```
#|

 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.

```
; prod0_CSXAdd.bm
;   . definition of circuits [assumes stringadd.bm] :
;     - if the circuit has only one line: OK without any hint
;  We MAY want to put the TOP0 hint, just for the induction, although
; for one line it probably collapes to the same induction (LEN X).
;     - if the circuit has more than one line:
;    - without hints: FAIL
;          - TOP00 is not definable, because of loops in the dependency graph!
;    - with TOPOR: OK
; 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...


;;; DEFINITION OF CIRCUITS:
#|
(setq sysd-prod '(sy-prod (x)
(Yprod S Times x Yprod2)
(Yprod2 R 0 Yprod)
))

(setq sysd-const0 '(sy-const0 (x)
(Yconst0 R 0 Yconst0)
))

(setq prod0_CSXA00 '(
|#

; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_times.bm: Times combinational element.
; U7-DONE

; no character function def since BM already knows about Times..

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

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

```
;; A2-Begin-S-TIMES
```

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

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

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

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

THEOREM: a2-ic-s-times
(len (x) = len (y))
$\rightarrow$ (s-times (i $(c_x,\ x)$, i $(c_y,\ y)$) = i $(c_x * c_y$, s-times $(x,\ y)))$

THEOREM: a2-lc-s-times
($\neg$ empty $(x)) \rightarrow$ (l (s-times $(x,\ y)$) = (l (x) $*$ l $(y)))$

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

THEOREM: a2-hc-s-times
(($\neg$ empty $(x)) \wedge$ (len (x) = len $(y)))$
$\rightarrow$ (h (s-times $(x,\ y)$) = (h (x) $*$ h $(y)))$

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

THEOREM: a2-bnc-s-times
(len (x) = len $(y)) \rightarrow$ (bn $(n$, s-times $(x,\ y)$) = s-times (bn $(n,\ x)$, bn $(n,\ y)))$

```
;; A2-End-S-TIMES

; eof:comb_times.bm
```

DEFINITION:
topor-sy-prod (ln)
= **if** ln = `'yprod` **then** `1`
elseif ln = `'yprod2` **then** `0`
else `0` **endif**

DEFINITION:
sy-prod $(ln,\ x)$
= **if** ln = `'yprod` **then** s-times $(x$, sy-prod (`'yprod2`, $x))$
elseif ln = `'yprod2`
then if empty (x) **then** E
else i (`0`, sy-prod (`'yprod`, p $(x))$) **endif**
else sfix (x) **endif**

```
;; A2-Begin-SY-PROD
```

THEOREM: a2-empty-sy-prod
empty (sy-prod $(ln,\ x)$) = empty (x)

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

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

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

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

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

; No TOPO def for 1 line sysds because it is not needed and confuses BM
```

DEFINITION:
sy-const0 (ln, x)
= **if** ln = `'yconst0`
then if empty (x) **then** E
else i (0, sy-const0 (`'yconst0`, p (x))) **endif**
else sfix (x) **endif**

```
;; A2-Begin-SY-CONST0
```

THEOREM: a2-empty-sy-const0
empty (sy-const0 (ln, x)) = empty (x)

THEOREM: a2-e-sy-const0
(sy-const0 (ln, x) = E) = empty (x)

THEOREM: a2-lp-sy-const0
len (sy-const0 (ln, x)) = len (x)

THEOREM: a2-lpe-sy-const0
eqlen (sy-const0 (ln, x), x)

THEOREM: a2-pc-sy-const0
p (sy-const0 (ln, x)) = sy-const0 (ln, p (x))

```
;; A2-End-SY-CONST0

;;; PROOF OF EQUIVALENCE:

; The key fact about SY-Yconst is that it equals the constant 0 function:
```

THEOREM: sy-const0-is-const
sy-const0 (`'yconst0`, x) = s-const (`0`, x)

```
; The key fact (bug) about prod0 is that both lines also equal const-0 sfun
; CRUCIAL NOTE: we only want the 1st equality, but in order for the induction
; proof to succeed, we need the stronger (global) statement.
```

THEOREM: prod0-is-const
(sy-prod (`'yprod`, x) = s-const (`0`, x))
$\wedge$ (sy-prod (`'yprod2`, x) = s-const (`0`, x))

```
; now the equality is trivial:
```

THEOREM: e_prodconst0
sy-prod (`'yprod`, x) = sy-const0 (`'yconst0`, x)

```
; eof: prod0_CSXA00.bm
;))
```

Index

a, 2
a2-bc-s-times, 3
a2-bnc-s-times, 3
a2-e-s-times, 2
a2-e-sy-const0, 4
a2-e-sy-prod, 4
a2-empty-s-times, 2
a2-empty-sy-const0, 4
a2-empty-sy-prod, 3
a2-hc-s-times, 3
a2-ic-s-times, 3
a2-lc-s-times, 3
a2-lp-s-times, 2
a2-lp-sy-const0, 4
a2-lp-sy-prod, 4
a2-lpe-s-times, 2
a2-lpe-sy-const0, 4
a2-lpe-sy-prod, 4
a2-pc-s-times, 3
a2-pc-sy-const0, 4
a2-pc-sy-prod, 4

b, 3
bn, 3

e, 2–4
e_prodconst0, 5
empty, 2–4
eqlen, 2, 4

h, 3

i, 3, 4

l, 2, 3
len, 2–4

p, 2–4
prod0-is-const, 5

s-const, 5
s-times, 2, 3
sfix, 3, 4
sy-const0, 4, 5
sy-const0-is-const, 5
sy-prod, 3–5

topor-sy-prod, 3