An ontology similarity algorithm for BioAgent

Rosario Culmone, Gloria Rossi and Emanuela Merelli
Dipartimento di Matematica e Informatica
Università di Camerino
via Madonna delle Carceri
62032 Camerino, Italy
email:{rosario.culmone, gloria.rossi, emanuela.merelli}@unicam.it

Abstract

In the last years, the Web has been evolving, transforming itself in a *semantic Web* [1]. The new Web aims at guaranteeing almost completely automatic access to information sources, by introducing ontologies [6, 7, 4] and using mobile agents [8]. Ontologies, representing agents' knowledge, will allow integration of heterogenous resources to support global information systems. The use of ontologies in agent framework is not an easy task, especially as a measurement of ontological similarity. Although several ontology models [3, 2] and several measurements for semantics similarity have been presented in the literature, there is a need for more sensitive measurements which can provide a degree of similarity between concepts in order to perform semantic matching. Such measurements should take into account the structure of the concepts description and the relationships between concepts. Starting from the graph-oriented model proposed in ONION [10], the present work proposes an algorithm to assess the semantic similarity between two concepts. The resulting data structure is used to represent agent knowledge, while the similarity algorithm compares concepts placed on different agent platforms. The proposed algorithm has been designed for a biological domain and developed for BioAgent [12], a mobile agent platform for distributed biological applications.

Ontology model

A lexicon $\mathcal{L}$ consists of a finite set of semantically meaningful Concepts, denoted by $\mathcal{C}$ and a finite set of Relationships $\mathcal{R}$, i.e. $\mathcal{L} = \mathcal{C} \cup \mathcal{R}$.

An ontology is a formal specification of a shared conceptualisation, that is, the knowledge structure that describes the semantics of an information source by using a lexicon $\mathcal{L}$.

Formally, an ontology can be represented by an ontological graph $G^o = (N, A, \lambda, \delta)$, a labelled directed graph in which the set of nodes N represents concepts and the set of arcs $A \subset N \times N \times \mathcal{R}$ represents relationships between concepts. The univocal association of a node to a concept is given by the $\lambda : N \rightarrow \mathcal{C}$ mapping function, whereas the association of arcs to relationships is given by a $\delta : A \rightarrow \mathcal{R}$ mapping function. The δ function is neither injective nor surjective mapping, allowing for unexpressed relationships in the ontological graph.

For ease of notation and without missing any meaning, the ontological graph will be addressed by $G^o = (N, A)$ in the sequel to this paper.

BioAgent

A BioAgent is a mobile agent framework designed to support the definition of distributed applications in a biological domain. An application consists of one or more agents, properly created to elaborate several tasks such as the retrieval and integration of heterogeneous information. Any information source is interfaced by a BioAgent platform, whose software architecture is sketched in Figure 1.

The *Core Layer* contains features exposed to *service agents* such as security and resource access and features exposed to *agents* such as mobility, communication, creation-cloning and domain descriptors.

The *Service Agents Layer* consists of a community of agents have been created to support the access to services locally available. Actually, the BioAgent prototype provides four services of general and wide applicability: *Broker Agent*, *Web Interface*, *Wrapper* and *Ontology*. The broker agent is a special agent which keeps trace of each running service agents and provides to any new agent the list of all locally ac-

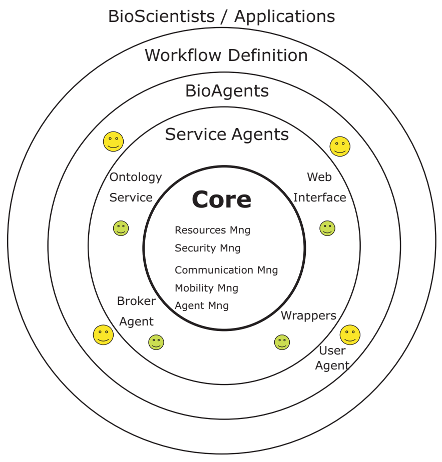

Figure 1: BioAgent Software Architecture

cessible services. The web interface service manages interactions between user and platform. The wrapper service allows access to local resources and the ontology service supports the use of global and local ontologies for semantically correct use of local data.

The *BioAgents Layer* concerns with the management of user agents.

The *Workflow Definition Layer* concerns the definition of a workflow in term of coordinated execution of multiple tasks or activities chosen by the user to perform an experiment.

Finally, the *BioScientists/Applications Layer* concerns the definition of distributed applications.

