This module verifies the correctness of the algorith used to implement *SeekPrefix* in the go-immutable-radix project. (https://*github.com*/hashicorp/go-immutable-radix)

SeekPrefix should move the iterator to the first value that has a certain prefix. All subsequent values should have that prefix, but no other ordering guarantees are made.

──────────────── MODULE *RadixSeekPrefix* ────────────────

EXTENDS *FiniteSets*, *Integers*, *Sequences*, *TLC*
INSTANCE *RadixTrees*

Set of characters to use for the alphabet of generated strings.
CONSTANT *Alphabet*

Length of input strings generated
CONSTANT *MinLength*, *MaxLength*
ASSUME
   $\land \{MinLength,\ MaxLength\} \subseteq Nat$
   $\land MinLength \leq MaxLength$
   $\land MinLength > 0$

Number of unique elements to construct the radix tree with. This is a set of numbers so you can test with inputs of multiple sizes.
CONSTANT *ElementCounts*
ASSUME *ElementCounts* $\subseteq Nat$

Inputs is the set of input strings valid for the tree.
$Inputs \triangleq$ UNION $\{[1 .. n \rightarrow Alphabet] : n \in MinLength .. MaxLength\}$

InputSets is the full set of possible inputs we can send to the radix tree.
$InputSets \triangleq \{T \in$ SUBSET $Inputs : Cardinality(T) \in ElementCounts\}$

──

TRUE iff *seq* is prefixed with prefix.
$HasPrefix(seq,\ prefix) \triangleq$
   $\land Len(seq) \geq Len(prefix)$
   $\land \forall\, i \in 1 .. Len(prefix) : seq[i] = prefix[i]$

Remove prefix from *seq*.
$TrimPrefix(seq,\ prefix) \triangleq [i \in 1 .. (Len(seq) - Len(prefix)) \mapsto seq[i + Len(prefix)]]$

──

SeekPrefix in pure TLA+ for verification purposes.
$Expected(root,\ search) \triangleq \{value \in Range(root) : HasPrefix(value,\ search)\}$

  --algorithm *seek_prefix*
variables
 $stack = \langle\rangle,$
 $input \in InputSets,$

1

$prefix \in Inputs,$
$root \; = RadixTree(input),$
$node = \{\},$
$search = \{\},$
$result \; = \{\} \, ;$

This entire algorith is almost 1:1 translated where possible from the
actual implementation in $iter.go$. That's the point: we're trying to verify
our algorithm is correct for all inputs.

begin
    I could've just set these variables in the initializer above but
    to better closely match the algorithm, I reset them here.
$Begin:$
  $stack := \langle \rangle \, ;$
  $node := root \, ;$
  $search := prefix \, ;$

$Seek:$
  while TRUE **do**
    if $Len(search) = 0$ **then**
      goto $Result \, ;$
    end if ;

  $CheckEdge:$
    if $\neg(search[1] \in$ DOMAIN $node.Edges)$ **then**
      goto $NoMatch \, ;$
    end if ;

  $GetEdge:$
    $node := node.Edges[search[1]] \, ;$

  $CheckPrefix:$
    if $HasPrefix(search, node.Prefix)$ **then**
      $search := TrimPrefix(search, node.Prefix) \, ;$
    elsif $HasPrefix(node.Prefix, search)$ **then**
      goto $Result \, ;$
    else
      goto $NoMatch \, ;$
    end if ;
  end while ;

$NoMatch:$
  $result := \{\} \, ;$
  goto $CheckResult \, ;$

$Result$:
  $result := Range(node)$;

$CheckResult$:
  assert $result = Expected(root, prefix)$;
end algorithm ;

BEGIN TRANSLATION ($chksum(pcal) =$ "$e56848a2$" $\wedge\ chksum(tla) =$ "$5b2d1a3e$")
VARIABLES $stack$, $input$, $prefix$, $root$, $node$, $search$, $result$, pc

$vars \triangleq \langle stack, input, prefix, root, node, search, result, pc \rangle$

