Revised OCL for Keyword and Transition Tables

Keyword Table

addEntry(keywd: String, tknType:TokenType)

pre-conditions:

not (self.entries->exists(e | e.keyword = keywd))
-- There is no entry in the table with the given keyword

post-conditions:

/* A new KeywordTableEntry instance kte was created
kte.keyword = keywd
kte.tokenType = tknType
kte was added to set of keyword table entries*/

Old Version:
self.entries->one(e | e.keyword = keywd
and e.tokenType = tknType)

New Version:
kte.oclIsNew() and
kte.oclIsTypeOf(KeyWordTableEntry) and
kte.keyword = keywd and
kte.tokenType = tknType and
self.entries->self.entries@pre->union(Set(kte))
Transition Table

addTransitionState(stateNumber: int)

pre-conditions:

not (transitionStates->exists(e | e.state = stateNumber)
    or finalStates->exists(e | e.state = stateNumber))

/* The state is not already in the table as either a
 transition state or as a final state */

post-conditions:

/* A new TransitionStateEntry e has been created
  e.state = state
  The set of transitionStates contains e
 */

Old Version
self.transitionStates->one(e | e.state = stateNumber)

New Version

e.oclIsNew() and
e.oclIsTypeOf(TransitionStateEntry) and
e.state = stateNumber and
self.transitionStates = self.transitionStates@pre->union(Set(e))

addFinalState(stateNumber: int, tknType: TokenType,
               isRetract: Boolean)

pre-conditions:

not (transitionStates->exists(e | e.state = stateNumber)
    or finalStates->exists(e | e.state = stateNumber))

/* The state is not already in the table as either a
 transition state or as a final state */

/* A new TransitionStateEntry e has been created
  e.state = state
  The set of transitionStates contains e
 */
post-conditions:
/* A new FinalState instance f was created
   f.state = stateNumber
   f.tokenType = tknType
   f.isRetraction = isRetract
   f is in the transition table's set of finalStates
*/

Old Version
self.finalStates->one(f | f.state = stateNumber
                        and f.tokenType = tknType
                        and f.isRetraction = isRetract)

New Version
f.oclIsNew() and
f.oclIsType(FinalState) and
f.state = stateNumber and
f.tokenType = tknType and
f.isRetraction = isRetract and
self.finalStates = self.finalStates@pre->union(Set(f))

addTransition(fromState: int, chrSet: CharSet, toState: int)

pre-conditions:
self.transitionStates->one(e | e.state = fromState)
-- fromState is a transition state

(self.transitionStates->union(self.finalStates))
    ->one(e | e.state = toState)
-- toState is a transition state or a final state

self.transitionStates(s | s.state=fromState).transitions
    ->forall(t | t.charSet.intersection(chrSet)
                ->isEmpty())
/* the parameter chrSet is disjoint from the character set
   for any transition for the fromState
*/
post-conditions:

/* A new Transition t has been created
   t.charset = charSet
   t.targetState = toState
   t has been added to the set of transitions for the
       TransitionStateEntry for fromState
*/

Old Version
(self.transitionStates->any(e | e.state =
   fromState)).transitions
   ->one(t | t.charset = chrSet
      and t.targetState = toState)

New Version
let e = self.transitionStates->any(e | e.state = fromState)
   -- There's only one such transition state
in e.transitions = e.transitions@pre->union(Set(t))