Petri Net-Based Ontology Analysis Method for Indoor Location-Based Service System

Jaegeol Yim, Jaehun Joo and Gyeyoung Lee

Dongguk University at Gyeongju Korea
{yim, givej, lky}@dongguk.ac.kr

Abstract

This paper presents a Petri net based analysis of ontology for indoor location-based services. The proposed method builds up a Petri net model for the given ontology at the first step. Then, it analyzes the Petri net model by running the simulation on it. The proposed method of building a Petri net model for the given ontology is given in the paper. Then algorithms of performing inferences on the Petri net are proposed. Our proposed method is applied on a simple ontology for a sample Indoor Location-based service.

Keywords: *Ontology; Petri net; Location-Based Service; Indoor LBS; Inference*

1. Introduction

This paper introduces a method that can transform an RDF (Resource Description Framework) model into a colored Petri net (CPN) model. Performing a simulation on the CPN, we can obtain the answers of an RDF query. The study demonstrates the practicality of the proposed method by introducing a simple ontology for an indoor location-based service described in the RDF and showing how the proposed method can be used in the moving object database system's query processing for the service.

The RDF is a family of World Wide Web Consortium (W3C) specifications that can be used as a general method of modeling information. An RDF model consists of an RDFS (RDF Schema) layer and an RDF layer. The vocabulary used in RDF models is defined in the RDFS layer. A user defines the vocabulary, specifies the properties to be applied to each object, and describes the relationships between objects in the RDFS. That is, a user represents his knowledge in the RDFS. This implies that a well-formed RDF model is a type of ontology. The basic component of an RDF layer is a statement consisting of a subject, a predicate, and an object. A statement expresses a known fact.

The Petri net was first introduced in 1962, and since then, it has been widely used in computer science and engineering for system performance tests, communication protocol consistency and validation tests, and so forth. The Petri net has been popular because constructing a Petri net is easy and because mathematical methods of analyzing Petri net models are readily available [1].

The colored Petri net is a modified Petri net; it was developed so that the explosive increment of the size of the Petri net can be avoided when a complex phenomenon is modeled. There are several software tools that can provide convenient environments for constructing a Petri net model and for performing simulations. Among such tools, CPNTools [2] is designed to run on a desktop computer. The present paper proposes a method that can transform an RDF model into a Petri net model. Applying the proposed

method, we build a sample CPN for an RDF model and perform simulations using the model to answer RDF queries.

An RDF model describes web resources in a machine-readable form. That is, it is possible to develop a software tool that interprets RDF models and inferences valuable information implied in those models. In other words, developing an RDF model is one thing, and developing an inference software tool is another. On the other hand, both the RDFS and RDF layers are mixed in a CPN model; the RDFS layer is reflected in the topology of the CPN model, whereas the RDF layer is reflected in the dynamics of the CPN model. As a result, we can extract inferential information by simulating CPN dynamics.

To demonstrate the practicality of the proposed method, a simple indoor location-based service system and an ontology for the system are considered. Applying the abovementioned transformation algorithm to the ontology, we construct a Petri net model. Then we introduce an inference engine for the Petri net model based on the ontology. We show that by using the inference engine, the moving object database of the indoor location-based service resolves ambiguities in queries.

2. Literature Review

This section provides a brief review of prior research results related to our subject.

2.1 RDF

XML (Extensible Markup Language) is a general language that can be used for defining markups. Therefore, XML documents are typically used to exchange data between applications. However, XML cannot represent data semantics. The RDF is a data model in which this shortcoming of XML is reinforced. That is, a user can define his or her own vocabulary, properties of objects, and relationships between the objects by using RDFS language.

The purpose of the RDF is to describe web resources, and the basic component of RDF is a statement consisting of a subject, a predicate, and an object. For example, “The name of the resource whose ID is ‘352’ is ‘David’” can be specified as follows:

<rdf:Description rdf:about=”352”>

<uni:name>David</uni:name>

</rdf:Description>,

where “uni” is a predefined name space. The subject of this statement is “352,” the predicate is the name, and the object is David. We assume here that the reader is familiar with the RDF. Please refer to References [3-8] for more information on the RDF.

