Automated Mathematical Induction

Principle of Mathematical Induction

- \( \forall I \) — an inference rule
- One way to use \( \forall I \):
  - Prove \( P(0) \)
  - Prove \( P(n+1) \) for arbitrary \( n \)
  - Takes care of \( P(1), P(2), P(3), \ldots \)
- Mathematical induction makes it easier
  - Proof of \( P(n+1) \) can cite \( P(n) \) as a reason
  - If you cite \( P(n) \) as a reason in proof of \( P(n+1) \), your proof relies on mathematical induction
  - If you don’t, your proof relies on \( \forall I \)

Example: Concatenation Conserves Length

Assume concatenation (\( \text{cons} \)) and length satisfy

\[
\begin{align*}
\text{cons}([], y) &= y \quad (1) \\
\text{cons}[x \mid xs], y] &= [x \mid \text{cons}(xs, y)] \quad (2) \\
\text{length}([]) &= 0 \quad (3) \\
\text{length}[x \mid y]\] &= 1 * \text{length}(y) \quad (4)
\end{align*}
\]

- Prove \( \forall xs, P(xs) \)
  where \( P(xs) = \text{\text{length}(\text{cons}(xs, y))] = \text{\text{length}(xs)} + \text{\text{length}(ys)} \)
- Base case: \( P([ ]) \) (uses (1) and (3))
- Inductive case: \( P(xs) \rightarrow P(x: xs) \)

Software Examples

- sum
- times
- gcd
- length
- cons
- reverse
- maximum
- vector addition
- insertion sort
- merge
- merge sort
- quick sort
- exponentiation
- binary tree search
- circuit boards
- compilers
- operating systems

Boyer and Moore’s Computational Logic System
- Written in lisp and prove properties of lisp programs
- Significant properties verified
  - Lots of examples in reasoning about software
  - Supporting reasoning tools are needed:
    - Simplification: rewriting
    - Deduction: resolution

Applications

- Mathematical proof checking
- The QED Project
- Computer chip verifications
- Software verification

Axioms about Sequences

- Algebraic laws of sequence construction
  \([ ] \) list
  \([ \ldots ] \) : int, list \rightarrow list
- Informally
  \([ x \mid x_1, x_2, \ldots ] \] \(= [x, x_1, x_2, \ldots ] \)
- Definition of concatenation
  \( \text{cons} \) : list, list \rightarrow list
  \(\text{cons}([], y) = y \quad (1)\)
  \(\text{cons}[x \mid xs, y] = [x \mid \text{cons}(xs, y)] \quad (2)\)
- Definition of length
  \( \text{length} \) : list \rightarrow integer
  \(\text{length}([]) = 0 \quad (3)\)
  \(\text{length}[x \mid y]\] \(= 1 * \text{length}(y) \quad (4)\)
- Can we prove the following?
  \(\text{length}([x \mid y]\] \(= \text{length}(x) + \text{length}(y) \quad (5)\)

Example: Concatenation Conserves Length

Assume concatenation (\( \text{cons} \)) and length satisfy

\[
\begin{align*}
\text{cons}([], y) &= y \quad (1) \\
\text{cons}[x \mid xs, y] &= [x \mid \text{cons}(xs, y)] \quad (2) \\
\text{length}([]) &= 0 \quad (3) \\
\text{length}[x \mid y]\] &= 1 * \text{length}(y) \quad (4)
\end{align*}
\]

- Prove \( \forall xs, P(xs) \)
  where \( P(xs) = \text{\text{length}(\text{cons}(xs, y))] = \text{\text{length}(xs)} + \text{\text{length}(ys)} \)
- Base case: \( P([ ]) \) (uses (1) and (3))
- Inductive case: \( P(xs) \rightarrow P(x: xs) \)

Software Examples

- sum
- times
- gcd
- length
- cons
- reverse
- maximum
- vector addition
- insertion sort
- merge
- merge sort
- quick sort
- exponentiation
- binary tree search
- circuit boards
- compilers
- operating systems

Boyer and Moore’s Computational Logic System
- Written in lisp and prove properties of lisp programs
- Significant properties verified
  - Lots of examples in reasoning about software
  - Supporting reasoning tools are needed:
    - Simplification: rewriting
    - Deduction: resolution

Applications

- Mathematical proof checking
- The QED Project
- Computer chip verifications
- Software verification

Axioms about Sequences

- Algebraic laws of sequence construction
  \([ ] \) list
  \([ \ldots ] \) : int, list \rightarrow list
- Informally
  \([ x \mid x_1, x_2, \ldots ] \] \(= [x, x_1, x_2, \ldots ] \)
- Definition of concatenation
  \( \text{cons} \) : list, list \rightarrow list
  \(\text{cons}([], y) = y \quad (1)\)
  \(\text{cons}[x \mid xs, y] = [x \mid \text{cons}(xs, y)] \quad (2)\)
- Definition of length
  \( \text{length} \) : list \rightarrow integer
  \(\text{length}([]) = 0 \quad (3)\)
  \(\text{length}[x \mid y]\] \(= 1 * \text{length}(y) \quad (4)\)
- Can we prove the following?
  \(\text{length}([x \mid y]\] \(= \text{length}(x) + \text{length}(y) \quad (5)\)
Mathematical Proof Checking

- Automated theorem provers do not "automate" math
- "Debugs" proofs
- Hard to use many proof checkers

The QED Project

- Effort of scientists from many laboratories and institutions
  - Will represent mathematical knowledge, technique
  - Based on a few pages of math
  - Still in early stages

The QED Project - Hoped Benefits

- Reduce mathematical "noise pollution."
- Speed publication of papers by taking focus off of proof checking. Referees can focus on relevance.
- Cultural monument to mathematics.

Chip Verification

- Formal vs. testbench
- Comparison verification
- NP-Complete
- IBM, Intel, AMD successes

Software Verification

- Hardware is more economically viable
- More design effort put into software
- => Software verification is viable
- Especially useful for critical applications: safety, e-commerce, military

Software Verification Paradox

- What will verify the verification program?
- Pragmatism does not demand ideal accuracy
- Significant improvement enough