Towards a Graph-based Data Model for Semantics Evolution

Xuhui Li[1,*], Dewang Sun[1], Mengchi Liu[2], Tieyun Qian[2], Feicheng Ma[1,*]
[1]School of Information Management, Wuhan University, Wuhan, China
[2]State Key Lab of Software Engineering, Wuhan University, Wuhan, China
[*]Corresponding Author

Abstract
Semantic information comes from the things being recognized and understood gradually, and thus it is often in evolution during the modeling process. Existing semantic models usually describe the objects and the relationships in an application-oriented way, which is unsuitable to reuse the schemas during the semantics evolution. In this paper, we propose a new graph-based semantic data model to overcome the limitation. SemGraph adopts a meaning-oriented approach to specifying the subjective view of the things and uses the certain meta-meaning relationships to build a graph-based semantic model. The model is simple but expressive, and is especially fit for the semantics evolution. We introduce the basic concepts and the essential mechanisms of the model, demonstrate its features with examples and typical cases of semantics evolution.

Keywords: Semantics Evolution; Semantic Data Model; Meaning-oriented Approach; Graph Data Modeling

Citation: Li, X., Sun, D., Liu, M., Qian, T. & Ma, F. (2017). Towards a Graph-based Data Model for Semantics Evolution. In iConference 2017 Proceedings, Vol. 2 (pp. 54-65). https://doi.org/10.9776/17010

Acknowledgements: This research is supported by the NSF of China under contract No.61272110, No.91646206, No.71420107026, No.61572376 and No.61272275 , and the China Postdoctoral Science Foundation under contract No. 2014M562070.

Contact: lixuhui@whu.edu.cn, fchma@whu.edu.cn.

1 Introduction

Semantic data model is always a central topic in knowledge and data engineering. Since the late 1970s, researchers have designed various models to specify semantic information in a way that computers can manage and manipulate. The semantic information of the things in the real world is often treated as the instantiation of various semantic concepts. These concepts are often interrelated by the certain meta-semantic relationships such as attribute function, aggregation, generalization/specialization, role, derivation, etc.(Hull & King, 1987; Peckham & Maryanski, 1988). Therefore, semantic information is recognized and understood with the concepts in which it is involved. How to present, organize and find the semantic information according to the concepts and their interrelationships is thus a fundamental issue for the data models to concern.

Early studies on semantic data modeling often explicitly or implicitly focus on describing the "entities" (or "objects" in some literatures) and the "relationships" among the entities, known as the Entity-Relationship models, and treat them as the "semantics" of the real world . The semantics embodied in the models is often presented as "attributes", i.e., the meaningful functions between the source and the target entities, and the "aggregation" which are essentially n-ary relationships of entities or subordinated "aggregation". The semantic models have been proposed following two philosophical approaches: one approach places an emphasis on explicitly type constructors especially the aggregates (Abiteboul & Hull, 1987), and the other stresses the use of the attributes(Hammer & McLeod, 1981; Shipman, 1981). The models such as enhanced ER modeling approaches (Thalheim, 2011) also adopt various ways of specifying the generalization/specialization of the concepts and the derived concepts.

The later studies are overwhelmingly affected by the rapid development of object-oriented paradigms, which mainly uses the attributes of the objects to describe the semantic relations to other objects(Atkinson et al., 1989; Cattell & Barry, 2000). Generally, these models often adopt taxonomical approach to classify the semantic information into various categories corresponding to various concepts, and use various semantic features such as the object identity, the aggregation, the generalization/specialization, the class hierarchies, the non-monotonic inheritance,etc., to describe and refine the concepts. Some early object-oriented models mainly concerned about the static aspects of the real world and normally required an object to be an instance of a most specific class. Then the literatures (Dahchour, Pirotte, & Zimányi, 2004; Liu & Hu, 2009), are more interested in describing the dynamic semantic features of the entities, e.g., transient roles under certain context, which is common in practical modeling.

Another kind of data models, the graph data models, attempt to overcome the limitations of the conventional models with respect to capturing the inherent graph structure of data appearing in application, such as hypertext or geographic information systems, where the interconnectivity of data is an important aspect(Angles & Gutiérrez, 2008). As the entities and their relationships are also the fundamental issues for these models, they are deeply influenced by the semantic data models, especially the object-oriented data models, and can present diverse semantic features using graphs or hyper-graphs(Hidders, 2003; Choudhury, Chaki, & Bhattacharya, 2006). Many of them can represent the attributes and aggregations, and some of them also adopt the inheritance and derivation. In fact, some semantic data models like IFO(Abiteboul & Hull, 1987) are designed as a graph-based one which can present most semantic features.

The recent decade has seen the popularity of NoSQL databases(datasets), e.g., the document databases and the graph databases. These databases(datasets) are to contain and manipulate the information involving more semantics than conventional relational databases, even though there is an insufficiency of the semantic modeling study for them. For the document databases managing the XML documents or JSON-like documents, the tags or the keys of the data elements are often regarded as semantic labels of the data, and the hierarchically organized data schema can also present the compound structure of the semantic concepts(Mani, Lee, & Muntz, 2001; Ahmed, Lenz, Liu, Robinson, & Ghafoor, 2008). On the other hand, the popular graph databases (Buerli & Obispo, 2012) use big graph datasets describing the entities and their relationships as the graph nodes and edges. However, these graph databases seldom deploy the aforementioned graph data models to present the semantic information(Angles, 2012). An exception is the RDF datasets deploying the triple-based attribute-oriented data models like RDFS(Brickley & Guha, 2014) to build the knowledge bases like the Semantic Web(Heath & Bizer, 2011).

