Generic Theories as Proof Strategies

Wilfred J. Legato
National Security Agency

Outline of Talk

- Background
- Loop Invariant Theory
- Tail Recursion Theory
- Alternative Induction Theory
- An Example
- Concluding Remarks

A Floyd-Hoare Triple

```
         { N>0 }
         X := N
         A := 0
LOOP:    A := A + X
         X := X -1
         X>0 => GO TO LOOP
         { A = N*(N+1)/2 }
```

A Floyd-Hoare Proof

- Find a loop invariant $R(A,X)$

  $X \geq 0 \wedge A+X*(X+1)/2=N*(N+1)/2$

- Prove: (1) $N>0 \Rightarrow R(0,N)$
  (2) $X>0 \wedge R(A,X) \Rightarrow R(A+X,X-1)$
  (3) $X \leq 0 \wedge R(A,X) \Rightarrow A=N*(N+1)/2$

- Use identity $X*(X+1)/2=X+(X-1)*X/2$

Weakest Precondition Model

- Denote the postcondition by Q(A,X,N)

- Mechanically derive

$$\begin{gathered} wp(A,X,N) \\ = \\ \text{if } X>1 \ wp(A+X,X-1,N) \text{ else } Q(A+X,X-1,N) \end{gathered}$$

- Prove: N>0 => wp(0,N,N)

Attempted Weakest Precondition Proof

- Induction patterned after wp yields:
- Base case:
  $N=1 \Rightarrow wp(0,N,N))$
- Induction step:
  $N>1 \wedge wp(0,N-1,N-1) \Rightarrow wp(0,N,N)$
- Expansion of $wp(0,N,N)$ to $wp(N,N-1,N)$ does not match the hypothesis.

Capturing Proof Strategies

- We can prove weakest preconditions with the same ease as Floyd-Hoare.
- We will identify alternative strategies to deal with finding suitable inductions when recursive functions are applied to specialized arguments.
- We will use generic theories to capture and apply these strategies.

Loop Invariant Theory

- The most general weakest precondition may be represented by:

  (wp s) = (if (b s) (qp s) (wp (sigma s)))

  where b is the loop exit predicate, sigma represents the loop body, and qp is the postcondition.

Loop Invariant Theory

- We constrain wp by:

  (b s) => (wp s)=(qp s)

  (not (b s)) => (wp (sigma s))=(wp s)

- Since these are treated as rewrite rules the order of the equalities matters.

Loop Invariant Theory

- We constrain a measure function:

  ```
  (o-p (measure s))
  (not (b s)) =>
  (o< (measure (sigma s)) (measure s))
  ```

  in order to allow inductive proofs about wp.

Loop Invariant Theory

- Finally, we constrain a loop invariant r by:

  (not (b s)) ∧ (r s) => (r (sigma s))   (2)
  (b s) ∧ (r s) => (qp s)   (3)

  from which we prove

  (r s) => (wp s)   (1)

- This characterizes wp as the weakest loop invariant.

Summary of the Tail Recursion Theory

- Our goal in this theory is to remove the "a" component of state from the tail recursive function

  ```
  (g a s) = (if (bb s)
                (qt a s)
              (g (rho a s) (tau s)))
  ```

  where tau is measure decreasing.

Summary of the Tail Recursion Theory

- We introduce an invariant rt to capture underlying assumptions of the theory.

  (not (bb s)) ∧ (rt a s) =>
  (rt (rho a s) (tau s))

Summary of the Tail Recursion Theory

- We introduce functions op, h and hs with properties that allow us to prove

```
(rt a s) =>
     (g a s)=(if (bb s)
                 (qt a s)
                 (qt (op a (h s) s) (hs s)))
```

Summary of the Alternative Induction Theory

- This theory uses two tail recursive functions

  (fn1 s)=(if (b1 s) (q1 s) (fn1 (sigma1 s)))
  (fn2 s)=(if (b2 s) (q2 s) (fn2 (sigma2 s)))

  together with a mapping id-alt from the domain of fn1 to the domain of fn2, and a loop invariant p on the domain of fn1.

Summary of the Alternative Induction Theory

- The key requirement in this theory is

  (not (b1 s)) $\wedge$ (p s) =>
     (id-alt (sigma1 s))=(sigma2 (id-alt s))

- When fn1 and fn2 are identical, this property states that id-alt and sigma1 commute.

