```
#|

 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.
|#

;; Matt Kaufmann

;; Koenig Tree Lemma events supporting ``An Extension of the
;; Boyer-Moore Theorem Prover to Support First-Order Quantification,''
;; to appear in JAR (1992?).  The DEFN-SK events have been replaced by
;; DCLs and ADD-AXIOMs, as shown.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Koenig's Lemma Events List
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; All this initial stuff is just to get the CONSTRAIN below accepted.
```

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

DEFINITION:
ones (n)
$=$ **if** $n \simeq 0$ **then nil**
else cons $(1, \text{ones}\,(n - 1))$ **endif**

DEFINITION:
all-ones (s)
$=$ **if** listp (s) **then** (car (s) $=$ 1) $\wedge$ all-ones (cdr (s))
else s $=$ **nil** **endif**

DEFINITION:
length (s)
$=$ **if** listp (s) **then** 1 $+$ length (cdr (s))
else 0 **endif**

DEFINITION:
subseq ($s1$, $s2$)
$=$ **if** $s1 = s2$ **then** **t**
elseif $s2 \simeq$ **nil** **then** **f**
else subseq ($s1$, cdr ($s2$)) **endif**

THEOREM: subseq-all-ones
(all-ones ($s1$) $\wedge$ subseq ($s2$, $s1$)) $\rightarrow$ all-ones ($s2$)

DEFINITION:
plistp (s)
$=$ **if** listp (s) **then** plistp (cdr (s))
else s $=$ **nil** **endif**

THEOREM: plistp-all-ones
all-ones (s) $\rightarrow$ plistp (s)

THEOREM: all-ones-ones
all-ones (ones (n))

THEOREM: ones-is-injective
(($i \in \mathbf{N}$) $\wedge$ ($j \in \mathbf{N}$) $\wedge$ ($i \neq j$)) $\rightarrow$ (ones (i) $\neq$ ones (j))

CONSERVATIVE AXIOM: koenig-intro
node-p (**nil**)
$\wedge$ (truep (node-p (s)) $\vee$ falsep (node-p (s)))
$\wedge$ (node-p (s)
$\rightarrow$ (node-p (cons (n, s)) $=$ ((0 $<$ n) $\wedge$ (succard (s) $\not<$ n))))
$\wedge$ ((node-p ($s1$) $\wedge$ subseq (s, $s1$)) $\rightarrow$ node-p (s))
$\wedge$ node-p (s-n (n))
$\wedge$ ((($i \in \mathbf{N}$) $\wedge$ ($j \in \mathbf{N}$) $\wedge$ ($i \neq j$)) $\rightarrow$ (s-n (i) $\neq$ s-n (j)))
$\wedge$ (($\neg$ plistp (s)) $\rightarrow$ ($\neg$ node-p (s)))

Simultaneously, we introduce the new function symbols *node-p*, *succard*, and *s-n*.

```
;; We want to define a function s-height which returns an element of a given height.
;; The next several events culminate in the following lemma:

;; (prove-lemma length-s-height (rewrite)
;;   (equal (length (s-height n)) (fix n)))
```

DEFINITION:
succ-aux $(s,\ n)$
$=$ **if** $n \simeq 0$ **then nil**
else cons (cons $(n,\ s)$, succ-aux $(s,\ n-1)$) **endif**

DEFINITION: successors (s) $=$ succ-aux $(s$, succard $(s))$

DEFINITION:
successors-list (ss)
$=$ **if** listp (ss)
then append (successors (car (ss)), successors-list (cdr (ss)))
else nil endif

DEFINITION:
level (n)
$=$ **if** $n \simeq 0$ **then** list (**nil**)
else successors-list (level $(n-1)$) **endif**

DEFINITION:
init-tree (n)
$=$ **if** $n \simeq 0$ **then** list (**nil**)
else append (level (n), init-tree $(n-1)$) **endif**

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

THEOREM: length-remove1
$(a \in x) \rightarrow$ (length (remove1 $(a,\ x)$) $<$ length (x))

DEFINITION:
first-non-member-index $(i,\ x)$
$=$ **if** s-n $(i) \in x$ **then** first-non-member-index $(1 + i$, remove1 (s-n (i), x))
else i **endif**

DEFINITION:
nthcdr $(n,\ s)$
$=$ **if** $n \simeq 0$ **then** s
else nthcdr $(n - 1,\ \text{cdr}(s))$ **endif**

DEFINITION:
s-height (n)
$=$ nthcdr (length (s-n (first-non-member-index (0, init-tree (n)))) $-\ n$,
s-n (first-non-member-index (0, init-tree (n))))

