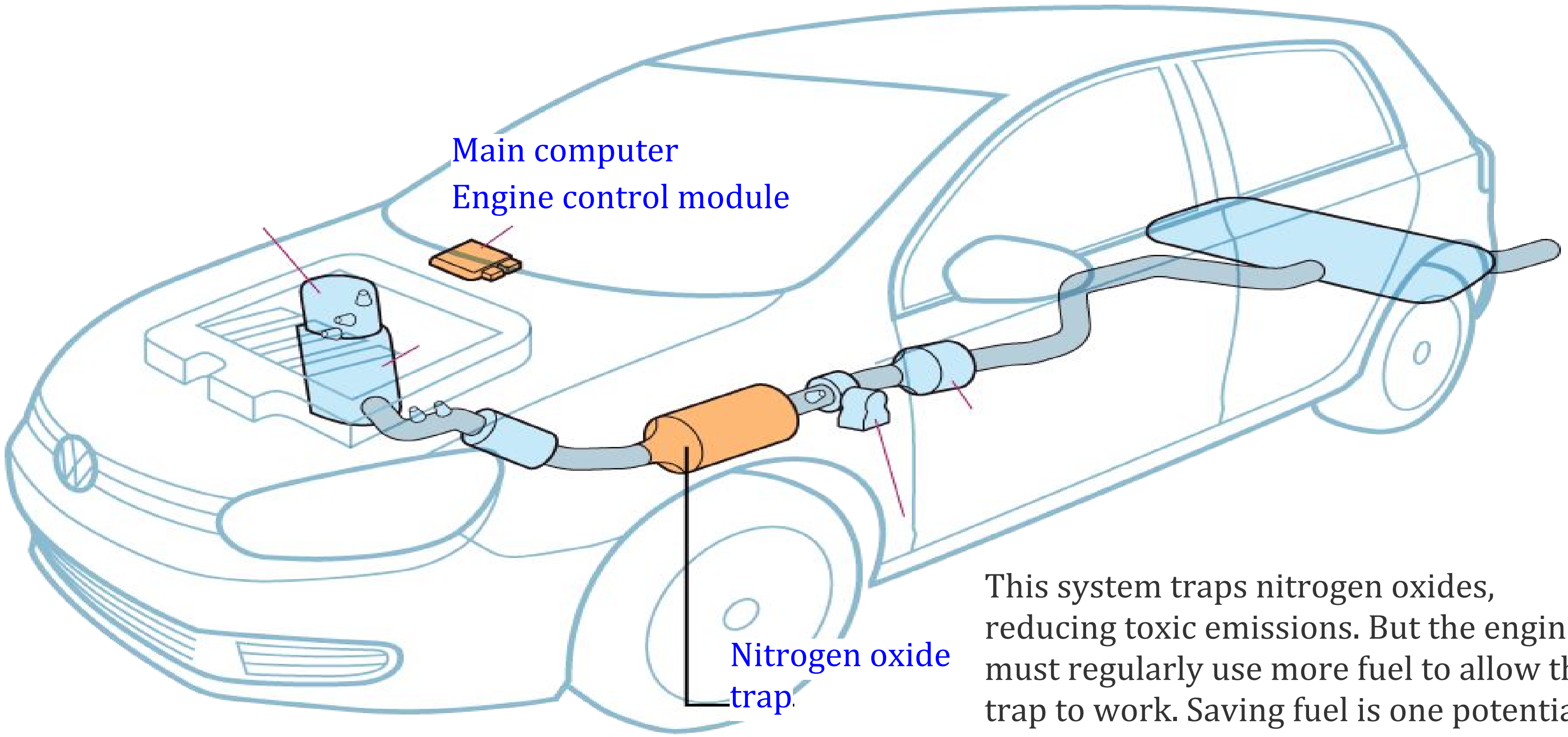
Develop and Test the C Program!

Why Would Programmers Cheat?

VW Emission Fraud

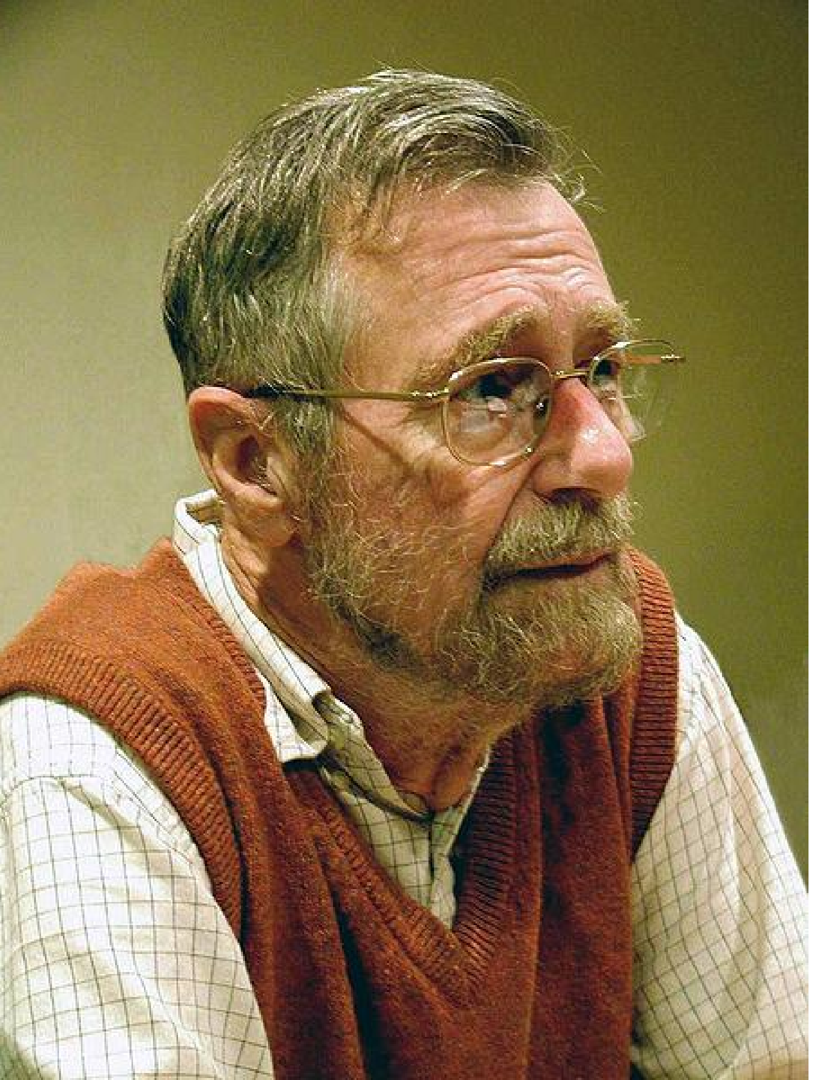
https://en.wikipedia.org/wiki/Volkswagen_emissions_scandal