The sample RDF model shown in Fig. 1 is from [3]. We refer to this model frequently in the present paper.

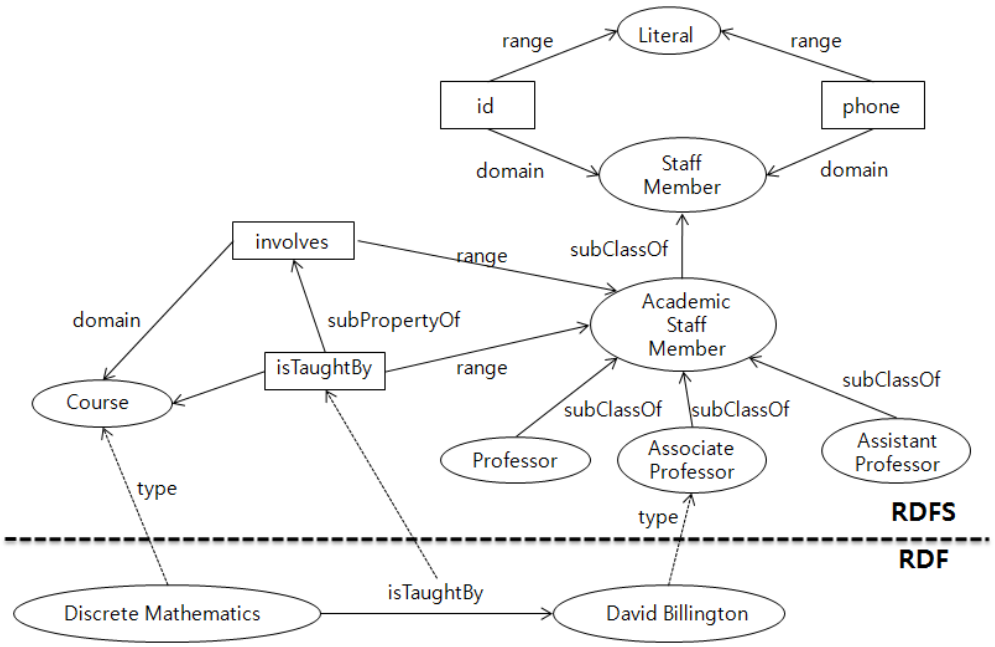


Fig. 1. RDF and RDFS Layers

2.2 Colored Petri Net

Petri (1962) was the first to introduce the concept of the Petri net. Since then, the Petri net has been widely used for modeling and analyzing concurrent, nondeterministic, asynchronous, parallel, and/or distributed information processing systems. A Petri net is both graphical and mathematical. That is, we can visually represent the information and data flow of systems and establish state equations and mathematical models governing the behavior of systems by using the Petri net.

A number of researchers have examined the applications and theory of Petri nets. Petri nets have been successfully applied in performance evaluation and communication protocols. Promising Petri net application areas include the modeling and analysis of distributed database systems, distributed software systems, flexible manufacturing control systems, concurrent and parallel programs, and so forth. We assume here that the reader is familiar with Petri nets. Please refer to References [1, 2, 10-12] for more information on Petri nets.

The sample colored Petri net shown in Fig. 2 and Table 1 is from [2]. The net represents a simple communication protocol in which a sender transmits a number of packets. Some of the packets (20% in this sample) can be lost on the way to the receiver. The lost packets are then resent by the sender. The colors (corresponding to the data types specified in programming languages), variables, and constants used in the Petri net are shown in Table 1. We refer to this model throughout this paper.

