

#|

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.

```
; serial.bm: a register, writable in parallel, and readable
; serially. This is Paillet example 3
;
; IMPORTANT NOTE: originally, we proved this multi-circuit WITHOUT
; EQ-LEN hyps, because we got lucky and all the inputs were
; Registered, so it didn't matter. We now go to the more general
; version for uniformity, even though it will be enormously more
; expensive, and will force EQ-LEN hyps in the correctness thms.
; It might be worth remembering though that for circuits where all
; inputs are immediately Registered, we can do away with the
; EQ-LEN hyp.
```

```
;
; OTHER IMPORTANT NOTE: all the comments below concerning various
; ways of phrasing the hypotheses were written when EMPTY was still
; ENABLED.
```

```
;;; CIRCUIT in SUGARED form: (after flattening out, yuck...)
```

```
#|
(setq sysd '(sy-SERIAL (xC x1 x2 x3)
(YC0 S const 0 xC)
(YM3 S mux xC x3 YC0)
(Y3 R 'a3 YM3)
(YM2 S mux xC x2 Y3)
(Y2 R 'a2 YM2)
(YM1 S mux xC x1 Y2)
(Y1 R 'a1 YM1)
))
```

```
(setq serial '(|#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_mux.bm: Mux combinational element, i.e. "if".
; U7-DONE
```

DEFINITION:

```
mux(u1, u2, u3)
= if u1 then u2
  else u3 endif
```

```
; everything below generated by: (bmcomb 'mux '() '(x1 x2 x3))
; with the EXCEPTIONS/HAND-MODIFICATIONS given below.
```

DEFINITION:

```
s-mux(x1, x2, x3)
= if empty(x1) then E
  else a(s-mux(p(x1), p(x2), p(x3)), mux(l(x1), l(x2), l(x3))) endif
```

```
; SMUX-is-SIF can make things much simpler on occasions:
```

THEOREM: smux-is-sif

```
s-mux(x1, x2, x3) = s-if(x1, x2, x3)
```

EVENT: Disable smux-is-sif.

```

; We take advantage of SMUX-is-SIF for all inductive proofs. To do so we
; HAND-MODIFY the code generated by Sugar to replace all the hints by
;   - A2-EMPTY, A2-PC replace hint with: ((enable smux-is-sif))
;   - A2-LP, A2-IC, A2-HC, A2-BC: ((enable smux-is-sif) (disable len))
;   - A2-BNC: ((enable smux-is-sif) (disable bn len))

```

```
;; A2-Begin-S-MUX
```

THEOREM: a2-empty-s-mux
 $\text{empty}(\text{s-mux}(x1, x2, x3)) = \text{empty}(x1)$

THEOREM: a2-e-s-mux
 $(\text{s-mux}(x1, x2, x3) = E) = \text{empty}(x1)$

THEOREM: a2-lp-s-mux
 $\text{len}(\text{s-mux}(x1, x2, x3)) = \text{len}(x1)$

THEOREM: a2-lpe-s-mux
 $\text{eqlen}(\text{s-mux}(x1, x2, x3), x1)$

THEOREM: a2-ic-s-mux
 $((\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{s-mux}(i(c.x1, x1), i(c.x2, x2), i(c.x3, x3)))$
 $= i(\text{mux}(c.x1, c.x2, c.x3), \text{s-mux}(x1, x2, x3)))$

THEOREM: a2-lc-s-mux
 $(\neg \text{empty}(x1)) \rightarrow (l(\text{s-mux}(x1, x2, x3)) = \text{mux}(l(x1), l(x2), l(x3)))$

THEOREM: a2-pc-s-mux
 $p(\text{s-mux}(x1, x2, x3)) = \text{s-mux}(p(x1), p(x2), p(x3))$

THEOREM: a2-hc-s-mux
 $((\neg \text{empty}(x1)) \wedge ((\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3))))$
 $\rightarrow (\text{h}(\text{s-mux}(x1, x2, x3)) = \text{mux}(\text{h}(x1), \text{h}(x2), \text{h}(x3)))$

```
;old: ((DISABLE MUX S-MUX) (ENABLE H LEN) (INDUCT (S-MUX X1 X2 X3)))
```

THEOREM: a2-bc-s-mux
 $((\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{b}(\text{s-mux}(x1, x2, x3)) = \text{s-mux}(\text{b}(x1), \text{b}(x2), \text{b}(x3)))$

```
;old: ((DISABLE MUX) (ENABLE B LEN) (INDUCT (S-MUX X1 X2 X3)))
```

