Algebra, Logic, Geometry: at the Foundation of CS

Tony Hoare
Theses

• Foundations of the Theory of Programming can be taught as an aid to practical programming throughout a degree course in Computing Science.

• A Program Development Environment for teaching should provide features similar to those of modern industrial tool chains.

• The level of Math required in the first practical course is that of High School courses in Algebra, Logic, and Geometry.
Summary

Part 1
Review of Boolean Algebra
Deductive Logic
Spatio-temporal Logic
Part 2
Sequential Composition
Concurrent Composition
Unifying Theories of Programming
1. Review of Boolean Algebra

relevant also for Mathematics, and Philosophy
and in CS for Hardware Design, and Program Development.
George Boole (1815-1864)

- Professor of Mathematics at Queen’s College Cork, Ireland
- Book: 1854 An Investigation of the Laws of Thought proposed the binary algebraic operators not, and, or, and a binary comparison for predicates: $≤$ (implies). These are the foundation for a deductive logic of propositions $(p, q, r, ... )$
George Boole (1815-1864)

• Professor of Mathematics at Queen’s College Cork, Ireland
• Book: 1854 An Investigation of the Laws of Thought proposed the binary algebraic operators not, and, or, and a binary comparison predicate: ≤ (implication) as the foundation of a deductive logic of propositions (p, q, r, ... )
Disjunction: $\lor$ (‘or’)

- Axioms: $\lor$ is associative, commutative and idempotent
- Theorem: distribution of $\lor$ through itself

\[(p \lor q) \lor r = (p \lor r) \lor (q \lor r)\]

Proof: $rhs = p \lor (r \lor (q \lor r))$ assoc

\[= p \lor ((q \lor r) \lor r)\] comm

\[= p \lor (q \lor (r \lor r))\] assoc

\[= p \lor (q \lor r)\] idem

\[= lhs\] assoc

Corollary: Rightward distribution (follows by comm)
Geometry: \((p \lor q)\)

Venn diagram
Implication: $p \leq r$

- Define $p \leq r$ as $r = p \lor r$
- $p$ implies $r$, $p$ is stronger than $r$, $r$ is weaker than $p$

- Geometry:
V is a weakening operator

• Theorem: $p \leq p \lor r$
• Proof: $p \lor r = (p \lor p) \lor r$ by idempotence
  
  \[= p \lor (p \lor r)\] by association

The theorem follows by definition of $\leq$

• Corollary: $p \leq r \lor p$ by commutation

Henceforth we omit brackets around associative operators, and proofs of theorems that follow by commutation.
2. Deductive Logic

Relevant for all branches of mathematics, science and engineering
The Aristotelian Syllogism

\[
\begin{align*}
\text{All men are animals} & \quad \text{All animals are mortal} \\
\hline
\text{All men are mortal}
\end{align*}
\]

• If the two antecedents above the line have been proved the consequent below the line is also provable
• To use a proof rule: first prove the antecedents and thereafter assume the consequent whenever required
• To validate a proof rule: first assume the antecedents and then use algebra prove the consequent
Aristotle 384-322 BC.

Founded the Lyceum in Athens, and lectured on

- sciences: physics, biology, zoology;
- aesthetics: poetry, theatre, music;
- ethics: politics, government, rhetoric;
- philosophy: metaphysics, logic, linguistics.

Recognised as the originator of logic and of classificatory biology, in which syllogisms are suited for deducing consequences from its classifications.
Rule of Proof by Cases

\[
\begin{array}{c}
p \leq r \\
q \leq r \\
\hline
(p \lor q) \leq r
\end{array}
\]

Validation: Assume the antecedents: \( r = p \lor r \) and \( r = q \lor r \)

\[
r = r \lor r
\]
the idempotence axiom

\[
= (p \lor r) \lor (q \lor r)
\]
by substitution for each \( r \)

\[
= (p \lor q) \lor r
\]
by distribution of \( \lor \) through \( \lor \)

The conclusion follows by the definition of \( \leq \)
Ordering: ≤

• Theorem: ≤ is a partial order

reflexive: $p \leq p$ by idempotence

transitive: $p \leq q$ $q \leq r$ $p \leq r$ by association

antisymmetric: $p \leq q$ $q \leq p$ $p = q$ by commutation
Covariance (monotonicity) of $\lor$

**Theorem:**

\[
\frac{p \leq q}{p \lor r \leq q \lor r}
\]

**Proof:**

\[
p \leq q \leq q \lor r \quad \text{and} \quad r \leq q \lor r
\]

The result follows by the proof rule for cases

A covariant operator preserves the ordering of each of its operands

‘strengthening a component can only strengthen the whole product’

