May 28, 2021

This document is part of the chapter "Labeled k-partite Graph for Statement Annotation". It presents the matching algorithm which is used in the chapter.

1 Algorithm

Algorithm Step 2.2: 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 ?var, obj is ?var then.
4:             if sub ?var is new then
5:                 Append the contents of sub_dict of result_dictionary with that of sub_dict of current_dictionary. And Add ?var to list in vars['sub']
6:             end if
7:             if sub is ?var and is previously encountered then
8:                 Find the position of previous encounter and select the corresponding_dictionary.
9:                 for each key in corresponding_dictionary do
10:                    if key exists in current_dictionary's sub_dict then
11:                        Make union of current_dictionary's sub_dict[key] and result_dictionary's corresponding_dictionary[key].
12:                        Store it as value of corresponding_dictionary[key].
13:                    end if
14:                    if key not found in sub_dict of current_dictionary then
15:                        Delete the key from corresponding_dictionary of result_dictionary.
16:                    end if
17:                end for
18:            end if
19:            if obj ?var is new then
20:                Append the contents of obj_dict of result_dictionary with that of obj_dict of current_dictionary.
21:                Add ?var to list in vars['obj']
22:            end if
23:            if obj is ?var and is previously encountered then
24:                Find the position of previous encounter and select the corresponding_dictionary.
25:                for each key in corresponding_dictionary do
26:                    if key exists in current_dictionary's obj_dict then
27:                        Make union of current_dictionary's obj_dict[key] and result_dictionary's corresponding_dictionary[key].
28:                        Store it as value of corresponding_dictionary[key].
29:                    end if
30:                    if key not found in obj_dict of current_dictionary then
31:                        Delete the key from corresponding_dictionary of result_dictionary.
32:                    end if
33:                end for
34:            end if
35:        end if
36:        if sub is given, obj is ?var then
37:            Keep the matching given subin result_dictionary's sub_dict.
38:            for each non-matching key in sub_dict do
39:                Remove sub from obj_dict of result_dictionary.
40:                Remove obj from uid-dict of result_dictionary.
41:            end for
42:            Remove the non-matching keys.
43:            if obj ?var is new then
44:                Append the contents of obj_dict of result_dictionary with that of obj_dict of current_dictionary and add ?var to list in vars['obj']
45:            end if
46:            if obj is ?var and is previously encountered then
47:                Find the position of previous encounter and select the corresponding_dictionary.
48:                for each key in corresponding_dictionary do
49:                    if key exists in current_dictionary's obj_dict then
50:                        Make union of current_dictionary's obj_dict[key] and result_dictionary's corresponding_dictionary[key].
51:                        Store it as value of corresponding_dictionary[key].
52:                    end if
53:                    if key not found in obj_dict of current_dictionary then
54:                        Delete the key from corresponding_dictionary of result_dictionary.
55:                    end if
56:                end for
57:            end if
58:        end if
```

```
59:         if sub is ?var, obj is given then
60:             Keep the matching given obj in result_dictionary's sub_dict.
61:             for each non-matching key in obj_dict do
62:                 Remove sub from sub_dict of result_dictionary.
63:                 Remove obj from uid-dict of result_dictionary.
64:             end for
65:             Remove the non-matching obj keys.
66:             if sub ?var is new then
67:                 Append the contents of sub_dict of result_dictionary with that of sub_dict of
    current_dictionary.
68:                 Add ?var to list in vars['sub']
69:             end if
70:             if subis ?var and is previously encountered then
71:                 Find the position of previous encounter.
72:                 Previously at sub, obj or uid position, select the corresponding_dictionary.
73:                 for each key in corresponding_dictionary do
74:                     if key exists in current_dictionary's sub_dict then
75:                         Make union of current_dictionary's sub_dict[key] and result_dictionary's
    corresponding_dictionary[key].
76:                         Store it as value of corresponding_dictionary[key].
77:                     end if
78:                     if key not found in sub_dict of current_dictionary then
79:                         Delete the key from corresponding_dictionary of result_dictionary.
80:                     end if
81:                 end for
82:             end if
83:         end if
84:         if sub is given, obj is given then
85:             Keep the matching given obj in result_dictionary's sub_dict.
86:             for each non-matching key in obj_dict do
87:                 Remove sub from sub_dict of result_dictionary.
88:                 Remove obj from uid-dict of result_dictionary.
89:             end for
90:             Remove the non-matching keys.
91:             Keep the matching given subin result_dictionary's sub_dict.
92:             for each non-matching key in sub_dict do
93:                 Remove sub from obj_dict of result_dictionary.
94:                 Remove obj from uid-dict of result_dictionary.
95:             end for
96:             Remove the non-matching keys.
97:         end if
98:         Start UID1 processing.
99:         if UID is ?var in user query then check:
100:             if UID ?var is new then
101:                 Append the contents of uid-dict of result_dictionary with that of uid-dict of
    current_dictionary.
102:                 Add ?var to list in vars['uid']
103:             end if
104:             if UID is ?var and is previously encountered then
105:                 Find the position of previous encounter.
106:                 Previously at sub, obj or uid position, select the corresponding_dictionary.
107:                 for each key in corresponding_dictionary do
108:                     if key exists in current_dictionary's uid-dict then
109:                         Make union of current_dictionary's uid-dict[key] and result_dictionary's
    corresponding_dictionary[key].
110:                         Store it as value of corresponding_dictionary[key].
111:                     end if
112:                     if key not found in uid-dict of current_dictionary then
113:                         Delete the key from corresponding_dictionary of result_dictionary.
114:                     end if
115:                 end for
116:             end if
117:         end if
```

```
118:         if UID1 is given then
119:             Keep the matching given UID in result_dictionary's uid-dict.
120:             for each non-matching key in uid-dict do
121:                 Remove sub from sub_dict of result_dictionary.
122:                 Remove obj from obj_dict of result_dictionary.
123:             end for
124:             Remove the non-matching keys.
125:         end if
126:         Predicate parameters processing.
127:         for each parameter do
128:             if parameter if ?var then
129:                 Check if parameter is new OR Check if parameter matches with previous parameter ?vars and match
130:             end if
131:             if parameter is given then
132:                 Match given parameter accordingly with result_dictionary.
133:             end if
134:             if parameter is not required then
135:                 No processing required.
136:             end if
137:         end for
138:         Start UID2 processing.
139:         if UID2 is ?var then
140:             Check if UID2 is new OR
141:             Check if UID2 matches with previous ?vars and match accordingly
142:         end if
143:         if UID2 is given then
144:             Match given UID2 with result_dictionary
145:         end if
146:     end for
147: end procedure
```