The above mentioned data models are quite expressive and comprehensively cover the various aspects of semantic modeling. However, these models are often designed in an application-oriented way. That is, the semantic concepts and their interrelationships, especially the taxonomical relationships, are well-designed in a top-down process according to the application requests. The models usually adopt a pragmatistic way to describe the semantic information, but these descriptions are not easy to be reused for other applications. For example, in describing a person concept, the designers usually use the attributes like "name", "birth-date" and "occupation". However, such information is only used as the social labels of a person under certain context, e.g., social security, and thus it is not actually natural for a physicial person who is just a human being. In practical data modeling, the semantic information about the entities and the relationships are often in evolution. During the evolution, new semantic information is generated from the old one through various data operations, but the semantic manipulations in existing models cannot sufficiently support these evolution requests. This insufficiency severely hinders the reuse of the semantic information, as would be shown in the next section.

In this paper, we propose a new graph-based semantic data model named SemGraph (standing for Semantics Graph) to overcome the inadequacy. SemGraph is a data model which uses meaning-oriented schemas to present interrelated concepts in a graph structure. The contribution of the paper lies in two folds. On one hand, we propose a meaning-oriented approach to modeling semantics. Existing approaches almost focus on the objects and the relationships, whereas our approach treats the subjective meaning of the things as the major source of semantic information. Therefore, our approach can simplify the data structure by adopting some simple and essential meta-semantic relations among the meanings. On the other hand, we design a set of mechanisms to work with the meta-semantic relationships. They can efficiently present the semantic information in the evolution and can effectively reuse the existing semantic schemas and data.

The remainder of this paper is organized as follows. Section 2 uses a scenario to show the motivation of exploring a new approach to semantic modeling for semantics evolution. Section 3 introduces the SemGraph data model, including its basic concepts and the four meta-semantic relationships which are useful for presenting the structure and the features of the semantic information. Section 4 further discusses the issues of representing semantics using SemGraph with the major mechanisms in common semantics evolution. Section 5 concludes the paper.

2 Motivation

2.1 A scenario of semantic data modeling

Let us consider the evolution of the semantic information in a teaching scenario where the semantic information of the entities and the relationships is gradually enriched as shown in Figure 1 (a)-(j).

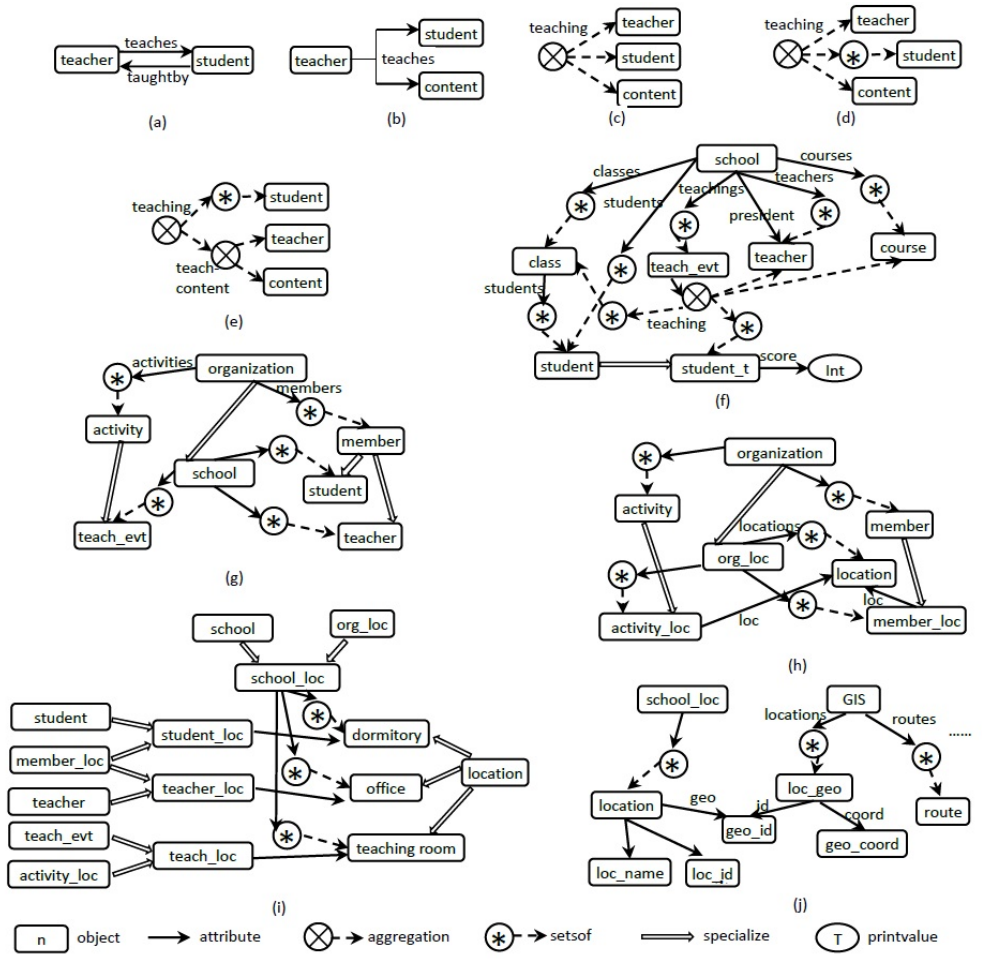

Figure 1: A scenario of modeling semantics evolution

In this scenario, the teachers and the students are initially treated as the people related to each other by the "*teaches*" and the "*taughtby*" relationships (Figure1 (a)). Then the teaching content is involved, and the figures (b) and (c) depict the attribute-based and the aggregation-based approaches to modeling the case. Since a teacher often teaches several students the same content, the 1:1 teaching relationship between the teachers and the students needs to be extended to 1:n as shown in (d). However, the current teaching relationship contains redundant couples of the teacher and the content which can be separated as a new nested relationship "*teaching content*", as shown in (e).

In a broader scope, a school, as shown in (f) is modeled to organize the teachers and the students and to separate the students into classes. Here a teacher plays the role of the president, and the courses are introduced as the official teaching content and a score is attached to a pair of student and course to assess his/her study.