THEOREM: a2-bnc-s-mux
 $((\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{bn}(n, \text{s-mux}(x1, x2, x3)) = \text{s-mux}(\text{bn}(n, x1), \text{bn}(n, x2), \text{bn}(n, x3)))$

```

;old: ((DISABLE MUX S-MUX))

;; A2-End-S-MUX

; eof:comb_mux.bm

```

DEFINITION:
topor-sy-serial(*ln*)

```

= if ln = 'yc0 then 1
  elseif ln = 'ym3 then 2
  elseif ln = 'y3 then 0
  elseif ln = 'ym2 then 1
  elseif ln = 'y2 then 0
  elseif ln = 'ym1 then 1
  elseif ln = 'y1 then 0
  else 0 endif

;Parameter found: 0 in: (YC0 S CONST 0 XC)

```

DEFINITION:
sy-serial(*ln*, *xc*, *x1*, *x2*, *x3*)

```

= if ln = 'yc0 then s-const(0, xc)
  elseif ln = 'ym3 then s-mux(xc, x3, sy-serial('yc0, xc, x1, x2, x3))
  elseif ln = 'y3
  then if empty(xc) then E
        else i('a3, sy-serial('ym3, p(xc), p(x1), p(x2), p(x3))) endif
  elseif ln = 'ym2 then s-mux(xc, x2, sy-serial('y3, xc, x1, x2, x3))
  elseif ln = 'y2
  then if empty(xc) then E
        else i('a2, sy-serial('ym2, p(xc), p(x1), p(x2), p(x3))) endif
  elseif ln = 'ym1 then s-mux(xc, x1, sy-serial('y2, xc, x1, x2, x3))
  elseif ln = 'y1
  then if empty(xc) then E
        else i('a1, sy-serial('ym1, p(xc), p(x1), p(x2), p(x3))) endif
  else sfix(xc) endif

;; A2-Begin-SY-SERIAL

```

THEOREM: a2-empty-sy-serial

$((\text{len}(xc) = \text{len}(x1)) \wedge (\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{empty}(\text{sy-serial}(ln, xc, x1, x2, x3)) = \text{empty}(xc))$

THEOREM: a2-e-sy-serial

$((\text{len}(xc) = \text{len}(x1)) \wedge (\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow ((\text{sy-serial}(ln, xc, x1, x2, x3) = \text{E}) = \text{empty}(xc))$

THEOREM: a2-lp-sy-serial

$((\text{len}(xc) = \text{len}(x1)) \wedge (\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{len}(\text{sy-serial}(ln, xc, x1, x2, x3)) = \text{len}(xc))$

THEOREM: a2-lpe-sy-serial

$((\text{len}(xc) = \text{len}(x1)) \wedge (\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow \text{eqlen}(\text{sy-serial}(ln, xc, x1, x2, x3), xc)$

THEOREM: a2-pc-sy-serial

$((\text{len}(xc) = \text{len}(x1)) \wedge (\text{len}(x1) = \text{len}(x2)) \wedge (\text{len}(x2) = \text{len}(x3)))$
 $\rightarrow (\text{p}(\text{sy-serial}(ln, xc, x1, x2, x3))$
 $\quad = \text{sy-serial}(ln, \text{p}(xc), \text{p}(x1), \text{p}(x2), \text{p}(x3)))$

;; A2-End-SY-SERIAL

;;; Circuit CORRECTNESS /Paillet:

; SPECIFICATION:

; Here we interpret Paillet as talking about last-chars implicitly

DEFINITION:

$\text{serial-spec-l}(xc, x1, x2, x3)$
 $= \text{if } \text{l}(\text{p}(xc)) \text{ then } \text{l}(\text{p}(x1))$
 $\quad \text{elseif } \text{l}(\text{p}(\text{p}(xc))) \text{ then } \text{l}(\text{p}(\text{p}(x2)))$
 $\quad \text{elseif } \text{l}(\text{p}(\text{p}(\text{p}(xc)))) \text{ then } \text{l}(\text{p}(\text{p}(\text{p}(x3))))$
 $\quad \text{else } 0 \text{ endif}$

; Here we intepret Paillet as really talking about streams (and
; correct for the missing initial values):

DEFINITION:

$\text{serial-spec}(xc, x1, x2, x3)$
 $= \text{i}('a1,$
 $\quad \text{s-if}(\text{p}(xc),$

```

    p(x1),
    i('a2,
      s-if(p(p(xc)),
          p(p(x2)),
          i('a3,
            s-if(p(p(p(xc))), p(p(p(x3))), s-const(0, p(p(p(xc))))))))))

```

; CORRECTNESS:

; note: we don't need EQ-LEN hyp here, although it was tried and
 ; didn't hurt.