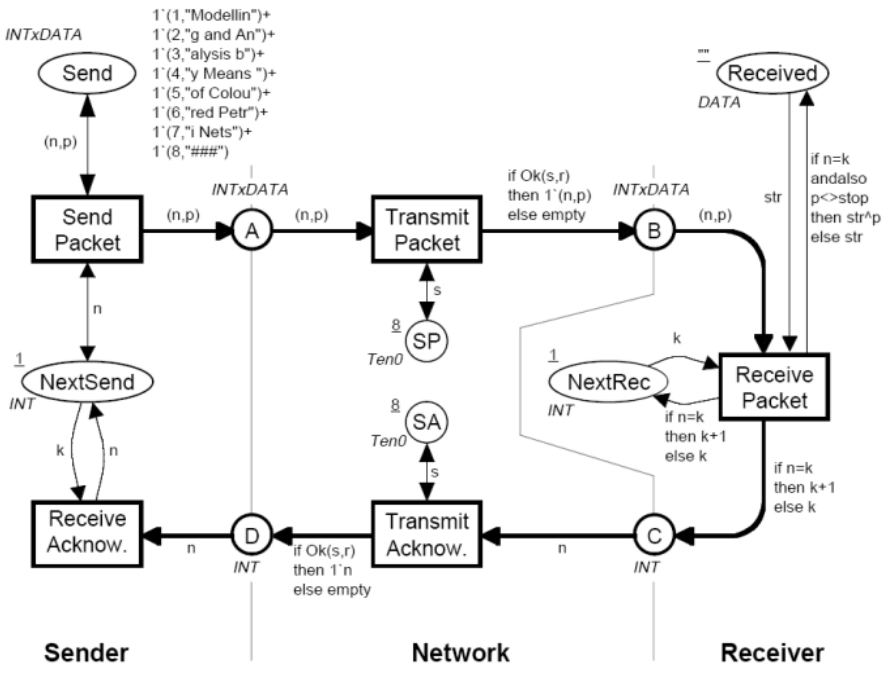


Fig. 2. Sample CPN: a Simple Communication Protocol

Table 1. Colors, Variables and Constants of the CPN in Fig. 2

```
color INT = int;
color DATA = string;
color INTxDATA = product INT * DATA;
var n, k: INT;
var p, str: DATA;
val stop = "###";
color Ten0 = int with 0. .10;
color Ten1 = int with 1. .10;
var s: Ten0; var r: Ten1;
fun Ok(s:Ten0, r:Ten1) = (r<=s);
```

2.3 Other Related Research Results

CPNTools [2] is a software tool that can be used to easily build CNP models and perform simulations. CPNTools has been used to develop a number of Petri net models and validate their performance. In the present study, we built a CPN model by using CPNTools and performed simulations to validate the proposed method.

Petri nets have been used in a wide range of areas. [13] proposed a ”generalized Petri net,” a type of modified Petri net for modeling rule-based systems. There are many formalized ways to represent Petri nets, including the metamodel and UML (Unified Modeling Language). [14] introduced a Petri net ontology written in UML, RDFS, and OWL (Web Ontology Language).

The present work shares many similarities with [15], which introduced a method of transforming OWL DL (Description Logic) into predicate/transition nets. Because the RDF model in the present paper is a subset of OWL DL, there is an overlap between the

objectives of [15] and those of the present paper. However, our paper proposes a method that can transform RDF models into CPNs, not into predicate/transition nets. Furthermore, we actually built a CPN model with CPNTools to verify the proposed method.

3. Our Method of Transformation

We propose a method that can transform an RDF model into a CPN. The proposed method consists of two phases: the transform of the RDFS layer and that of the RDF layer. In the "transform of the RDFS layer" phase, we identify the core classes, map them onto CPN places, and identify the core properties. For each property, we introduce a transition so that the tokens at the place corresponding to the subject of the property can be transferred to the place corresponding to the object of the property. We then identify the domains and ranges of properties, and for each pair of the domain and range of a property, we introduce a transition that transfers an ordered pair of tokens: the first one from the place corresponding to the domain to the place corresponding to the property at a time and the second one from the place corresponding to the range to the same. This completes the topology of the CPN model.