The sequel of the paper (Section 1) shows how to apply the graph-oriented ontology model to the BioAgent framework. Section 2 introduces the new similarity algorithm used to compare ontology's concepts in the *Ontology service*. Section 3 describes an application example of the use of the similarity algorithm in a biological domain.

1 Ontology and BioAgent

Any BioAgent platform utilizes a common lexicon $\mathcal{L}$ in terms of set of primitive concepts $\mathcal{C}$ and set of relationships $\mathcal{R}$, modelled by the same *primitive ontological graph* $G^o = (N, A)$. For the sake of simplicity, this first version of the ontology model allows only an a priori stated set $\mathcal{R}$ of relationships. As an example, let be $\mathcal{L} = \mathcal{C} \cup \mathcal{R}$ the set of concepts and relationships taken from a part of molecular biology ontology which has been developed for the TAMBIS project [5] and expressed in OIL [9, 11]. Let be $\mathcal{C}$ ={Protein, Ribozyme, RNA, DNA, Genomic-DNA, Nucleic-acid, Ribonucleotide, Deoxinucleotide, Nucleotide, Macromolecule} and $\mathcal{R}$ = {Subclass-of, Polymer-of, Transcribed-from, Translated-to}. Figure 2 shows the primitive ontological graph G^o representing concepts and relationships of that part of molecular biology ontology.

In the ontological graph, the set of concepts is completely represented by nodes, while any relationship is represented by a graph's arc only if there are two concepts to be related. In the specific example, the two relationships, Transcribed-from and Translated-to, are not represented. This modelling feature will allow an a posteriori verification of the existence of concepts that satisfy an hypothetical relationship.

Thus every platform can add new local concepts, widening its lexicon and obtaining a derived lexicon $\mathcal{L}_1 = \mathcal{C} \cup \mathcal{C}_1 \cup \mathcal{R}$, where $\mathcal{C}_1$ is the set of new concepts created over the set of existing relationships $\mathcal{R}$ and linked to primitive or derived concepts. Adding a new concept to the ontological graph corresponds to the creation of a new node to embed into the existing graph. Formally this means creating a new node n, and every arc a corresponding to a possible relationship with existing nodes. The *derived ontological graph* becomes $G^o_1 = (N \cup N_1, A \cup A_1)$, where N_1 is the set of new nodes and A_1 the set of new arcs. The derived ontological graph will represent the *glocal* (global plus local) formal knowledge managed by *Ontology Service* at the Service Agent Layer of BioAgent architecture.

Figures 3 and 4 show two ontological graphs derived from the primitive ontological graph G^o in Figure 2. The two derived graphs have been generated over two different platforms; G^o_1 as a consequence of the addition of concepts $\mathcal{C}_1$ ={Messenger-RNA, Complement-DNA} in the first platform, and G^o_2 as a consequence of the addition of concepts $\mathcal{C}_2$ ={mRNA, cDNA, Enzyme} in the second platform.

Since a bioagent is created to perform a set of tasks or activities by moving over a set of places; starting from a glocal knowledge of the domain of the first place, the agent will use G^o_1 for local concepts and G^o for the meaning of the common concepts, ensuring that the meaning remains the same for each place. After the agent completes his task in the place one, he will prepare to move to the second place and so on. Anywhere he arrives, he will attempt to establish the correspondence between the concepts known in the first place and those known locally.

Before leaving the first place, for any local con-

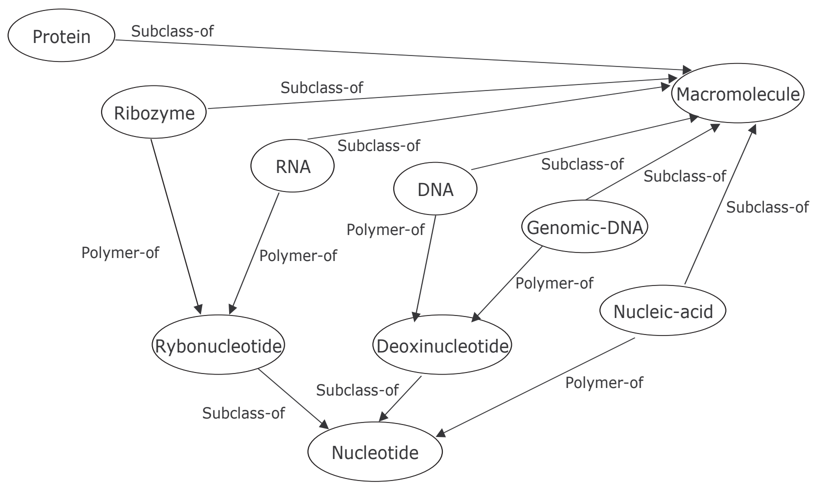

Figure 2: Primitive ontological graph G^o.