(and weakening similarly)
3. Spatio-temporal Logic

for non-metric reasoning about what happens in space and in time
William of Occam (1287-1347)

- Franciscan friar, Scholar at Merton College Oxford
- excommunicated (1328) rehabilitated (1359)
- Occam’s razor: entities should not be postulated beyond necessity
- Book: Summa Logicae (1323)

with operators for implication, disjunction, conjunction, causation; and temporal operators while (|), and then (;)
Discrete geometry

interval in time

extent in space
The empty box [ ] does nothing.

interval in time

extent in space
Time and space co-ordinates
Multiple pen recorder.
Diagram of \((p;q)\)
Diagram of \((p \mid q)\)
Summary of algebraic laws

• Both sequential composition and concurrent composition are associative and have [ ] as unit
‘proof’: consider the diagrams
• Both distribute (leftward and rightward) through disjunction
This allows the implementation to make the choice of which disjunct to execute either at compile time or only just before execution the disjunction ...
... or at any time between.
The small print

• A term is defined only if all its operands are defined
• \( (p \lor q) \) and \( p \leq q \) are defined only if \( p \) and \( q \) have the same events
• \( (p; q) \) and \( (p|q) \) are defined only if their events are disjoint.
• \( \text{events}(p; q) = \text{events}(p|q) = \text{events}(p) \cup \text{events}(q) \)
Part 2.

Some of the material presented in the second part are not suitable for an introductory programming course.
4. Sequential Composition

The algebraic axioms for sequential composition validate the relevant proof rules.
Algebraic Axioms for ;

- [ ] (aka null, skip) describes a region in which nothing happens
- ; is associative and has unit [ ]
- ; distributes through ∨ (both leftward and rightward):
  
e.g., \( p;(q \lor q') = (p;q \lor p;q') \)

Distribution justifies giving ; a precedence stronger than ∨
Example of distribution

• Theorem:

\[
\frac{p; q \leq r}{p'; q \leq r}
\]
\[
(p \lor p'); q \leq r
\]

• Assume (1) \( r = p; q \lor r \) and (2) \( r = p'; q \lor r \)

Therefore

\[
\begin{align*}
  r &= p; q \lor p'; q \lor r \\
  &= (p \lor p'); q \lor r \\
  &= \text{consequent of the rule (by definition of } \leq )
\end{align*}
\]

substitute (2) in (1) ; distributes thru \( \lor \)
Proof rule for sequential composition

• Theorem:

\[
\frac{p; q \leq m \quad m; r \leq t}{p; q; r \leq t}
\]

• Assume (1) \( m = p; q \lor m \) and (2) \( t = m; r \lor t \)

Therefore 
\[
t = (p; q \lor m); r \lor t
\]
\[
= p; q; r \lor (m; r \lor t)
\]
\[
= p; q; r \lor t
\]

substitute (1) in (2) 
distribute \( r \) thru \( \lor \) 
substitute back by (2)
Rules of consequence

The theorem

\[
\frac{p; q \leq m}{m; r \leq t} \quad \frac{p; q; r \leq t}{p; q; r \leq t}
\]

has corollaries

\[
\frac{p \leq m}{m; r \leq t} \quad \frac{p; r \leq t}{p; r \leq t}
\]

\[
\frac{p; q \leq m}{t \leq m} \quad \frac{p; q \leq t}{p; q \leq t}
\]

Proof: (1) by substitution of [ ] for \( q \); (2) just by transitivity
Hoare triple

Consider the proposition $p; q \leq r$. It means that if $p$ describes the interval from the start of $r$ to the start of $q$, and $q$ describes the interval from the end of $p$ to the end of $r$, then $r$ correctly describes the whole of $(p; q)$. This is the intended meaning of the Hoare triple.

Define $\{p\} q \{r\}$ as $p; q \leq r$.
Verification Rules for ;

• By substitution of the definition of the triple into the Proof Rule for ;
  \[
  \{p\} q \{m\} \quad \{m\} r \{t\} \\
  \{p\} q;r \{t\}
  \]
  which is the Hoare rule for ;

• The two corollaries give:
  \[
  \{p\} q \{m\} \quad m \leq t \\
  \{p\} q \{t\}
  \]
  \[
  p \leq m \quad \{m\} r \{t\} \\
  \{p\} r \{t\}
  \]
  which are the Hoare rules of Consequence
Milner transition \( r \rightarrow q \)

• One way of executing \( r \) is to execute \( p \) first, saving \( q \) as a continuation for subsequent execution.

