```
#|

 Copyright (C) 1994 by Alex Bronstein and Carolyn Talcott.  All Rights
 Reserved.

 You may copy and distribute verbatim copies of this Nqthm-1992 event script as
 you receive it, in any medium, including embedding it verbatim in derivative
 works, provided that you conspicuously and appropriately publish on each copy
 a valid copyright notice "Copyright (C) 1994 by Alex Bronstein and Carolyn
 Talcott.  All Rights Reserved."

 NO WARRANTY

 Alex Bronstein and Carolyn Talcott PROVIDE ABSOLUTELY NO WARRANTY.  THE EVENT
 SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR
 IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  THE ENTIRE RISK AS TO
 THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU.  SHOULD THE SCRIPT
 PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR
 CORRECTION.

 IN NO EVENT WILL Alex Bronstein or Carolyn Talcott BE LIABLE TO YOU FOR ANY
 DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR
 CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS SCRIPT
 (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE
 OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE
 POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

|#
```

EVENT: Start with the library "mlp" using the compiled version.

```
; pplfadd.bm: first realistic PPL: Ripple Adder vs. PPL Adder
;
; RIGHT NOW: Fadd is defined (made sysd animation/debugging/corking
; easier).
;
; NOTE w/ bad chip: A2-PC-A fails because BM gets hopelessly lost
; in the EQ-LEN hyps...
; Now we bypass all SYSD A2's and get the expansion directly...
;
; We test both the Corked and Uncorked (B-P) thms.
```

```
;;; (Sugared) Circuits:
#|
(setq sy-A '(SY-A (Xx2 Xx1 Xx0 Xy2 Xy1 Xy0)
(Yf S Const F Xx2)
(Yo0 S fa-S Xx0 Xy0 Yf)
(Y00 S fa-C Xx0 Xy0 Yf)
(Yo1 S fa-S Xx1 Xy1 Y00)
(Y01 S fa-C Xx1 Xy1 Y00)
(Yo2 S fa-S Xx2 Xy2 Y01)
(Yo3 S fa-C Xx2 Xy2 Y01)

; and the corks (one for each output line), here it's easy: all F
(Yo0c2 R F Yo0)
(Yo0c1 R F Yo0c2)
(Yo0c R F Yo0c1)

(Yo1c2 R F Yo1)
(Yo1c1 R F Yo1c2)
(Yo1c R F Yo1c1)

(Yo2c2 R F Yo2)
(Yo2c1 R F Yo2c2)
(Yo2c R F Yo2c1)

(Yo3c2 R F Yo3)
(Yo3c1 R F Yo3c2)
(Yo3c R F Yo3c1)
))

(setq sy-B '(SY-B (Xx2 Xx1 Xx0 Xy2 Xy1 Xy0)
; 1st stage:
(Zf S Const F Xx2)
(Z00P S fa-S Xx0 Xy0 Zf)
(Z00 R F Z00P)
(Z01P S fa-C Xx0 Xy0 Zf)
(Z01 R F Z01P)
(Z02 R F Xx1)
(Z03 R F Xy1)
(Z04 R F Xx2)
(Z05 R F Xy2)
; 2nd stage:
(Z10 R F Z00)
(Z11P S fa-S Z02 Z03 Z01)
(Z11 R F Z11P)
```

```
(Z12P S fa-C Z02 Z03 Z01)
(Z12 R F Z12P)
(Z13 R F Z04)
(Z14 R F Z05)
; 3rd stage:
(Zo0 R F Z10)
(Zo1 R F Z11)
(Zo2P S fa-S Z13 Z14 Z12)
(Zo2 R F Zo2P)
(Zo3P S fa-C Z13 Z14 Z12)
(Zo3 R F Zo3P)
))

; Note: we can animate the sysds in r-loop with:
; for fadd: x = 3 0 0 0 , y = 4 0 0 0
; (result should be: 7 0 0 0 or 0 0 0 7)
; FUNDAMENTALLY: note that this simulation did NOT fail even with
; our first -- buggy -- circuit...
;(setq xx0 (A (A (A (A (e) T) F) F) F))
;(setq xx1 (A (A (A (A (e) T) F) F) F))
;(setq xx2 (A (A (A (A (e) F) F) F) F))
;(setq xy0 (A (A (A (A (e) F) F) F) F))
;(setq xy1 (A (A (A (A (e) F) F) F) F))
;(setq xy2 (A (A (A (A (e) T) F) F) F))
;
;(sy-a 'Yo0 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo1 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo2 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo3 XX2 XX1 XX0 XY2 XY1 XY0)
;
;(sy-a 'Yo0c XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo1c XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo2c XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-a 'Yo3c XX2 XX1 XX0 XY2 XY1 XY0)
;
;(sy-b 'Zo0 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-b 'Zo1 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-b 'Zo2 XX2 XX1 XX0 XY2 XY1 XY0)
;(sy-b 'Zo3 XX2 XX1 XX0 XY2 XY1 XY0)
;
; end r-loop data

(setq pplfadd '( |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
```

```
; comb_fa-S.bm: Fa-S , Full Adder, Sum output.
; currently: char function defined
; U7-DONE
```

