퍼지 및 지능시스템학회 논문지 2007, Vol. 17. No. 1, pp. 123-129

Knowledge Conversion between Conceptual Graph Model and Resource Description Framework

Jin Sung Kim

School of Business Administration, Jeonju University
Hyoja-Dong 3-1200, Wansan-Ku, Jeonju, Jeonbuk, 560-759, Korea
Tel: +82-63-220-2932, Fax: +82-63-220-2787, E-mail: kimjs@jj.ac.kr

Abstract

On the Semantic Web, the content of the documents must be explicitly represented through metadata in order to enable contents-based inference. In this study, we propose a mechanism to convert the Conceptual Graph (CG) into Resource Description Framework (RDF). Quite a large number of representation languages for representing knowledge on the Web have been established over the last decade. Most of these researches are focused on design of independent knowledge description. On the Semantic Web, however, a knowledge conversion mechanism will be needed to exchange the knowledge used in independent devices. In this study, the CG could give an entire conceptual view of knowledge and RDF can represent that knowledge on the Semantic Web. Then the CG-based object oriented PROLOG could support the natural inference based on that knowledge. Therefore, our proposed knowledge conversion mechanism will be used in the designing of Semantic Web-based knowledge representation and inference systems.

Key Words : Conceptual graph, Inference, Ontology, PROLOG, RDF, Semantic Web

1. Introduction

The Semantic Web is recognized as a fabulous information repository, with a number of heterogeneous knowledge and information resources [4]. In recent years, therefore, Semantic Web technology and its applications have been given tremendous attention by technologist, businesses, and trend watchers. On the Semantic Web, the semantics of the documents must be explicitly represented through semantic metadata and ontology in order to enable semantic-contents-guided search. During the development of the Semantic Web-based ontology, the most important is the description of the knowledge and resources. But the existing keyword-based knowledge/resource representations do not take into account the knowledge sharing through the Semantic Web. Most of knowledge representations and inference systems expressed in one form usually cannot be directly incorporated or converted into other knowledge representation forms [1].

In Knowledge-Based Systems (KBS) applications, domain knowledge is encoded by a wide variety of knowledge representation languages for sharing and reusing [1]. In addition, sharing and reusing knowledge expressed in different representations becomes more important as the use of KBS continues to expand. Such the diversity of knowledge representation forms results in knowledge system developers choosing on formalism over others for a specific application.

In this paper, we will discuss the conversion of knowledge representation formats between two well-known knowledge representation standards: the Conceptual Graph (CG) and the Resource Description Framework (RDF).

Some former researches for CG and RDF pointed out that CG and RDF are closely associated in concepts, syntax, and semantics [1, 4]. CG and RDF are designed to represent domain knowledge, or systematic semantics for knowledge sharing and reuse. They are both graph-oriented models depicted by *nodes* connected with *arcs*: in CGs, *concept* nodes are connected by *conceptual relationship* arcs; in RDF, *resource* nodes are connected with *property* arcs [1].

The purpose of this study is to knowledge sharing and converting between knowledge models expressed in different representations CG and RDF. In this paper, therefore, we apply the CG programming paradigm and RDF/XML. In the experiment PROLOG+CG will provide a useful basis and conceptual view to illustrate and develop our ideas due to its simple semantics and widespread familiarity [5]. The ROLOG+CG is a *conceptual* and an *object-oriented extension* of PROLOG [3].

접수일자 : 2006년 11월 20일
완료일자 : 2007년 2월 2일

2. Research Background

2.1. Conceptual Graph

Conceptual Graph (CG) is a method of knowledge representation developed by Sowa [7] based on Charles Peirce's Existential Graphs and semantic networks of artificial intelligence (AI) [8]. CG has a direct mapping to and from natural language and a graphic notation designed for human readability. Therefore, CG models express meaning in a form that is logically precise, humanly readable, and computationally tractable. Many popular graphic notations and structures ranging from type hierarchies to entity-relationship or state transition diagrams can be viewed as special cases of CGs [8].

