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.
$Inputs \triangleq \text{UNION } \{[1 \,..\, n \to Alphabet] : n \in MinLength \,..\, MaxLength\}$

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

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
$stack = \langle\rangle$,
$input \in InputSets$,

```
  prefix ∈ 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.

  Source: https://github.com/hashicorp/go-immutable-radix/blob/f63f49c0b598a5ead21c5015fb4d08fe7e3c21ea/iter.go#L16
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 := ⟨⟩ ;
  node := root ;
  search := prefix ;

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

  CheckEdge:
    if ¬(search[1] ∈ 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" $\land\ chksum(tla) =$ "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
$\land\ stack = \langle\rangle$
$\land\ input \in InputSets$
$\land\ prefix \in Inputs$
$\land\ root = RadixTree(input)$
$\land\ node = \{\}$
$\land\ search = \{\}$
$\land\ result = \{\}$
$\land\ pc =$ "Begin"

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

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

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

$GetEdge \triangleq$ $\land\ pc =$ "GetEdge"
$\land\ node' = node.Edges[search[1]]$
$\land\ pc' =$ "CheckPrefix"
$\land$ UNCHANGED $\langle stack, input, prefix, root, search, result \rangle$

$CheckPrefix \triangleq$ $\land\ pc =$ "CheckPrefix"
$\land$ IF $HasPrefix(search, node.Prefix)$

$$
\begin{array}{l}
\quad \text{THEN} \ \land \ search' = TrimPrefix(search,\ node.Prefix) \\
\qquad\quad \land \ pc' = \text{“Seek”} \\
\quad \text{ELSE} \ \land \ \text{IF} \ HasPrefix(node.Prefix,\ search) \\
\qquad\qquad\quad \text{THEN} \ \land \ pc' = \text{“Result”} \\
\qquad\qquad\quad \text{ELSE} \ \land \ pc' = \text{“NoMatch”} \\
\qquad\qquad \land \ \text{UNCHANGED} \ search \\
\land \ \text{UNCHANGED} \ \langle stack,\ input,\ prefix,\ root,\ node,\ result \rangle
\end{array}
$$

$$
\begin{array}{rl}
NoMatch \ \triangleq & \land \ pc = \text{“NoMatch”} \\
& \land \ result' = \{\} \\
& \land \ pc' = \text{“CheckResult”} \\
& \land \ \text{UNCHANGED} \ \langle stack,\ input,\ prefix,\ root,\ node,\ search \rangle
\end{array}
$$

$$
\begin{array}{rl}
Result \ \triangleq & \land \ pc = \text{“Result”} \\
& \land \ result' = Range(node) \\
& \land \ pc' = \text{“CheckResult”} \\
& \land \ \text{UNCHANGED} \ \langle stack,\ input,\ prefix,\ root,\ node,\ search \rangle
\end{array}
$$

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

Allow infinite stuttering to prevent deadlock on termination.

$$Terminating \ \triangleq \ pc = \text{“Done”} \ \land \ \text{UNCHANGED} \ vars$$

$$
\begin{array}{rl}
Next \ \triangleq & Begin \lor Seek \lor CheckEdge \lor GetEdge \lor CheckPrefix \lor NoMatch \\
& \lor \ Result \lor CheckResult \\
& \lor \ Terminating
\end{array}
$$

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

$$Termination \ \triangleq \ \Diamond(pc = \text{“Done”})$$

END TRANSLATION

* Modification History
* Last modified *Wed Jun* 30 11:46:15 *PDT* 2021 by *mitchellh*
* Created *Wed Jun* 30 10:05:52 *PDT* 2021 by *mitchellh*