February 23, 2021

This document is part of the paper "RDF^M: An Alternative Approach for Representing, Storing, and Maintaining Meta-knowledge in Web of Data." It presents the whole Algorithm used for matching the variables in the query and builds the final result.

The matching function takes input in the form of a user-entered query and predicate dictionaries. It returns the result linked to variables as a list.

Two sets of dictionaries are maintained; result set and current set for query processing. An additional dictionary vars[] is maintained to keep track of variables in user queries.

The current table is loaded from predicate-dictionary for matching purposes. The result table is filtered out based on the position of ?var to match with its previously encountered position in the user query. After processing each query, the variable in the user query is stored vars['?var'].

1 Algorithm

Algorithm 1 Step 3.2: matching_function

Input: line send by Algorithm 1.
Output: Result send back to Algorithm 1.

```
1: procedure EXECUTE line containing user_ query
2:     for each line do
3:         if sub is variable, obj is variable then.
4:             if subject variable is new then
5:                 Append the contents of sub-dict of result-table with that of sub-dict of current-table.
   And Add variable to list in vars['sub']
6:             if sub is variable and is previously encountered then
7:                 Find the position of previous encounter and select the associated dictionary.
8:                 for each key in associated-dict do
9:                     if key exists in current table's sub-dict then
10:                        Make union of current-table's sub-dict[key] and result table's associated-dict[key].
11:                        Store it as value of associated-dict[key].
12:                    if key not found in sub-dict of current-table then
13:                        Delete the key from associated-dict of result-table.
14:            if object variable is new then
15:                Append the contents of obj-dict of result-table with that of obj-dict of current-table.
16:                Add variable to list in vars['obj']
17:            if obj is variable and is previously encountered then
18:                Find the position of previous encounter and select the associated dictionary.
19:                for each key in associated-dict do
20:                    if key exists in current table's obj-dict then
21:                        Make union of current-table's obj-dict[key] and result table's associated-dict[key].
22:                        Store it as value of associated-dict[key].
23:                    if key not found in obj-dict of current-table then
24:                        Delete the key from associated-dict of result-table.
25:        if sub is given, obj is variable then
26:            Keep the matching given sub in result-table's sub-dict.
27:            for each non-matching key in sub-dict do
28:                Remove subject from obj-dict of result-table.
29:                Remove object from uid-dict of result-table.
30:            Remove the non-matching keys.
31:            if object variable is new then
32:                Append the contents of obj-dict of result-table with that of obj-dict of current-table
   and add variable to list in vars['obj']
33:            if obj is variable and is previously encountered then
34:                Find the position of previous encounter and select the associated dictionary.
35:                for each key in associated-dict do
36:                    if key exists in current table's obj-dict then
37:                        Make union of current-table's obj-dict[key] and result table's associated-dict[key].
38:                        Store it as value of associated-dict[key].
39:                    if key not found in obj-dict of current-table then
40:                        Delete the key from associated-dict of result-table.
```

```
41:         if sub is variable, object is given then
42:             Keep the matching given obj in result-table's sub-dict.
43:             for each non-matching key in obj-dict do
44:                 Remove subject from sub-dict of result-table.
45:                 Remove object from uid-dict of result-table.
46:             Remove the non-matching obj keys.
47:             if subject variable is new then
48:                 Append the contents of sub-dict of result-table with that of sub-dict of current-table.
49:                 Add variable to list in vars['sub']
50:             if sub is variable and is previously encountered; [comparing it with vars[]] then
51:                 Find the position of previous encounter.
52:                 Eg: Previously at sub, obj or uid position, select the associated dictionary.
53:                 for each key in associated-dict do
54:                     if key exists in current table's sub-dict then
55:                         Make union of current-table's sub-dict[key] and result table's associated-
    dict[key].
56:                         Store it as value of associated-dict[key].
57:                     if key not found in sub-dict of current-table then
58:                         Delete the key from associated-dict of result-table.
59:         if sub is given, object is given then
60:             Keep the matching given obj in result-table's sub-dict.
61:             for each non-matching key in obj-dict do
62:                 Remove subject from sub-dict of result-table.
63:                 Remove object from uid-dict of result-table.
64:             Remove the non-matching keys.
65:             Keep the matching given sub in result-table's sub-dict.
66:             for each non-matching key in sub-dict do
67:                 Remove subject from obj-dict of result-table.
68:                 Remove object from uid-dict of result-table.
69:             Remove the non-matching keys.
70:         Start UID1 processing.
71:         if UID is variable in user query then check:
72:             if UID variable is new then
73:                 Append the contents of uid-dict of result-table with that of uid-dict of current-table.
74:                 Add variable to list in vars['uid']
75:             if UID is variable and is previously encountered; [comparing it with vars[]] then
76:                 Find the position of previous encounter.
77:                 Eg: Previously at sub, obj or uid position, select the associated dictionary.
78:                 for each key in associated-dict do
79:                     if key exists in current table's uid-dict then
80:                         Make union of current-table's uid-dict[key] and result table's associated-
    dict[key].
81:                         Store it as value of associated-dict[key].
82:                     if key not found in uid-dict of current-table then
83:                         Delete the key from associated-dict of result-table.
```

```
84:         if UID1 is given then
85:             Keep the matching given UID in result-table's uid-dict.
86:             for each non-matching key in uid-dict do
87:                 Remove subject from sub-dict of result-table.
88:                 Remove object from obj-dict of result-table.
89:             Remove the non-matching keys.
90:         Predicate parameters processing.
91:         for each parameter do
92:             if parameter if variable then
93:                 Check if parameter is new OR Check if parameter matches with previous parameter
    variables and match
94:             if parameter is given then
95:                 Match given parameter accordingly with result table.
96:             if parameter is not required then
97:                 No processing required.
98:         Start UID2 processing.
99:         if UID2 is variable then
100:            Check if UID2 is new OR
101:            Check if UID2 matches with previous variables and match accordingly
102:        if UID2 is given then
103:            Match given UID2 with result table
```