CHAPTER 1

THE NEW APPROACH

As it was discussed and depicted, in chapter **??** the main problems of the previous approach are:

- Faulty inclusion of the credibility values for the fuzzy rules, into the similarity evaluation function.
- The final result of the method converging to the default values defined between an arbitrary predicate and the identity predicate.
- Flawed filtering mechanism of the search algorithm.
- No means for utilizing the information which comes from shared similarity concepts in similarity predicate pairings.

Thus hereby we propose a new methodology which in general lines:

- Has a sound evaluation algorithm that prevents credibility values from dominating the final result.
- Does not introduce any extra knowledge which is not originally in the knowledge base, so avoids problems like the final result converging to erroneous values.
- Has a solid search algorithm which is able to utilize all of the information that the knowledge base contain, including similarity concepts that appear in more than one fuzzy rule.
- And as the predicate trees are maintained in their original forms, algorithm is able to work effectively on the complete structure emitting the need of filtering some parts of the data, which could potentially consist of crucial information.

We present the method in the following way:

We lay the fundamentals for a generic mathematical formulation of the main focus of the research. Then the application of the methodology in logic programming is inspected. Lastly, final section sheds light on the comparison of the current and prior approaches via observing solid examples.

1.1 Problem Description

The first step for our approach is generalizing the research's main focus, finding the similarity proximity of two predicates via giving a mathematical foundation. As seen in chapter **??**, the solution of the problem relies heavily on the predefined similarity relations between subconcepts. Moreover it was displayed that by expanding the rules in the knowledge bases, the construction of the predicate trees resulted in a graph representation of the original domain. This problem of matching subconcepts in two trees can easily describe the precise meaning of the matching with a mathematical definition. Informally, one may define matching between two predicates in a knowledge base of a logic program as follows: given some specific node corresponding to concept of our interest in the knowledge base- call it a, two concepts (i.e. a and candidate node) can be matched in the corresponding trees if there is a mapping between some specific subset of nodes which are reachable from a in first tree and some subset of nodes which are reachable from candidate node in the second tree.

1.1.1 Preliminaries

A function f is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output.

A *directed graph* (digraph) is an ordered pair *(V, E)* , where V is a set and E is a binary relation on V .

In a digraph *(V, E)*, the elements of V are called vertices, and the elements of E are called the edges of the digraph.

A digraph *(V',E')* is a subgraph of a digraph *(V, E)* if $V' \subseteq V$ and $E' \subseteq E$.

In a digraph (V, E), a path from a vertex u to a vertex u' is a sequence $\langle v_0, v_1, ..., v_k \rangle$ of vertices such that $u = v_0$ and $u' = v_k$,and $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq k$; we call v_k as the terminal node of the path. If there is a path from a vertex u to a vertex v, then we say that v is reachable from u. If V' is a subset of V, a path from u to v whose vertices belong to V' is a path from u to v in V'. If there exists a path from u to v in V' , v is reachable from u in V'.

Let $G_0 = (V_0, E_0)$ be a digraph whose vertices are labeled by a function $f_0 : V_0 \rightarrow VL_0$ where VL_0 is the vertex labels of G_0 and let $G_1 = (V_1, E_1)$ be a graph whose vertices are labeled by a function $f_1 : V_1 \rightarrow VL_1$ where VL_1 is the vertex labels of G_1. Let vm be a function that maps the label of a vertex in VL_0 to a set of vertex labels in VL_1.

We say that a terminal node v_0 matches a terminal node v_1, iff

$$VL_1(v_1) \in vm(VL_0(v_0)).$$

Informally a terminal node v_0 matches a terminal node v_1, when the vertex label of v_1 is in the vertex label set that the vertex label of v_0 is mapped to by vm.

We say that an arbitrary node v_0 matches an arbitrary node v_1, iff

For each element u_0 of a subset of vertices reachable from v_0 via a path p_0, there exists a vertex u_1 reachable from v_1 via a path p_1 in G_1 such that u_0 matches u_1.

