Content Selection Algorithm

1. **Algo 1: Syntactic adequacy between the set of tokens and an item of axiom prerequisites (retriveTextsToCheck)**
   Obtaining the syntactically valid form of an item of axiom prerequisites based on the definitions of the axiom prerequisites and the slots.

Brief description: The principle of this algorithm is to reconstitute the table forming the path to check all possibilities based on the definitions of the tokens. A token is represented by a slot specification (SlotSpec) and a list of ontology elements (ontoElts). For instance, one can specify at the level of the slot specification the classes to exclude, the route of the research space, etc. From those specifications, the instance of axiom prerequisites to be checked are formed and will be verified for semantic and logical correctness.

Inputs: the set of tokens: `Tokens tokens` (The tokens contain some specifications restricting the possible candidates to replace the slots.) and the item of axiom prerequisites: `String text` to check (constituted by an axiom with slots)

```
Let axPreqInsts as AxPreqInsts
// list of axioms to check w.r.t the semantic/logical correctness
length ← getLengthFrom(tokens)
// such that the table can contain the combination of all ontology elements
array[][] ← array[size(tokens)][length]
array ← populateArray(tokens) // populating the array such that all combinations of all
// syntactically valid ontology elements are considered
j ← 0
while (j < length) do
// This loop is to devise the syntactically valid axioms based on the slot specifications.
     Let axPreqInst as AxPreqInst // a syntactically correct axiom
     newText ← copy(text)
     i ← 0
     Let assocs as Associations
     // a list of correspondences between the tokens and the ontology elements
     foreach token in tokens
         // This loop is to form the valid axiom based on the list of tokens.
         newText ← newText.replace(token.getTextSlot(), array[i][j])
         // specifying the slots from valid ontology elements (syntactically)
         AssociationTokenOntoElt assoc
         // a correspondence between the tokens and the ontology elements
         assoc.setToken(token)
         assoc.setOntologyElt(array[i][j])
         assocs.add(assoc)
         i++
     end foreach
     axPreqInst.setText(newText)
     // the syntactically correct axiom to be checked semantically
     axPreqInst.setAssociations(assocs)
     // assocs represents the associations between the ontology elements and
     // the slots to be replaced (useful for the NLG part of the algorithm)
     axPreqInsts.add(axPreqInst)
     j++
end while
return axPreqInsts
```

Outputs: `axPreqInsts`: the list of possible concrete items of axiom prerequisites (Ontology elements replace the slots) and the correspondences between the slots and the ontology elements that are stored

2. Semantic and logical adequacy between the syntactically correct axiom prerequisites and the content in the ontology

Brief description: Obtaining the semantically and logically valid form of axioms based on the definitions of the axiom prerequisites, the slots and the content of the ontology.

a. Algo 2: The recursive algorithm for selecting the items of axiom prerequisites:

Inputs:

