#|

|#

EVENT: Start with the library "mlp" using the compiled version.

```
; pplinc3.bm is our 1st PIPELINE proof.

;;; (Sugared) Circuits:
#|
(setq A '(SY-A (x)
(Y1 S Inc x)
(Y2 S Inc Y1)
(Y3 S Inc Y2)
; and the cork:
(Yc2 R 2 Y3)
(Yc1 R 1 Yc2)
```

```
(Yout R 0 Yc1)
))

(setq B '(SY-B (x)
(Z1 S Inc x)
(Z2 R 0 Z1)
(Z3 S Inc Z2)
(Z4 R 0 Z3)
(Z5 S Inc Z4)
(Zout R 0 Z5)
))

(setq pplinc3 '( |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_inc.bm: INCrement combinational element
; U7-DONE
```

DEFINITION: $\text{inc}(u) = (1 + u)$

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

DEFINITION:
s-inc (x)
= **if** empty (x) **then** E
else a (s-inc (p (x)), inc (l (x))) **endif**

```
;; A2-Begin-S-INC
```

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

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

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

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

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

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

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

THEOREM: a2-hc-s-inc
$(\neg\ \text{empty}\,(x)) \rightarrow (\text{h}\,(\text{s-inc}\,(x)) = \text{inc}\,(\text{h}\,(x)))$

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

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

```
;; A2-End-S-INC
```

```
; eof:comb_inc.bm
```

DEFINITION:
topor-sy-a (*ln*)
= **if** *ln* = `'y1` **then** `1`
elseif *ln* = `'y2` **then** `2`
elseif *ln* = `'y3` **then** `3`
elseif *ln* = `'yc2` **then** `0`
elseif *ln* = `'yc1` **then** `0`
elseif *ln* = `'yout` **then** `0`
else `0` **endif**

DEFINITION:
sy-a (*ln*, *x*)
= **if** *ln* = `'y1` **then** s-inc (*x*)
elseif *ln* = `'y2` **then** s-inc (sy-a (`'y1`, *x*))
elseif *ln* = `'y3` **then** s-inc (sy-a (`'y2`, *x*))
elseif *ln* = `'yc2`
then if empty (*x*) **then** E
else i (`2`, sy-a (`'y3`, p (*x*))) **endif**
elseif *ln* = `'yc1`
then if empty (*x*) **then** E
else i (`1`, sy-a (`'yc2`, p (*x*))) **endif**
elseif *ln* = `'yout`
then if empty (*x*) **then** E
else i (`0`, sy-a (`'yc1`, p (*x*))) **endif**
else sfix (*x*) **endif**

```
;; A2-Begin-SY-A
```

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

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

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

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

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

```
;; A2-End-SY-A
```

DEFINITION:
topor-sy-b (ln)
= **if** ln = 'z1 **then** 1
elseif ln = 'z2 **then** 0
elseif ln = 'z3 **then** 1
elseif ln = 'z4 **then** 0
elseif ln = 'z5 **then** 1
elseif ln = 'zout **then** 0
else 0 **endif**

DEFINITION:
sy-b (ln, x)
= **if** ln = 'z1 **then** s-inc (x)
elseif ln = 'z2
then if empty (x) **then** E
else i (0, sy-b ('z1, p (x))) **endif**
elseif ln = 'z3 **then** s-inc (sy-b ('z2, x))
elseif ln = 'z4
then if empty (x) **then** E
else i (0, sy-b ('z3, p (x))) **endif**
elseif ln = 'z5 **then** s-inc (sy-b ('z4, x))
elseif ln = 'zout
then if empty (x) **then** E
else i (0, sy-b ('z5, p (x))) **endif**
else sfix (x) **endif**

```
;; A2-Begin-SY-B
```

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

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

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

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

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

```
;; A2-End-SY-B

;;; CORRECTNESS PROOF (hand generated, dreamer!):

; EQ-A-B: just like in CorrSL, since there are no loops, straight unfolding
; should work, as long as Brain's normalization is strong enough...
; Note about the hint:
;   - STR-add1-len-P2 (and hence LEN) came from CorrSL.
;   - at first we disabled S-INC thinking that it was irrelevent, but it
;   IS necessary, since it affects the value of the cork.
```

THEOREM: eq-a-b
sy-b (`'zout`, x) = sy-a (`'yout`, x)

```
; eof: pplinc3.bm
;))
```

Index