At this stage, we find that the school is an organization consisting of the members, and an organization is often specialized to the one with some geographical locations for their members to work and to take certain activities like meetings or teachings, as shown in (g) and (h). Correspondingly, we need to consider the school again with the locations including the offices for the teachers, the dormitories for the students and the teaching rooms for the teaching events; after that, the school model would be associated with a geo-information model to do some geo-sensitive works, as in (i) and (j). A practical use of such an information system might be a request on the geo-information of a school, e.g., to seek a free teaching room for an optional course in the mid of a term. This task can be fulfilled by a certain geo-information sorting algorithm to choose the minimum expense of time for the students and the teacher to arrive at the destination teaching room from their dormitory rooms or office.

2.2 Limitations of conventional semantic data models

The above scenario might not be difficult to model for some semantic models if the complete knowledge and the application requests, e.g., the geo-information of a school and the teaching room selection, are known in advance. However, it is often not the case in practical application modeling. We usually only have partial information about the world in the beginning and cannot see the whole scene until the model is almost done (or even worse). A common request for the modeling approach is that they can provide a proper evolution mechanism which can keep up with the evolution of our recognition to the real world. That is, the model of the partial world can be reused as much as possible in the future. However, for the existing semantic models, there are some issues not properly addressed, as listed below, which hinder the models to be evolved and be reused in practice.

a. Extending attributes. The attributes denoting the binary relationships between the entities, such as "*teaches*" or "*taughtby*" in Figure 1-(a), often need to be extended to contain more information like the "*teaching content*" in Figure 1-(b)(c). This extension can be carried out by extending the attribute as the multi-valued one (as in (b)), which is adopted by some models like FDM(Shipman, 1981) or RDFS(Brickley & Guha, 2014). However, the inverse relationship like "*taughtby*" and the new relation introduced by the extension, e.g., the one between the student and the teaching content, are not easy to represent, and the possible further extension would make the model more complicated. Another solution is to use the aggregation approach to redesign the extended concepts (like in (c)). Both solutions severely restrict reusing the previous modeling result.

b. Projecting aggregations. The aggregations gather the related entities as n-ary relations. These relations, e.g., the teaching relation in Figure 1-(d), are formed in a pragmatistic way and often need to be adapted for new situations such as in Figure 1-(e). However, existing models usually provide the mechanisms, such as the subtyping or the nested aggregation, to extend or combine the aggregations. But they seldom provide a mechanism to project an aggregation, just like the projection operation in the relational model, into smaller parts which represent subordinate and original semantic units.

c. Representing context-dependent concepts. Many modeling approaches recognize the importance of the context-dependent concepts, i.e., the concepts which only has actual meaning in certain situations. Some early models like IFO use the specialization mechanism to specify the context-dependent concepts, meanwhile the common way in the recent studies is the role-based approach using the context-dependent roles associated with other entities. These two approaches focus on representing different aspects of the context-dependent concepts. Some concepts like a person being a teacher in a school are static and should be treated as context-dependent subtypes, and the others like a person being a president of a school are dynamic and should be treated as a role. However, existing models seem seldom use a consistent mechanism to integrate the two approaches.

d. Specifying logical dependency. The semantic concepts are usually dependent to one another logically, and this dependency is important in representing knowledge and searching latent semantic information. For example, from the 1:n teaching relationship in Figure 1-(c), the simpler 1:1 teaching relationship in (b) should be easily inferred. Such logic dependency should be checked by the model to guarantee the completeness of the semantic information. However, existing models seldom deploy proper mechanism to specify and check it. Some models allow users to manually specify the logical dependency as user-defined constraints, but these constraints are optional and the specification is legal either with or without the constraints. Therefore, a database with a sound model might contain incomplete data logically.

e. Specifying the generalization. As shown in Figure 1-(f) and (g), the specialization is often required to be dynamically specified during the modeling process, so as to enable the generalization of existing semantic types, e.g., generalizing the “organization” concept from “school”. However, the existing models often only allow the specialization of the concepts be statically declared in their definition, and thus the generalization of the concepts have to be restrained to the union of the concepts.

f. Presenting some important constructed types. Existing semantic models often support certain kinds of type construction, like the “union” and the “product” (through aggregation). The “intersection” and the “natural join” of the semantic types, which are common and important in building the links of semantics objects as shown in Figure 1-(i) and (j), are often presented through the multi-inheritance mechanism or the derived types of the product using the constraints to simulate the general join operation. However, the current multi-inheritance mechanisms focus on combining existing concepts, e.g., the “*member_loc*” and the “*teacher*” in Figure 1-(i), but seldom consider the cases of the intersection involved in other semantic modeling issues such as generalization and attributes. On the other hand, the instances of the derived types are apt to be affected by the update propagation of the joined values, but we are often just interested in the statically related objects sharing common attribute targets as in Figure 1-(j).

Due to the above inadequacy of the semantic models in presenting the evolving semantic information, we propose a new graph-based model named SemGraph to meet the requests of frequent semantics evolution.

3 The SemGraph Model

3.1 Basic concepts of SemGraph

SemGrpah adopts a meaning-oriented approach to semantic data modeling. In SemGraph we don’t directly describe the objective entities and relationships, instead we use a “meaning” to denote the subjective view in our mind when we think about a thing, e.g., an entity, a relationship, an event, etc. For example, to describe a school entity, we don’t consider what the school is, but dwell on the information about the school from a certain viewpoint, e.g., teaching or geo-information. A “meaning” is a (partial) understanding of a thing, and the related “meanings” from various aspects can form an overall and considerate understanding of the thing.

As the fundamental semantic units, the “meaning” are interrelated to model complex semantic information in a directed graph structure called semantics graph, as the name “SemGraph” indicates. In a semantics graph, the nodes represent the “meanings”, and the edges represent the meta-semantic relationships between the meanings. A meaning node has a label, e.g., “*a:teaching*”, denoting its meaning type and thus the node a is treated as an instance of the type “*teaching*”. A meaning type is specified in a schema indicating how its instance nodes should be linked with other nodes. For brevity, we often call a meaning type a “*meaning*” and a meaning node a “*node*”.