- **int** `idAxItem`: the id of the item in the axiom prerequisites to investigate
- **Tokens** `consideredTokens`: the tokens considered in the axiom prerequisites
- **AxPreq** `axPreq`: the items of the axiom prerequisites
- **Step** `step`: the paths for obtaining the right answer
- **ResultAxs** `resultAxs`: the valid axioms with respect to the axiom prerequisites
- **MyOnto**: representing the ontology used

```
getValidAxiom(int idAxItem, Tokens consideredTokens, AxPreq axPreq, Step step, ResultAxs
resultAxs)
// This recursive algorithm is to find if a list of axioms is semantically/logically valid,
// corresponding to axiom prerequisites. Since it is a recursive-based algorithm, the next
// case is only investigated if the current axiom is valid.

    AxPreqItem axPreqItem ← axPreq.get(idAxItem)
    // obtaining the item of axiom prerequisites from an id
    Let tks as Tokens
    foreach textSlot in axPreqItem.getTextSlots()
    // for each slot in the item of axiom prerequisites
    // just limited to the slots that are used in the current
    // item of the axiom prerequisites
         Let token as Token
         token ← consideredTokens.getToken(textSlot)
         if (size(token.getOntoElts())>0) then // considering the token if it is not empty
             tks.add(token)
         end if
    end foreach

   Let axPreqInsts as AxPreqInsts // instances of items of axiom prerequisites
   axPreqInsts ← retrieveTextsToCheck(tks, axPreqItem.getText())
   // by using Algo 1, obtaining all syntactically valid axioms from the tokens 'tks' and
   // the item of axiom prerequisites 'axPreqItem' to be checked by the reasoner
   idAxItem++ // moving to the next item of axiom prerequisites

   foreach inst in axPreqInsts
      MyOnto.getManchesterOWLSyntaxParser().setStringToParse(inst.getText())
      // initialising the parsing of the syntactic valid axiom
      // MyOnto represents the ontology and permits the use of the reasoning over it.
      Let owlAxiom as OWLAxiom
      // a structure that can represent the instance of axiom prerequisites
      owlAxiom ← MyOnto.getManchesterOWLSyntaxParser().parseAxiom()
      // parsing the axiom (in Manchester syntax) to object: 'owlAxiom'
      boolean entailed ← MyOnto.getOWLReasoner().isEntailed(owlAxiom)
      // checking by the use of a reasoner if the ontology entails the axiom in: 'owlAxiom'
```

```
    if (entailed) then
       Let newTokens as Tokens
       newTokens ← copy(consideredTokens)
       // copying the tokens for the specific path
       Let associations as Associations // set of pairs of slots and ontology elements
       associations ← inst.getAssociations()
       // obtaining the associations between the tokens and the ontology elements

       foreach assoc in associations
           Let token as Token
           token ← newTokens.getToken(assoc.getToken().getTextSlot())
           token.clearOntoElts() // removing ontology elements for a particular token
           token.addOntoElt(assoc.getOntologyElt())
           // specifying the correct ontology element for a specific token
       end foreach

       if (idAxItem == size(axPreq)) then
       // stop condition: if there is no more item of axiom prerequisites to investigate
           Let resultAx as ResultAx // an instance of result
           resultAx.setAxPreqInst(inst)
           resultAxs.add(resultAx)
           resultAx.setPaths(step.getResultAxs())
           // record the path of the final result,
           // and the whole result is stored in 'resultsAxs'
      else if
           Let resAx as ResultAx // an instance of result
           resAx.setAxPreqInst(inst)
           Let newStep as Step
           // Step is a data structure linking parent and child to store the path
           // resolving the axiom prerequisites

           newStep.setParent(step)
           newStep.setText(inst.getText())
           newStep.setResultAx(resAx)
           // They are details of the path.

           getValidAxiom(idAxItem, newTokens, axPreq, newStep, resultAxs)
           // Since this is not the end of the investigation, Algo 2 recursively finds if
           // the next item of the axiom prerequisites is entailed by the ontology or
           // not.

      end if
    end if
  end foreach
```

Outputs: `ResultAxs resultAxs`: the valid axioms with respect to the axiom prerequisites

b. Algo 3: The content selection algorithm for the items of axiom prerequisites:

Inputs:

- `AxPreq axPreq`: the items of the axiom prerequisites
- `Tokens tokens`: the tokens used in the axiom prerequisites
- `MyOnto`: representing the ontology used

```
contentSelection(AxPreq axPreq, Tokens tokens)
// finding the valid semantic axioms for specific axiom prerequisites
    Let resultAxs as ResultAxs
    // the valid axioms with respect to the axiom prerequisites (initially empty)
    Let step as Step // step is to store the path (initially empty)
    idAxItem ← 0 // starting from the first item of the axiom prerequisites
    getValidAxiom(0, tokens, axPreq, step, resultAxs)
    // by using Algo 2: getting the valid axioms with respect to the axiom prerequisites,
    // the slot specifications and the content of the ontology
    return resultAxs
```

Outputs: `ResultAxs resultAxs`: the valid axioms with respect to the axiom prerequisites:

- the set of valid axioms (`axPreqInst`),
- the associations between the slots and the ontology elements (`token`, containing the `associations`), and
- the explanation (`step`).

c. The content selection algorithm for the set of items of axiom prerequisites:

For the set of axiom prerequisites, **`contentSelection`** described in **Algo 3** is used for each 'axiom prerequisites' to get the results.