• Define \( r \rightarrow^q p \) as \( p;q \leq r \).
Operational rules for $r \Rightarrow p\, m \Rightarrow q\, t$

$$
\begin{array}{c}
  r \Rightarrow m \\
  m \Rightarrow t \\
\hline
  r \Rightarrow t
\end{array}
$$

the Milner rule for $r \Rightarrow p; q\, t$

• The two corollaries are

$$
\begin{array}{c}
  m \leq r \\
  m \Rightarrow t \\
\hline
  r \Rightarrow t
\end{array}
\quad
\begin{array}{c}
  r \Rightarrow m \\
  t \leq m \\
\hline
  r \Rightarrow t
\end{array}
$$

like Milner’s ‘rules of structural equivalence’, with $\equiv$ replaced by $\leq$
5. Concurrent Composition

| has the same laws as ; . An additional Interchange axiom permits a concurrent program to be executed sequentially by interleaving.
Algebraic Axioms for |

• | is associative and has unit []
• | distributes through ∨
• \((p \mid q);(p' \mid q') \leq (p; p') \mid (q; q')\) (the interchange axiom)
  • The rhs and the lhs differ by interchange of operators (; with |),
  • and of operands (p' with q)

Theorems:  
  \(p; q' \leq p \mid q'\) by interchange, with \(p' = q = []\) 
  \(q; p' \leq p' \mid q\) similarly, with \(q' = p = []\)

Hence \(p; q \lor q; p \leq p \mid q\) by the rule for cases
\[(p \mid q);(p' \mid q') \leq (p;p') \mid (q;q')\]

**Theorems**

\[
\begin{align*}
(p \mid q);q' & \leq p \mid (q;q') & p' = [ ] \\
p;(p' \mid q') & \leq (p;p') \mid q' & q = [ ] \\
q;(p' \mid q') & \leq p' \mid (q;q') & p = [ ] \\
(p \mid q);p' & \leq (p;p') \mid q & q' = [ ]
\end{align*}
\]

**Corollary:** \[p; q; q' \leq (p \mid q);q' \leq p \mid (q;q')\]

All four are proved by substitution of [ ] for a different variable. They are known as small interchange laws (or frame laws in separation logic).
Interleaving longer strings

• Let $x, y, z, w, a, b, c, d$ be characters representing single events
• Let us omit $;$ in strings except for emphasis. Thus

$$xyzw = x; y; z; w$$
Example of Interleaving

\[
abcd \mid xyzw \\
\vdash (a;bcd) \mid (xy;zw) \quad \text{is the rhs of interchange}
\]
\[
\vdash (a \mid xy);(bcd \mid zw) \quad \text{associativity (twice)}
\]
\[
\vdash (a \mid x;y);(bcd \mid zw) \quad \text{interchange}
\]
\[
\vdash (a \mid x);y;(b \mid zw);cd \quad \text{associativity (twice)}
\]
\[
\vdash (a \mid x);y;(b \mid zw);cd \quad \text{small interchange (twice)}
\]
\[
\vdash xayzbwcd \quad \text{similarly}
\]

Each step of the proof reduces length of same-coloured strings. Termination is assured when this is no longer possible.
Basic Principle of Concurrent Programming

• Every interleaving which preserves the order of the operands of all sequential and of all concurrent compositions is reachable by strengthening applications of the interchange axiom.

• first proved for Turing machines by simulation (interpretation)

• a direct algebraic proof (omitted) uses structural induction.
6. Unifying Theories of Concurrency

We repeat for concurrent programs the same unification achieved before for sequential programming.
Interchange Rule (O’Hearn)

\[
p; q \leq r \quad \quad \quad \quad \quad \quad p'; q' \leq r' \\
(p \mid p'); (q \mid q') \leq (r \mid r')
\]

The rule tells how to prove a complicated concurrent theorem by splitting it into two proofs of two much simpler sequential theorems, with only three variables in each.

Theorem: This rule is equivalent to the Interchange axiom

Proof: next two slides
The rule implies axiom

\[
\begin{align*}
p; q &\leq r & p'; q' &\leq r' \\
\Rightarrow (p \parallel p'); (q \parallel q') &\leq (r \parallel r')
\end{align*}
\]

(concurrency rule)

Proof: Replace \( r \) by \( p; q \) and \( r' \) by \( p'; q' \) throughout

The antecedents are true by the reflexivity of \( \leq \) and the conclusion is:

\[
(p \parallel p'); (q \parallel q') \leq (p; q) \parallel (p'; q')
\]

which is the interchange law
The axiom implies the rule

Assume the antecedents of the rule: $p;q \leq r'$ and $p';q' \leq r'$

\[
(p;q) \mid (p';q') \leq (r \mid r')
\]

(covariance of $\mid$ twice)

\[
(p \mid p') ; (q \mid q') \leq (p;q) \mid (p';q')
\]

(interchange axiom)

