#|

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, 1 (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
(¬ empty (x)) → (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
\(\neg \text{empty}(x) \rightarrow (h(s\text{-del}(val, x)) = \text{del}(val, h(x)))\)

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

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

;; 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:
\(s\text{-plus}(x, y) = \begin{cases} 
\text{E} & \text{if empty}(x) \\
\text{a}(s\text{-plus}(p(x), p(y)), l(x) + l(y)) & \text{else} 
\end{cases}\)  

;; A2-Begin-S-PLUS

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

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

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

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

THEOREM: a2-ic-s-plus
\((\text{len}(x) = \text{len}(y)) \rightarrow (s\text{-plus}(i(c_x, x), i(c_y, y)) = i(c_x + c_y, s\text{-plus}(x, y)))\)
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

; 1st circuit:

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

Definition: y2 (x) = s-del ('a1, y1 (x))

Definition: y3 (x) = y1 (x)

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

Definition: y5 (x) = s-del ('a2, y4 (x))

Definition: w1 (x) = s-plus (y2 (x), y5 (x))

; 2nd circuit:

Definition: z1 (x) = s-del ('a1, x)
Definition: \( z_2(x) = x \)

Definition:
\[
z_3(x) = \begin{cases} 
\text{if} & \text{empty}(x) \quad \text{then} \ E \\
\text{else} & i(\text{'a}_1, z_2(p(x))) \quad \text{endif}
\end{cases}
\]

Definition: \( z_4(x) = s\text{-del}(\text{'a}_2, z_3(x)) \)

Definition: \( z_5(x) = s\text{-plus}(z_1(x), z_4(x)) \)

Definition:
\[
w_2(x) = \begin{cases} 
\text{if} & \text{empty}(x) \quad \text{then} \ E \\
\text{else} & i(2, z_5(p(x))) \quad \text{endif}
\end{cases}
\]

;; PROOF:

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

;;; Correctness (equivalence):

Theorem: lw1w2
\[
\text{stringp}(x) \rightarrow (w_1(x) = w_2(x))
\]

; eof: corr_CIXA00.bm
Index

a, 2, 3
a2-bc-s-del, 3
a2-bc-s-plus, 4
a2-bnc-s-del, 3
a2-bnc-s-plus, 4
a2-e-s-del, 2
a2-e-s-plus, 3
a2-empty-s-del, 2
a2-empty-s-plus, 3
a2-hc-s-del, 3
a2-hc-s-plus, 4
a2-ic-s-del, 2
a2-ic-s-plus, 3
a2-lc-s-del, 2
a2-lc-s-plus, 4
a2-lp-s-del, 2
a2-lp-s-plus, 3
a2-lpe-s-del, 2
a2-lpe-s-plus, 3
a2-pc-s-del, 2
a2-pc-s-plus, 4

s-plus, 3–5
stringp, 5
w1, 4, 5
w2, 5
y1, 4
y2, 4
y3, 4
y4, 4
y5, 4

z1, 4, 5
z2, 5
z3, 5
z4, 5
z5, 5

b, 3, 4
bn, 3, 4
del, 2, 3
e, 2–5
empty, 2–5
eqlen, 2, 3

h, 3, 4

i, 2–5

l, 2–4
len, 2–4
lw1w2, 5

p, 2–5

s-del, 2–5