THEOREM: nthcdr-subseq
(length $(s) \not< n$) $\rightarrow$ subseq (nthcdr $(n,\ s)$, s)

THEOREM: node-p-nthcdr
(node-p (s) $\wedge$ (length $(s) \not< n$)) $\rightarrow$ node-p (nthcdr $(n,\ s)$)

THEOREM: lessp-difference-1
$(x < (x - n)) = \mathbf{f}$

THEOREM: node-p-s-height
node-p (s-height (n))

THEOREM: length-nthcdr
length (nthcdr $(n,\ s)$) $=$ (length $(s) - n$)

THEOREM: first-non-member-index-lessp
first-non-member-index $(i,\ x) \not< i$

THEOREM: s-n-first-non-member-index-not-equal
$(i \in \mathbf{N})$
$\rightarrow$ (s-n (first-non-member-index $(1 + i$, remove1 (s-n (i), x))) $\neq$ s-n (i))

THEOREM: member-remove1
$(a \neq b) \rightarrow ((a \in$ remove1 $(b,\ x)) = (a \in x))$

THEOREM: s-n-first-non-member-index
$(i \in \mathbf{N}) \rightarrow$ (s-n (first-non-member-index $(i,\ x)$) $\notin x$)

THEOREM: member-append
$(a \in$ append $(x,\ y)) = ((a \in x) \vee (a \in y))$

THEOREM: member-cons-succ-aux
(cons $(z,\ v) \in$ succ-aux $(v,\ n)$) $= ((0 < z) \wedge (n \not< z))$

THEOREM: node-p-cons-lemma
($\neg$ node-p (s)) $\rightarrow$ ($\neg$ node-p (cons $(n,\ s)$))

THEOREM: node-p-cons
$\text{node-p}(\text{cons}(n,\ s)) = (\text{node-p}(s) \wedge (0 < n) \wedge (\text{succard}(s) \not< n))$

DEFINITION:
$\text{all-length-n}(ss,\ n)$
$= \ \textbf{if}\ \text{listp}(ss)\ \textbf{then}\ (\text{length}(\text{car}(ss)) = n) \wedge \text{all-length-n}(\text{cdr}(ss),\ n)$
$\quad \textbf{else}\ \textbf{t}\ \textbf{endif}$

THEOREM: all-length-n-append
$\text{all-length-n}(\text{append}(ss1,\ ss2),\ n)$
$= \ (\text{all-length-n}(ss1,\ n) \wedge \text{all-length-n}(ss2,\ n))$

THEOREM: all-length-n-succ-aux
$(\text{length}(s) = n) \rightarrow \text{all-length-n}(\text{succ-aux}(s,\ k),\ 1 + n)$

THEOREM: all-length-n-successors-list
$\text{all-length-n}(ss,\ n) \rightarrow \text{all-length-n}(\text{successors-list}(ss),\ 1 + n)$

THEOREM: length-0
$(\text{length}(s) = 0) = (s \simeq \textbf{nil})$

DEFINITION:
$\text{member-level-induction}(s,\ n)$
$= \ \textbf{if}\ n \simeq 0\ \textbf{then}\ \textbf{t}$
$\quad \textbf{else}\ \text{member-level-induction}(\text{cdr}(s),\ n - 1)\ \textbf{endif}$

THEOREM: succ-aux-listp
$(\neg\ \text{listp}(s)) \rightarrow (s \notin \text{succ-aux}(z,\ n))$

THEOREM: successors-list-listp
$(\neg\ \text{listp}(s)) \rightarrow (s \notin \text{successors-list}(ss))$

THEOREM: member-succ-aux
$(s \in \text{succ-aux}(x,\ n)) \rightarrow (\text{cdr}(s) = x)$

THEOREM: member-successors-list-successors-list-witness
$(s \in \text{successors-list}(ss))$
$= \ ((\text{cdr}(s) \in ss) \wedge (s \in \text{successors}(\text{cdr}(s))))$

THEOREM: member-level
$((n \in \mathbf{N}) \wedge \text{node-p}(s)) \rightarrow ((s \in \text{level}(n)) = (\text{length}(s) = n))$

THEOREM: member-init-tree
$\text{node-p}(s) \rightarrow ((s \in \text{init-tree}(n)) = (n \not< \text{length}(s)))$

THEOREM: length-s-non-member-index
$(i \in \mathbf{N}) \rightarrow (n < \text{length}\,(\text{s-n}\,(\text{first-non-member-index}\,(i,\ \text{init-tree}\,(n)))))$