THEOREM: serial-correct-1

```

((¬ empty(xc))
  ∧ (¬ empty(p(xc)))
  ∧ (¬ empty(p(p(xc))))
  ∧ (¬ empty(p(p(p(xc))))))
→ (l(sy-serial('y1, xc, x1, x2, x3)) = serial-spec-1(xc, x1, x2, x3))

```

; Note: we shouldn't need the EQ-LEN hyp here, since it's just an unfolding..

THEOREM: serial-correct

```

(¬ empty(p(p(p(xc))))
→ (sy-serial('y1, xc, x1, x2, x3) = serial-spec(xc, x1, x2, x3))

```

; NOTE that above we have a choice of how we phrase the hypothesis:

```

; 1: (and (not (empty xC)) (not (empty (p xC)))
;       (not (empty (p (p xC)))) (not (empty (p (p (p xC)))))
;   is highly redundant but says everything needed and so solves
;   in 1 step.
; 2: (not (empty (p (p (p xC))))) concise, -> many cases (but
;   LESS time!)
; 3: (not (empty (Pn 3 xC))) concise, -> same # cases as 2, but
;   more time.

```

; Rewrite lemmas such as:

```

;(prove-lemma not-empty-Pn (rewrite)
;(equal (not (empty (Pn n x)))
;       (if (zerop n)
;           (not (empty x))
;           (and (not (empty x))
;                (not (empty (Pn (sub1 n) (P x)))))))
;)

```

; although true, have no effect on the hypothesis expansion,

```

; unfortunately..

; Another property listed as "correctness" in Paillet is:
; Note that here we have translated P .. into L P .., because
; if we try to understand this last Paillet property as speaking of
; streams, then the Hypothesis: P3 C = 1 and P2 C = P C = 0
; doesn't make any sense!!!
; in fact he acknowledges that "these computations are supposed to
; be made in a temporal interval corresponding to one cycle, but
; this interval is not indicated in the calculus to avoid too much
; notation". Formally of course, we don't have that luxury...

; Again, we don't need the EQ-LEN hyp, although when we tested it,
; it threw BM into a loop, until we DISABLED LEN; this trick might
; carry over!!

```

THEOREM: serial-correct-specialcase-1

```

((l(p(xc)) = f)
 ^ (l(p(p(xc))) = f)
 ^ (l(p(p(p(xc)))) = t)
 ^ (¬ empty(xc))
 ^ (¬ empty(p(xc)))
 ^ (¬ empty(p(p(xc))))
 ^ (¬ empty(p(p(p(xc))))))
→ ((l(sy-serial('y1, xc, x1, x2, x3)) = l(p(p(p(x3))))))
   ^ (l(p(sy-serial('y1, xc, x1, x2, x3))) = l(p(p(p(x2))))))
   ^ (l(p(p(sy-serial('y1, xc, x1, x2, x3)))) = l(p(p(p(x1))))))

```

```

; Note above that using the (redundant) hypothesis:
; (not (empty xC)) (not (empty (p xC))) (not (empty (p (p xC))))
; (not (empty (p (p (p xC))))))
; makes the proof instantaneous, since otherwise BM goes through
; eliminations to realize the "equal" hyps imply it.

```

```

; eof: serial.bm
;))

```

Index

a, 2
a2-bc-s-mux, 3
a2-bnc-s-mux, 4
a2-e-s-mux, 3
a2-e-sy-serial, 5
a2-empty-s-mux, 3
a2-empty-sy-serial, 5
a2-hc-s-mux, 3
a2-ic-s-mux, 3
a2-lc-s-mux, 3
a2-lp-s-mux, 3
a2-lp-sy-serial, 5
a2-lpe-s-mux, 3
a2-lpe-sy-serial, 5
a2-pc-s-mux, 3
a2-pc-sy-serial, 5

b, 3
bn, 4

e, 2–5
empty, 2–7
eqLen, 3, 5

h, 3

i, 3, 4, 6

l, 2, 3, 5–7
len, 3–5

mux, 2, 3

p, 2–7

s-const, 4, 6
s-if, 2, 6
s-mux, 2–4
serial-correct, 6
serial-correct-l, 6
serial-correct-specialcase-l, 7
serial-spec, 5, 6
serial-spec-l, 5, 6
sfix, 4
smux-is-sif, 2
sy-serial, 4–7
topor-sy-serial, 4