So

\[
(p \mid p') ; (q \mid q') \leq (r \mid r')
\]

(by transitivity of $\leq$)

Therefore

\[
\begin{align*}
\frac{p;q \leq r}{(p \mid p') ; (q \mid q') \leq (r \mid r')} & \quad \frac{p';q' \leq r'}{(p \mid p') ; (q \mid q') \leq (r \mid r')} \quad \text{(the interchange rule)}
\end{align*}
\]
Summary

\[ \frac{p; q \leq r \quad p'; q' \leq r'}{(p \parallel p'); (q \parallel q') \leq (r \parallel r')} \]

copied from an earlier slide

\[ \frac{\{p\} q \{r\} \quad \{p'\} q' \{r'\}}{\{p \parallel p'\} q \parallel q' \quad \{r \parallel r'\}} \]

translated to Hoare triples

\[ \frac{r \xrightarrow{q} p \quad r' \xrightarrow{q'} p'}{(r \parallel r') \xrightarrow{(q \parallel q')} (p \parallel p')} \]

translated to Milner transitions

are all equivalent to the exchange law
Applications to Programming

• Many interpreters and compilers for programming languages are specified by an operational semantics expressed as Milner Transitions.

• Most program analysers and proof tools for sequential languages follow a verification semantics expressed as Hoare Triples.

• The geometry is useful for students developing an intuition about the effect of running a program.

• As a log of execution of a test program, it helps to locate and diagnose an error, and to decide where in the program to correct it.
The challenge for the next fifteen years

• to provide students with a programming environment similar to those provided in the software industry
  • based on algebraic, logical and geometric theories
• to provide textbooks for initial practical programming courses
  • exploiting these theories to meet deadlines and win marks
• and for domain-specific languages for later courses
  • and for application projects developed by teamwork

The aim is to educate graduates who understand and can exploit advances software technology throughout their professional careers.
Further reading

• Consult my website  www.cl.cam.ac.uk/~carh4/

• Lecture 1. Geometric theory of program testing
• Lecture 2. Algebra for program transformation
• Lecture 3 is an early version of today’s lecture
Acknowledgements to my Colleagues

• at Oxford University: Jean-Raymond Abrial, Richard Bird, Steve Brookes, Carroll Morgan, Bill Roscoe, Dana Scott, Joe Stoy,
• at Microsoft Research: Jade Alglave, Ernie Cohen, Byron Cook, Rustan Leino, Matt Parkinson, Jon Pincus, Wolfram Schulte
• also: Nada Amin (Cambridge), Zhe Hou (Griffith), Jay Misra (Austin)
• my recent coauthors: Diego Marmsoler, Bernhard Moeller, Peter O’Hearn, Stephan van Staden, Georg Struth, Ian Wehrman, John Wickerson
• my heroes: Edsger W. Dijkstra, Robin Milner, Bertrand Russell
• my family: Jill Hoare and Jonathan Lawrence
End of main lecture

Discussion follows
Discussion: Algebra

Opposing views
In Praise of Algebra

• Simple, elegant, reusable, tractable by people and by machines,

• Algebraic transformation is essential in the top-down design of application system architecture by successive refinement

• It is also used in compilation, optimisation, refactoring, obfuscation, and automatic generation of program code

• Algebra unifies theories which underlie a range of programming tools, It is clearly essential for their correct interworking

• It also essential for the introduction of Theory into Computer Science education
Limitations of Algebra

• It has insufficient expressive power: no quantification.
• Logic can also specify and verify interfaces between components of a program.
• It cannot specify basic commands
• Logic specifies basic commands (assignment, input, output, ...)
• It has no negation: it cannot prove that a formula is not a theorem
• Geometry is a model of both algebra and logic. It provides test cases for incorrect programs and counterexamples for false conjectures

• see  https://www.cl.cam.ac.uk/~carh4/
Isaac Newton (1642-1726)

Communication with Richard Gregory (1694)

“Our [my] specious [falsely convincing] algebra [the infinitesimal calculus] is fit enough to find out [has some heuristic value], but entirely unfit to consign to writing and commit to posterity [it cannot and must not be published].”

(with translation to Modern English)

Newton’s proofs were geometric, establishing properties of the Keplerian ellipses that describe the orbits of the planets
Bertrand Russell (1872 – 1970)

The method of “postulating” what we want has many advantages; they are the same as the advantages of theft over honest toil. Let us leave them to others and proceed with our honest toil.

Introduction to Mathematical Philosophy.

Russell then refused to postulate the existence of real numbers (such as the sqrt of 2), and proceeded to model them by the Dedekind cut.
Gottfried Leibniz (1646-1716)

• calculemus 

Let us calculate (symbolically)