

#|

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.

```
; ppltcpum.bm is a variation of ppltcpu which treats ProgMemory like SRC.  
;           All it requires is a an additional trivial lemma for the trivial loop.  
;  
; We use the hands-off PPL proving policies developped there, but  
; check both theorems with Explicit and Implicit cork.  
  
;;; (Sugared) Circuits:  
#|  
(setq sy-A '(SY-A (x)  
(Ypc R 0 Ypcn)  
(Ypcn S inc Ypc)
```

```

(Ypr R 'Ipr Ypr)
(Yin S Ummx Ypc Ypr)
(Yi S Ui Yin)
(Ye S Ue Yi)
(Yout R 0 Ye)
))

(setq sy-B '(SY-B (x)
(Ypc R 0 Ypcn)
(Ypcn S inc Ypc)
(Ypr R 'Ipr Ypr)
(Yin S Ummx Ypc Ypr)
(Yin2 R 0 Yin)
(Yi S Ui Yin2)
(Yi2 R 0 Yi)
(Ye S Ue Yi2)
(Yout R 0 Ye)
))

(setq ppltcpum '(|#
; 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:
 $s\text{-inc}(x)$
 $= \text{if } \text{empty}(x) \text{ then } E$
 $\quad \text{else } a(s\text{-inc}(p(x)), \text{inc}(l(x))) \text{ endif}$
;; A2-Begin-S-INC

THEOREM: a2-empty-s-inc
 $\text{empty}(s\text{-inc}(x)) = \text{empty}(x)$

THEOREM: a2-e-s-inc
 $(s\text{-inc}(x) = E) = \text{empty}(x)$

THEOREM: a2-lp-s-inc
 $\text{len}(s\text{-inc}(x)) = \text{len}(x)$

THEOREM: a2-lpe-s-inc
 $\text{eqlen}(\text{s-inc}(x), x)$

THEOREM: a2-ic-s-inc
 $\text{s-inc}(\text{i}(c_x, x)) = \text{i}(\text{inc}(c_x), \text{s-inc}(x))$

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

; comb_ummx.bm: Ummx combinational element (= fun2)
 ; U7-DONE

; arbitrary Char-Fun of arity 2:

EVENT: Introduce the function symbol *ummx* of 2 arguments.

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

DEFINITION:
 $\text{s-ummx}(x, y)$
 $= \text{if empty}(x) \text{ then E}$
 $\text{else a}(\text{s-ummx}(\text{p}(x), \text{p}(y)), \text{ummx}(\text{l}(x), \text{l}(y))) \text{endif}$

;; A2-Begin-S-UMMX

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

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

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

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

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

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

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

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

THEOREM: a2-bc-s-ummx
 $(\text{len}(x) = \text{len}(y)) \rightarrow (\text{b}(\text{s-ummx}(x, y)) = \text{s-ummx}(\text{b}(x), \text{b}(y)))$

THEOREM: a2-bnc-s-ummx
 $(\text{len}(x) = \text{len}(y)) \rightarrow (\text{bn}(n, \text{s-ummx}(x, y)) = \text{s-ummx}(\text{bn}(n, x), \text{bn}(n, y)))$

; ; A2-End-S-UMMX

```

; eof:comb_ummx.bm

; comb_ui.bm: Ui combinational element (= fun1)
; U7-DONE

; arbitrary Char-Fun of arity 1:

```

EVENT: Introduce the function symbol *ui* of one argument.

