```
#|

 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_CSXA00.bm
;   . definition of circuits [assumes stringadd.bm] :
;       - without hints: FAIL
; - with TOPO: OK, but proofs blows up (understood) or fails because
;  of lack of sysd- expansion...
; - with TOPOR: OK
;   . proof:
;  - with 2nd order combinationals:
; - NO expand hint: FAIL
;  - expand hint: OK
; - adding the 2nd order SYSD thms changes nothing to the equivalence
```

```
; proof.
; NOTE: the above comments date back to the hand-generation time, when we
;       were still trying to FIND a way to feed things to BM.  They are kept
;       here for historical purposes only...

; 1st circuit:
#|
(setq sysd-A '(sy-corrA (x)
(Y1 R 'a1 x)
(Y2 S Del 'a1 Y1)
(Y3 S id Y1) ;useless, but what the hell, let's try it anyway..
(Y4 R 'a2 Y3)
(Y5 S Del 'a2 Y4)
(W1 S Plus Y2 Y5)
))

; 2nd circuit:

(setq sysd-B '(sy-corrB (x)
(Z1 S Del 'a1 x)
(Z3 R 'a1 x)
(Z4 S Del 'a2 Z3)
(Z5 S Plus Z1 Z4)
(W2 R 2 Z5)
))
|#
;(setq corr_CSXA00 '(
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_id.bm: Id (identity) combinational element
; U7-DONE
;      Yes, it's probably quite useless, but it's a good test of the
;      proof generation algorithms, and as the Romans found out the
;      hard way, it sometimes help to have 0 around!

; SUGAR NOTE: an extra lemma about it is given at the end of this file...
```

Definition: $\mathrm{id}\,(u) = u$

```
; Everything (until ;; A2-End...) below generated by: (bmcomb 'id '() '(x))
```

Definition:
s-id (x)
$=$ **if** empty (x) **then** E

else a (s-id (p (x)), id (l (x))) **endif**

`;; A2-Begin-S-ID`

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

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

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

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

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

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

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

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

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

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

`;; A2-End-S-ID`

`;; Hand-generated (potentially useful) lemmas`

THEOREM: sid-is-sfix
s-id (x) = sfix (x)

EVENT: Disable sid-is-sfix.

```
; eof:comb_id.bm

; 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
($\neg$ empty (x)) $\rightarrow$ (h (s-del (val, x)) = del (val, 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:
$\text{s-plus}(x,\ y)$
$= \quad \textbf{if}\ \text{empty}(x)\ \ \textbf{then}\ \text{E}$
$\quad\ \ \textbf{else}\ \text{a}(\text{s-plus}(\text{p}(x),\ \text{p}(y)),\ \text{l}(x) + \text{l}(y))\ \textbf{endif}$

```
;; 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
$\mathrm{p}(\text{s-plus}(x,\ y)) = \text{s-plus}(\mathrm{p}(x),\ \mathrm{p}(y))$

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

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

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
```

DEFINITION:
topor-sy-corra (*ln*)
= **if** *ln* = 'y1 **then** 0
elseif *ln* = 'y2 **then** 1
elseif *ln* = 'y3 **then** 1
elseif *ln* = 'y4 **then** 0
elseif *ln* = 'y5 **then** 1
elseif *ln* = 'w1 **then** 2
else 0 **endif**

```
;Parameter found: 'A1 in: (Y2 S DEL 'A1 Y1)
;Parameter found: 'A2 in: (Y5 S DEL 'A2 Y4)
```

DEFINITION:
sy-corra (*ln*, *x*)
= **if** *ln* = 'y1
then if empty (*x*) **then** E
else i ('a1, p (*x*)) **endif**
elseif *ln* = 'y2 **then** s-del ('a1, sy-corra ('y1, *x*))
elseif *ln* = 'y3 **then** s-id (sy-corra ('y1, *x*))
elseif *ln* = 'y4
then if empty (*x*) **then** E
else i ('a2, sy-corra ('y3, p (*x*))) **endif**
elseif *ln* = 'y5 **then** s-del ('a2, sy-corra ('y4, *x*))
elseif *ln* = 'w1 **then** s-plus (sy-corra ('y2, *x*), sy-corra ('y5, *x*))
else sfix (*x*) **endif**

```
;; A2-Begin-SY-CORRA
```

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

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

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

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

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

```
;; A2-End-SY-CORRA
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
```

DEFINITION:
topor-sy-corrb (ln)
= **if** ln = 'z1 **then** 1
elseif ln = 'z3 **then** 0
elseif ln = 'z4 **then** 1
elseif ln = 'z5 **then** 2
elseif ln = 'w2 **then** 0
else 0 **endif**

```
;Parameter found: 'A1 in: (Z1 S DEL 'A1 X)
;Parameter found: 'A2 in: (Z4 S DEL 'A2 Z3)
```

DEFINITION:
sy-corrb (ln, x)
= **if** ln = 'z1 **then** s-del ('a1, x)
elseif ln = 'z3
then if empty (x) **then** E
else i ('a1, p (x)) **endif**
elseif ln = 'z4 **then** s-del ('a2, sy-corrb ('z3, x))
elseif ln = 'z5 **then** s-plus (sy-corrb ('z1, x), sy-corrb ('z4, x))
elseif ln = 'w2
then if empty (x) **then** E
else i (2, sy-corrb ('z5, p (x))) **endif**
else sfix (x) **endif**

```
;; A2-Begin-SY-CORRB
```

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

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

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

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

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

```
;; A2-End-SY-CORRB

;;; CORRECTNESS (equality):

; Lw1w2: still requires the big expansion hint...
```

THEOREM: lw1w2
stringp (x) $\rightarrow$ (sy-corra ('w1, x) = sy-corrb ('w2, x))

```
; eof: corr_CSXA00.bm
;))
```

Index