Again if we are to state the meaning in an informal manner, two arbitrary nodes v_0 and v_1 match, if and only if there exists a set of vertices that are reachable from v_0, and each of these vertices match to some vertex which are reachable from v_1.

1.1.2 Vertex Matching Problem

We define an Vertex Matching Problem *(VMP)* with the following input:

- Two finite, polynomially bounded directed graphs: $G_0 = (V_0, E_0)$, $G_1 = (V_1, E_1)$
- Vertex Labeling Functions: $f_0 : V_0 \rightarrow VL_0$, $f_1 : V_1 \rightarrow VL_1$
- Vertex Labeling Match Function: $vm : VL_0 \rightarrow 2^{VL_1}$
- Vertices to be matched: $v_0 \in V_0$, $v_1 \in V_1$

1.2 Algorithm in the Domain of Fuzzy Logic

The application of the generic formalism presented in section **??** consists of three steps:

- Construction of the predicate tree
- Search algorithm for similar predicates
- The similarity evaluation function

In a nutshell, firstly the predicate trees are constructed via utilizing the fuzzy rules of the program. Then the search algorithm locates similar predicate pairings and collects the corresponding values. Finally the evaluation algorithm computes the resulting similarity proximity value for the predicate pair of interest.

The topics are inspected in detail in the following subsections respectively.

1.2.1 Construction of the Predicate Tree

The process for building the predicate trees follow the same methodology as the one depicted in section **??**. Thus as a brief recap we might state that, for every fuzzy rule in the knowledge base in the following form:

$$p_c(\vec{t}) \xleftarrow{c,F_c} F(p_1(\vec{t_1}), ..., p_n(\vec{t_n}))$$

we expand the head of the rule p_c via adding the predicates that the body of the fuzzy rule contain, i.e. $p_1, ..., p_n$, as its children in the directed tree graph. The task continues in a recursive manner until all the rules are exhausted.

No modifications are done on the original predicate trees at any step of the algorithm. Hence the process of building the trees is very straightforward.

1.2.2 Search and Evaluation

Similarity searching process heavily builds on the generic *vertex matching algorithm* given in section **??**. The only modification is that instead of searching exact matches between the subconcepts, the focus is on finding ones for which a fuzzy similarity rule exists in the particular program.

So if we are to re-state the methodology with the minor modification, when observing the similarity between predicates p_1 and p_2, for every vertex v_1 reachable from p_1 via a path t_1, the algorithm looks for a vertex v_2 reachable from p_2 via a path t_2 where the similarity degree between v_1 and v_2 is defined. The values of these relations are collected in the set *sm* and similarity degree *sd* between p_1 and p_2 is evaluated with the following equation:

$$sd(p_1, p_2) = (1 - |cred_{p_1} - cred_{p_2}|)\mathbf{OP}(\frac{\sum_{i=1}^{n} sm_i}{n}) \tag{1.1}$$

where $cred_1$ and $cred_2$ are the credibility values of the fuzzy rules that contain p_1 and p_2 in their heads respectively, OP is the fuzzy operator that is again defined by those rules, and n is the cardinality of the set *sm*.

Mind that as the focus is on two fuzzy rules at a given time, there might me two distinct OP values for the corresponding rules. In those cases, the generic product operation defined on real number is taken as the default OP value.

1.3 Improvements Over Related Work

The purpose of the section is depicting the contributions the new methodology introduces, specifically the improvements over the related work, the *Structured Based Measurement* that was introduced by Lu (**?**). As mentioned in the beginning of this chapter, main sources of advancement are the way the predicate trees are constructed, the methodology for searching the predicate trees, and the structure of the evaluation algorithm. We will re-visit the examples that had been presented in chapter **??** in order to observe how the new methodology compare to *SBM* and also see some extension that it introduces to the domain of interest which gives birth to some interesting features for the framework.