cept, $c \in \mathcal{C}_1$, the bioagent selects in G^o_1 the subgraph corresponding to the representation of the concept c. This subgraph represents knowledge the agent has acquired in the starting place and he which he will bring with him during the working journey. Generally, in th e kth place, given a derived ontological graph $G^o_k = (N \cup N_k, A \cup A_k)$ and a concept $c \in N_k$, the subgraph of G^o_k corresponding to the c concept is denoted as $S^c_k = (M_k, E_k, r_c)$, where $M_k \subseteq N \cup N_k$, $E_k \subseteq \{r_c\} \times M_k$ and $r_c \in N \cup N_k$.

Once in a new place, the bioagent must compare the subgraphs he brings with him, with those contained into the local derived ontological graph G^o_i representing the knowledge available in the platform i. In the example, moving from place 1 to place 2, the agent will compare the two subgraphs selected from G^o_1, corresponding to the concepts Complement-DNA, Messenger-RNA shown in Figure 5 with those contained in G^o_2 shown in Figure 6.

The similarity degree between concepts represented in the two graphs will be established by the following similarity function.

2 Similarity function

In this section, a function to asses the semantics similarity between concepts is proposed. Given the two concepts c and s, the similarity degree between c and s depends on the number of identical relationships existing with the same concepts.

Thus, the similarity degree between two concepts c and s locally expressed in places 1 and i respectively, represented by the two subgraphs $S^c_1 = (M_1, E_1, r_c)$ and $S^s_i = (M_i, E_i, r_s)$, will be assessed by the similarity function F. The function F is defined in terms of the function f. The function f is used to determine the existence of a given relationship, associated to concept c, in the set of relationships of the concept s, where the two concepts c and s are directly related to primitive concepts. The function f will be generalized to assess the similarity between concepts having relationships with derived concepts as well.

The function f can be more formally expressed as $f : E_1 \times r_s \to \{0, 1\}$. Given a generic arc $(r_c, j) \in E_1$ and the node r_s we define f as

$$f((r_c, j), r_s) = \begin{cases} 1, & \text{if } \exists k \in M_i | (r_s, k) \in E_i \wedge \\ & \lambda(j) = \lambda(k) \wedge \delta(r_c, j) = \delta(r_s, k) \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The similarity function $F : \{r_c\} \times \{r_s\} \to [0, 1]$ between two nodes r_c and r_s corresponding to the similarity between the two concepts c and s, is defined as

$$F(r_c, r_s) = \frac{\sum\limits_{j:(r_c,j)\in E} f((r_c, j), r_s)}{\tau(r_s) + \tau(r_c) - \sum\limits_{j:(r_c,j)\in E} f((r_c, j), r_s)} \tag{2}$$

where τ is the function used to determine the node cardinality of the set of outgoing arcs of a given node. Then, the similarity function F determines the similarity degree between two concepts c and s directly related to primitive ones.

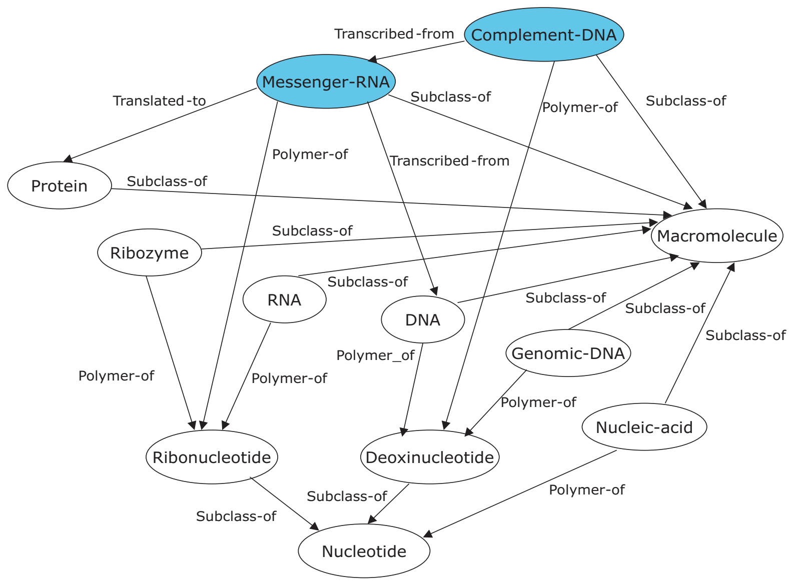

Figure 3: Derived ontological graph G_1^o on platform 1.