Summary of the Alternative Induction Theory

- This theory allows us to prove

  (p s) => (fn1 s)=(fn2 (id-alt s))

- Notice that when fn1 and fn2 are the same and id-alt is measure decreasing, id-alt defines an alternative induction.

Example Using the Loop Invariant Theory

- State consists of bytes A, N and flag C.

$\{ N>0 \wedge NS=N \wedge N*(N+1)/2<256 \}$

```
       LDA #0     load A immediate with 0
       CLC        clear the carry flag
LOOP:  ADC N      add with carry N to A
       DEC N      decrement N by 1
       BNE LOOP   branch if N>0 to LOOP
```

$\{ A=NS*(NS+1)/2 \}$

Example Using the Loop Invariant Theory

- The weakest precondition at LOOP is

```
(defun wp-loop (n a c ns)
    (declare (xargs :measure (dec n)))
    (if (equal (dec n) 0)
        (equal (mod (+ c a n) 256)
                    (floor (* ns (1+ ns)) 2))
      (wp-loop (dec n)
                 (mod (+ c a n) 256)
                 (floor (+ c a n) 256)
                 ns)))
```

Example Using the Loop Invariant Theory

- Where dec is defined by

```
(defun dec (n)
    (if (zp n) 255 (1- n)))
```

- The weakest precondition at the beginning of the program is

```
(defun wp-1 (n ns)
    (wp-loop n 0 0 ns))
```

Example Using the Loop Invariant Theory

- The proof goal is

```
(defthm wp-loop-is-correct
  (implies (and (not (zp n))
                (equal ns n)
                (< (floor (* n (1+ n)) 2)
                   256))
           (wp-1 n ns)))
```

The Automated Loop Invariant Proof

```
;;; Define the loop invariant

(defun sum-invariant (n a c ns)
  (and (not (zp n))
       (< (+ a (floor (* n (1+ n)) 2)) 256)
       (natp a)
       (equal c 0)
       (natp ns)
       (equal (+ a (floor (* n (1+ n)) 2))
              (floor (* ns (1+ ns)) 2))))
```

The Automated Loop Invariant Proof

```
;;; Instantiate the theory

(defthm wp-sum-loop-invariant
  (implies (sum-invariant n a c ns)
           (wp-loop n a c ns))
  :hints ((loop-invariant-hint ; computed hint
          'wp-loop             ; concrete weakest precondition
          '(sum-invariant n a c ns))))

(defthm wp-loop-is-correct
  (implies (and (not (zp n))
                (equal ns n)
                (< (floor (* n (1+ n)) 2) 256))
           (wp-1 n ns)))
```

A Comparison of the Theories

- We compare the theories on the sum program and the following multiply program.

$\{ F1=F1SAVE \wedge F1<256 \wedge F2<256 \wedge LOW<256 \}$

```
        LDX #8       load the X register immediate with 8
        LDA #0       load the A register immediate with 0
LOOP    ROR F1       rotate F1 right circular through carry
        BCC ZCOEF    branch on carry clear to ZCOEF
        CLC          clear the carry flag
        ADC F2       add with carry F2 to the contents of A
ZCOEF   ROR A        rotate A right circular through carry
        ROR LOW      rotate LOW right circular through carry
        DEX          decrement the X register by 1
        BNE LOOP     branch if X is non-zero to LOOP
```

$\{ LOW + 256*A = F1SAVE*F2 \}$

A Comparison of the Theories

- We count the number of supporting lemmas needed to prove the two programs with each of the generic theories. We use Robert Krug’s September 2003 modified ACL2 and arithmetic-4 proof library as well as NQTHM with my modularithmetic-98 proof library.

A Comparison of the Theories

Theorem Count for the Sum Program

	Generalization	Loop Invariant	Tail Recursion	Alt. Induction
ACL2	3	4	5	4
NQTHM	2	4	4	3

Theorem Count for the Multiply Program

	Generalization	Loop Invariant	Tail Recursion	Alt. Induction
ACL2	6	12	18	11
NQTHM	8	11	19	11

Conclusions

- Without automation generic theories are cumbersome to use compared with straight forward generalization.
- With automation they are effective means for high level proof structuring.
- Based upon these examples, ACL2 and NQTHM are roughly comparable in their ability to support arithmetic proofs over the naturals.