```
#|

 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.

```
; corr_CIXA00.bm
;   . definition of circuits:
; w/ stringadd: OK
; w/ stringins:
;   . proof of equivalence:
; w/ stringadd: OK! get: LP theorems, IC theorems, and then Rewrite!
; w/ stringins:
; NOTE: one of the original experiments, NOT sugar generated.

;;; DEFINITION OF CIRCUITS:
```

```
; Register: NOT NEEDED in expanded version
;(defn R (u x)
;  (if (empty x) (e) (I u (p x))))

; combinationals (defs & thms) needed for circuits:
; comb_del.bm: Delta combinational element, parametrized.
; U7-DONE
```

DEFINITION:
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
$(\neg\ \text{empty}\,(x)) \rightarrow (\text{h}\,(\text{s-del}\,(val,\ x)) = \text{del}\,(val,\ \text{h}\,(x)))$

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

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

```
;; 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:
$\text{s-plus}\,(x,\ y)$
$= \quad$ **if** $\text{empty}\,(x)$ **then** E
else $\text{a}\,(\text{s-plus}\,(\text{p}\,(x),\ \text{p}\,(y)),\ \text{l}\,(x) + \text{l}\,(y))$ **endif**

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

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

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

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

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

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

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

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

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

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

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

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

; eof:comb_plus.bm


; 1st circuit:
```

DEFINITION:
y1 (x)
= **if** empty (x) **then** E
else i ('a1, p (x)) **endif**

DEFINITION: $\text{y2}\,(x) = \text{s-del}\,(\texttt{'a1},\ \text{y1}\,(x))$

DEFINITION: $\text{y3}\,(x) = \text{y1}\,(x)$

DEFINITION:
y4 (x)
= **if** empty (x) **then** E
else i ('a2, y3 (p (x))) **endif**

DEFINITION: $\text{y5}\,(x) = \text{s-del}\,(\texttt{'a2},\ \text{y4}\,(x))$

DEFINITION: $\text{w1}\,(x) = \text{s-plus}\,(\text{y2}\,(x),\ \text{y5}\,(x))$

```
; 2nd circuit:
```

DEFINITION: $\text{z1}\,(x) = \text{s-del}\,(\texttt{'a1},\ x)$

DEFINITION: $\mathrm{z2}\,(x) = x$

DEFINITION:
$\mathrm{z3}\,(x)$
$=$ **if** empty (x) **then** E
else i (`'a1`, z2 (p (x))) **endif**

DEFINITION: $\mathrm{z4}\,(x) =$ s-del (`'a2`, z3 (x))

DEFINITION: $\mathrm{z5}\,(x) =$ s-plus (z1 (x), z4 (x))

DEFINITION:
$\mathrm{w2}\,(x)$
$=$ **if** empty (x) **then** E
else i (2, z5 (p (x))) **endif**

```
;;; PROOF:

;;; no 2nd order for sysd lines, because we are not in CSX..

;;; Correctness (equivalence):
```

THEOREM: lw1w2
stringp $(x) \rightarrow (\mathrm{w1}\,(x) = \mathrm{w2}\,(x))$

```
; eof: corr_CIXA00.bm
```

Index