SemGraph allows four kinds of meta-semantic relationships: the ”*composition*”, the ”*specialization*”, the ”*reference*” and the ”*equivalence*” relationships respectively denoted by the symbols ”$\rightarrow$”, “$\Rightarrow$” ,”$\dashrightarrow$” and ”$\Leftrightarrow$” between pairs of nodes. Fig.2 depicts a semantics graph describing some simplified information about the school, the teaching, the teachers and the students.

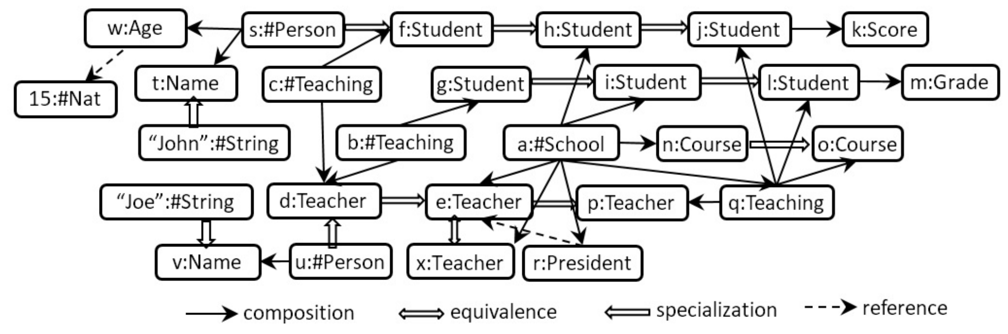

Figure 2: A semantics graph of simplified school information

In a “composition” relationship, e.g., “*b:#Teaching*$\rightarrow$*e:Teacher*”, the source node a represents a composite one and the target node b represents a component meaning (or component for short) node of a. The composite node can have one or more component nodes, e.g., there might be another relation "*b:#Teaching→ g:student*". The component *m'* of the meaning *m* is also denoted as *m.m'*. In the figure and the following description, the meaning "*m.m'*" is often abbreviated as "*m'*", e.g., "*e:Teacher*" is the abbreviation of "*e:#School.Teacher*".

In a "specialization" relationship, the target meaning specializes the source meaning. For example, in the relation "*s:#Person⇒f:Student*", the meaning "*Teaching.Student*" called the *sub meaning* is a specialized kind of the meaning "*#Person*" called the *super meaning*. The source node and the target node are thus respectively called the *super node* and the *sub node*.

In a "reference" relationship, the source meaning represents the reference to the target meaning and the target meaning node can be dynamically altered. For example, the reference relationship "*w:Age-->15:#Nat*" indicates that the age *w* should be a reference to the natural number *15*; and the relationship "*r:President-->e:Teacher*" indicates that the president role *r* be played by the teacher *e*.

The "equivalence" relationship indicates that the two meaning nodes be semantically "equivalent". For example, "*e:Teacher⇔x:Teacher*" indicates that the two nodes *e* and *x* represent the same "*teacher*" meaning of the same person. Such equivalence is common in object oriented programming, e.g., Java, where an *equals()* method is always provided for a class to specify how the two instances of the same class are equivalent to each other.

The following subsections would dwell on the basic features of SemGraph using the above meta-semantic relationships. In this paper the features are mainly introduced with the examples, please refer to (Li, 2016) for a more formal description of SemGraph.

3.2 Composition of meanings

SemGraph uses a simple but expressive composition mechanism to specify how the component meanings constitute a composite meaning. A meaning not being the component of another meaning is called a "root" meaning and is denoted with a "#" prefix. A component meaning can be composed of lower-level component meanings and thus forming a composition hierarchy. A semantic graph containing a composite node and all its descendant nodes in the composition hierarchy (maybe empty) is called a unit graph. It is required that the composition hierarchy be a directed acyclic graph, and so do the unit graphs. For example, the root meaning "*#School*" consists of the component meanings "*Teaching*", "*Teacher*", "*Student*" and "*Course*" meanwhile "*Teaching*" consists of the components "*Teacher*", "*Course*" and "*Student*".

The composition of the meanings indicates a semantical dependency, that is, the component meanings make sense only under the context of the composite meaning, and thus the composite node is also called the "context" of its component nodes. This feature makes SemGraph different from aggregation-oriented models where the components are loosely associated and do not depend on the aggregation. The semantic dependency is required to be determined by the users from the beginning stage of modeling and be statical during the semantics evolution, which enhances the reusability of the schemas.

For example, for the concepts "*Teacher*", "*Student*" and "*Teaching*" involved in specifying the activity "a teacher teaches a student", the meanings "*Teacher*" and "*Student*" depend on "*Teaching*" because they are two agents of the activity, and thus they should be the components of "*#Teaching*". However, for a school teacher, the "teacher" concept makes sense only in the school and thus it can be defined as the component meaning "*Teacher*" of the "*#School*" meaning.

The schema of a composite meaning is specified as the combination of the components with certain modifiers and occurrence operators. For example, the composition information of the "*#Teaching*" and the "*#School*" meanings are specified as follows:

```
meaning #Teaching -> {public Teacher, public Student}
meaning #School -> {Teacher[*], Student[*], Course[*],
                    Coursegrade -> {A | B | C | D},
                    Teaching -> { Teacher, Student[*]->{Score | Grade}, Course, Location?}  }
```

The components of a composite meaning can be modified as "local" and "public" indicating its sharable scope. A "public" component can be shared by multiple context nodes. For example, a public "*#Teaching.Teacher*" node can the component of multiple context nodes of "*#Teaching*". However, a "local" component which is often omitted on default, e.g., "*#School.Teacher*", only pertains to one context node.