1.3.1 Inclusion of Credibilities in Similarity Evaluation Function

The first difference of the approaches come with the way they utilize the credibility values of the fuzzy rules in the similarity evaluation function for the predicates. As mentioned in the previous chapter, the way that *SBM* adopts proves to be problematic when the branching factor is small. Since it's directly summed with the values of similarity pairs in the numerator of the equation, when the cardinality n is small, the values of credibilities simply overshadow of similarity pairs.

Example 1.3.1
The following simple example displays one such scenario:

$$\begin{array}{ll} good\_basketball\_player: & (Player) \\ bad\_basketball\_player: & (Player) \\ good\_technique: & (Player) \\ egoism: & (Player) \end{array}$$

$$\begin{array}{lll} good\_basketball\_player(X) & \overset{0.95,.}{\longleftarrow} prod & good\_technique(X). \\ bad\_basketball\_player(X) & \overset{0.9,.}{\longleftarrow} prod & egoism(X)\ . \end{array}$$

$$Sim(good\_technique, egoism) = 0.1$$

The predicate trees are the same for both of the approaches since there is no need for expansion in this case as they're structurally equivalent. The built trees are as follows:

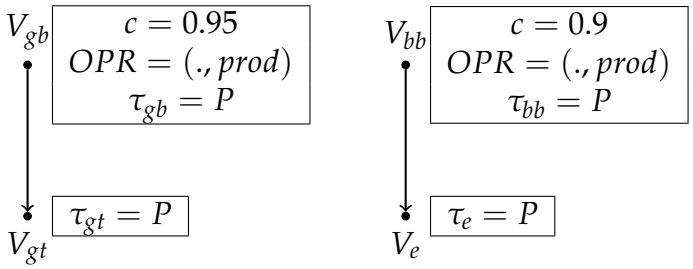

Figure 1.1: Predicate trees for *good_basketball_player* and *bad_basketball_player*

By inspecting the example, one should **not** expect the predicates *good_basketball_player* and *bad_basketball_player* to be similar at all.

This is indeed the case for our approach. Let us evaluate the similarity equation in order to demonstrate this:

In addition to the variables of the equation, we display the similar predicate pairs collected via the search algorithm in the set pm.

$cred_{p_1} = 0.95$, $cred_{p_2} = 0.9$, $OP = prod$, $pm = \{$ (good_technique, egoism)$\}$, $sm = \{0.1\}$, $n = 1$.

When the values are input in the equation:

$$\begin{aligned} sd(p_1, p_2) &= (1 - |cred_{p_1} - cred_{p_2}|)\boldsymbol{OP}(\frac{\sum_{i=1}^{n} sm_i}{n}) \\ &= (1 - |0.95 - 0.9|).(\frac{0,1}{1}) \\ &= 0.095 \\ &\cong \mathbf{0.1} \end{aligned} \tag{1.2}$$

Our method evaluates *good_basketball_player* and *bad_basketball_player* as not similar, which is what we expect by using common sense. In contrast with that, as seen before, *SBM* produces unexpected results. To illustrate this affirmation we show the results *SBM* obtains:

M_s = { (good_technique, egoism)}, $c^a = 0.95$, $c^b = 0.9$, $n = 1$.

$$\begin{aligned} M_v &= \frac{\sum_{i=1}^{1} v_i + 1 - |c^a - c^b|}{n+1} \\ &= \frac{0.1 + 1 - |0.95 - 0.9|}{1+1} \\ &= \frac{1.05}{2} \\ &\cong \mathbf{0.53} \end{aligned} \tag{1.3}$$

So on contrary to what was expected, *SBM* calculated *good_basketball_player* and *bad_basketball_player* to be somewhat similar. As mentioned earlier, this problem is caused by the fact that the equation does not integrate the credibility values in a proper way, thus in some cases(when the branching factor is low) the emphasize is so high that it dominates the similarity values between the subpredicates.