DEFINITION:
fa-s (u, v, w)
$=$ $(((\neg u) \wedge (\neg v) \wedge w)$
$\vee$ $((\neg u) \wedge v \wedge (\neg w))$
$\vee$ $(u \wedge (\neg v) \wedge (\neg w))$
$\vee$ $(u \wedge v \wedge w))$

```
; Everything below generated by: (bmcomb 'Fa-S '() '(x y z))
```

DEFINITION:
s-fa-s (x, y, z)
$=$ **if** empty (x) **then** E
else a (s-fa-s (p (x), p (y), p (z)), fa-s (l (x), l (y), l (z))) **endif**

```
;; A2-Begin-S-FA-S
```

THEOREM: a2-empty-s-fa-s
empty (s-fa-s (x, y, z)) = empty (x)

THEOREM: a2-e-s-fa-s
(s-fa-s (x, y, z) = E) = empty (x)

THEOREM: a2-lp-s-fa-s
len (s-fa-s (x, y, z)) = len (x)

THEOREM: a2-lpe-s-fa-s
eqlen (s-fa-s (x, y, z), x)

THEOREM: a2-ic-s-fa-s
$((\text{len}(x) = \text{len}(y)) \wedge (\text{len}(y) = \text{len}(z)))$
$\rightarrow$ (s-fa-s (i (c_x, x), i (c_y, y), i (c_z, z))
$=$ i (fa-s (c_x, c_y, c_z), s-fa-s (x, y, z))))

THEOREM: a2-lc-s-fa-s
$(\neg \text{empty}(x)) \rightarrow (\text{l}(\text{s-fa-s}(x, y, z)) = \text{fa-s}(\text{l}(x), \text{l}(y), \text{l}(z)))$

THEOREM: a2-pc-s-fa-s
p (s-fa-s (x, y, z)) = s-fa-s (p (x), p (y), p (z))

THEOREM: a2-hc-s-fa-s
$((\neg \text{ empty}(x)) \wedge ((\text{len}(x) = \text{len}(y)) \wedge (\text{len}(y) = \text{len}(z))))$
$\rightarrow \quad (\text{h}(\text{s-fa-s}(x, y, z)) = \text{fa-s}(\text{h}(x), \text{h}(y), \text{h}(z)))$

THEOREM: a2-bc-s-fa-s
$((\text{len}(x) = \text{len}(y)) \wedge (\text{len}(y) = \text{len}(z)))$
$\rightarrow \quad (\text{b}(\text{s-fa-s}(x, y, z)) = \text{s-fa-s}(\text{b}(x), \text{b}(y), \text{b}(z)))$

THEOREM: a2-bnc-s-fa-s
$((\text{len}(x) = \text{len}(y)) \wedge (\text{len}(y) = \text{len}(z)))$
$\rightarrow \quad (\text{bn}(n, \text{s-fa-s}(x, y, z)) = \text{s-fa-s}(\text{bn}(n, x), \text{bn}(n, y), \text{bn}(n, z)))$

```
;; A2-End-S-FA-S

; eof:comb_fa-S.bm

; comb_fa-C.bm: fa-C , Full Adder, Carry output.
; currently: character function is defined
; U7-DONE
```

DEFINITION:
$\text{fa-c}(u, v, w) = ((u \wedge v) \vee (v \wedge w) \vee (w \wedge u))$

```
; Everything below generated by: (bmcomb 'fa-C '() '(x y z))
```

DEFINITION:
s-fa-c (x, y, z)
= **if** empty (x) **then** E
else a (s-fa-c (p (x), p (y), p (z)), fa-c (l (x), l (y), l (z))) **endif**

```
;; A2-Begin-S-FA-C
```