SemGraph adopts a regular-expression-like mechanism to specify how the component nodes can occur in its context. It uses the conjunctive modifier ",", the disjunctive modifier "|", the optional modifier "?" and the multiple modifier "[*]" for the component meanings to indicate their instance meanings' occurrence. For example, in the composite meaning "*#School.Teaching*", the occurrence declaration "*(Teacher, Course)*" indicates that the teacher and the course should occur conjunctively once in the context, and "*(Score|Grade)*" indicates that either the "*Score*" or the "*Grade*" meaning occurs once, "*Location?*" indicates that a "*Location*" meaning would occur or not, and "*Student[*]*" indicates that one or more student meanings occur.

3.3 Specialization and reference

SemGraph deploys the specialization relationship to build the taxonomy of meanings and to enrich the sub meanings with more features. A sub meaning node would inherit all its super node's component nodes (if exists), meanwhile it can develop new component nodes which usually specialize other meaning nodes (including the inherited component nodes) allowed to access. The sub node is often relatively independent to its super node unless the super meaning being specified with a "!" symobl (to be introduced later), because a sub node just represents a subjective extension to the super node and thus the existence of the sub node should not affect the super node itself. A meaning schema often embeds the specialization relationship declaration into the composition and contains certain constraints to specify its features.

For example, the following schemas show the specialization version of "*#Teaching*" and "*#School*".

```
meaning #Entity -> {Name <= #String}
meaning #Person <= #Entity -> {Age <-- #Nat}
meaning #Teaching -> {public Teacher <= #Person, public Student <= #Person, public Content}
meaning #School -> {Teacher[*] <= #Person with {this.Age >= 18}, Student[*] <= #Person, ...,
                    Teaching -> {
                       Teacher <= #Teaching.Teacher & .Teacher,
                       Student[*] <= #Teaching.Student & .Student ->
                          {Score <= #Int | Grade <= .Coursegrade },
                       Course <= #Teaching.Content &.Course, Location? }
                       with { virtual #Teaching (-> $x, ->$y, ->$z) :-
                              this (-> Teacher <= $x:#Teaching.Teacher,
                                    -> Student <= $y:#Teacheing.Student,
                                    -> Course <= #Teaching.Content) } } }
```

As the example shows, the "*Teacher*" and the "*Student*" components of "*#Teaching*" are declared as the sub meanings of "*#Person*", and so do these components of "*#School*". In the "*#School.Teaching*" meaning, "*Teacher*", "*Student*" and "*Course*" are declared as the intersection sub meanings, i.e., the meaning indicating the intersection of multiple super meanings, of the corresponding components of "*#Teaching*" and "*#School*". For example, the specialization declaration "*Teacher ⇐ (#Teaching.Teacher & .Teacher)*" indicates that a "*#School.Teaching.Teacher*" component node has a super node of "*#Teaching.Teacher*" and a super node of "*#School.Teacher*" meanwhile the two super nodes should have a common "*#Person*" super node. The intersection meaning would be introduced in more detail in the next section.

The constraints of a meaning indicates how to validate it and its related meanings. The constraints are appended with a meaning using a "with" clause in the schema declaration. There are two kinds of constraints: one kind is the validating constraints which impose restrictions on the properties of its (inherited) components. For example, a "*#School.Teacher*" has a constraint rule "*this.Age >=18*", which means that its "*Age*" component value be not less than 18. The validating constraints enable specifying the derived meanings in SemGraph. A derived meaning is the sub meaning of an existing one. For example, a derived meaning "*big_school*" can be defined as the school has over 1,000 students. Due to the independency of the sub node to the super node, the derived meanings can be dynamically appended to the super node and be deleted if the derivation condition is no longer satisfied.

The other kind of constraints are to guarantee the completeness of the related meanings nodes. The so-called complementary constraints are in the form of "p_1 *:-* p_2" where p_1 and p_2 are two patterns indicating the specialization relationships among the nodes of the specific meanings. For example, in a "*#School.Teaching*" node, the component nodes "*Teacher*", "*Student*" and "*Course*" are assumed to be sub nodes of the certain "*virtual #Teaching*" nodes. These virtual nodes are not materialized in the graph, but they can be queried through the rewriting mechanisms derived from the constraint. Furthermore, the virtual "*#Teaching*" nodes involved in these component nodes are required to be complete, i.e., each virtual node have necessary materialized components in the graph to form a complete unit graph. For example, each component node belonging to a certain "*#Teaching*" node are covered in the constraint, and each virtual "*#Teaching*" node has its components fully determined by the constraint. The completeness of the virtual nodes and the associated unit graphs should be determined by the schema constraints. SemGraph deploys a certain checking mechanism to guarantee that the schema has been equipped with the proper complementary constraints to avoid the incomplete information(Li, 2016).

The reference relationship is a variant of specialization which allows the super node being altered. This is analogous to the reference types in the programming languages. Here the sub meaning and the super meanings are respectively called the "delegate" and the "host" meanings. A delegate meaning can inherit the components of the host, and these components can be the hosts of other components in the delegate. However, since the host is alterable, the delegate meaning should not put restriction on the host and its components in its constraints. but these components cannot be specialized as the new ones in the delegate meaning.

An important use of the reference relationship is to implement the role models. It has the following features which are exactly required by a role model: a delegate node can refer to mutable host nodes; the delegate meaning can be a composite one and be extended, and the extended component meanings only make sense under the context of the delegate meaning. For example, in a school the president is obviously a role and thus can be defined as the component meaning "*President→{Office, Telephone}*" of the "*#School*" meaning with the reference relationship "*President⇢Teacher*". Here the component meanings "*Office*" and "*Telephone*" only pertain to the president role rather than the teacher who plays the role.

The reference relationship can also be used to emulate the conventional O-O mechanism, as the mutable fields of an object can be treated as delegates and its values correspond to hosts. For example, the meaning "*#Person.Age*" can be declared as a delegate of the "*#Nat*", which means the age of a person refers to a mutable natural number value.