In the CG model the knowledge is divided into two parts: the *terminological knowledge* (*support*) which contains the "ground" vocabulary and the *assertional knowledge* which consist of a set of conceptual graphs built by means of the terminological knowledge [6]. The *support* provides the ground vocabulary used to build the domain knowledge base: the type of *concepts* used, the *instances* of these types, and the types of *relations* (*conceptual relationships*) linking the concepts. The concepts can be linked by means of relations and the support contains the set of *conceptual relation types*.

Agent (agnt) is an example of relation; it is a binary relation allowing one to link an *Action* to a *Germ* (which are both concept types). The third set of the support is the *set of individual markers*, which represent the instances of the concepts. For example, *Boy*, *Girl* or *Employee* can be an instance of *Person*. The generic marker (1 or more than that, noted *) is a particular marker referring to an unspecified instance of a concept [6].

A conceptual graph is composed of: (i) a set of *concept vertices* (noted in rectangles) which represent the entities, attribute, states, events; (ii) a set of *relation vertices* (noted in ovals) which express the nature of the relationship between concepts; (iii) a set of *edges* linking relation vertices to concept vertices; (iv) a *label* for each vertex or edge [6].

To make CGs machine-process-able, Conceptual Graph Interchange Form (CGIF) is employed for encoding knowledge represented by CGs into machine-readable character strings [1]. Another CG model representation is Linear Form (LF) which makes CGs more human-readable. The following is an example illustrating three CG forms to represent the meaning of "Jin is going to Seoul by train" [1]

(1) Display Form:

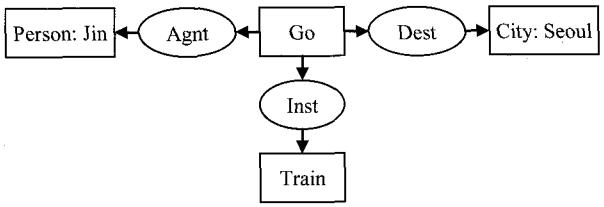

Fig. 1. A CG display form for "Jin is going to Seoul by train."

The linear form for CGs is intended as a more compact notation than Display Form, but with good human readability. It is exactly equivalent in expressive power to the abstract syntax and the Display Form. Following is the LF for Fig. 1:

(2) LF:

[Go] –
(Agnt)-> [Person: Jin]
(Dest) -> [City: Seoul]
(Inst) -> [Train].

For communication between machines, the CGIF has a simpler syntax and a more restricted character set. Following is the CGIF for Fig. 1:

(3) CGIF:

(Go [*x] (Agnt ?x [Person: Jin]) (Dest ?x [City: Seoul]) (Inst ?x [Train])

The CG has a similar syntax with LF. Following is the PROLOG+CG form for Fig. 1:

(4) CG:

[Go] -
- Agnt -> [Person: Jin],
- Dest -> [City: Seoul],
- Inst -> [Vehicle: Train].

2.2. Resource Description Framework

Resource Description Framework (RDF) is a general purpose knowledge representation framework and RDF schema (RDF-S) is the RDF vocabulary description language to be used to provide a further description mechanism to define classes or groups of related resources and the relationships between the resources [1]. RDF developed by W3C is based on an underlying model with triples made of *resource*, *property*, and *value* [4].

- A *resource* is an entity accessible by an URI on the Semantic Web (e.g. an XML document). Resources are the elements described by RDF statements.

- A *property* defines a binary relation between resources and/or its atomic values. A property enables us to attach detailed information to resources, and provide descriptions for resources.

- A *value* can be either a character string or a resource. Reification of resources using property and values enables us to transform the tripe into a resource.

RDF is also graph model that is depicted as a directed labeled graph.

An RDF *statement* specifies a *value* for a *property* of a *resource*. The *statement* is represented by a node for the *subject*, a node for the *object*, and an arc for the *predicate* that is directed from the subject node to the object node. Each statement in RDF is a triple corresponding to a single arc with a beginning node and an ending node. The principle idea of RDF is that all the things are naturally described by *object-attribute-value* triples. In the triple, *resources* are described using *properties* which have *values*. One characteristic that that distinguishes RDF from other knowledge representation language is that RDF uses Uniform Resource Identification (URI) references as 'resources' or 'properties' [1].