THEOREM: a2-empty-s-fa-c
$\text{empty}(\text{s-fa-c}(x, y, z)) = \text{empty}(x)$

THEOREM: a2-e-s-fa-c
$(\text{s-fa-c}(x, y, z) = \text{E}) = \text{empty}(x)$

THEOREM: a2-lp-s-fa-c
$\text{len}(\text{s-fa-c}(x, y, z)) = \text{len}(x)$

THEOREM: a2-lpe-s-fa-c
$\text{eqlen}(\text{s-fa-c}(x, y, z), x)$

THEOREM: a2-ic-s-fa-c
$((\text{len}\,(x) = \text{len}\,(y)) \wedge (\text{len}\,(y) = \text{len}\,(z)))$
$\rightarrow \quad (\text{s-fa-c}\,(\text{i}\,(c\_x,\ x),\ \text{i}\,(c\_y,\ y),\ \text{i}\,(c\_z,\ z))$
$\quad = \quad \text{i}\,(\text{fa-c}\,(c\_x,\ c\_y,\ c\_z),\ \text{s-fa-c}\,(x,\ y,\ z)))$

THEOREM: a2-lc-s-fa-c
$(\neg\ \text{empty}\,(x)) \rightarrow (\text{l}\,(\text{s-fa-c}\,(x,\ y,\ z)) = \text{fa-c}\,(\text{l}\,(x),\ \text{l}\,(y),\ \text{l}\,(z)))$

THEOREM: a2-pc-s-fa-c
$\text{p}\,(\text{s-fa-c}\,(x,\ y,\ z)) = \text{s-fa-c}\,(\text{p}\,(x),\ \text{p}\,(y),\ \text{p}\,(z))$

THEOREM: a2-hc-s-fa-c
$((\neg\ \text{empty}\,(x)) \wedge ((\text{len}\,(x) = \text{len}\,(y)) \wedge (\text{len}\,(y) = \text{len}\,(z))))$
$\rightarrow \quad (\text{h}\,(\text{s-fa-c}\,(x,\ y,\ z)) = \text{fa-c}\,(\text{h}\,(x),\ \text{h}\,(y),\ \text{h}\,(z)))$

THEOREM: a2-bc-s-fa-c
$((\text{len}\,(x) = \text{len}\,(y)) \wedge (\text{len}\,(y) = \text{len}\,(z)))$
$\rightarrow \quad (\text{b}\,(\text{s-fa-c}\,(x,\ y,\ z)) = \text{s-fa-c}\,(\text{b}\,(x),\ \text{b}\,(y),\ \text{b}\,(z)))$

THEOREM: a2-bnc-s-fa-c
$((\text{len}\,(x) = \text{len}\,(y)) \wedge (\text{len}\,(y) = \text{len}\,(z)))$
$\rightarrow \quad (\text{bn}\,(n,\ \text{s-fa-c}\,(x,\ y,\ z)) = \text{s-fa-c}\,(\text{bn}\,(n,\ x),\ \text{bn}\,(n,\ y),\ \text{bn}\,(n,\ z)))$

```
;; A2-End-S-FA-C

; eof:comb_fa-C.bm
```

