```
#|

 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.

```
; corrSL.bm is the FULL version of the Saxe/Leiserson retimed
; correlator.
; PROOF HISTORY:
; - very interestingly, a bug in the first definition I entered was
; shown by BM reducing a case to:
;          (IMPLIES (AND (STRINGP X)
;                        (NOT (EQUAL X (E)))
;                        (EQUAL (P X) (E))
;                        (NOT (EQUAL 'A2 (L X))))
;                   (EQUAL 4 (PLUS 1 1 0 1))).
; which is obviously false, and describes precisely the
```

```
; counterexample!
; - a first correctness proof (brute expansion) was obtained in
; LESS THAN 1 HOUR, and that includes finding out the definition
; was wrong!

;;; (Sugared) Circuits:
#|
(setq A '(SY-A (x)
(Y1 R 'a0 x)
(Y2 S Del 'a0 Y1)
(Y3 R 'a1 Y1)
(Y4 S Del 'a1 Y3)
(Y5 R 'a2 Y3)
(Y6 S Del 'a2 Y5)
(Y7 R 'a3 Y5)
(Y8 S Del 'a3 Y7)
(Y9 S Plus Y6 Y8)
(Y10 S Plus Y4 Y9)
(Yout S Plus Y2 Y10)
))

(setq B '(SY-B (x)
(Z1 R 'a0 x)
(Z2 S Del 'a0 Z1)
(Z3 R 'a1 Z1)
(Z4 S Del 'a1 Z3)
(Z5 S Del 'a2 Z3)
(Z6 R 'a2 Z3)
(Z7 S Del 'a3 Z6)
(Z8 S Plus Z5 Z7)
(Z9 R 2 Z8)
(Z10 S Plus Z4 Z9)
(Zout S Plus Z2 Z10)
))

(setq corrSL '(  |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_del.bm: Delta combinational element, parametrized.
; U7-DONE
```

DEFINITION:
$\mathrm{del}\,(val,\, u)$
$=$ **if** $val = u$ **then** 1

else `0` **endif**

```
; Everything below generated by SUGAR with:    (bmcomb 'del '(val) '(x))
```

DEFINITION:
s-del $(val,\ x)$
= **if** empty (x) **then** E
else a (s-del $(val,\ $ p $(x))$, del $(val,$ l $(x)))$ **endif**

```
;; A2-Begin-S-DEL
```

THEOREM: a2-empty-s-del
empty (s-del $(val,\ x)$) = empty (x)

THEOREM: a2-e-s-del
(s-del $(val,\ x)$ = E) = empty (x)

THEOREM: a2-lp-s-del
len (s-del $(val,\ x)$) = len (x)

THEOREM: a2-lpe-s-del
eqlen (s-del $(val,\ x)$, x)

THEOREM: a2-ic-s-del
s-del $(val,$ i $(c\_x,\ x))$ = i (del $(val,\ c\_x)$, s-del $(val,\ x)$)

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

THEOREM: a2-pc-s-del
p (s-del $(val,\ x)$) = s-del $(val,$ p $(x))$

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

THEOREM: a2-bc-s-del
b (s-del $(val,\ x)$) = s-del $(val,$ b $(x))$

THEOREM: a2-bnc-s-del
bn $(n,$ s-del $(val,\ x))$ = s-del $(val,$ bn $(n,\ x))$

```
;; A2-End-S-DEL

; eof:comb_del.bm
```

```
; comb_plus.bm: Plus combinational element.
; U7-DONE

;   no character function definition since BM already knows about Plus..

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

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

```
;; A2-Begin-S-PLUS
```

THEOREM: a2-empty-s-plus
empty (s-plus $(x,\ y)$) = empty (x)

THEOREM: a2-e-s-plus
(s-plus $(x,\ y)$ = E) = empty (x)

THEOREM: a2-lp-s-plus
len (s-plus $(x,\ y)$) = len (x)

THEOREM: a2-lpe-s-plus
eqlen (s-plus $(x,\ y)$, x)

THEOREM: a2-ic-s-plus
(len (x) = len (y))
→ (s-plus (i $(c\_x,\ x)$, i $(c\_y,\ y)$) = i $(c\_x$ + c_y, s-plus $(x,\ y)$))

THEOREM: a2-lc-s-plus
(¬ empty (x)) → (l (s-plus $(x,\ y)$) = (l (x) + l (y)))

THEOREM: a2-pc-s-plus
p (s-plus $(x,\ y)$) = s-plus (p (x), p (y))

THEOREM: a2-hc-s-plus
((¬ empty (x)) ∧ (len (x) = len (y)))
→ (h (s-plus $(x,\ y)$) = (h (x) + h (y)))

THEOREM: a2-bc-s-plus
(len (x) = len (y)) → (b (s-plus $(x,\ y)$) = s-plus (b (x), b (y)))

THEOREM: a2-bnc-s-plus
(len (*x*) = len (*y*)) → (bn (*n*, s-plus (*x*, *y*)) = s-plus (bn (*n*, *x*), bn (*n*, *y*)))

```
;; A2-End-S-PLUS

; eof:comb_plus.bm
```