$Init \triangleq$ Global variables
           $\wedge\ stack = \langle\rangle$
           $\wedge\ input \in InputSets$
           $\wedge\ prefix \in Inputs$
           $\wedge\ root\ = RadixTree(input)$
           $\wedge\ node = \{\}$
           $\wedge\ search = \{\}$
           $\wedge\ result\ = \{\}$
           $\wedge\ pc =$ "Begin"

$Begin \triangleq\ \wedge\ pc =$ "Begin"
           $\wedge\ stack' = \langle\rangle$
           $\wedge\ node' = root$
           $\wedge\ search' = prefix$
           $\wedge\ pc' =$ "Seek"
           $\wedge$ UNCHANGED $\langle input, prefix, root, result \rangle$

$Seek \triangleq\ \wedge\ pc =$ "Seek"
         $\wedge$ IF $Len(search)\ = 0$
              THEN $\wedge\ pc' =$ "Result"
              ELSE $\wedge\ pc' =$ "CheckEdge"
         $\wedge$ UNCHANGED $\langle stack, input, prefix, root, node, search, result \rangle$

$CheckEdge \triangleq\ \wedge\ pc =$ "CheckEdge"
             $\wedge$ IF $\neg(search[1]\ \in$ DOMAIN $node.Edges)$
                  THEN $\wedge\ pc' =$ "NoMatch"
                  ELSE $\wedge\ pc' =$ "GetEdge"
             $\wedge$ UNCHANGED $\langle stack, input, prefix, root, node, search, result \rangle$

$GetEdge \triangleq\ \wedge\ pc =$ "GetEdge"

$$\land\ node' = node.Edges[search[1]]$$
$$\land\ pc' = \text{``CheckPrefix''}$$
$$\land\ \text{UNCHANGED }\langle stack,\ input,\ prefix,\ root,\ search,\ result\rangle$$

$CheckPrefix \triangleq$
$$\land\ pc = \text{``CheckPrefix''}$$
$$\land\ \text{IF } HasPrefix(search,\ node.Prefix)$$
$$\text{THEN }\ \land\ search' = TrimPrefix(search,\ node.Prefix)$$
$$\land\ pc' = \text{``Seek''}$$
$$\text{ELSE }\ \land\ \text{IF } HasPrefix(node.Prefix,\ search)$$
$$\text{THEN }\ \land\ pc' = \text{``Result''}$$
$$\text{ELSE }\ \land\ pc' = \text{``NoMatch''}$$
$$\land\ \text{UNCHANGED } search$$
$$\land\ \text{UNCHANGED }\langle stack,\ input,\ prefix,\ root,\ node,\ result\rangle$$

$NoMatch \triangleq$
$$\land\ pc = \text{``NoMatch''}$$
$$\land\ result' = \{\}$$
$$\land\ pc' = \text{``CheckResult''}$$
$$\land\ \text{UNCHANGED }\langle stack,\ input,\ prefix,\ root,\ node,\ search\rangle$$

$Result \triangleq$
$$\land\ pc = \text{``Result''}$$
$$\land\ result' = Range(node)$$
$$\land\ pc' = \text{``CheckResult''}$$
$$\land\ \text{UNCHANGED }\langle stack,\ input,\ prefix,\ root,\ node,\ search\rangle$$

$CheckResult \triangleq$
$$\land\ pc = \text{``CheckResult''}$$
$$\land\ Assert(result = Expected(root,\ prefix),$$
$$\text{``Failure of assertion at line 104, column 3.''})$$
$$\land\ pc' = \text{``Done''}$$
$$\land\ \text{UNCHANGED }\langle stack,\ input,\ prefix,\ root,\ node,\ search,$$
$$result\rangle$$

Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq pc = \text{``Done''} \land \text{UNCHANGED } vars$

$Next \triangleq Begin \lor Seek \lor CheckEdge \lor GetEdge \lor CheckPrefix \lor NoMatch$
$$\lor\ Result \lor CheckResult$$
$$\lor\ Terminating$$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$Termination \triangleq \Diamond(pc = \text{``Done''})$

END TRANSLATION

---

* Modification History
* Last modified *Wed Jun 30 12:04:00 PDT* 2021 by *mitchellh*
* Created *Wed Jun 30 10:05:52 PDT* 2021 by *mitchellh*