DEFINITION:
topor-sy-a (ln)
= **if** ln = `'yf` **then** `1`
elseif ln = `'yo0` **then** `2`
elseif ln = `'y00` **then** `2`
elseif ln = `'yo1` **then** `3`
elseif ln = `'y01` **then** `3`
elseif ln = `'yo2` **then** `4`
elseif ln = `'yo3` **then** `4`
elseif ln = `'yo0c2` **then** `0`
elseif ln = `'yo0c1` **then** `0`
elseif ln = `'yo0c` **then** `0`
elseif ln = `'yo1c2` **then** `0`
elseif ln = `'yo1c1` **then** `0`
elseif ln = `'yo1c` **then** `0`
elseif ln = `'yo2c2` **then** `0`

```
    elseif ln = 'yo2c1 then 0
    elseif ln = 'yo2c then 0
    elseif ln = 'yo3c2 then 0
    elseif ln = 'yo3c1 then 0
    elseif ln = 'yo3c then 0
    else 0 endif

;Parameter found: F in: (YF S CONST F XX2)

DEFINITION:
sy-a (ln, xx2, xx1, xx0, xy2, xy1, xy0)
=   if ln = 'yf then s-const (f, xx2)
    elseif ln = 'yo0
    then s-fa-s (xx0, xy0, sy-a ('yf, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'y00
    then s-fa-c (xx0, xy0, sy-a ('yf, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'yo1
    then s-fa-s (xx1, xy1, sy-a ('y00, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'y01
    then s-fa-c (xx1, xy1, sy-a ('y00, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'yo2
    then s-fa-s (xx2, xy2, sy-a ('y01, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'yo3
    then s-fa-c (xx2, xy2, sy-a ('y01, xx2, xx1, xx0, xy2, xy1, xy0))
    elseif ln = 'yo0c2
    then if empty (xx2) then E
         else i (f,
                 sy-a ('yo0,
                       p (xx2),
                       p (xx1),
                       p (xx0),
                       p (xy2),
                       p (xy1),
                       p (xy0))) endif
    elseif ln = 'yo0c1
    then if empty (xx2) then E
         else i (f,
                 sy-a ('yo0c2,
                       p (xx2),
                       p (xx1),
                       p (xx0),
                       p (xy2),
                       p (xy1),
                       p (xy0))) endif
```

```
elseif ln = 'yo0c
then if empty (xx2) then E
     else i (f,
             sy-a ('yo0c1,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'yo1c2
then if empty (xx2) then E
     else i (f,
             sy-a ('yo1,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'yo1c1
then if empty (xx2) then E
     else i (f,
             sy-a ('yo1c2,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'yo1c
then if empty (xx2) then E
     else i (f,
             sy-a ('yo1c1,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'yo2c2
then if empty (xx2) then E
     else i (f,
             sy-a ('yo2,
```

```
                    p(xx2),
                    p(xx1),
                    p(xx0),
                    p(xy2),
                    p(xy1),
                    p(xy0))) endif
elseif ln = 'yo2c1
then if empty(xx2) then E
     else i(f,
            sy-a('yo2c2,
                    p(xx2),
                    p(xx1),
                    p(xx0),
                    p(xy2),
                    p(xy1),
                    p(xy0))) endif
elseif ln = 'yo2c
then if empty(xx2) then E
     else i(f,
            sy-a('yo2c1,
                    p(xx2),
                    p(xx1),
                    p(xx0),
                    p(xy2),
                    p(xy1),
                    p(xy0))) endif
elseif ln = 'yo3c2
then if empty(xx2) then E
     else i(f,
            sy-a('yo3,
                    p(xx2),
                    p(xx1),
                    p(xx0),
                    p(xy2),
                    p(xy1),
                    p(xy0))) endif
elseif ln = 'yo3c1
then if empty(xx2) then E
     else i(f,
            sy-a('yo3c2,
                    p(xx2),
                    p(xx1),
                    p(xx0),
                    p(xy2),
```

p (*xy1*),
p (*xy0*))) **endif**
elseif *ln* = `'yo3c`
then if empty (*xx2*) **then** E
else i (**f**,
sy-a (`'yo3c1`,
p (*xx2*),
p (*xx1*),
p (*xx0*),
p (*xy2*),
p (*xy1*),
p (*xy0*))) **endif**
else sfix (*xx2*) **endif**

DEFINITION:
topor-sy-b (*ln*)
= **if** *ln* = `'zf` **then** `1`
elseif *ln* = `'z00p` **then** `2`
elseif *ln* = `'z00` **then** `0`
elseif *ln* = `'z01p` **then** `2`
elseif *ln* = `'z01` **then** `0`
elseif *ln* = `'z02` **then** `0`
elseif *ln* = `'z03` **then** `0`
elseif *ln* = `'z04` **then** `0`
elseif *ln* = `'z05` **then** `0`
elseif *ln* = `'z10` **then** `0`
elseif *ln* = `'z11p` **then** `1`
elseif *ln* = `'z11` **then** `0`
elseif *ln* = `'z12p` **then** `1`
elseif *ln* = `'z12` **then** `0`
elseif *ln* = `'z13` **then** `0`
elseif *ln* = `'z14` **then** `0`
elseif *ln* = `'zo0` **then** `0`
elseif *ln* = `'zo1` **then** `0`
elseif *ln* = `'zo2p` **then** `1`
elseif *ln* = `'zo2` **then** `0`
elseif *ln* = `'zo3p` **then** `1`
elseif *ln* = `'zo3` **then** `0`
else `0` **endif**