http://www.vw-emissions-fraud.com/



This system traps nitrogen oxides, reducing toxic emissions. But the engine must regularly use more fuel to allow the trap to work. Saving fuel is one potential reason that Volkswagen's software could have been altered to make cars pollute more...

http://www.nytimes.com/interactive/2015/business/international/vw-diesel-emissions-scandal-explained.html?_r=0



Program Synthesis.
Guarded Commands.
Nondeterminacy.

“... The first moral of the story is that program testing can be used very effectively to show the presence of bugs but never to show their absence.”

Program Testing

Testing aims to show *incorrectness*, rather than *correctness*.

When a test fails, it succeeds in revealing an error.

- When a considerable number of tests fail to detect bugs, our confidence in the program increases, even if the correctness cannot be established.

Program Testing

Testing aims to show *incorrectness*, rather than *correctness*.

When a test fails, it succeeds in revealing an error.

- When a considerable number of tests fail to detect bugs, our confidence in the program increases, even if the correctness cannot be established.

Code coverage requires that certain parts of the code be exercised.

Program Testing versus Verification

Testing aims to show *incorrectness*, rather than *correctness*.

When a test fails, it succeeds in revealing an error.

When a considerable number of tests fail to detect bugs, our confidence in the program increases, even if the correctness cannot be established.

This is in contrast with formal methods where a program is proved to be correct without executing it!

Specifications and Programs

Specification is a rigorous definition of the *relation between inputs and outputs*.

  Mathematically precise definition of *what* is required.

Program is an *implementation* of a solution that meets the specification.

  There will be more than one correct implementation for the same specification.

Program Verification

Verification rigorously establishes if a program is correct with respect to the specification.

Employs methods of mathematics and logic.

Verification may be used to establish interesting properties of a program.

Challenges of Programming Languages*

Ambiguous and unspecified semantics.

Overly complex and complicated semantics.

Hard to reason. Hard to use confidently.

Offer no or very little assistance for verification methods.

Most are artefacts of the 60s, 70s, 80s and 90s!

* those regularly taught in typical academic institutes, and used in most software shops.

Demands of the 21st Century

Dependable systems.

Safety-critical systems.

Systems that are correct by construction.

Verified Software!

Do We Know Enough?

We do have rich experience building large, complex systems.

The theory and tools are out there.

> We need more.
>
> We need to build more!

There is a great need to educate programmers!

Assigning Meaning to Programs

"The semantic definition of a particular set of command types, then, is a rule for constructing, for any command *c* a *verification condition* $V_c(P; Q)$ on the antecedents and consequents of *c*."

An Axiomatic Basis for Computer Programming

"This involves elucidation of sets of axioms and rules of inference which can be used in proofs of th properties of programming languages."

First Order Logic

Predicates: odd(x), <, >, =,

Functions: abs, min, ...

Propositional logic symbols: ∧, ∨, ¬, ⇒

Quantifiers: ∀, ∃

Inference Rules!

More Examples!

My approach at the School of Information Sciences, Manipal.

- C programming language
- Data Structures

More Dafny Samples.

From Programs to Processes

Reactive Programs

A reactive program's role is to maintains an ongoing interaction with its environment rather than to compute a final value and terminate.

Concurrency is a fundamental element in reactive programs.

- By definition a reactive program runs concurrently with its environment.

Examples of Reactive Programs

Web Servers and Web Browsers.

Operating Systems.

Embedded real-time systems.

Computer Games.

Properties of Reactive Programs

Safety properties

“nothing bad ever happens”

Liveness properties

“something good will eventually happen”

Properties of Reactive Programs

Safety properties

“nothing bad ever happens”

no deadlocks, no abnormal termination, ...

Liveness properties

“something good will eventually happen”

progress, fairness, no livelocks, ...

Properties of Reactive Programs

Safety properties

> no deadlocks, no abnormal termination, ...

Liveness properties

> progress, fairness, no livelocks, …

These properties are defined only over infinite execution sequences of reactive programs.

Thank You!