```

; Everything below generated by: (bmcomb 'Ui '() '(x))

```

DEFINITION:
 $\text{s-ui}(x)$
 $= \text{if } \text{empty}(x) \text{ then E}$
 $\text{else a}(\text{s-ui}(\text{p}(x)), \text{ui}(\text{l}(x))) \text{ endif}$

; ; A2-Begin-S-UI

THEOREM: a2-empty-s-ui
 $\text{empty}(\text{s-ui}(x)) = \text{empty}(x)$

THEOREM: a2-e-s-ui
 $(\text{s-ui}(x) = E) = \text{empty}(x)$

THEOREM: a2-lp-s-ui
 $\text{len}(\text{s-ui}(x)) = \text{len}(x)$

THEOREM: a2-lpe-s-ui
 $\text{eqlen}(\text{s-ui}(x), x)$

THEOREM: a2-ic-s-ui
 $\text{s-ui}(\text{i}(c_x, x)) = \text{i}(\text{ui}(c_x), \text{s-ui}(x))$

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

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

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

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

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

; ; A2-End-S-UI

; eof:comb_ue.bm

; comb_ue.bm: Ue combinational element (= fun1)
; U7-DONE

; arbitrary Char-Fun of arity 1:

EVENT: Introduce the function symbol ue of one argument.

; Everything below generated by: (bmcomb 'Ue '() '(x))

DEFINITION:

$s\text{-ue}(x)$
= **if** $\text{empty}(x)$ **then** E
else $a(s\text{-ue}(p(x)), ue(l(x)))$ **endif**
;; A2-Begin-S-UE

THEOREM: a2-empty-s-ue
 $\text{empty}(s\text{-ue}(x)) = \text{empty}(x)$

THEOREM: a2-e-s-ue
 $(s\text{-ue}(x) = E) = \text{empty}(x)$

THEOREM: a2-lp-s-ue
 $\text{len}(s\text{-ue}(x)) = \text{len}(x)$

THEOREM: a2-lpe-s-ue
 $\text{eqlen}(s\text{-ue}(x), x)$

THEOREM: a2-ic-s-ue
 $s\text{-ue}(i(c_x, x)) = i(ue(c_x), s\text{-ue}(x))$

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

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

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

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

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

;; A2-End-S-UE

; eof:comb_ue.bm

DEFINITION:

```
topor-sy-a (ln)
= if ln = 'ypc then 0
  elseif ln = 'pcn then 1
  elseif ln = 'pr then 0
  elseif ln = 'in then 1
  elseif ln = 'i then 2
  elseif ln = 'e then 3
  elseif ln = 'out then 0
  else 0 endif
```

DEFINITION:

```
sy-a (ln, x)
= if ln = 'pc
  then if empty (x) then E
    else i(0, sy-a ('pcn, p(x))) endif
  elseif ln = 'pcn then s-inc(sy-a ('pc, x))
  elseif ln = 'pr
  then if empty (x) then E
    else i('ipr, sy-a ('pr, p(x))) endif
  elseif ln = 'in then s-ummx(sy-a ('pc, x), sy-a ('pr, x))
  elseif ln = 'i then s-ui(sy-a ('in, x))
  elseif ln = 'e then s-ue(sy-a ('i, x))
  elseif ln = 'out
  then if empty (x) then E
    else i(0, sy-a ('e, p(x))) endif
  else sf(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)) = 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 = 'ypc then 0
  elseif ln = 'ypcn then 1
  elseif ln = 'ypr then 0
  elseif ln = 'yin then 1
  elseif ln = 'yin2 then 0
  elseif ln = 'yi then 1
  elseif ln = 'yi2 then 0
  elseif ln = 'ye then 1
  elseif ln = 'yout then 0
  else 0 endif
```

DEFINITION:

```
sy-b (ln, x)
=  if ln = 'ypc
  then if empty (x) then E
    else i (0, sy-b ('ypcn, p (x))) endif
  elseif ln = 'ypcn then s-inc (sy-b ('ypc, x))
  elseif ln = 'ypr
  then if empty (x) then E
    else i ('ipr, sy-b ('ypr, p (x))) endif
  elseif ln = 'yin then s-ummx (sy-b ('ypc, x), sy-b ('ypr, x))
  elseif ln = 'yin2
  then if empty (x) then E
    else i (0, sy-b ('yin, p (x))) endif
  elseif ln = 'yi then s-ui (sy-b ('yin2, x))
  elseif ln = 'yi2
  then if empty (x) then E
    else i (0, sy-b ('yi, p (x))) endif
  elseif ln = 'ye then s-ue (sy-b ('yi2, x))
  elseif ln = 'yout
  then if empty (x) then E
    else i (0, sy-b ('ye, 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