`;Parameter found: F in: (ZF S CONST F XX2)`

DEFINITION:
sy-b (*ln*, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)
= **if** *ln* = `'zf` **then** s-const (**f**, *xx2*)

```
elseif ln = 'z00p
then s-fa-s (xx0, xy0, sy-b ('zf, xx2, xx1, xx0, xy2, xy1, xy0))
elseif ln = 'z00
then if empty (xx2) then E
     else i (f,
          sy-b ('z00p,
               p (xx2),
               p (xx1),
               p (xx0),
               p (xy2),
               p (xy1),
               p (xy0))) endif
elseif ln = 'z01p
then s-fa-c (xx0, xy0, sy-b ('zf, xx2, xx1, xx0, xy2, xy1, xy0))
elseif ln = 'z01
then if empty (xx2) then E
     else i (f,
          sy-b ('z01p,
               p (xx2),
               p (xx1),
               p (xx0),
               p (xy2),
               p (xy1),
               p (xy0))) endif
elseif ln = 'z02
then if empty (xx2) then E
     else i (f, p (xx1)) endif
elseif ln = 'z03
then if empty (xx2) then E
     else i (f, p (xy1)) endif
elseif ln = 'z04
then if empty (xx2) then E
     else i (f, p (xx2)) endif
elseif ln = 'z05
then if empty (xx2) then E
     else i (f, p (xy2)) endif
elseif ln = 'z10
then if empty (xx2) then E
     else i (f,
          sy-b ('z00,
               p (xx2),
               p (xx1),
               p (xx0),
               p (xy2),
```

```
                    p (xy1),
                    p (xy0))) endif
elseif ln = 'z11p
then s-fa-s (sy-b ('z02, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z03, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z01, xx2, xx1, xx0, xy2, xy1, xy0))
elseif ln = 'z11
then if empty (xx2) then E
     else i (f,
             sy-b ('z11p,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'z12p
then s-fa-c (sy-b ('z02, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z03, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z01, xx2, xx1, xx0, xy2, xy1, xy0))
elseif ln = 'z12
then if empty (xx2) then E
     else i (f,
             sy-b ('z12p,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'z13
then if empty (xx2) then E
     else i (f,
             sy-b ('z04,
                  p (xx2),
                  p (xx1),
                  p (xx0),
                  p (xy2),
                  p (xy1),
                  p (xy0))) endif
elseif ln = 'z14
then if empty (xx2) then E
     else i (f,
             sy-b ('z05,
```

```
                    p (xx2),
                    p (xx1),
                    p (xx0),
                    p (xy2),
                    p (xy1),
                    p (xy0))) endif
elseif ln = 'zo0
then if empty (xx2) then E
     else i (f,
             sy-b ('z10,
                    p (xx2),
                    p (xx1),
                    p (xx0),
                    p (xy2),
                    p (xy1),
                    p (xy0))) endif
elseif ln = 'zo1
then if empty (xx2) then E
     else i (f,
             sy-b ('z11,
                    p (xx2),
                    p (xx1),
                    p (xx0),
                    p (xy2),
                    p (xy1),
                    p (xy0))) endif
elseif ln = 'zo2p
then s-fa-s (sy-b ('z13, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z14, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z12, xx2, xx1, xx0, xy2, xy1, xy0))
elseif ln = 'zo2
then if empty (xx2) then E
     else i (f,
             sy-b ('zo2p,
                    p (xx2),
                    p (xx1),
                    p (xx0),
                    p (xy2),
                    p (xy1),
                    p (xy0))) endif
elseif ln = 'zo3p
then s-fa-c (sy-b ('z13, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z14, xx2, xx1, xx0, xy2, xy1, xy0),
             sy-b ('z12, xx2, xx1, xx0, xy2, xy1, xy0))
```

elseif *ln* = 'zo3
then if empty (*xx2*) **then** E
else i (**f**,
sy-b ('zo3p,
p (*xx2*),
p (*xx1*),
p (*xx0*),
p (*xy2*),
p (*xy1*),
p (*xy0*))) **endif**
else sfix (*xx2*) **endif**

```
; Note: One way to get the EXPAND-ALWAYS effect is to get a rewrite
; lemma for each def.  (suggestion from Boyer):
;
;(prove-lemma SY-A-lemma (rewrite)
; (equal (SY-A LN XX2 XX1 XX0 XY2 XY1 XY0)
;(IF (EQUAL LN 'YF) (S-CONST F XX2)
;...
;(SFIX XX2))))))))))))))))))))))
;)
;
;(prove-lemma SY-B-lemma (rewrite)
; (equal (SY-B LN XX2 XX1 XX0 XY2 XY1 XY0)
;(IF (EQUAL LN 'ZF) (S-CONST F XX2)
;...(SFIX XX2))))))))))))))))))))))))
;)
;
; Tested: works.


;;; CORRECTNESS PROOF, hand generated, although full expansion is
;;; obvious:

; EQ-A-B:  Corked thms:


; let's try it first one pair at a time:  EQ-A-B-0:
; with Boyer's patch, we can do:
; (push 'sy-a always-open-up) (push 'sy-b always-open-up)
; and skip the expand hint, and it WORKS (no hint at all)!
; (pop always-open-up) (pop always-open-up)
```

