```
#|

 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.

```
;;;                          bibo_exp.bm
;;;
;;; Experiments with type conversions, standard commutative squares, to
;;; get a feel for these issues.  Some of these theorems may turn out
;;; useful in the future, in which case they should end up in Brain,
;;; probably in th_types.
;;;
;;; Clearly, no sugar involved.
;;;
;;; Name convention: "isa" means "is almost", i.e. up to type conversion.
```

```
; (setq bibo_exp '(

; comb_bor.bm: Binary Or combinational element
; U7-DONE
```

DEFINITION:
bor $(u,\ v)$
$=$ **if** $(u = 0) \wedge (v = 0)$ **then** 0
else 1 **endif**

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

DEFINITION:
s-bor $(x,\ y)$
$=$ **if** empty (x) **then** E
else a (s-bor (p (x), p (y)), bor (l (x), l (y))) **endif**

```
;; A2-Begin-S-BOR
```

THEOREM: a2-empty-s-bor
empty (s-bor $(x,\ y)$) = empty (x)

THEOREM: a2-e-s-bor
(s-bor $(x,\ y)$ = E) = empty (x)

THEOREM: a2-lp-s-bor
len (s-bor $(x,\ y)$) = len (x)

THEOREM: a2-lpe-s-bor
eqlen (s-bor $(x,\ y)$, x)

THEOREM: a2-ic-s-bor
(len (x) = len (y))
$\rightarrow$ (s-bor (i $(c\_x,\ x)$, i $(c\_y,\ y)$) = i (bor $(c\_x,\ c\_y)$, s-bor $(x,\ y)$))

THEOREM: a2-lc-s-bor
$(\neg$ empty $(x)) \rightarrow$ (l (s-bor $(x,\ y)$) = bor (l (x), l (y)))

THEOREM: a2-pc-s-bor
p (s-bor $(x,\ y)$) = s-bor (p (x), p (y))

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

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

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

```
;; A2-End-S-BOR

; eof:comb_bor.bm


; BOR-ISA-OR is trivially proved (straight rewrites) and useless because
; it refers to non-recursive head: bibo ; and in fact it does not trigger
; in the next theorem.
```

THEOREM: bor-isa-or
$\text{bibo}\,(\text{bor}\,(\text{bobi}\,(u),\, \text{bobi}\,(v))) = (u \vee v)$

```
; SBOR-ISA-SOR requires induction, and difficulty depends on hypothesis:
;   - when no eqlen hyp is given, requires 16 cases, and non-trivial rewriting
;     for the non-eqlen cases. Time: 41s
;   - with: (equal (len x) (len y)), reduces to 5 cases and 7s, same induction.
;   - with: (eqlen x y), gets better induction scheme, 4 cases and 8s.
; of course, we keep the theorem in its most general form.
```

THEOREM: sbor-isa-sor
$\text{s-bibo}\,(\text{s-bor}\,(\text{s-bobi}\,(x),\, \text{s-bobi}\,(y))) = \text{s-or}\,(x,\, y)$

```
; some trivial type-checking experiments:
```

THEOREM: bor-0
$\text{bitp}\,(v) \rightarrow (\text{bor}\,(0,\, v) = v)$

```
; eof: bibo_exp.bm
;))
```

Index