During the "transform of the RDF layer" phase, RDF statements are reflected in the CPN. For each statement, we identify its subject, predicate, and object and then do the following. First, if the predicate is ref:type, then we introduce a transition that generates a token representing the subject and adds it to the place corresponding to the object.

Second, if the object is a container (e.g., rdf:Bag, rdf:Seq, or rdf:Alt), then for each case, we do the following. For rdf:Bag, we separate the statement into many statements and go to the third step. For example,

```
<rdf:Description rdf:about="318">
<uni:coursesTaught>
    <rdf:Bag>
      <rdf:_1 rdf:resource="C111"/>
      <rdf:_2 rdf:resource="C112"/>
    </rdf:Bag>

  …
```

is separated into two statements: "318 teaches C111" and "318 teaches C112". In the CPN, a domain is always a finite multiset. Therefore, the resulting CPN actually corresponds to rdf:Collection.

For rdf:Seq, we put all the elements of rdf:Seq into a list in order. For example, given

```
<rdf:Seq>
  <rdf:li rdf:resource="352"/>
  <rdf:li rdf:resource="318"/>

</rdf:Seq>,
```

we can build the CPN shown in Fig. 3. 1`[ ] in Fig. 3 represents an empty list. The empty list, 1`1, and C111 substitute Seq, i, and CID, respectively, when the transition of the figure fires. As a result, the list and 1`1 are changed into ["C111"] and 1`2, respectively. Firing the transition for the second time changes the list to ["C111" "C112"].

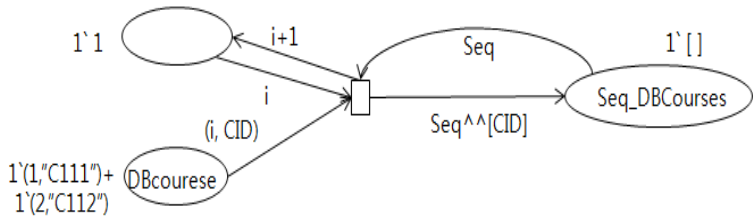


Fig. 3. A CPN Representing rdf:Seq

For rdf:Alt, by using the OK function of the CPN in Fig. 2, we make the transition to produce one of the tokens corresponding to the elements of rdf:Alt. For example, given the following rdf:Alt, we make the OK function return "true" or "false" with equal probability by replacing "8" next to SP in Fig. 2 with "5" so that the transition produces a token with "352" or "318" evenly as follows:

```
<rdf:Alt>
  <rdf:li rdf:resource="352"/>
  <rdf:li rdf:resource="318"/>
</rdf:Alt>
```

Third, we finally map the subject, predicate, and object onto places ps, pp, and po, respectively, and map the sentence onto the transition ts. Then we put a guard on ts so that it consumes a token x from ps and a token y from po and adds a token of (x, y) to pp when it fires. In this manner, we can also transform rdf:resource attributes and reifications into a CPN. Our transformation method is summarized in Table 2.

Table 2. Algorithm for Transforming an RDF Model into a CPN

Input: an RDF model
Output: CPN model

1. Identify the core classes, resources, classes, literals, properties, and statements of the RDFS and map them onto CPN places.
2. Represent the properties of RDFS:subClassOf and subPropertyOf by introducing a transition that transfers a token from the place corresponding to the subject class/property to the place corresponding to the object class/property when it fires.
3. Identify the domains and ranges of RDFS properties. For each property and its domain and range, we introduce a transition that consumes a token x from the place corresponding to the domain and a token y from the place corresponding to the range and adds a token of (x, y) to the place corresponding to the property.
4. For each RDF statement consisting of a subject, a predicate, and an object, we map its subject, predicate, and object onto places and the statement onto a transition. Then we put a guard to the transition so that it consumes tokens x and y from the places corresponding to the subject and the object, respectively, and adds a token of (x, y) to the place corresponding to the predicate.
   (A) In case of the predicate rdf:type, instead of mapping the predicate onto a place, we introduce a transition that transfers a token representing the subject to the place corresponding to the object class.
   (B) In case of the "Container," we represent the Container by using the approach explained earlier in the discussion about the "transform of the RDF layer" phase.

