```
#|

 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.

```
; pplfun3.bm is our 2nd PIPELINE proof: it generalizes Inc to FUN1.
;
; Note: we run into trouble w/ R-loop corking, because of the DCL'd function.
; A partial kludge is to:
; (LOAD "Comb/comb_fun1_s.bm") , i.e. with the _s extension, getting an actual
; symbolic value (a list).  But we really need full BY NAME evaluation..

;;; (Sugared) Circuits:
#|
(setq A '(SY-A (x)
(Y1 S Fun1 x)
```

```
(Y2 S Fun1 Y1)
(Y3 S Fun1 Y2)
; and the cork:
(Yc2 R (FUN1 (FUN1 0)) Y3)
(Yc1 R (FUN1 0) Yc2)
(Yout R 0 Yc1)
))

(setq B '(SY-B (x)
(Z1 S Fun1 x)
(Z2 R 0 Z1)
(Z3 S Fun1 Z2)
(Z4 R 0 Z3)
(Z5 S Fun1 Z4)
(Zout R 0 Z5)
))

(setq pplfun3 '( |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_fun1.bm: Fun1 combinational element
; U7-DONE

; arbitrary Char-Fun of arity 1:
```

EVENT: Introduce the function symbol *fun1* of one argument.

```
; Everything below generated by: (bmcomb 'Fun1 '() '(x))
```

DEFINITION:
s-fun1 (x)
= **if** empty (x) **then** E
else a (s-fun1 (p (x)), fun1 (l (x))) **endif**

```
;; A2-Begin-S-FUN1
```

THEOREM: a2-empty-s-fun1
empty (s-fun1 (x)) = empty (x)

THEOREM: a2-e-s-fun1
(s-fun1 (x) = E) = empty (x)

THEOREM: a2-lp-s-fun1
len (s-fun1 (x)) = len (x)

THEOREM: a2-lpe-s-fun1
eqlen (s-fun1 (x), x)

THEOREM: a2-ic-s-fun1
s-fun1 (i (c_x, x)) = i (fun1 (c_x), s-fun1 (x))

THEOREM: a2-lc-s-fun1
($\neg$ empty (x)) $\rightarrow$ (l (s-fun1 (x)) = fun1 (l (x)))

THEOREM: a2-pc-s-fun1
p (s-fun1 (x)) = s-fun1 (p (x))

THEOREM: a2-hc-s-fun1
($\neg$ empty (x)) $\rightarrow$ (h (s-fun1 (x)) = fun1 (h (x)))

THEOREM: a2-bc-s-fun1
b (s-fun1 (x)) = s-fun1 (b (x))

THEOREM: a2-bnc-s-fun1
bn (n, s-fun1 (x)) = s-fun1 (bn (n, x))

```
;; A2-End-S-FUN1

; eof:comb_fun1.bm
```

DEFINITION:
topor-sy-a (ln)
= **if** ln = 'y1 **then** 1
elseif ln = 'y2 **then** 2
elseif ln = 'y3 **then** 3
elseif ln = 'yc2 **then** 0
elseif ln = 'yc1 **then** 0
elseif ln = 'yout **then** 0
else 0 **endif**

DEFINITION:
sy-a (ln, x)
= **if** ln = 'y1 **then** s-fun1 (x)
elseif ln = 'y2 **then** s-fun1 (sy-a ('y1, x))
elseif ln = 'y3 **then** s-fun1 (sy-a ('y2, x))
elseif ln = 'yc2
then if empty (x) **then** E
else i (fun1 (fun1 (0)), sy-a ('y3, p (x))) **endif**
elseif ln = 'yc1

then if empty (*x*) **then** E
else i (fun1 (0), sy-a ('yc2, p (*x*))) **endif**
elseif *ln* = 'yout
then if empty (*x*) **then** E
else i (0, sy-a ('yc1, p (*x*))) **endif**
else sfix (*x*) **endif**

```
;; A2-Begin-SY-A
```

THEOREM: a2-empty-sy-a
empty (sy-a (*ln*, *x*)) = empty (*x*)

THEOREM: a2-e-sy-a
(sy-a (*ln*, *x*) = E) = empty (*x*)

THEOREM: a2-lp-sy-a
len (sy-a (*ln*, *x*)) = len (*x*)

THEOREM: a2-lpe-sy-a
eqlen (sy-a (*ln*, *x*), *x*)

THEOREM: a2-pc-sy-a
p (sy-a (*ln*, *x*)) = sy-a (*ln*, p (*x*))

```
;; A2-End-SY-A
```

DEFINITION:
topor-sy-b (*ln*)
= **if** *ln* = 'z1 **then** 1
elseif *ln* = 'z2 **then** 0
elseif *ln* = 'z3 **then** 1
elseif *ln* = 'z4 **then** 0
elseif *ln* = 'z5 **then** 1
elseif *ln* = 'zout **then** 0
else 0 **endif**

DEFINITION:
sy-b (*ln*, *x*)
= **if** *ln* = 'z1 **then** s-fun1 (*x*)
elseif *ln* = 'z2
then if empty (*x*) **then** E
else i (0, sy-b ('z1, p (*x*))) **endif**
elseif *ln* = 'z3 **then** s-fun1 (sy-b ('z2, *x*))
elseif *ln* = 'z4

then if empty (x) **then** E
else i (0, sy-b ('z3, p (x))) **endif**
elseif ln = 'z5 **then** s-fun1 (sy-b ('z4, x))
elseif ln = 'zout
then if empty (x) **then** E
else i (0, sy-b ('z5, p (x))) **endif**
else sfix (x) **endif**

```
;; A2-Begin-SY-B
```

THEOREM: a2-empty-sy-b
empty (sy-b (ln, x)) = empty (x)

THEOREM: a2-e-sy-b
(sy-b (ln, x) = E) = empty (x)

THEOREM: a2-lp-sy-b
len (sy-b (ln, x)) = len (x)

THEOREM: a2-lpe-sy-b
eqlen (sy-b (ln, x), x)

THEOREM: a2-pc-sy-b
p (sy-b (ln, x)) = sy-b (ln, p (x))

```
;; A2-End-SY-B

;;; CORRECTNESS PROOF (hand generated, dreamer!):


; EQ-A-B:
;   We start out directly with the expansion list form pplinc3.
```

THEOREM: eq-a-b
sy-b ('zout, x) = sy-a ('yout, x)

```
; eof: pplfun3.bm
;))
```

Index