In order to be machine-process-able, RDF employs eXtensible Markup Language (XML) as its normative interchange syntax. Therefore, RDF has an XML syntax and can be seen as an object-oriented formalism for metadata statements. These metadata can rely on common ontology represented using RDF Schema (RDF-S). The vocabulary used in RDF triples can be defined using RDF-S, by a hierarchy of classes and a hierarchy of properties. RDF-S offers three core classes [4]:

- *Resource* (class of all objects),
- *Property* (class of all properties),
- *Class* (class of all classes).

RDF-S defines type systems similar to the type systems of object-oriented programming languages define the classes in terms of the properties that the instances of the classes [1]. Contrary to object-oriented and/or frame-based knowledge representation languages, however, properties are defined globally and not encapsulated in class definitions. They can be specialized using the subPropertyOf relations. For example, rdfs:Class and rdfs:Property are used to define the class of resources that are RDF classes and/or RDF properties; and rdfs:subClassOf and rdfs:subPropertyOf are used as properties to define relationship between RDF-S classes.

Here the example is used and represented in RDF and RDF-S. An RDF file must declare namespace to describe statements and description or resources.

Header of the RDF file is as follows:

<rdf:RDF xml:lang="en"
xmlns: rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns: rdfs ="http://www.w3.org/2000/01/rdf-schema#">

Fig. 2 shows an RDF representation in graph format.

(1) Graph format:

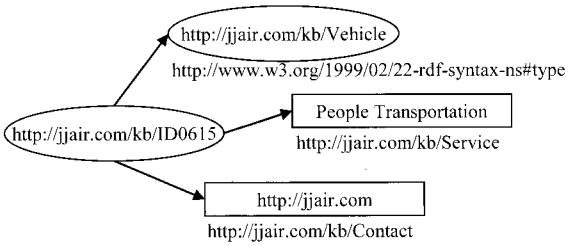

Fig. 2. An RDF represented in graph format

The following is an example of RDF/XML statement/ instance to be represented.

(2) RDF/XML format:

```
<? xml version = 1.0 ?>
<rdf:Description ID="AirService">
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:vehicle = "http://jjair.com/vehicle">
<rdfs:Class rdf:about = "http://jjair.com/vehicle/Airplane" />
<rdf:Property rdf:about = "http://jjair.com/vehicle/Airplane
  /Service" />
<rdf:Property rdf:about = "http://jjair.com/vehicle/Airplane
  /Contact" />
<vehicle:Airplane rdf:about = http://jjair.com/vehicle/ID0615>
   <vehicle:Service>People Transportation </vehicle:service>
   <vehicle:name>Boeing 747</vehicle:name>
   <vehicle:Contact>http://jjair.com</vehicle:Contact>
</vehicle:Airplane>
</rdf:Description>
</rdf:RDF>
```

The above two representations have equivalent meaning. Then the triple format for Fig. 2 is as follows:

(3) Triple format:

<http://jjair.com/vehicle/ID0615><http://jjair.com/vehicle/Service>"People Transportation"
<http://jjair.com/vehicle/ID0615><http://jjair.com/vehicle/name>"Boeing 747"
<http://jjair.com/vehicle/ID0615><http://jjair.com/vehicle/Contact>"http://jjair.com"

2.3. CGs and RDF conversion

Both CGs and RDF are graph models that include nodes connected by arcs, where nodes represent entities, and arcs represent relationships between entities. Although designed for different purposes, RDF models and CG models are both graph models and are very similar [1, 6]. The model of CG formalism is based on (1) a support made of a concept type lattice and of a relation type set, a set of individual markers enabling the designation of instances, a conformity relation between markers and types, and (2) a base of conceptual graphs built on this support. It therefore seems natural to translate a) the RDF statements into a base of CG-facts b) the hierarchy of classes appearing in an RDF-S into a concept type hierarchy in CG, and c) hierarchy of properties appearing in a RDF-S into a relation type hierarchy in CGs. From the knowledge representation language point of view, CGs and RDF are both designed as simple data models to derive formal semantics from the abstract syntax represented by the graphs, and to provide a basis for context-based semantic reasoning. Therefore, it is possible to convert CGs to RDF while retaining semantics, and vice versa. We researched the available software on manipulating CGs and RDF/XML, and found that the two Java API-based tools Jena [9] and NOTIO [10]. Then the CPE, CG Mars Langer, PROLOG+CG, and CoGITaNT [2] also have been used to construct some successful CGs and RDF/XML applications.

