applications

communication protocols
processors (CPUs)
kernel of a secure distributed operating system
compilers
safety-critical: medical systems, nuclear control
railway automated control
aerospace — attitude monitors
instrumentation systems
telephone and internet switching systems
airplane cabin communications

any software that must be correct
**programs are**

commands to a computer $\rightarrow$ execution

mathematical expressions $\rightarrow$ theory of programming

**why theory?** $\rightarrow$ proof, calculation, precision, understanding

**formal theory** = formalism + rules of proof, calculation, manipulation

formal $\neq$ careful, detailed
informal $\neq$ sloppy, sketchy

formal = using formulas (mathematical expressions)
informal = using a natural language (English)
start informal (with discussion)

end formal (with program)

then test, but

how do you know if the program is working?

what about the inputs you didn't test?

proof tells whether program is correct for all inputs

proof / verification after development

program development, with proof at each step

program modification, with proof
other theories

Hoare triples  \( P\{S\}R \) or  \( \{P\}S\{R\} \)

Dijkstra's weakest preconditions  \( wp(S, R) \)

Vienna Development Method (VDM)

Z and B

temporal logic  \( \Diamond \)

process algebras (CSP, CCS, mu-calculus, pi-calculus, ...)

event traces, interleaved histories

model checking

  exhaustive automated testing

  up to 10^{60} states \( \approx 2^{200} \) states \( \approx 200 \) bits \( \approx 6 \) variables

  abstraction, proof (not automated)
this theory

simpler

just binary (boolean) expressions

more general

includes terminating and nonterminating computation

includes sequential and parallel computation

includes stand-alone and interactive computation

includes time and space bounds and real time

includes probabilistic computations

prerequisites

some programming, any language

assignment statement, if-statement
TEXTBOOK available FREE at www.cs.utoronto.ca/~hehner