DEFINITION:
topor-sy-a (*ln*)
= **if** *ln* = 'y1 **then** 0
elseif *ln* = 'y2 **then** 1
elseif *ln* = 'y3 **then** 0
elseif *ln* = 'y4 **then** 1
elseif *ln* = 'y5 **then** 0
elseif *ln* = 'y6 **then** 1
elseif *ln* = 'y7 **then** 0
elseif *ln* = 'y8 **then** 1
elseif *ln* = 'y9 **then** 2
elseif *ln* = 'y10 **then** 3
elseif *ln* = 'yout **then** 4
else 0 **endif**

```
;Parameter found: 'A0 in: (Y2 S DEL 'A0 Y1)
;Parameter found: 'A1 in: (Y4 S DEL 'A1 Y3)
;Parameter found: 'A2 in: (Y6 S DEL 'A2 Y5)
;Parameter found: 'A3 in: (Y8 S DEL 'A3 Y7)
```

DEFINITION:
sy-a (*ln*, *x*)
= **if** *ln* = 'y1
then if empty (*x*) **then** E
else i ('a0, p (*x*)) **endif**
elseif *ln* = 'y2 **then** s-del ('a0, sy-a ('y1, *x*))
elseif *ln* = 'y3
then if empty (*x*) **then** E
else i ('a1, sy-a ('y1, p (*x*))) **endif**
elseif *ln* = 'y4 **then** s-del ('a1, sy-a ('y3, *x*))
elseif *ln* = 'y5
then if empty (*x*) **then** E
else i ('a2, sy-a ('y3, p (*x*))) **endif**
elseif *ln* = 'y6 **then** s-del ('a2, sy-a ('y5, *x*))
elseif *ln* = 'y7
then if empty (*x*) **then** E

else i ('a3, sy-a ('y5, p (x))) **endif**
elseif ln = 'y8 **then** s-del ('a3, sy-a ('y7, x))
elseif ln = 'y9 **then** s-plus (sy-a ('y6, x), sy-a ('y8, x))
elseif ln = 'y10 **then** s-plus (sy-a ('y4, x), sy-a ('y9, x))
elseif ln = 'yout **then** s-plus (sy-a ('y2, x), sy-a ('y10, x))
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
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
```

DEFINITION:
topor-sy-b (ln)
= **if** ln = 'z1 **then** 0
elseif ln = 'z2 **then** 1
elseif ln = 'z3 **then** 0
elseif ln = 'z4 **then** 1
elseif ln = 'z5 **then** 1
elseif ln = 'z6 **then** 0
elseif ln = 'z7 **then** 1
elseif ln = 'z8 **then** 2
elseif ln = 'z9 **then** 0
elseif ln = 'z10 **then** 2
elseif ln = 'zout **then** 3
else 0 **endif**

```
;Parameter found: 'A0 in: (Z2 S DEL 'A0 Z1)
;Parameter found: 'A1 in: (Z4 S DEL 'A1 Z3)
;Parameter found: 'A2 in: (Z5 S DEL 'A2 Z3)
;Parameter found: 'A3 in: (Z7 S DEL 'A3 Z6)
```

DEFINITION:
sy-b (ln, x)
= **if** ln = 'z1
then if empty (x) **then** E
else i ('a0, p (x)) **endif**
elseif ln = 'z2 **then** s-del ('a0, sy-b ('z1, x))
elseif ln = 'z3
then if empty (x) **then** E
else i ('a1, sy-b ('z1, p (x))) **endif**
elseif ln = 'z4 **then** s-del ('a1, sy-b ('z3, x))
elseif ln = 'z5 **then** s-del ('a2, sy-b ('z3, x))
elseif ln = 'z6
then if empty (x) **then** E
else i ('a2, sy-b ('z3, p (x))) **endif**
elseif ln = 'z7 **then** s-del ('a3, sy-b ('z6, x))
elseif ln = 'z8 **then** s-plus (sy-b ('z5, x), sy-b ('z7, x))
elseif ln = 'z9
then if empty (x) **then** E
else i (2, sy-b ('z8, p (x))) **endif**
elseif ln = 'z10 **then** s-plus (sy-b ('z4, x), sy-b ('z9, x))
elseif ln = 'zout **then** s-plus (sy-b ('z2, x), sy-b ('z10, x))
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!):
```

```
; The following works, but is rather inelegant, and painful to
; scale.
; Note that STR-add1-len-P2 saves an elimination,and hence cuts the
; proof complexity, and run time in 3.
```

THEOREM: eq-a-b
stringp$(x) \rightarrow$ (sy-b ('zout, x) = sy-a ('yout, x))

```
; The next attempt uses SMARTS instead of strength:
; FIRST we build the equalities which are in some sense
; syntactically guaranteed to be true, since they relate to the
; "unchanged" part of the circuit:
```

THEOREM: eq-z1-y1
stringp$(x) \rightarrow$ (sy-b ('z1, x) = sy-a ('y1, x))

THEOREM: eq-z2-y2
stringp$(x) \rightarrow$ (sy-b ('z2, x) = sy-a ('y2, x))

THEOREM: eq-z3-y3
stringp$(x) \rightarrow$ (sy-b ('z3, x) = sy-a ('y3, x))

THEOREM: eq-z4-y4
stringp$(x) \rightarrow$ (sy-b ('z4, x) = sy-a ('y4, x))

```
; SECOND we prove the only tricky fact about the circuit:
```

THEOREM: eq-z9-y9
stringp$(x) \rightarrow$ (sy-b ('z9, x) = sy-a ('y9, x))

```
; THIRD we complete the list of trivial equalities..
```

THEOREM: eq-z10-y10
stringp$(x) \rightarrow$ (sy-b ('z10, x) = sy-a ('y10, x))

```
; yielding the desired correctness statement!
```

THEOREM: eq-zout-yout
stringp$(x) \rightarrow$ (sy-b ('zout, x) = sy-a ('yout, x))

```
; eof: corrSL.bm
;))
```

Index