If we apply F to all concepts $s \in C_2$ directly related to primitive concepts, we will obtain the concept $\bar{s}$ most similar to c, taking that which corresponds to $F(r_c, r_{\bar{s}}) = \max\limits_{s} F(r_c, r_s)$.

Concepts c and $\bar{s}$ can be added to the primitive set of concepts with the associated similarity degree only for that platform, thus obtaining a new primitive set of nodes $N \cup \bar{N}$, where $\bar{N}$ contains all nodes corresponding to the new concepts added.

The next step is to redefine the function f in order to determine the similarity degree between two concepts c and s not only directly related to the primitive ones, but also related to the derived concept we have added with the associated similarity degree.

In this case we will have the two subgraphs $S_1^c = (M_1, E_1, r_c)$ where $M_1 \subseteq N \cup \bar{N}$ and $E_1 \subseteq \{r_c\} \times M_1$, and $S_i^s = (M_i, E_i, r_s)$ where $M_i \subseteq N \cup \bar{N}$ and $E_i \subseteq \{r_s\} \times M_i$.

We need only redefine the function f expressed in Equation (1) as

$$f((r_c, j), r_s) = \begin{cases} 1, & \exists k \in N \mid (r_s, k) \in E_i \wedge \\ & \lambda(j) = \lambda(k) \wedge \delta(r_c, j) = \delta(r_s, k), \\ & j \in N \\ & \\ F(j,k), & \exists k \in \bar{N} \mid (r_s, k) \in E_i \wedge \\ & \lambda(j) = \lambda(k) \wedge \delta(r_c, j) = \delta(r_s, k), \\ & j \in \bar{N} \\ & \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Where $\lambda(j) = \lambda(k), j, k \in \bar{N}$ is valid if the concept associated to j is the most similar to the concept associated to k. The function F remains exactly the same, and the computation proceeds as in the first example. Figure 7 contains the complete similarity algorithm.

A simple application of the algorithm to the example is described in the next section.

3 Application example

The example described in this section refers to the ontological graphs G_1^o and G_2^o of Figures 3 and 4.

A bioagent is created on platform 1 with the task of retrieving information about Complement-DNA. The bioagent must search platform 2 for the concept Complement-DNA defined on platform 1 (see Figures 5 and 6).

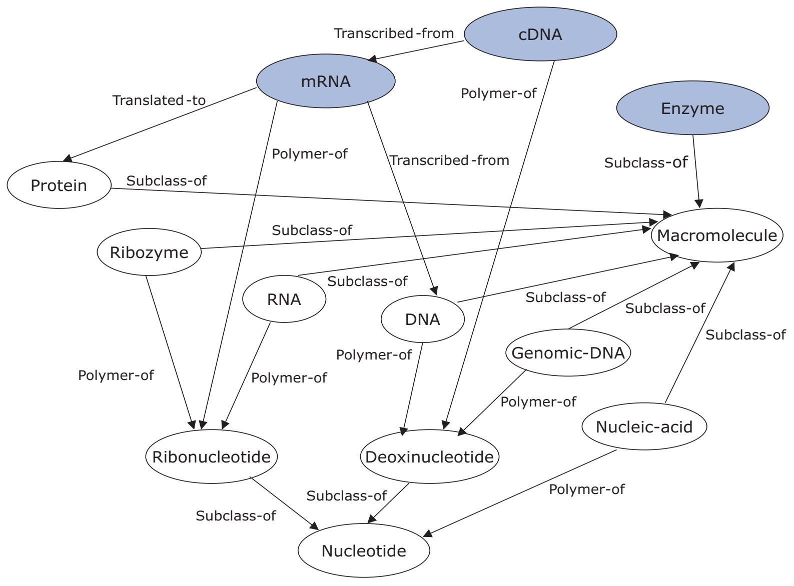

Figure 4: Derived ontological graph G_2^o on platform 2.

Nodes and arcs of the subgraphs are indicated by corresponding labels in the lexicon.

The agent extracts the subgraph of Complement-DNA (Figure 5) from the derived ontological graph G_1^o and searches for the concept on G_2^o.

Complement-DNA is linked to the derived node Messenger-RNA and to the primitive nodes {Deoxinucleotide, Macromolecule}. Messenger-RNA is directly linked with only the primitive nodes M={Protein, Ribonucleotide, DNA, Macromolecule}.