As displayed by the example, our approach handles this issue without any problems as it introduces the confidence values via the operator that is defined by the fuzzy rules themselves.

1.3.2 Convergence to Identity Predicate

We had stated that in order to compare two predicates, *SBM* needs them to have the same tree structure. And in order to accomplish this when they are not equal, the missing branches and leaves of the smaller tree are filled with identity predicates. Moreover a default similarity value is defined between an arbitrary predicate and the identity predicate.

In the same sense that it was discussed earlier, regrettably this introduces a couple of problems concerning the precision of the algorithm's final result. Similar to the effect that credibility values had in the previous section, this time we may have such an overwhelming impact from the default similarity values that is defined between the identity predicate and an arbitrary predicate. We will especially encounter this kind of scenarios when one tree has high branching factor compared to the other one.

Example 1.3.2
Once more we may see an example for that kind of case in the following program:

$$\begin{array}{ll} classy\_restaurant: & (Restaurant) \\ good\_restaurant: & (Restaurant) \\ well\_trained\_waiters: & (Restaurant) \\ expensive\_inventory: & (Restaurant) \\ has\_good\_service: & (Restaurant) \\ has\_healthy\_food: & (Restaurant) \\ has\_tasty\_food: & (Restaurant) \\ has\_nice\_surroundings: & (Restaurant) \\ has\_high\_reputation: & (Restaurant) \end{array}$$

$$\begin{array}{lll} classy\_restaurant(X) & \overset{1.0,.}{\longleftarrow} prod & well\_trained\_waiters(X), expensive\_inventory(X). \\ good\_restaurant(X) & \overset{0.95,.}{\longleftarrow} prod & has\_healthy\_food(X), has\_good\_service(X), \\ & & has\_nice\_surroundings(X), has\_high\_reputation(X), \\ & & has\_tasty\_food(X). \end{array}$$

$$Sim(well\_trained\_waiters, has\_good\_service) = 0.9$$

$$Sim(expensive\_inventory, has\_nice\_surroundings) = 0.8$$

Regarding to the program we should expect the predicates *classy_restaurant* and *good_restaurant* to have a high similarity degree.

Again let's start evaluating the result with our new approach. Our algorithm does not require the predicates to be expanded so the tree are in their original form.

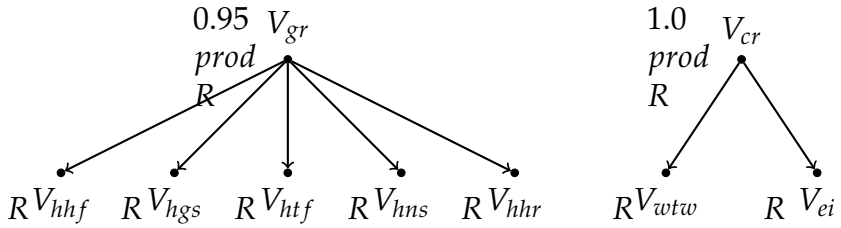

Figure 1.2: Predicate trees in original form

Then the values for the variables of the equation:

$cred_{p_1}$ = 0.95 , $cred_{p_2}$ = 1, OP = *prod*, pm = { (well_trained_waiters, has_good_service), (expensive_inventory, has_nice_surroundings)}, sm = {*0.9, 0.8*}, n = 2.

And finally the evaluation:

$$\begin{aligned} sd(p_1, p_2) &= (1 - |cred_{p_1} - cred_{p_2}|)\boldsymbol{OP}(\frac{\sum_{i=1}^{n} sm_i}{n}) \\ &= (1 - |0.95 - 1|).(\frac{0.9 + 0.8}{2}) \\ &\cong \mathbf{0.81} \end{aligned} \tag{1.4}$$

As the result seems high enough in harmony with the expectation, indeed we can be satisfied with the evaluated degree of similarity.

Thus we may pursue by constructing the predicate trees for *SBM*. As the two trees are not structurally equal, the smaller tree, the predicate tree of *classy_restaurant* must be expanded.

