

#|

Copyright (C) 1994 by Computational Logic, Inc. All Rights Reserved.

This script is hereby placed in the public domain, and therefore unlimited editing and redistribution is permitted.

NO WARRANTY

Computational Logic, Inc. PROVIDES 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 Computational Logic, Inc. 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.

|#

```
;; Here is a proof of correctness of mergesort. The
;; main events are marked with '!!!'.
```

EVENT: Start with the initial **nqthm** theory.

DEFINITION:

```
length(x)
=  if listp(x) then 1 + length(cdr(x))
    else 0 endif
```

```
;; [in r-loop, try:]
#|
*(cons 3 (cons 4 (cons 7 nil)))
'(3 4 7)
*(length '(3 4 7))
|#
```

```
#|
;;[try without hint]
```

```

(defn merge (l m)
  (if (not (listp l))
      m
      (if (not (listp m))
          l
          (if (lessp (car l) (car m))
              (cons (car l) (merge (cdr l) m))
              (cons (car m) (merge l (cdr m)))))))
  ((lessp (plus (length l) (length m))))))
|#

```

DEFINITION:

```

merge(l, m)
=  if ¬ listp(l) then m
   elseif ¬ listp(m) then l
   elseif car(l) < car(m) then cons(car(l), merge(cdr(l), m))
   else cons(car(m), merge(l, cdr(m))) endif

```

DEFINITION:

```

odds(l)
=  if ¬ listp(l) then nil
   else cons(car(l), odds(cddr(l))) endif

```

```

#|
(defn mergesort (l)
  (if (not (listp l))
      nil
      (if (not (listp (cdr l)))
          l
          (merge (mergesort (odds (cdr l)))
                  (mergesort (odds l))))))
  ((lessp (length l))))
|#

```

```

#|
(defn mergesort (l)
  (if (not (listp l))
      nil
      (if (not (listp (cdr l)))
          l
          (merge (mergesort (odds (cdr l)))
                  (mergesort (odds l))))))
  ((lessp (length l))))
|#

```

```

#|
(prove-lemma mergesort-helper (rewrite)
  (implies (and (listp l)
                 (listp (cdr l)))
            (equal (lessp (sub1 (length (odds l)))
                        (length (cdr l)))
                    t)))

|#

;; still wasn't enough, so we prove:

THEOREM: mergesort-helper
(listp (l)  $\wedge$  listp (cdr (l)))
 $\rightarrow$  (((length (odds (l)) - 1) < length (cdr (l))) = t)
 $\wedge$  (((length (odds (cdr (l))) - 1) < length (cdr (l))) = t))

DEFINITION:
mergesort (l)
= if  $\neg$  listp (l) then nil
  elseif  $\neg$  listp (cdr (l)) then l
  else merge (mergesort (odds (cdr (l))), mergesort (odds (l))) endif

;; [try (mergesort '(3 7 8 2 9 4 7)) in r-loop]

DEFINITION:
sortedp (x)
= if listp (x)
  then if listp (cdr (x))
        then (car (cdr (x))  $\not\prec$  car (x))  $\wedge$  sortedp (cdr (x))
        else t endif
  else t endif

;; !!! FIRST MAIN THEOREM -- note that the subgoal
;; (IMPLIES (AND (SORTEDP B) (SORTEDP U))
;;           (SORTEDP (MERGE U B)))
;; is generated automatically!

THEOREM: sortedp-mergesort
sortedp (mergesort (x))

DEFINITION:
occur (a, x)
= if listp (x)
  then if a = car (x) then 1 + occur (a, cdr (x))
        else occur (a, cdr (x)) endif
  else 0 endif

```

```

#|
;; Want to prove the following, but need lemmas.
;; Use the proof-checker to try to find them.  Suggests
;; OCCUR-MERGE pretty quickly
(prove-lemma occur-mergesort (rewrite)
  (equal (occur a (mergesort x))
    (occur a x)))
|#

THEOREM: occur-merge

$$\text{occur}(a, \text{merge}(x, y)) = (\text{occur}(a, x) + \text{occur}(a, y))$$


;; Now back into VERIFY.... prover goes into an induction in PROVE
;; call, so we abort.  Use
#|
(INSTRUCTIONS INDUCT PROVE PROVE PROMOTE
  (DIVE 1 2)
  X TOP
  (S LEMMAS)
  (DIVE 1 1)
  = NX = TOP
  (DROP 3 4))
|#

;; in proof-checker.

THEOREM: plus-occur-odds

$$(\text{listp}(x) \wedge \text{listp}(\text{cdr}(x))) \rightarrow ((\text{occur}(a, \text{odds}(\text{cdr}(x))) + \text{occur}(a, \text{odds}(x))) = \text{occur}(a, x))$$


;; !!! SECOND MAIN THEOREM; see last event for permutationp version

THEOREM: occur-mergesort

$$\text{occur}(a, \text{mergesort}(x)) = \text{occur}(a, x)$$


;; Events to show facts about permutationp:

DEFINITION:
remove1(a, x)
= if listp(x)
  then if car(x) = a then cdr(x)
    else cons(car(x), remove1(a, cdr(x))) endif
  else x endif

```

DEFINITION:

$\text{badguy}(x, y)$
= **if** $\text{listp}(x)$
 then if $\text{car}(x) \in y$ **then** $\text{badguy}(\text{cdr}(x), \text{remove1}(\text{car}(x), y))$
 else $\text{car}(x)$ **endif**
 else 0 endif

DEFINITION:

$\text{subbagp}(x, y)$
= **if** $\text{listp}(x)$ **then** $(\text{car}(x) \in y)$
 $\wedge \text{subbagp}(\text{cdr}(x), \text{remove1}(\text{car}(x), y))$
 else t endif

THEOREM: member-occur

$(a \in x) = (0 < \text{occur}(a, x))$

THEOREM: occur-remove1

$\text{occur}(a, \text{remove1}(b, x))$
= **if** $a = b$ **then** $\text{occur}(a, x) - 1$
 else $\text{occur}(a, x)$ **endif**

THEOREM: subbagp-wit-lemma

$\text{subbagp}(x, y) = (\text{occur}(\text{badguy}(x, y), y) \not< \text{occur}(\text{badguy}(x, y), x))$

DEFINITION: $\text{permutationp}(x, y) = (\text{subbagp}(x, y) \wedge \text{subbagp}(y, x))$

;; !!! REVISED VERSION OF SECOND MAIN THEOREM

THEOREM: permutationp-mergesort

$\text{permutationp}(\text{mergesort}(x), x)$

Index

badguy, 5

length, 1, 3

member-occur, 5

merge, 2–4

mergesort, 3–5

mergesort-helper, 3

occur, 3–5

occur-merge, 4

occur-mergesort, 4

occur-remove1, 5

odds, 2–4

permutationp, 5

permutationp-mergesort, 5

plus-occur-odds, 4

remove1, 4, 5

sortedp, 3

sortedp-mergesort, 3

subbagp, 5

subbagp-wit-lemma, 5