THEOREM: length-s-height
$\text{length}\,(\text{s-height}\,(n)) = \text{fix}\,(n)$

EVENT: Disable s-height.

```
;; End of s-height excursion.

;; Our goal:

;(prove-lemma konig-tree-lemma nil
;  (and (node-p (k n))
;       (implies (not (lessp j i))
;                (subseq (k i) (k j)))
;       (equal (length (k n)) (fix n))))

#| The original DEFN-SK event here was processed as follows:

>(defn-sk inf (s)
  ;; says that s has arbitrarily high successors
  (forall big-h (exists big-s
    (and (subseq s big-s)
         (node-p big-s)
         (lessp big-h (length big-s))))))


Adding the Skolem axiom:
      (AND (IMPLIES (AND (SUBSEQ S BIG-S)
                         (NODE-P BIG-S)
                         (LESSP (BIG-H S) (LENGTH BIG-S)))
                    (INF S))
           (IMPLIES (NOT (AND (SUBSEQ S (BIG-S BIG-H S))
                              (NODE-P (BIG-S BIG-H S))
                              (LESSP BIG-H
                                     (LENGTH (BIG-S BIG-H S)))))
                    (NOT (INF S)))).


     As this is a DEFN-SK we can conclude that:
      (OR (TRUEP (INF S)) (FALSEP (INF S)))
is a theorem.
```

```
[ 0.2 0.0 0.0 ]
INF

>
|#
```

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

EVENT: Introduce the function symbol *big-h* of one argument.

EVENT: Introduce the function symbol *big-s* of 2 arguments.

AXIOM: inf-intro

$$\begin{array}{l} ((\text{subseq}\,(s,\ \textit{big-s}) \wedge \text{node-p}\,(\textit{big-s}) \wedge (\text{big-h}\,(s) < \text{length}\,(\textit{big-s}))) \\ \ \rightarrow \quad \text{inf}\,(s)) \\ \wedge \quad ((\neg\,(\text{subseq}\,(s,\ \text{big-s}\,(\textit{big-h},\ s)) \\ \qquad\qquad \wedge \quad \text{node-p}\,(\text{big-s}\,(\textit{big-h},\ s)) \\ \qquad\qquad \wedge \quad (\textit{big-h} < \text{length}\,(\text{big-s}\,(\textit{big-h},\ s))))) \\ \qquad \rightarrow \quad (\neg\,\text{inf}\,(s))) \end{array}$$

AXIOM: inf-boolean

$$\text{truep}\,(\text{inf}\,(s)) \vee \text{falsep}\,(\text{inf}\,(s))$$

```
;; The following three events were generated mechanically.  They are
;; useful especially for applying the Skolem axioms for INF inside the
;; proof-checker, via the macro command SK*.
```

EVENT: Disable inf-intro.

THEOREM: inf-suff

$$\begin{array}{l} (\text{subseq}\,(s,\ \textit{big-s}) \wedge \text{node-p}\,(\textit{big-s}) \wedge (\text{big-h}\,(s) < \text{length}\,(\textit{big-s}))) \\ \rightarrow \quad \text{inf}\,(s) \end{array}$$

THEOREM: inf-necc

$$\begin{array}{l} (\neg\,(\text{subseq}\,(s,\ \text{big-s}\,(\textit{big-h},\ s)) \\ \qquad \wedge \quad \text{node-p}\,(\text{big-s}\,(\textit{big-h},\ s)) \\ \qquad \wedge \quad (\textit{big-h} < \text{length}\,(\text{big-s}\,(\textit{big-h},\ s))))) \\ \rightarrow \quad (\neg\,\text{inf}\,(s)) \end{array}$$

DEFINITION:
next $(s,\ max)$
$=$ **if** $max \simeq 0$ **then** cons $(0,\ s)$
elseif inf (cons $(max,\ s)$) **then** cons $(max,\ s)$
else next $(s,\ max - 1)$ **endif**

```
;; We want to show that NEXT gives us a successor with infinitely many
;; successors.

; INF-IMPLIES-INF-NEXT:
;(implies (and (node-p s)
;              (inf s))
;         (inf (next s (succard s))))

;; Note that if some successor of s has infinitely many successors, so
;; does (NEXT S (SUCCARD S)).  This is the lemma
;; INF-CONS-IMPLIES-INF-NEXT below.  But first note:
```

THEOREM: inf-implies-node-p
$\text{inf}(s) \rightarrow \text{node-p}(s)$

THEOREM: not-inf-zerop
$(i \simeq 0) \rightarrow (\neg\ \text{inf}(\text{cons}(i,\ s)))$