There is just one level of expansion. Since at every level the M_s that leads to the highest degree of similarity is chosen, the corresponding decision set will be: M_{ds} = { *(well_trained_waiters, has_good_service), (expensive_inventory, has_nice_surroundings), (ID, has_healthy_food), (ID, has_tasty_food), (ID, has_high_reputation)*}

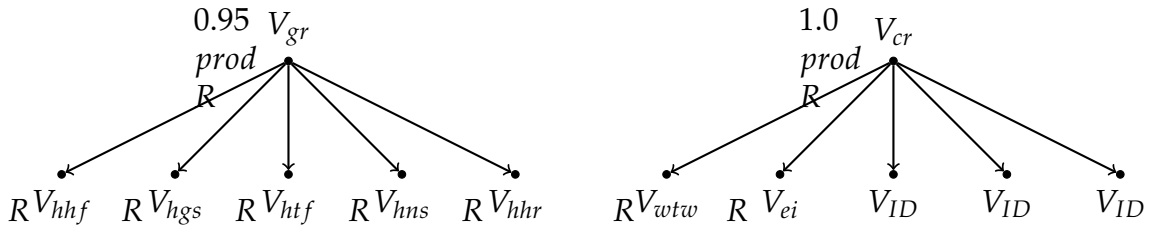

Figure 1.3: Predicate trees in expanded form for *SBM*

One thing that we should pay attention to is that, since in this example, *SBM* needed the introduction of the *Identity Predicate*, a default similarity value between an arbitrary predicate and the identity predicate should also be defined. Assume *Sim(ID, p) = 0.3* for any arbitrary predicate.

The rest of the variable values are as follows:
$c^a = 0.95$, $c^b = 1$, $n = 5$.

$$\begin{aligned} M_v &= \frac{\sum_{i=1}^{5} v_i + 1 - |c^a - c^b|}{n+1} \\ &= \frac{2.9 + 1 - |0.95 - 1|}{5+1} \\ &\cong \mathbf{0.55} \end{aligned} \tag{1.5}$$

In this case the algorithm was not able to successfully evaluate and conclude that the predicates are actually closely related. This distortion was caused because of the relatively big branching factor difference and the identity predicates introduced for the missing one. As one can see, in *SBM* as ***the missing number of nodes in one tree increases, the similarity degree calculated by the algorithm converges to the default values*** that is defined between an arbitrary predicate and the identity predicate.

Since our algorithm does not introduce any such external knowledge, and just use the original information of the knowledge base, it does not have such shortcomings.

1.3.3 Wrong Filtering

As one may remember from the earlier discussions from the previous chapter, there is one other faulty behavior of *SBM* which we suspect to be caused from the relaxed assumption of the knowledge bases. After every level of expansion, *SBM* checks every predicate pair between the two trees with respect to their resulting similarity degree. Before the next level of expansion is pursued, a filtering is done on the tree by selecting the best pair combination on the level. The problem with this approach is that the information on a prior level is incomplete, and thus wrong steps can be taken when filtering that will cause the loss of crucial information for the main focus, *i.e.* comparing the similarity values of the main predicates.

Since the previous thesis work does not include any examples with predicate trees deeper then just one level, we try to demonstrate the problem with the following example:

Example 1.3.3
The program consists of following type declarations and rules:

$$
\begin{array}{ll}
modern\_city: & (City) \\
livable\_city: & (City) \\
life\_expectancy: & (Society) \\
birth\_rate: & (Society) \\
social\_welfare: & (Society) \\
\#of\_schools & (Society) \\
quality\_of\_academic\_staff & (Society) \\
\#of\_teachers & (Society) \\
compulsory\_schooling\_length & (Society) \\
educated\_society: & (Society) \\
\#of\_healthy\_individuals & (Society) \\
literacy\_rate: & (Society) \\
high\_population: & (Society)
\end{array}
$$