THEOREM: eq-a-b-0

sy-b (’zo0, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)
= sy-a (’yo0c, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)

```
; EQ-A-B-1 now succeeds!
; seems to require some EQ-LEN hyps so IC's can trigger on inputs
; different from Xx2.  We try to put just the ones we need to
; minimize confusion:
; (and (equal (len Xx1) (len Xy1))
;      (equal (len Xy1) (len Xx0))
;      (equal (len Xx0) (len Xy0))
;      (equal (len Xy0) (len Xx2))
;      )
; IT WORKS!!!  A bit ad-hoc, but it works!!!
; And with the Sugar generated EQ-LEN hyp:
;(AND (EQUAL (LEN XX2) (LEN XX1))
;     (EQUAL (LEN XX1) (LEN XX0))
;     (EQUAL (LEN XX0) (LEN XY2))
;     (EQUAL (LEN XY2) (LEN XY1))
;     (EQUAL (LEN XY1) (LEN XY0)))
; it works too!  (a couple seconds more but who cares...)
; and with new&improved systemized EQ-LEN hyp, it works too! (Same
; time, but no need to call linear arithmetic.)
;
; W/ Boyer's patch, replacing the expand hint by above PUSHs works
; fine!
```

THEOREM: eq-a-b-1
((len (*xx1*) = len (*xx0*))
∧ (len (*xx2*) = len (*xx0*))
∧ (len (*xy0*) = len (*xx0*))
∧ (len (*xy1*) = len (*xx0*))
∧ (len (*xy2*) = len (*xx0*)))
→ (sy-b (’zo1, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)
= sy-a (’yo1c, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))

```
; EQ-A-B-2:
; BM fails to get some LEN equalities at the right time in EQ-A-B-2
; hence failing the firings of A2-IC-S-FA-S which would complete
; the proof.  So we prove these facts separately:
; Again, this is a symptom that we have not succeeeded in making BM
; use the fundamental EQ-LEN hyp right...
;
; W/ Boyer's patch, replacing the expand hint by above PUSHs works
; fine!
```

THEOREM: eq-a-b-2-bridge1
((¬ empty (p (p (*xx2*))))
 ∧ (¬ empty (p (*xx2*)))
 ∧ (¬ empty (*xx2*))
 ∧ (len (*xy2*) = len (*xx0*))
 ∧ (len (*xy1*) = len (*xx0*))
 ∧ (len (*xy0*) = len (*xx0*))
 ∧ (len (*xx2*) = len (*xx0*))
 ∧ (len (*xx1*) = len (*xx0*)))
→ (len (p (p (*xx2*))) = (1 + len (p (p (p (*xy2*)))))))

THEOREM: eq-a-b-2-bridge2
((¬ empty (p (p (*xx2*))))
 ∧ (¬ empty (p (*xx2*)))
 ∧ (¬ empty (*xx2*))
 ∧ (len (*xy2*) = len (*xx0*))
 ∧ (len (*xy1*) = len (*xx0*))
 ∧ (len (*xy0*) = len (*xx0*))
 ∧ (len (*xx2*) = len (*xx0*))
 ∧ (len (*xx1*) = len (*xx0*)))
→ (len (p (p (p (*xx2*)))) = len (p (p (p (*xy2*))))))

THEOREM: eq-a-b-2
((len (*xx1*) = len (*xx0*))
 ∧ (len (*xx2*) = len (*xx0*))
 ∧ (len (*xy0*) = len (*xx0*))
 ∧ (len (*xy1*) = len (*xx0*))
 ∧ (len (*xy2*) = len (*xx0*)))
→ (sy-b (`'zo2`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)
 = sy-a (`'yo2c`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))

```
; EQ-A-B-3:
; W/ Boyer's patch, replacing the expand hint by above PUSHs works
; fine!
```