4. Examples of Transforms

This section illustrates the transformation of an RDF model into a CPN.

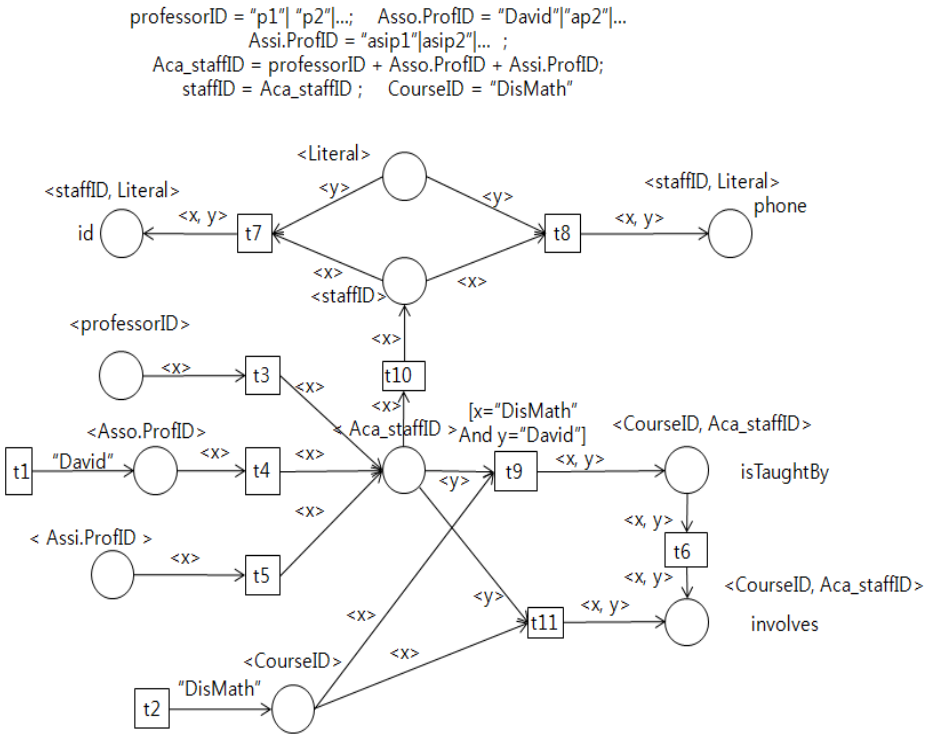


Fig. 4. A CPN Model for the RDF Model in Fig. 1

4.1 A CPN Model of an RDF Model

A CPN model for the RDF model shown in Fig. 1 is shown in Fig. 4. The topology of the CPN in Fig. 4 is constructed during the "transform of the RDFS layer" phase (Steps 1 to 3 of the algorithm shown in Table 2), except for the source transitions (t1 and t2). In Step 1, we identify all the classes (e.g., Professor, Academic_staff, Literal, etc.) and properties (e.g., ID, Phone, isTaughtBy, etc.) and introduce a place for each. In Step 2, we introduce the transitions for subClassOf (t3, t4, t5, …) and subPropertyOf (t6). In Step 3, we introduce t7 and t8 to represent the domains and ranges.

In Step 4, we introduce t1 and t2 to represent that David is a professor and DisMath is a course. We then put a guard to represent "DisMath isTaughtBy David."

