Automated Mathematical Induction

Principle of Mathematical Induction

- \( \forall I \) — an inference rule with \( \forall n. P(n) \) as its conclusion
- 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 \( \forall I \)

Axioms about Sequences

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

Assume concatenation (cons), and length satisfy

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

\[\Box\text{Prove } \forall xs. P(xs)\]
where \(P(xs) \equiv \forall ys. \text{length}(\text{cons}(xs, ys)) = \text{length}(xs) + \text{length}(ys)\)

\[\Box\text{Base case: } P([ ]) \quad \{(\text{uses } (1) \text{ and } (3))\}\]

\[\Box\text{Inductive case: } P(xs) \rightarrow P(x: xs) \quad \text{length}(xs) \rightarrow \text{integer}\]

\[
= \text{length}([x | \text{cons}(xs, ys)] \\
= 1 + \text{length}([x | \text{cons}(xs, ys)] \\
= 1 + (\text{length}(xs) + \text{length}(ys)) \quad (\text{induction hypothesis, } P(xs) ) \\
= \text{length}(xs) + \text{length}(ys) \quad (+ \text{assoc}) \\
= \text{length}(x | xs) + \text{length}(ys) \quad (4) \text{ backward}
\]

Software Examples

- sum
- times
- gcd
- length
- cons
- reverse
- maximum
- vector addition

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
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.

---

Copyright © 2000 by Rex Page
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