3.4 Equivalence

The equivalence relationship is actually an isomorphism between two unit graphs. It is useful both at the meaning schema level and at the semantic graph level. The equivalence between the schemas is used to declare two meaning types originated from different situations are equivalent to each other. Therefore, the equivalence declaration should specify the correspondence between the components of the two meanings.

The equivalence between the nodes in a semantic graph is often more interesting and important in semantics evolution. A meaning node can have partial information, i.e., lacking some components, during the semantics evolution. For example, when we try to resolve the case like "a school teacher teaches the course 101" in a semantic graph, we can build a "*#School.Teacher*" and a "*#Person*" meaning is created accordingly, but his/her "*Name*" and "*Age*" is not determined, and there might be another "*#Person*" node which is later determined to be equivalent to the former "*#Person*" node. To determine the equivalence between the nodes is an important semantic computation where recognizing and aligning the equivalent things are common in building semantic graphs in practice.

The equivalence relationships in semantic graph are not always physically represented as the equivalence edges. Sometimes they can be inferred based on the schemas and the nodes' components, which prevents the graph from being trivially complicated with all the equivalence or specialization edges. The meanings whose equivalence can be (recursively) inferred are called *intensional* meanings, which are universally used to denote the value types, e.g., integers and strings. For example, the "*#Person.Age*" meaning can involve numerous specialization edges from all the "*#Person.Age*" nodes to their "*#Nat*" nodes. Instead, the specialization edges nodes can be replaced with a certain value node of "*#Nat*", and the equivalences among the value nodes are determined through certain computations. The physical equivalence edges are thus transformed to the logical equivalence relations in data management.

4 Representing semantics using SemGraph

Based on the four meta-semantic relationships listed above and certain associated mechanisms, SemGraph can easily specify various kinds of semantic information and its evolution.

4.1 Dependency of meanings

SemGraph concerns the dependencies of meanings from both the context aspect and the logic aspect. Representing context-dependent concepts is straightforward in SemGraph. As mentioned previously, composite meanings are supposed to be the context of the components. Therefore, the components always represent the context-dependent concepts. These concepts can also be treated as the roles under the context. For example, for the case of a component "*Teacher*" of a composite meaning "*#School*" specializes a meaning "*#Person*", it indicates that a person play the role of teacher in a school. When the component's super meaning is alterable, i.e., the reference relationship is considered, the role becomes dynamic, as the case of the "President" component described previously.

Logical dependency is mainly specified by the specialization relationship and the related constraints, as shown in Section 3.3. A specialization relationship indicating the taxonomy of meanings represents a basic logical dependency between the meanings. Furthermore, the validating constraints and the complementary constraints provide practical mechanisms to specify the subtle logical relationships and the restrictions between the meanings. For example, the logical dependency between the 1:n teaching relationship in Figure 1-(c) and the 1:1 one in Figure 1-(b) is easily and completely addressed with the specialization and the constraints described in Section 3.3.

4.2 Type Construction

Constructing the compound semantic types to present complex meanings is the most important task for a semantic data model. Existing data models usually provide some common constructors like the product (or aggregation) and the group, meanwhile some models also try to use the preliminary generalization mechanism to implement the union type.

SemGraph can easily implement various type constructors, since the composition relation and the component occurrence mechanisms has already laid the foundation of the product(or aggregation), the group and the union types. SemGraph especially provides a product type constructor "ˆ", a group type constructor "*" and a union type constructor "||", to explicitly declare a compound meaning type of other root meanings. (A compound meaning is a special composite meaning generated by the constructors.) For example, the compound meaning "*#Schoolˆ#GIS*" consists of two components "*(#Schoolˆ#GIS).School*" and "*(#Schoolˆ#GIS).GIS*" which are respectively the sub meanings of the two root meanings "*#School*" and "*#GIS*", and the two components should occur conjunctively. For the compound meaning "**#School*", the component meaning "*(*#School).School*" should be declared to have multiple instances. For the compound meaning "*#School||#Institute*", it is the super meaning of "#School" and "#Institute" whose components should occur disjunctively in the union.

SemGraph does not support the compound meaning of the components of different root meanings, because the existence the components always assume the existence of their root meanings and thus the compound of their root meanings should be declared previously. However, the components under the same context can be composed with the type constructors.

Although the compound meanings can easily implement the functions of the conventional type constructors, they are different in their intension. The latter like the aggregation often represents the loose associations of the member entities or relations, and the member types don't depend on the compound semantic types. However, the former is declared as the context under which the components can make sense. Therefore we restrict the components of the product or group meaning to be the sub meanings of the original root meanings.

Intersection of semantic concepts is usually considered by the logic-based semantic models, e.g., the ontology models, and is seldom directly supported by the data models. Some data models can indirectly implement the limited features of the intersection through multiple inheritance. SemGraph introduces a special intersection compound meaning using an intersection constructor "&". For an intersection meaning

l_a*&*l_b, the meanings l_a and l_b are both the sub meanings of a certain meaning l, and l_a*&*l_b is automatically declared as the sub meaning of l_a and l_b respectively. The l_a*&*l_b meaning inherits the components of its super meanings (through a certain mechanism avoiding the name conflict). Recursively, under the context l_a*&*l_b the certain intersection component meanings l_1*&*l_2 can be declared where the meaning l_1 and l_2 are respectively the components of l_a and l_b and they are both the sub meanings of a component *l'* of *l*. At the instance level, a node of l_1*&*l_2 certainly shares the same super-node of *l'* as l_1 and l_2.

For example, for the meanings "*#School*" and "*#Institute*" which are both the sub meanings of "*#Organization*", the component meaning "*#Organization.Member*" is specialized as the meanings "*#School.Teacher*" and "*#Institute.Researcher*", and thus in the intersection "*#School&#Institute*" a new component "*Teacher&Researcher*" can be declared. In the semantic graph, if a node of "*#Organization*" has a sub node of "*#School*" and a sub node of "*#Institute*", they would have a sub node of "*#School&#Institute*" which not only inherits nodes of the components "*Teacher*" and "*Researcher*" but also has the new nodes of "*Teacher&Researcher*" which specialize the "*Teacher*" and the "*Researcher*" nodes being of the common "*Member*" super nodes.

