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 TOPO hint, just for the induction, although
;   for one line it probably collapses to the same induction (LEN X).
;   - if the circuit has more than one line:
;     - without hints: FAIL
;     - TOPO0 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 EM. They are kept
;;; 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)), 1(x) * 1(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
\[(\text{len} (x) = \text{len} (y)) \rightarrow (\text{s-times} (i (c_x, x), i (c_y, y)) = i (c_x * c_y, \text{s-times} (x, y)))\]

Theorem: a2-lc-s-times
\[(\neg \text{empty} (x)) \rightarrow (1(\text{s-times} (x, y)) = (1(x) * 1(y)))\]

Theorem: a2-pc-s-times
\[\text{p} (\text{s-times} (x, y)) = \text{s-times} (\text{p} (x), \text{p} (y))\]

Theorem: a2-hc-s-times
\[((\neg \text{empty} (x)) \land (\text{len} (x) = \text{len} (y)))\] \[\rightarrow (\text{h} (\text{s-times} (x, y)) = (\text{h} (x) * \text{h} (y)))\]

Theorem: a2-bc-s-times
\[(\text{len} (x) = \text{len} (y)) \rightarrow (\text{b} (\text{s-times} (x, y)) = \text{s-times} (\text{b} (x), \text{b} (y)))\]

Theorem: a2-bnc-s-times
\[(\text{len} (x) = \text{len} (y)) \rightarrow (\text{bn} (n, \text{s-times} (x, y)) = \text{s-times} (\text{bn} (n, x), \text{bn} (n, y)))\]

;; 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
\( (\text{sy-prod}(ln, x) = e) = \text{empty}(x) \)

Theorem: a2-lp-sy-prod
\( \text{len}(\text{sy-prod}(ln, x)) = \text{len}(x) \)

Theorem: a2-lpe-sy-prod
\( \text{eqlen}(\text{sy-prod}(ln, x), x) \)

Theorem: a2-pc-sy-prod
\( p(\text{sy-prod}(ln, x)) = \text{sy-prod}(ln, p(x)) \)

;; 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:
\[
sy-const0(ln, x) = \begin{cases} 
  \text{if } ln = 'yconst0 
  & \text{then if empty}(x) \text{ then } e 
  & \text{else } i(0, \text{sy-const0}('yconst0, p(x))) \text{ endif} 
  & \text{else } \text{sfix}(x) \text{ endif} 
\end{cases}
\]

;; A2-Begin-SY-CONST0

Theorem: a2-empty-sy-const0
\( \text{empty}(\text{sy-const0}(ln, x)) = \text{empty}(x) \)

Theorem: a2-e-sy-const0
\( (\text{sy-const0}(ln, x) = e) = \text{empty}(x) \)

Theorem: a2-lp-sy-const0
\( \text{len}(\text{sy-const0}(ln, x)) = \text{len}(x) \)

Theorem: a2-lpe-sy-const0
\( \text{eqlen}(\text{sy-const0}(ln, x), x) \)

Theorem: a2-pc-sy-const0
\( p(\text{sy-const0}(ln, x)) = \text{sy-const0}(ln, p(x)) \)

;; 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))
∧ (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-bmc-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-ic-s-times, 3
a2-ic-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