The concepts derived on platform 2 are $\mathcal{C}_\in$={cDNA, mRNA, Enzyme}. For each concept derived on platform 2, the agent extracts the corresponding (Figure 6).

The subgraph of Complement-DNA is compared to all these subgraphs and the agent compute F(Complement-DNA, mRNA) $=$ 0 and F(Complement-DNA, Enzyme) $= 0.333$.
To evaluate F(Complement-DNA, cDNA) the agent must search for the concept corresponding to Messenger-RNA. The agent extracts the subgraph corresponding to Messenger-RNA and repeats the comparison computing F(Messenger-RNA, Enzyme) $=$ 0.333 and F(Messenger-RNA, cDNA) $=$ 0 and F(Messenger-RNA, mRNA) $=$ 0.75, which yields mRNA as the concept most similar to Messenger-RNA, with a similarity degree of 0.75.

The agent is now able to complete the confrontation between Complement-DNA and cDNA, computing F(Complement-DNA, cDNA) $=$ 0.538, which yields cDNA as the concept most similar to Complement-DNA, with a similarity degree of 0.538.

The agent has thus obtained the similarity between Complement-DNA on platform 1 and cDNA on platform 2. He may now extract information about Complement-DNA from platform 2, having learned that cDNA on platform 2 could be Complement-DNA on platform 1.

4 Conclusions and Future Work

The model we have presented here is being installed on the BioAgent platform as an agent service to support user agents performing experimental tasks on Bioscientists' behalf. The ontology service will provide the following Java methods:

- public Graph getSubGraph(Node node)
- public Set getNoPrimitiveNodes()
- public NodeValue f(Edge e, Node n, Graph O2)
- public NodeValue F(Node node1, Graph O2)
- public class Similarity

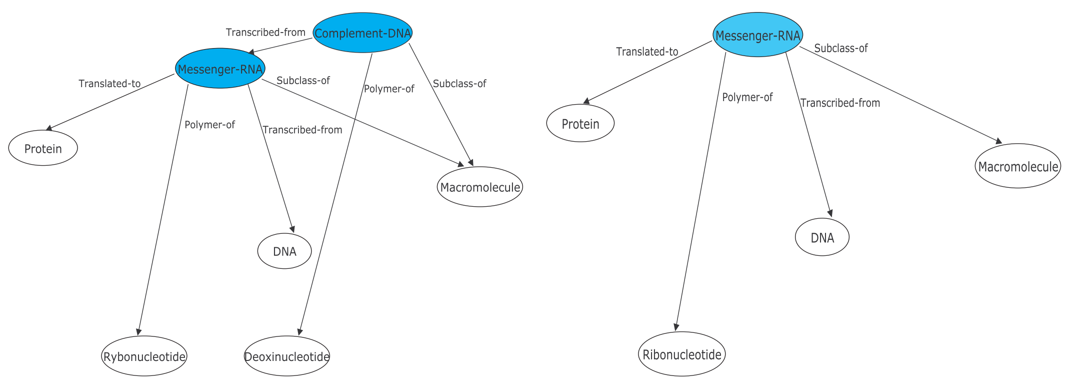

Figure 5: Subgraphs for Complement-DNA and Messenger-RNA concepts

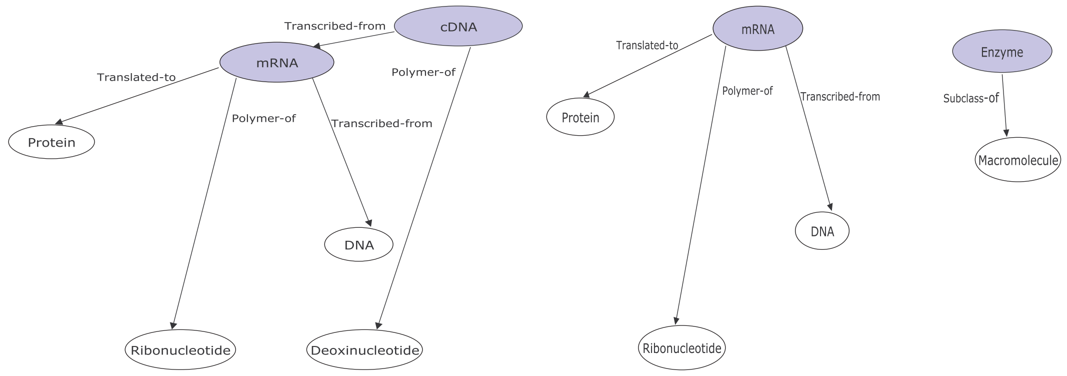

Figure 6: Subgraphs for cDNA, mRNA, Enzyme concepts

- public class Ontology

Future work concern testing and verification of the proposed algorithm as well as how reliable, congruent and correct the ontological model we have created is in representing knowledge of biological agents.