Corby et al. [4] proposed an approach for translating RDF to CG. In this study, they implemented a prototype using the NOTIO CG platform and the VRP RDF parser from ICS Forth [6]. Yaho and Etzkorn [1] developed an automated CG to RDF converter. Since their propose is to propose a cooperative ontology for translating knowledge model represented in CGs into knowledge models represented in RDF, they incorporated NOTIO, the CGs' handling API, and the Jena, the RDF handling API, into their automatic converter: NOTIO packages are use to parse CG models in CGIF and build CG models in memory; and Jena packages are used to build RDF models and string the models into RDF/XML after the models being transferred.

In this study, to propose a knowledge conversion mechanism for CG-RDF/XML converting, we will use the PROLOG+CG, the CGs' handling API.

3. Transformation of CG and RDF

In this section, we will describe a framework that performs a conversion of CGs to RDF (and RDF to CGs) models and discuss the experimental results attained by using the framework to translate some typical examples.

The similarity between the CGs and the RDF makes it possible to convert a knowledge representation in the CG model to a knowledge representation in the RDF model, and vice versa. A basic RDF statement says something like: 'The producer of the resource found at http://jj.shmall.com/ID9027 is SJEC.' it can be stated as a triple by this way:

producer("http://jj.shmall.com/ID9027", "SJEC")

Several statements can be written about the same resource, for example:

product("http://jj.shmall.com/ID9027", "PDPTV-A100")
date("http://jj.shmall.com/ID9027", "20061015")

(1) Graph format:

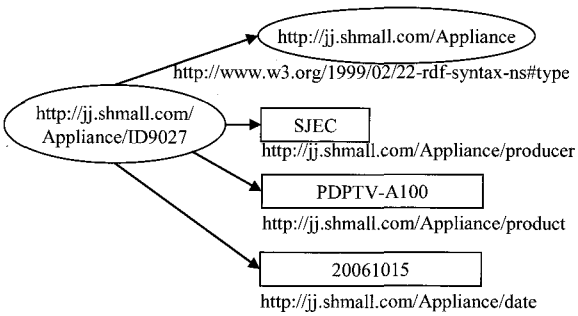

Fig. 3. A graph format for the product ID9027

Written with the RDF/XML syntax:

(2) RDF/XML format:

```
<rdf:Description about = "http://jj.shmall.com/ID9027">
  <producer>SJEC</producer>
  <product>PDPTV-A100</product>
  <date>20061015</date>
</rdf:Description>
```

This can be interpreted in PROLOG+CG as:

(3) CG format:

[Resource: "jj.shmall.com/Appliance/ID9027"]-
- producer -> [Literal: SJEC],
- product -> [Literal: "PDPTV-A100"],
- date -> [Literal: "20061015"].

The principle of the mapping from RDF to CG relies on considering an RDF description as an instance of a resource CG concept type and the associated properties as relations of this concept. The designator of a resource is the URI of the resource itself [6].

To show the mapping of nested RDF descriptions, we can express that the value of the 'company' property is itself a description of another resource:

```
<rdf:Description about = "http://jj.shmall.com/ID9027">
<product>PDPTV-A100</product>
<producer>SJEC</producer>
<company>
<rdf:Description about = "http://jj.shmall.com/SJEC">
    <industry>Appliance</industry>
 </rdf:Description>
</company>
</rdf:Description>
```

The RDF description will be translated into the following PROLOG+CG:

```
[Resource: "jj.shmall.com/ID9027"] -
- product -> [Literal: "PDPTV-A100"],
- producer -> [Literal: SJEC],
- company -> [Resource: "jj.shmall.com/SJEC"],
- industry -> [Literal: Appliance].
```

The previous RDF description can be typed as a description of a goods, the class of Goods being itself defined in an RDF-S.

```
<rdf:RDF
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns: rdfs = "http://jj.ac.kr/kb/iccs#">
   <ns:Goods rdf:about = "http://jj.shmall.com/ID9027">
     <ns:producer>SJEC</ns:producer>
       <ns:product>PDPTV-A100</ns:product>
</ns:Goods>
</rdf:RDF>
```

The next is an example mapping a CG concept type hierarchy to an RDF-S class hierarchy.