To avoid the possible combinatorial explosion of the compound meaning types produced by the type constructors, SemGraph deploys a declare-to-use policy. That is, the compound meanings with the type constructors only make sense if they are explicitly declared in the schema. The nodes of a compound meaning are virtual on default, and the nodes of its user-defined sub meaning are physical in the semantic graph. Furthermore, since the intersection meaning only works as the two super meanings l_a and l_b have the common ancestor l, the amount of the intersection node and its component nodes can be restricted in the polynomial space.

With the aid of intersection, SemGraph can easily present the natural join operation. For the two meanings *m* and *m'* where the components *m.x* and *m'.y* share a common ancestor super meaning *z*, the natural join of *m* and *m'* through *m.x* joining *m'.y* is presented as the sub meaning of the product meaning *m^m'* where there exists a component *m.x&m'.y*, and thus is denoted as $m\hat{}m'_{m.x\&m'.y}$. Since *m.x&m'.y* shares the super meaning *z* with *m.x* and *m'.y*, the join meaning $m\hat{}m'_{m.x\&m'.y}$ can exactly combine *m* and *m'* using *m.x&m'.y* as the joint. For example, the case in Figure 1-(j) can be specified as the natural join of the meanings "*#School_loc*" and "*#GIS*" using a certain joint meaning "*#School_loc.location.geo_id & #GIS.loc_geo.geo_id*", assuming that the "*#GIS*" meaning be defined following the above guidelines.

4.3 Generalization

The specialization relationships between meaning nodes are often established as the sub nodes are instantiated. However, SemGraph also supports creating sub nodes from super nodes by declaring the specialization with the "!" symbol. The relationship "A ⇐ !B" means that for every node of the super meaning B, there does exist a node of the sub meaning A. This mechanism is used to declare generalization of existing meanings.

The following schema shows an example of declaring a meaning as the generalization of an existing meaning.

```
meaning #Organization -> { Member[*], Head --> .Member}
meaning #School_O <= !#School & #Organization -> {
            !.Teacher & .Member, !.Student & .Member, !.President & .Head}
```

In this example, an "*#Organization*" meaning is defined after the "*#School*" meaning. Then a new meaning "*#School_O*" is declared as the intersection meaning of "*#School*" and "*#Organization*", and especially, the components are specified as the intersection meanings of the corresponding component in the super meanings. Since the "*#School*" and its components are decorated with the "!" symbol in "*#School_O*", every node of "*#School*" is required to correspond to a node of "*#School_O*" and so do their components. As "*#School_O*" is also a sub meaning of "*#Organization*", the generalization of "*#School*" from the viewpoint of "*#Organization*" is actually formed because every component of "*#School*" corresponds to a sub component of the one in "*#Organization*".

4.4 Representing and extending attributes

Many conceptual modeling approaches use attributes to denote binary relationships between entities. In SemGraph, the attributes can be emulated easily using the generalization based on a binary relationship meaning "*#BiRel*". The following example specifies how to represent the "*#Teaching*" meaning as a binary relationship "*#Teaches*".

```
meaning #BiRel -> { First <= #Entity, Second <= #Entity }
meaning #Teaches <= !#Teaching & #BiRel -> { !.Teacher & .First, !.Student & .Second }
```

As the example shows, the "*#Teaching*" meaning corresponds to a special case, i.e., "*#Teaches*", of the generalized meaning "*#BiRel*". Therefore, "*#Teaches*" can be used as an attribute of "*#Teaching.Teacher*", and its inverse relationship "*#TaughtBy*" can be defined similarly.

Using the common specialization and the above generalization mechanisms, attributes can be easily extended. On one hand, to extend an existing attribute, e.g., "*#Teaches*", we can just extend the meaning with new component like "*Content*". On the other hand, to create a new attribute from the existing meaning, e.g., "student studies content", the attribute like "*#Studies*" can be specified with another "*!#Teaching & #BiRel*" where the intersection component meanings "*!.Student & .First*" and "*!.Content & .Second*" exist.

4.5 Projecting composite meaning

The components in a composite meaning sometimes have subtle interrelationships. The tightly-coupled components can form a new composite meaning which is a projection of the original composite meaning, as described previously. For example, in Figure 1-(e) the "teacher-content" couple is recognized as a new meaning and needs to be specified separately.

The projection of a composite meaning is also a variant of generalization. Since the components in the projection meaning have the same information as the ones in the original meaning, they need to share the content with the original components through the intersection meanings.

The following schemas show the projection example of Figure 1-(e).

```
meaning #Teaching -> {
            public Teacher <= #Person -> {ID <= #String},
            public Student <= #Person -> {ID <= #String},
            public Content -> {Name <= #String} }
meaning #Teach_content -> {
            public Teacher_p <= #Person -> {ID_p <= #String},
            public Content_p -> {Name_p <= #String}
meaning #Teaching_proj <= !#Teaching & #Teaching_content  -> {
            !.Teacher & .Teacher_p -> { !.ID & .ID_p }
            !.Content & .Content -> { !.Name & .Name_p} }
```

In this example, the intersection meaning of the components "*Teacher*" and "*Teacher_p*" exists and further their components "*ID*" and "*ID_p*" also have intersection meaning. Therefore, in a "*.Teacher&.Teacher_p*" node, the "*ID*" and the "*ID_p*" components specialize the same *#String* node, i.e., they have the same value. Similarly for the "*Content*" and "*Content_p*" components. As a "*#Teaching_proj*" node is mandatorily instantiated by its "*#Teaching*" super node , its "*#Teach_content*" super node should have the information according to the"*#Teaching*" node, which actually forms a projection of original node.