$$
\begin{array}{lll}
livable\_city(X) & \xleftarrow{0.8,.} prod & literacy\_rate(X), \#of\_healthy\_individuals(X). \\
literacy\_rate(X) & \xleftarrow{1.0,.} prod & compulsory\_schooling\_length(X), \#of\_teachers(X). \\
modern\_city(X) & \xleftarrow{0.7,.} prod & educated\_society(X), high\_population(X), \\
 & & social\_welfare(X). \\
educated\_society(X) & \xleftarrow{1.0,.} prod & \#of\_schools(X), quality\_of\_academic\_staff(X). \\
high\_population(X) & \xleftarrow{1,0,.} prod & birth\_rate(X), life\_expectancy(X).
\end{array}
$$

$$Sim(compulsory\_schooling\_length, \#of\_schools) = 0.9$$

$$Sim(\#of\_teachers, quality\_of\_academic\_staff) = 0.85$$

$$Sim(literacy\_rate, social\_welfare) = 0.4$$

$$Sim(\#of\_healthy\_individuals, life\_expectancy) = 0.7$$

Focus of interest is evaluating the similarity degree of *livable_city* and *modern_city*. Let's start calculating this value via the new algorithm.

By know we know that our new methodology does not need to modify the trees or do calculations of *mid-results*. Thus let's write down the input needed by the similarity evaluation equation, and then solve the equation for these values.

$cred_{p_1} = 0.7$, $cred_{p_2} = 0.8$, $OP = prod$, pm = { (compulsory_schooling_length, #of_schools), ((#of_teachers, quality_of_academic_staff), (literacy_rate, social_welfare), (#of_healthy_individuals, life_expectancy) }, $sm = \{0.9, 0.85, 0.4, 0.7\}$, $n = 4$.

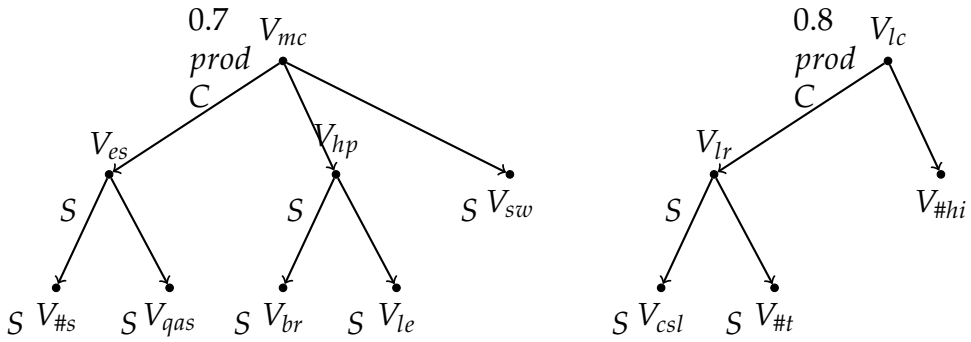

Figure 1.4: Predicate trees of *livable_city* and *modern_city* in original form

$$\begin{aligned} sd(p_1, p_2) &= (1 - |cred_{p_1} - cred_{p_2}|)\boldsymbol{OP}(\frac{\sum_{i=1}^{n} sm_i}{n}) \\ &= (1 - |0.7 - 0.8|).((\frac{\sum_{i=1}^{4} sm_i}{4}) \\ &\cong \mathbf{0.64} \end{aligned} \tag{1.6}$$

Then as usual we follow the steps of *SBM*.

Figure 1.5: Predicate trees after one level of expansion in *SBM*

After the first expansion is done, again an identity predicate is introduced for the missing leaf node in the predicate tree with the root *livable_city*. The problem arise in this step. As we have discussed before, *SBM* makes a filtering of the node-pairs before the next expansion, with respect to the similarity proximities of the pairs in this level. And once again as we have mentioned, this as most of the information hidden in the lower levels is not apparent to the algorithm yet, filtering can cause neglecting some important paths of the tree.