; SOME ANIMATION:
; (setq x5T (A (A (A (A (A (e) T) T) T) T) T))
;*(sy-A 'Ye x5t)
; (A (A (A (A (A (E) '(UE (UI (UMMX 0 IPR)))))))
;           '(UE (UI (UMMX 1 IPR))))
;           '(UE (UI (UMMX 2 IPR))))
;           '(UE (UI (UMMX 3 IPR))))
;           '(UE (UI (UMMX 4 IPR))))
;*(sy-B 'Ye x5t)
; (A (A (A (A (A (E) '(UE 0)))
;           '(UE (UI 0)))
;           '(UE (UI (UMMX 0 IPR))))
;           '(UE (UI (UMMX 1 IPR))))
;           '(UE (UI (UMMX 2 IPR))))
;*
; This takes care of the PC loop:

THEOREM: eq-pc
sy-b ('ypc, x) = sy-a ('ypc, x)

; This takes care of the PR loop:

THEOREM: eq-pr
sy-b ('ypr, x) = sy-a ('ypr, x)

; EQ-A-B-exp: explicit cork

THEOREM: eq-a-b-exp
sy-b ('ye, x)
= if empty (x) then E

```

```
else i(ue(0),
      if empty(p(x)) then E
      else i(ue(ui(0)), sy-a('ye, p(p(x)))) endif) endif
; w/ implicit cork:
```

THEOREM: eq-a-b-imp
 $(\neg \text{empty}(p(x))) \rightarrow (\text{b}(\text{b}(\text{sy-b}('ye, x))) = \text{sy-a}('ye, p(p(x))))$

```
; eof: ppltcpum.bm
;)
```

Index

- a, 2–4, 6
- a2-bc-s-inc, 3
- a2-bc-s-ue, 6
- a2-bc-s-ui, 5
- a2-bc-s-ummx, 4
- a2-bnc-s-inc, 3
- a2-bnc-s-ue, 6
- a2-bnc-s-ui, 5
- a2-bnc-s-ummx, 4
- a2-e-s-inc, 2
- a2-e-s-ue, 6
- a2-e-s-ui, 5
- a2-e-s-ummx, 4
- a2-e-sy-a, 7
- a2-e-sy-b, 9
- a2-empty-s-inc, 2
- a2-empty-s-ue, 6
- a2-empty-s-ui, 5
- a2-empty-s-ummx, 3
- a2-empty-sy-a, 7
- a2-empty-sy-b, 8
- a2-hc-s-inc, 3
- a2-hc-s-ue, 6
- a2-hc-s-ui, 5
- a2-hc-s-ummx, 4
- a2-ic-s-inc, 3
- a2-ic-s-ue, 6
- a2-ic-s-ui, 5
- a2-ic-s-ummx, 4
- a2-lc-s-inc, 3
- a2-lc-s-ue, 6
- a2-lc-s-ui, 5
- a2-lc-s-ummx, 4
- a2-lp-s-inc, 2
- a2-lp-s-ue, 6
- a2-lp-s-ui, 5
- a2-lp-s-ummx, 4
- a2-lp-sy-a, 7
- a2-lp-sy-b, 9
- a2-lpe-s-inc, 3
- a2-lpe-s-ue, 6
- a2-lpe-s-ui, 5
- a2-lpe-s-ummx, 4
- a2-lpe-sy-a, 7
- a2-lpe-sy-b, 9
- a2-pc-s-inc, 3
- a2-pc-s-ue, 6
- a2-pc-s-ui, 5
- a2-pc-s-ummx, 4
- a2-pc-sy-a, 7
- a2-pc-sy-b, 9
- b, 3–6, 10
- bn, 3–6
- e, 2–10
- empty, 2–10
- eq-a-b-exp, 9
- eq-a-b-imp, 10
- eq-pc, 9
- eq-pr, 9
- eqlen, 3–7, 9
- h, 3–6
- i, 3–8, 10
- inc, 2, 3
- l, 2–6
- len, 2, 4–7, 9
- p, 2–10
- s-inc, 2, 3, 7, 8
- s-ue, 6–8
- s-ui, 4, 5, 7, 8
- s-ummx, 3, 4, 7, 8
- sfix, 7, 8
- sy-a, 7, 9, 10
- sy-b, 8–10
- topor-sy-a, 7
- topor-sy-b, 8

ue, 5, 6, 10

ui, 4, 5, 10

ummx, 3, 4