Pseudo code for reusing cooking recipes (straight-forward retrieval and substitutional adaptation)

Mirjam Minor
Goethe University Frankfurt
Institute for Informatics
D-60325 Frankfurt am Main, Germany
minor@cs.uni-frankfurt.de

August 27, 2013

```
1  main( setOfIngredients, title, caseBase )
2  // create query:
3      if( !empty( setOfIngredients ) )
4          query.ingredients := setOfIngredients;
5      if( !empty( title ) )
6          query.title := title;
7  // find best matching case:
8      highestSimResult := 0;
9      foreach( case ∈ caseBase ) {
10         sim := compare( query, case );
11         if( sim > highestSimResult ) {
12             highestSimResult := sim;
13             bestMatchingCase := case;
14         }
15     }
16 // adapt case if required:
17     if( !empty( bestMachingCase ) &&
18                     highestSimResult < THRESH )
19         bestMatchingCase :=
20                     adapt( query, bestMatchingCase );
21
22     return bestMatchingCase;
23 function compare( query, case )
24 // compare titles:
25     titleWordsQ := splitString( query.title );
26     titleWordsC := splitString( case.title );
27     commonTitleWords := titleWordsQ ∩ titleWordsC;
28     titleSim := noOfElements( commonTitleWords );
29 // compare ingredients:
30     ingSim := 0;
31     foreach( ingQ ∈ query.ingredients ) {
32         maxLocalSim := 0;
33         foreach( ingC ∈ case.ingredients ) {
34             localSim := computeLocalSim( ingQ, ingC );
35             if( localSim > maxLocalSim )
36             maxLocalSim := localSim;
37         }
38         ingSim := ingSim + maxLocalSim;
39     }
40 // aggregate results:
41     return ( titleSim + ingSim );
42
43 function computeLocalSim( ingQ, ingC )
44     return ( DEPTH_OF_TAXONOMY /
45                         pathLength( ingQ, ingC ) );
46
```

```
function adapt( query, case )
// determine missing ingredients from query:
    commonIngs := query.ingredients ∩ case.ingredients;
    missingIngs := query.ingredients \ commonIngs;
    adaptationCandidates := case.ingredients \
                                        commonIngs;
// find substitutes and adapt case:
    foreach( ingQ ∈ missingIngs ) {
        maxLocalSim := 0;
        foreach( ingC ∈ adaptationCandidates ) {
            localSim := computeLocalSim( ingQ, ingC );
            if( localSim > maxLocalSim ) {
                maxLocalSim := localSim;
                substitutable := ingC;
            }
        }
        if( maxLocalSim > SUBST_THRESH ) {
            replace( case.ingredients, substitutable, ingQ );
            adaptationCandidates :=
                adaptationCandidates \ substitutable;
        }
    }
    return case;
```