For instance at this point, between the node pairs, only a similarity relation between the predicates *literacy_rate* and *social_welfare* is defined. The algorithm continues expanding, via selecting the *middle set* with the highest similarity value at the current value as the *decision middle set*. So in this example, at the first level M_{ds} = { (V_{sw} , V_{lr}), (V_{hp} , V_{ID}) } or M_{ds} = { (V_{sw} , V_{lr}), (V_{es} , V_{ID}) } as they prove to be the best of the M_s sets.

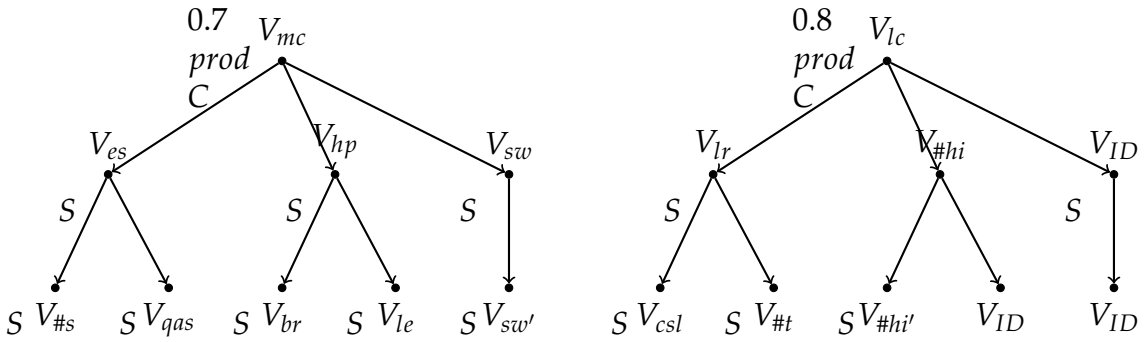

Figure 1.6: Predicate trees after two levels of expansion in *SBM*

However this local optimization approach eliminates other fruitful pair combinations such as *social_welfare* and *educated_society*. In *Figure 6* this pair corresponds to the left subtrees of the main predicate trees. These prove to be the most similar subconcepts of the two as there are two leaf node pairs for which the similarity relation is defined via values *0.9* and *0.85*.

All in all, wrong choice for filtering causes comparing wrong pairs of subpredicates in the end and thus a distorted final similarity degree between the predicates of interest.

1.3.4 Shared Similarity Concepts

In section **??**, we had discussed the incapability of the *SBM* realizing the information where a specific concept is included in more then one similarity relations as the methodology seeks for a one to one mapping between the subconcepts of the main predicates. Our algorithm does not suffer from the same problem as it may collect and process all distinct information from each different fuzzy similarity rules.

Example 1.3.4
Let us once again observe the following simple program:

$$\begin{array}{ll} touristic\_place: & (Land) \\ nice\_destinationt: & (Land) \\ cultural\_venues: & (Sight) \\ natural\_wonders: & (Sight) \\ many\_sights: & (Sight) \\ good\_weather: & (Temperature) \end{array}$$

$$touristic\_place(X) \xleftarrow{1.0,.} prod \quad cultural\_venues(X), natural\_wonders(X).$$
$$nice\_destination(X) \xleftarrow{1.0,.} prod \quad good\_weather(X), many\_sights(X).$$

$$Sim(cultural\_venues, many\_sights) = 0.7$$

$$Sim(natural\_wonders, many\_sights) = 0.7$$

In section **??** we had proved that the algorithm is bound to miss one of the similarity relations and thus the corresponding evaluation algorithm could not utilize the complete data.

Let us tackle the problem with our methodology:

$cred_{p_1} = 1$, $cred_{p_2} = 1$, $OP = prod$, $pm = \{$ (cultural_venues, many_sights), (natural_wonders, many_sights)$\}$, $sm = \{0.7, 0.7\}$, $n = 2$.