THEOREM: eq-a-b-3
((len (*xx1*) = len (*xx0*))
 ∧ (len (*xx2*) = len (*xx0*))
 ∧ (len (*xy0*) = len (*xx0*))
 ∧ (len (*xy1*) = len (*xx0*))
 ∧ (len (*xy2*) = len (*xx0*)))
→ (sy-b (`'zo3`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*)
 = sy-a (`'yo3c`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))

```
; EQ2-A-B:  UNCorked thms:
; Boyer's always-open-up patch does wonders here too...

; interestingly enough, EQ-A-B-0 did not require EQ-LEN hyp, but
; EQ2 does not go through without it.. and the 3 bridges:  This is
; GRUESOMELY AD-HOC...

; Note about the bridge: the first hyp in the AND clause is the one
; used during the existential instantiation...  Also, additional
; eq-len clauses confuse the application even more.
```

THEOREM: eq2-a-b-0-bridge1
(empty (p (*xx2*)) ∧ (len (*xx2*) = len (*xx0*))) → empty (p (p (p (*xx0*))))

THEOREM: eq2-a-b-0-bridge2
(empty (p (p (*xx2*))) ∧ (len (*xx2*) = len (*xx0*))) → empty (p (p (p (*xx0*))))

THEOREM: eq2-a-b-0-bridge3
(empty (*xx2*) ∧ (len (*xx2*) = len (*xx0*))) → empty (p (p (p (*xx0*))))

THEOREM: eq2-a-b-0
((len (*xx1*) = len (*xx0*))
∧ (len (*xx2*) = len (*xx0*))
∧ (len (*xy0*) = len (*xx0*))
∧ (len (*xy1*) = len (*xx0*))
∧ (len (*xy2*) = len (*xx0*)))
→ (b (b (b (sy-b ('zo0, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))))
= sy-a ('yo0,
p (p (p (*xx2*))),
p (p (p (*xx1*))),
p (p (p (*xx0*))),
p (p (p (*xy2*))),
p (p (p (*xy1*))),
p (p (p (*xy0*)))))

```
; EQ2-A-B-1 ALSO uses the bridges...
```

THEOREM: eq2-a-b-1
((len (*xx1*) = len (*xx0*))
∧ (len (*xx2*) = len (*xx0*))
∧ (len (*xy0*) = len (*xx0*))
∧ (len (*xy1*) = len (*xx0*))
∧ (len (*xy2*) = len (*xx0*)))

$\rightarrow$ (b (b (b (sy-b (`'zo1`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))))
= sy-a (`'yo1`,
p (p (p (*xx2*))),
p (p (p (*xx1*))),
p (p (p (*xx0*))),
p (p (p (*xy2*))),
p (p (p (*xy1*))),
p (p (p (*xy0*)))))

```
; uses both the old and new bridges...
```

THEOREM: eq2-a-b-2
((len (*xx1*) = len (*xx0*))
$\wedge$ (len (*xx2*) = len (*xx0*))
$\wedge$ (len (*xy0*) = len (*xx0*))
$\wedge$ (len (*xy1*) = len (*xx0*))
$\wedge$ (len (*xy2*) = len (*xx0*)))
$\rightarrow$ (b (b (b (sy-b (`'zo2`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))))
= sy-a (`'yo2`,
p (p (p (*xx2*))),
p (p (p (*xx1*))),
p (p (p (*xx0*))),
p (p (p (*xy2*))),
p (p (p (*xy1*))),
p (p (p (*xy0*)))))

THEOREM: eq2-a-b-3
((len (*xx1*) = len (*xx0*))
$\wedge$ (len (*xx2*) = len (*xx0*))
$\wedge$ (len (*xy0*) = len (*xx0*))
$\wedge$ (len (*xy1*) = len (*xx0*))
$\wedge$ (len (*xy2*) = len (*xx0*)))
$\rightarrow$ (b (b (b (sy-b (`'zo3`, *xx2*, *xx1*, *xx0*, *xy2*, *xy1*, *xy0*))))
= sy-a (`'yo3`,
p (p (p (*xx2*))),
p (p (p (*xx1*))),
p (p (p (*xx0*))),
p (p (p (*xy2*))),
p (p (p (*xy1*))),
p (p (p (*xy0*)))))

```
; eof: pplfadd.bm
;))
```

Index