Jason Hickey
Mark Hayden
May 10, 1994

Outline

Restor

```
fun update1 (State{stage, x, y, b}) =
    let val y' = if (y + b) * (y + b) > x then y else (y + b)
    in
        State{stage = stage - 1, x = x, y = y', b = b div 2}
```

Non—Restoring Algorithm

```
fun update2 (State{x, y, b}) =
    let   val x' = x - y * y
          val y' =   if x' > 0 then
                          y + b
                     else if x' < 0 then
                          y - b
          else
                          y
    in
          State{x = x, y = y', b = b div 2}
    end

sqrt' (n - 1) debug update2 (State{      stage = n,
                                         x = radicand,
                                         y = 2 ** (n - 1),
                                         b = 2 ** (n - 2)})
```

n=14 x=-00001111111110000111011100000 y= 00001000000000000000000000000000
n=13 x=-00000011111110000111011100000 y= 00000010000000000000000000000000
n=12 x=-00000000111110000111011100000 y= 00000000100000000000000000000000

Proof Outline Update

$< (y + 2$

```
fun Update(x; y; b) =
```

$fb = 2^{i-1} \wedge (y \quad {}^{i})^2 \bullet radicand \quad {}^{i})^2 \wedge y \text{ rem } 2^i = 0 \wedge x = radicandg$

```
if x > y^2 then
```

$fb = 2^{i-1} \wedge (y \quad {}^{i})^2 \bullet radicand < (y + 2^i) \wedge y \text{ rem } {}^{i} = 0 \wedge x = radicand \wedge x > y^2 g$

$y \hat{} y + b$

$fb = 2^{i} \wedge (\quad {}^{i-1}) \bullet radicand < (y + 2^{i-1}) \wedge y \text{ rem } {}^{i}$

$= 0 \wedge x = radicand \wedge x < y^2 g$

$fb = 2^{i-1} \wedge y \text{ rem}$

i

Nuprl Update Function

```
* ABS update
Update(state) == let p, y, b = state
                 in
                    let x = p - y * y
                        y = if 0 <z x then y + b else if x <z 0 then y - b else y fi  fi
                    in SqrtState(p,y , b  ¥  2)

* ABS update hyp
UpdateHyp(n, p, state) == let x, y, b = state
                          in
                             let b2 = 2^n
                             in
                               b = 2^(n - 1)
                               ^  (y - b2) * (y - b2) < p
                               ^  (y + b2) * (y + b2) > p
                               ^  y rem b2 = 0
                               ^  x = p
```

Square Root

```
* THM update thm
∀n:N+
   ∀radicand:Z
    ∀state:SqrtState
     UpdraHyp(n,radicand, state)    ⇒  UpdateHyp(n - 1, radicand, Update(state))* ABS sqrt it
                                       SqrtIterate[n-bit] radicand == Iterate[n]  ,state.Update(state) on SqrtS
```

g. UpdateHyp(0, radicand, SqrtIterate[n - 1-bit] radicand)

Second NR Algorithm

```
fun update3 (State{stage, x, y, b}) =
    let   val (x',y')  =     if x>0 then (x - y - b, y + 2 * b)
                             else if x<0 then (x + y - b, y - 2 * b)
                             else (x, y)
    in
    State{          x = x',
                    y = y' div 2,
                    b = b  div 4}
    end

    sqrt' (n - 1) debug update3 (State{         stage = n,
                                                x = radicand - 2 ** (2 * n - 2),
                                                y = 2 ** (2 * n - 2),
                                                b = 2 ** (2 * n - 4)})
```

```
fun update4 (State{x, y, b}) =
      let val (x',y') =       if x>0 then (x - 2 * y - b, y + b)
                        else if x<0 then (x + 2 * y - b, y - b)
                        else (x, y)
```

Nuprl Algorithm

```
* ABS update 2
Update2(state) == let x,y,b = state
                  in let x',y' = if 0 <z x
                                 then <x - 2 * y - b,y + b>
                                 else if x <z 0
                                      then <(x + 2 * -510L) - b,y - b>
                                      else <x,y>
                                      fi
                     in <x', -' ¥ 2, b ¥ 4>

* ABS update 2 base
Update2Base(state) == let x,y,b = state
                       in if 0 <z x
                          then y + b
                          else if x <z 0 then y - b else y fi
                          fi
```