Acknowledgements

The authors would like to acknowledge the invaluable contributions of Leonardo Mariani.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[2] Sowa J. F.. Building, sharing, and merging ontologies. Technical report, 2001.

[3] A. Farquhar, R. Fikes, and J. Rice. The ontolingua server: A tool for collaborative ontology construction. Technical Report 96-26, Knowledge Systems Laboratory, Stanford University, September 1996.

[4] D. Fensel. *Ontologies: a silver bullet for Knowledge Management and Electronic Commerce.* Springer, 2001.

[5] C. A. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. G. Baker, M. Peim, and A. Brass. Transparent access to multiple bioinformatics information sources. *IBM Systems Journal*, 40(2):532–551, 2001.

[6] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report 93-04, Knowledge Systems Laboratory, Stanford University, 1993.

[7] N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N.J.I. Mars, editor, *Towards Very Large Knowledge Bases*. IOS Press, 1995.

[8] J. Hendler. Agents and the semantic Web. *IEEE Intelligent Systems*, 16(1):30–37, January/ February 2001.

[9] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. The ontology inference layer OIL. Technical report, Free University of Amsterdam, 2000.

[10] P. Mitra, G. Wiederhold, and M. L. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Extending Database Technology*, pages 86–100, 2000.

[11] R. Stevens. An ontology of molecular biology and the questions that can be performed on the resources containing data about molecular biology. http://img.cs.man.ac.uk/stevens/tambis-oil.html.

[12] The BioAgent project. http://www.bioagent.net/.

Given a concept $c \in \mathcal{C}_1$:

```
**Algorithm** FindSimilar($Concept : c$) **returns** $Concept$
    { $r_c$ = Node corresponding to the concept $c$ on platform 1}
    **begin**
        extractSubGraph($r_c$);
        for each not primitive concept $s \in \mathcal{C}_2$ on platform 2 **do**
            { $r_s$ = Node corresponding to the concept $s$ on platform 2}
            **begin**
                extractSubGraph($r_s$);
                Compare subgraphs $S_1^c$ and $S_2^s$: compute $F(r_c, r_s)$;
            **end**
        max = $\max_{s \in \mathcal{C}_2} F(r_c, r_s)$;
        **end**
    { $\bar{s}$ such that $F(r_c, r_{\bar{s}})$=max is the concept similar to $c$ on platform 2}
    **return** $\bar{s}$ such that $F(r_c, r_{\bar{s}})$=max;
    **end**
```

```
**Algorithm** extractSubGraph($Node : r_c$) **returns** $Subgraph$
    **begin**
        $S_k^c$ = Subgraph corresponding to concept $c$ on platform $k$-th;
        add the node $r_c$ to $S_k^c$;
        for each arc $(r_c, j)$ outgoing from $r_c$ **do**
            **begin**
                add the arc $(r_c, j)$ to $S_k^c$;
                add the node $j$ to $S_k^c$;
                **if**$j$ not primitive **then**
                  extractSubGraph($j$);
                **end**
        **return** $S_k^c$;
    **end**
```

```
**Algorithm** $F(Nodes : r_c, r_s)$ **returns** $value\ in$ $[0, 1]$
    **begin**
        for each arc $(r_c, j)$ outgoing from $r_c$ **do**
            sum = sum + f($(r_c, j), r_s$);
        $F$ = sum / (size(outgoingArcs($r_c$)) + size(outgoingArcs($r_s$))- sum)
        **return** $F$;
    **end**
```

```
**Algorithm** $f(Arc : a = (r_c, j), Node : r_s)$ **returns** $value\ in$ $[0, 1]$
    **begin**
        for each arc $h = (r_s, k)$ outgoing from $r_s$ **do**
            **if** $(\delta(a) = \delta(h)))$ **then**
                **if** $((\lambda(j) = \lambda(k))$ **then**
                    **if** $j, k$ primitive **then**
                        f = 1;
                **else if** $j, k$ not primitive **then**
                        $f = F(j, k)$;
            **else**
                 $f = 0$;
        **return** $f$;
    **end**
```

Figure 7: Similarity algorithm