CG: Customer < Person

RDF:

```
<rdfs:Class rdf:about = "http://jj.ac.kr/kb/concepts/Person" />
   <rdfs:Class rdf:about = "http://jj.ac.kr/kb/concepts/
   Customer">
<rdf:subClassOf rdf:resource = "http://jj.ac.kr/kb/concepts/
   Person" />
</rdfs:Class>
```

The following example shows the knowledge representation and inference by using CGs and RDF/XML. It contains the information for buyers and sellers located in two different countries.

(1) Knowledge representation in CGs:

```
Universal > COMPANY, COUNTRY, PRODUCT, FOUND.
COMPANY > BUYER, SELLER.
SELLER = "SJEC".
SELLER = "KGSE".
BUYER = "UGM".
COUNTRY = "KOREA".
COUNTRY = "USA".
PRODUCT = "ID9027".
```

```
[SELLER: x]<-MEMB-[COUNTRY: "KOREA"] :-
  [COMPANY:x]<-AGNT-[FOUND]-LOC->[COUNTRY:
"KOREA"].

[SELLER: x]<-MEMB-[COUNTRY: "KOREA"] :-
   [COMPANY: x]<-OEM-[COMPANY: y],
   [SELLER: y]<-MEMB-[COUNTRY: "KOREA"].

[SELLER: x]<-MEMB-[COUNTRY: "KOREA"] :-
   [COMPANY:x]<-RCPT-[MERGE]-LOC->[COUNTRY:
"KOREA"].

[BUYER: x]<-MEMB-[COUNTRY: "USA"] :-
   [COMPANY: x]<-AGNT-[FOUND]-LOC->[COUNTRY: "USA"].

[BUYER: x]<-MEMB-[COUNTRY: "USA"] :-
   [COMPANY:x]<-RCPT-[ACQUISITION]-LOC->[COUNTRY:
"USA"].

[COMPANY: "SJEC"]-
   -SUBSD->[SELLER: "KGSE"],
```

<-AGNT-[FOUND]-LOC->[COUNTRY: "KOREA"].

[PRODUCT: "ID9027"]-MADEBY->[SELLER: "SJEC"].

[COMPANY:"KGSE"]<-AGNT-[FOUND]-LOC->[COUNTRY: "KOREA"].
[COMPANY:"UGM"]<-AGNT-[FOUND]-LOC->[COUNTRY: "USA"].

The following statement shows the queries and results executed by PROLOG+CG.

(2) Inference results:

?- [COMPANY: x]<-MEMB-[COUNTRY: y].

{x = "SJEC", y = "KOREA"}
{x = "KGSE", y = "KOREA"}
{x = "UGM", y = "USA"}

?- [SELLER: x]<-MEMB-[COUNTRY: y].

{x = "SJEC", y = "KOREA"}
{x = "KGSE", y = "KOREA"}

?- [BUYER: x]<-MEMB-[COUNTRY: y].

{x = "UGM", y = "USA"}

?- [PRODUCT: x]-MADEBY->[SELLER: y].

{x = "ID9027", y = "SJEC"}

?- [PRODUCT: y]-MADEBY->[COMPANY: "SJEC"].

{y = "ID9027"}

The following RDF/XML statement shows the sample conversion of the above CGs.

