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.
**Definition:**
\[ \text{bor}(u, v) = \begin{cases} 0 & \text{if } (u = 0) \land (v = 0) \text{ then } 0 \\ 1 & \text{else} \end{cases} \]

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

**Definition:**
\[ \text{s-bor}(x, y) = \begin{cases} \text{E} & \text{if empty}(x) \text{ then } \text{E} \\ \text{s-bor}(\text{p}(x), \text{p}(y)), \text{bor}(\text{l}(x), \text{l}(y))) & \text{else} \end{cases} \]

;; A2-Begin-S-BOR

**Theorem:** a2-empty-s-bor
\[ \text{empty}(\text{s-bor}(x, y)) = \text{empty}(x) \]

**Theorem:** a2-e-s-bor
\[ (\text{s-bor}(x, y) = \text{E}) = \text{empty}(x) \]

**Theorem:** a2-lp-s-bor
\[ \text{len}(\text{s-bor}(x, y)) = \text{len}(x) \]

**Theorem:** a2-lpe-s-bor
\[ \text{eqlen}(\text{s-bor}(x, y), x) \]

**Theorem:** a2-ic-s-bor
\[ (\text{len}(x) = \text{len}(y)) \rightarrow (\text{s-bor}(\text{i}(c_x, x), \text{i}(c_y, y)) = \text{i}(\text{bor}(c_x, c_y), \text{s-bor}(x, y))) \]

**Theorem:** a2-lc-s-bor
\[ (\neg \text{empty}(x)) \rightarrow (\text{l}(\text{s-bor}(x, y)) = \text{bor}(\text{l}(x), \text{l}(y))) \]

**Theorem:** a2-pc-s-bor
\[ \text{p}(\text{s-bor}(x, y)) = \text{s-bor}(\text{p}(x), \text{p}(y)) \]

**Theorem:** a2-hc-s-bor
\[ ((\neg \text{empty}(x)) \land (\text{len}(x) = \text{len}(y))) \rightarrow (\text{h}(\text{s-bor}(x, y)) = \text{bor}(\text{h}(x), \text{h}(y))) \]

2
Theorem: a2-bc-s-bor
(len (x) = len (y)) → (b (s-bor (x, y)) = s-bor (b (x), b (y)))

Theorem: a2-bnc-s-bor
(len (x) = len (y)) → (bn (n, s-bor (x, y)) = s-bor (bn (n, x), 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
bibo (bor (bobi (u), bobi (v))) = (u ∨ 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
s-bibo (s-bor (s-bobi (x), s-bobi (y))) = s-or (x, y)

; some trivial type-checking experiments:

Theorem: bor-0
bitp (v) → (bor (0, v) = v)

; eof: bibo_exp.bm
;;
Index

a, 2
a2-bc-s-bor, 3
a2-bmc-s-bor, 3
a2-e-s-bor, 2
a2-empty-s-bor, 2
a2-hc-s-bor, 2
a2-ic-s-bor, 2
a2-ic-s-bor, 2
a2-lc-s-bor, 2
a2-lp-s-bor, 2
a2-lpe-s-bor, 2
a2-pc-s-bor, 2

b, 3
bibo, 3
bitp, 3
bn, 3
bobi, 3
bor, 2, 3
bor-0, 3
bor-isa-or, 3

e, 2
empty, 2
eqlen, 2

h, 2

i, 2

l, 2
len, 2, 3

p, 2

s-bibo, 3
s-bobi, 3
s-bor, 2, 3
s-or, 3
sbor-isa-sor, 3