5 Conclusion

Semantic information comes from the things being recognized and understood gradually, and thus it is always in evolution during the modeling. In this paper, we have proposed a new graph-based semantic data model named SemGraph to overcome the limitations of existing semantic data models in semantics evolution. SemGraph deploys a meaning-oriented approach to specify the subjective view of the things and uses the certain meta-meaning relations to build a graph-based semantic model, which is simple but expressive, and especially fit for the semantics evolution.

Our work on the SemGraph model is in the preliminary stage. Based on the meta-semantic relationships and the mechanisms listed in the paper, we are studying the proper approaches to physically present and store the semantic graphs and the expressive mechanisms for presenting more complex semantics such as regular graph structures and temporal information. Further, we would introduce the new meta-semantic relations to represent the knowledge rules and the natural language semantics to build a knowledge base. A full-fledge query language is also under development to explore the graph of semantic information and extract meaningful results.

References

Abiteboul, S., & Hull, R. (1987). IFO: A formal semantic database model. *ACM Trans. Database Syst.*, *12*(4), 525–565. Retrieved from http://doi.acm.org/10.1145/32204.32205 doi: 10.1145/32204.32205

Ahmed, W. M., Lenz, D., Liu, J., Robinson, J. P., & Ghafoor, A. (2008). Xml-based data model and architecture for a knowledge-based grid-enabled problem-solving environment for high-throughput biological imaging. *IEEE Transactions on Information Technology in Biomedicine*, *12*(2), 226–240.

Angles, R. (2012). A comparison of current graph database models. In *Workshops proceedings of the IEEE 28th international conference on data engineering, ICDE 2012, arlington, va, usa, april 1-5, 2012* (pp. 171–177). Retrieved from http://dx.doi.org/10.1109/ICDEW.2012.31 doi: 10.1109/ICDEW.2012.31

Angles, R., & Gutiérrez, C. (2008). Survey of graph database models. *ACM Comput. Surv.*, *40*(1). Retrieved from http://doi.acm.org/10.1145/1322432.1322433 doi: 10.1145/1322432.1322433

Atkinson, M. P., Bancilhon, F., DeWitt, D. J., Dittrich, K. R., Maier, D., & Zdonik, S. B. (1989). The object-oriented database system manifesto. In *DOOD* (pp. 223–240).

Brickley, D., & Guha, R. (2014). Rdf schema 1.1. w3c recommendation (25 february 2014). *World Wide Web Consortium*.

Buerli, M., & Obispo, C. (2012). The current state of graph databases. *Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@ calpoly. edu*, 1–7.

Cattell, R. G. G., & Barry, D. K. (2000). *The object data standard: ODMG 3.0.* Morgan Kaufmann.

Choudhury, S., Chaki, N., & Bhattacharya, S. (2006). GDM: A new graph based data model using functional abstractionx. *J. Comput. Sci. Technol.*, *21*(3), 430–438. Retrieved from http://dx.doi.org/10.1007/s11390-006-0430-0 doi: 10.1007/s11390-006-0430-0

Dahchour, M., Pirotte, A., & Zimányi, E. (2004). A role model and its metaclass implementation. *Inf. Syst.*, *29*(3), 235–270. Retrieved from http://dx.doi.org/10.1016/S0306-4379(03)00029-2 doi: 10.1016/S0306-4379(03)00029-2

Hammer, M., & McLeod, D. (1981). Database description with SDM: A semantic database model. *ACM Trans. Database Syst.*, *6*(3), 351–386. Retrieved from http://doi.acm.org/10.1145/319587.319588 doi: 10.1145/319587.319588

Heath, T., & Bizer, C. (2011). *Linked data: Evolving the web into a global data space.* Morgan & Claypool Publishers. Retrieved from http://dx.doi.org/10.2200/S00334ED1V01Y201102WBE001 doi: 10.2200/S00334ED1V01Y201102WBE001

Hidders, J. (2003). Typing graph-manipulation operations. In *Database theory - ICDT 2003, 9th international conference, siena, italy, january 8-10, 2003, proceedings* (pp. 391–406). Retrieved from http://dx.doi.org/10.1007/3-540-36285-1_26 doi: 10.1007/3-540-36285-1_26

Hull, R., & King, R. (1987). Semantic database modeling: Survey, applications, and research issues. *ACM Comput. Surv.*, *19*(3), 201–260. Retrieved from http://doi.acm.org/10.1145/45072.45073 doi: 10.1145/45072.45073

Li, X. (2016). A meaning-oriented approach to semantic data modeling. *arXiv preprint arXiv:1609.03346*.

Liu, M., & Hu, J. (2009). Information networking model. In *Conceptual modeling - ER 2009, 28th international conference on conceptual modeling, gramado, brazil, november 9-12, 2009. proceedings* (pp. 131–144). Retrieved from http://dx.doi.org/10.1007/978-3-642-04840-1_12 doi: 10.1007/978-3-642-04840-1_12

Mani, M., Lee, D., & Muntz, R. R. (2001). Semantic data modeling using XML schemas. In *Conceptual modeling - ER 2001, 20th international conference on conceptual modeling, yokohama, japan, november 27-30, 2001, proceedings* (pp. 149–163). Retrieved from http://dx.doi.org/10.1007/3-540-45581-7_13 doi: 10.1007/3-540-45581-7_13

Peckham, J., & Maryanski, F. J. (1988). Semantic data models. *ACM Comput. Surv.*, *20*(3), 153–189. Retrieved from http://doi.acm.org/10.1145/62061.62062 doi: 10.1145/62061.62062

Shipman, D. W. (1981). The functional data model and the data language DAPLEX. *ACM Trans. Database Syst.*, *6*(1), 140–173. Retrieved from http://doi.acm.org/10.1145/319540.319561 doi: 10.1145/319540.319561

Thalheim, B. (2011). The enhanced entity-relationship model. In *Handbook of conceptual modeling - theory, practice, and research challenges* (pp. 165–206). Retrieved from http://dx.doi.org/10.1007/978-3-642-15865-0_6 doi: 10.1007/978-3-642-15865-0_6