THEOREM: inf-cons-implies-inf-next
$(\text{node-p}(s) \land \text{inf}(\text{cons}(i,\ s)) \land (n \not< i)) \rightarrow \text{inf}(\text{next}(s,\ n))$

```
;; Our goal now is to apply this lemma by proving that
;; (inf (cons i s)) for some i <= (succard s).
```

DEFINITION:
all-big-h $(s,\ n)$
$=$ **if** $n \simeq 0$ **then** $1 + \text{length}(s)$
else big-h (cons $(n,\ s)$) $+$ all-big-h $(s,\ n - 1)$ **endif**

THEOREM: all-big-h-length
$\text{length}(s) < \text{all-big-h}(s,\ n)$

THEOREM: all-big-h-lessp
$((0 < i) \land (n \not< i))$
$\rightarrow \quad ((\text{big-h}(\text{cons}(i,\ s)) < \text{all-big-h}(s,\ n)) = \mathbf{t})$

```
;; Here's a function which tells us which way s first branches on its
;; way to extending to s1.
```

DEFINITION:
first-branch (s, $s1$)
$=$ **if** $s = \text{cdr}(s1)$ **then** car ($s1$)
elseif $s1 \simeq$ **nil** **then** 0
else first-branch (s, cdr ($s1$)) **endif**

THEOREM: subseq-cons-first-branch
$(\text{subseq}(s, x) \wedge (s \neq x)) \rightarrow \text{subseq}(\text{cons}(\text{first-branch}(s, x), s), x)$

THEOREM: length-non-equal
$(\text{length}(x) < \text{length}(y)) \rightarrow ((x = y) = \mathbf{f})$

THEOREM: first-branch-ok-for-succard
$(\text{subseq}(s, \textit{big-s}) \wedge \text{node-p}(\textit{big-s}) \wedge (s \neq \textit{big-s}))$
$\rightarrow \quad ((\text{first-branch}(s, \textit{big-s}) \in \mathbf{N})$
$\wedge \quad (0 < \text{first-branch}(s, \textit{big-s}))$
$\wedge \quad (\text{succard}(s) \not< \text{first-branch}(s, \textit{big-s})))$

THEOREM: all-big-h-lessp-linear
$((0 < i) \wedge (\text{succard}(s) \not< i))$
$\rightarrow \quad (\text{big-h}(\text{cons}(i, s)) < \text{all-big-h}(s, \text{succard}(s)))$

EVENT: Disable all-big-h-lessp.

THEOREM: inf-implies-inf-next
$(\text{node-p}(s) \wedge \text{inf}(s)) \rightarrow \text{inf}(\text{next}(s, \text{succard}(s)))$

DEFINITION:
k (n)
$=$ **if** $n \simeq 0$ **then nil**
else next (k ($n - 1$), succard (k ($n - 1$))) **endif**

THEOREM: subseq-nil
subseq (**nil**, x) $=$ plistp (x)

THEOREM: node-p-implies-plistp
$\text{node-p}(s) \rightarrow \text{plistp}(s)$

THEOREM: inf-nil
inf (**nil**)

EVENT: Disable node-p-implies-plistp.

THEOREM: konig-tree-lemma-1
inf (k (n))

THEOREM: length-next
$\text{inf}(x) \rightarrow (\text{length}(\text{next}(s,\ n)) = (1 + \text{length}(s)))$

THEOREM: konig-tree-lemma-2
$\text{length}(\text{k}(n)) = \text{fix}(n)$

THEOREM: subseq-next
$\text{subseq}(s1,\ s2) \rightarrow \text{subseq}(s1,\ \text{next}(s2,\ n))$

THEOREM: konig-tree-lemma-3
$(j \not< i) \rightarrow \text{subseq}(\text{k}(i),\ \text{k}(j))$

THEOREM: konig-tree-lemma
$$\begin{aligned} & \text{node-p}(\text{k}(n)) \\ \wedge\quad & ((j \not< i) \rightarrow \text{subseq}(\text{k}(i),\ \text{k}(j))) \\ \wedge\quad & (\text{length}(\text{k}(n)) = \text{fix}(n)) \end{aligned}$$

```
;; or, if one prefers:
```

THEOREM: konig-tree-lemma-again
$$\begin{aligned} & (n \in \mathbf{N}) \\ \rightarrow\quad & (\text{node-p}(\text{k}(n)) \\ & \quad \wedge\quad ((j \not< i) \rightarrow \text{subseq}(\text{k}(i),\ \text{k}(j))) \\ & \quad \wedge\quad (\text{length}(\text{k}(n)) = n)) \end{aligned}$$

Index