(3) RDF/XML

```
  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/COMPANY" />
  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/COUNTRY" />
  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/PRODUCT" />
  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/FOUND" />
  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/BUYER">
  <rdfs:subClassOf rdf:resource =
"http://jj.shmall.concepts/kb/relations/COMPANY" />
  </rdfs:Class>

  <rdfs:Class rdf:about =
"http://jj.shmall.concepts/kb/concepts/SELLER">
   <rdfs:subClassOf rdf:resource =
"http://jj.shmall.concepts/kb/relations/COMPANY" />
  </rdfs:Class>

  <rdf:Statement>
   <rdf:object>
    <concepts:FOUND rdf:about =
"http://jj.shmall.concepts/kb/concepts/any/FOUND" />
   </rdf:object>
   <rdf:subject>
    <concepts:SELLER rdf:about =
"http://jj.shmall.concepts/kb/concepts/any/COMPANY" />
   </rdf:subject>
   <rdf:predicate rdf:resource =
"http://jj.shmall.concepts/kb/relations/AGNT" />
  </rdf:Statement>

  <rdf:Statement>
   <rdf:object>
    <concepts:FOUND rdf:about =
"http://jj.shmall.concepts/kb/concepts/any/FOUND" />
   </rdf:object>
   <rdf:subject>
    <concepts:SELLER rdf:about =
"http://jj.shmall.concepts/kb/relations/LOC" />
   </rdf:subject>
   <rdf:predicate rdf:resource =
"http://jj.shmall.concepts/kb/concepts/any/COUNTRY" />
  </rdf:Statement>
... .
```

4. Conclusions and future work

In this study, we have presented an approach towards conversion of CGs and RDF/XML. We have also illustrated our point using CG and RDF/XML based knowledge representation framework. Therefore, we found it is true both theoretically and experimentally that "there is a huge overlap, making two technologies (CGs and RDF) very comparable and inter-workable" [1, 6]. Our knowledge conversion approach takes advantages of the CG formalism [1, 4, 6], Object-oriented Programming, and Logic Programming [3]. Our experience

with PROLOG+CG and RDF/XML shows this proposal is feasible, and a similar strategy should apply to any knowledge representation ontology.

In the future, the challenge is to study a logic programming and inference based on PROLOG+CG and OWL. Since OWL is used as an open standard for developing large scale ontologies on the Semantic Web, in our future research we plan to study an intelligent query language for OWL statements and the mapping to appropriate CG models.

References

[1] Yao, H. and Etzkorn, L., Automated conversion between different knowledge representation formats, *Knowledge-Based* Systems, 19, 404-412, 2006.

[2] Sowa, J.F., Conceptual graph standard, working document: ISO/JTC1/SC 32/WG2 N 000, 2001-04-02, http://www.jfsowa.com/cg/cgstand.htm, 2001.

[3] Kabbaj, A. and Petersen, U., PROLOG+CG version 2.0 user's manual, http://prologpluscg.sourceforge.net, 2006.

[4] Corby, O., Dieng, R., Hébert, C., A conceptual graph model for W3C resource description framework, *Proceedings of the 8th International Conference on Conceptual Structures (ICCS '2000)*, Berlin, August 2000.

[5] Loke, S.W., Declarative programming of integrated peer-to-peer and Web bases systems: the case of Prolog, *The Journal of Systems and Software*, 79, 523-536, 2006.

[6] Dibie-Barthélemy, J., Haemmerlé, O., and Salvat, E., A semantic validation of conceptual graphs, *Knowledge-Based Systems*, 19, 489-510, 2006.

[7] Sowa, J.F., Conceptual structures: Information processing in minds and machines, Addison-Wesley, Reading, MA, 1984.

[8] Kayed, A. and Colomb, R.M., Using BWW model to evaluate building ontologies in CGs formalism, Information Systems, 30, 379-398, 2005.

[9] Jena, A Semantic Web framework for Java, http://jena.sourceforge.net, accessed Sep. 11, 2006.

[10] Southey, F. and Linders, J.G., Notio – A Java API for developing CG tools, Proceedings of the 9th International Conference on Conceptual Structures (ICCS '99), Springer-Verlag, 1999.

저 자 소 개

김진성(Kim, Jin Sung)

김진성 (金珍成)은 현재, 전주대학교 경영학부 조교수로 재직 중이다. 주요 관심분야는 퍼지이론과 인공지능 기법을 이용한 지능형의사결정지원과 시멘틱 웹 서비스 등이다.

TEL : 063-220-2932
FAX : 063-220-2787
E-mail : kimjs@jj.ac.kr