And finally the evaluation:

$$\begin{aligned} sd(p_1, p_2) &= (1 - |cred_{p_1} - cred_{p_2}|)\mathbf{OP}(\frac{\sum_{i=1}^{n} sm_i}{n}) \\ &= (1 - |1 - 1|).(\frac{0.7 + 0.7}{2}) \\ &\cong \mathbf{0.7} \end{aligned} \tag{1.7}$$

As foreseen, the search algorithm were able to realize both of the similarity relations and hence the evaluation algorithm had concluded the accurate similarity degree, which is 0.7 for the case.

Figure 1.7: Predicate trees for *nice_destination* and *touristic_place*

1.3.5 Credibility of Similarity Relations

The domain of interest can be enriched by computing not only the similarity value between predicates, but also the credibility value this similarity has via the introduction of the *fuzzy similarity predicates* with *fuzzy credibility values*.

In the previous research all fuzzy similarity relations are defined without credibility values as if we assume that the similarity computed was always the expected one. So the following type of notation,

$$Sim(good\_technique, swift\_shot) = 0.7$$

can be replaced by:

$$Sim(good\_technique, swift\_shot) \quad \xleftarrow{1.0} 0.7.$$

This extension could be handled by our algorithm with just a small modification in the evaluation equation. In the part where the similarity degree of the collected pairs'

are summed, we should normalize each of these values by the rule's credibility. Thus the algorithm is re-written as:

$$sd(p_1, p_2) = (1 - |cred_{p_1} - cred_{p_2}|)\mathbf{OP}(\frac{\sum_{i=1}^{n}(sm_i \times w_i)}{n}) \tag{1.8}$$

Example 1.3.5
Let's revisit Ex. 2.3., but this time suppose we are sure of the similarity relations with credibility values of 0.7 and 0.6 respectively.

$$\begin{array}{ll} classy\_restaurant: & (Restaurant) \\ good\_restaurant: & (Restaurant) \\ well\_trained\_waiters: & (Restaurant) \\ expensive\_inventory: & (Restaurant) \\ has\_good\_service: & (Restaurant) \\ has\_healthy\_food: & (Restaurant) \\ has\_tasty\_food: & (Restaurant) \\ has\_nice\_surroundings: & (Restaurant) \\ has\_high\_reputation: & (Restaurant) \end{array}$$

$$\begin{array}{lll} classy\_restaurant(X) & \overset{1.0,.}{\longleftarrow} prod & well\_trained\_waiters(X), expensive\_inventory(X). \\ good\_restaurant(X) & \overset{0.95,.}{\longleftarrow} prod & has\_healthy\_food(X), has\_good\_service(X), has\_tasty\_food(X) \\ & & has\_nice\_surroundings(X), has\_high\_reputation(X). \end{array}$$

$$\begin{array}{ll} Sim(well\_trained\_waiters, has\_good\_service) & \overset{0.7}{\longleftarrow} 0.9. \\ Sim(expensive\_inventory, has\_nice\_surroundings) & \overset{0.6}{\longleftarrow} 0.8. \end{array}$$

The only change in our search procedure is that, now the collected similarity values are stored together with their associated credibility value.

Thus the new result from the modified evaluation algorithms follows as:

$$\begin{aligned} sd(p_1, p_2) &= (1 - |cred_{p_1} - cred_{p_2}|)\mathbf{OP}(\frac{\sum_{i=1}^{n}(sm_i \times w_i)}{n}) \\ &= (1 - |0.95 - 1|).(\frac{(0.9 \times 0.7) + (0.8 \times 0.6)}{2}) \\ &\cong \mathbf{0.53} \end{aligned} \tag{1.9}$$

Note that the evaluated similarity value decreased to *0.53* from it's original value of *0.81*. This was expected as the similarity relations' credibility decreased to *0.7* and *0.6* from *1.0*, since they used to be treated as if they were facts.