4.2 rdf:resource Attribute

The rdf:resource attribute is addressed in Step 4 of our algorithm. The following descriptions with rdf:resource can be transformed into the CPN shown in Fig. 5:

<rdf:Description rdf:about="CIT1111">
<uni:isTaughtBy rdf:resource="949318"/>
</rdf:Description>

```
<rdf:Description rdf:about=”949318>
   <uni:name>David Billington</uni:name>
</rdf:Description>
```

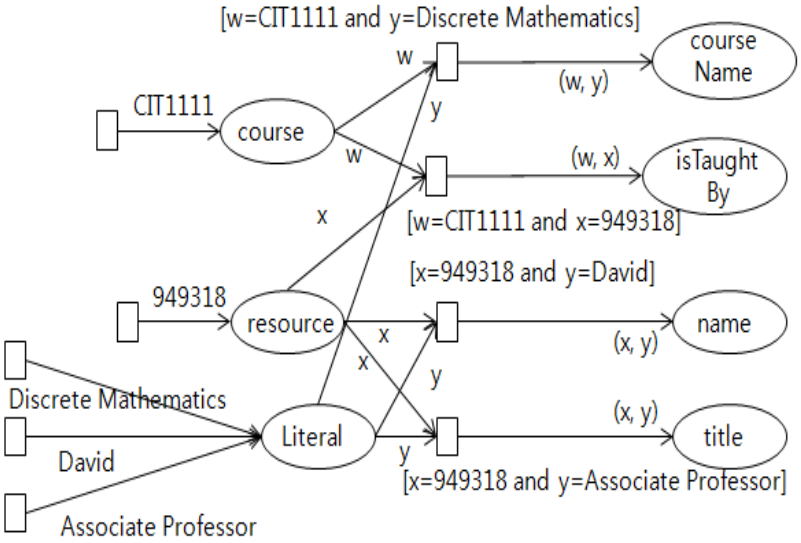


Fig. 5. CPN Representation of a Sample rdf:Resource Attribute

4.3 Transformation of a Structured Object

Fig. 6 is a sample structured object from [4]. The figure shows that the address of the staff whose ID is 85740 consists of four fields (city, street, state, and postal code). This can be transformed into the CPN shown in Fig. 7.

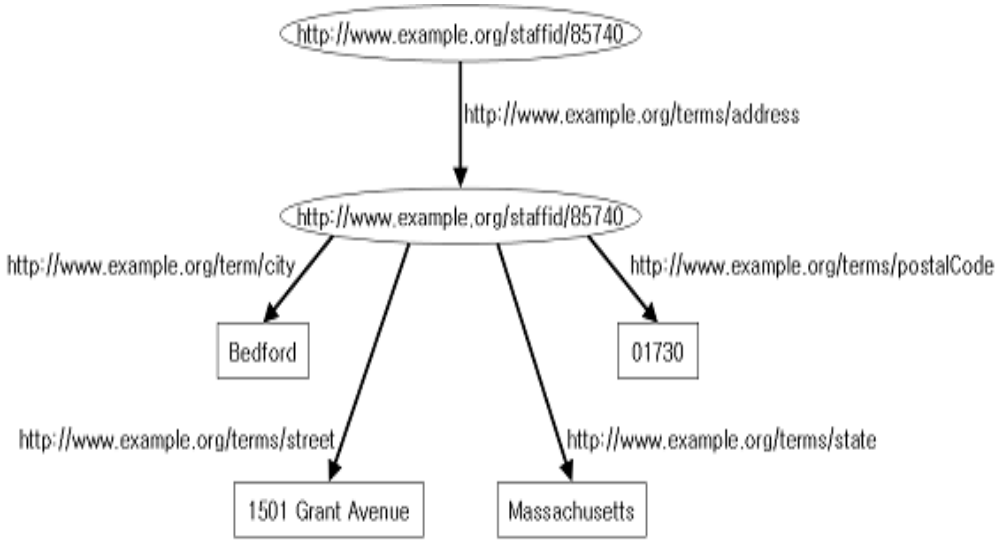


Fig. 6. A Sample Structured Object

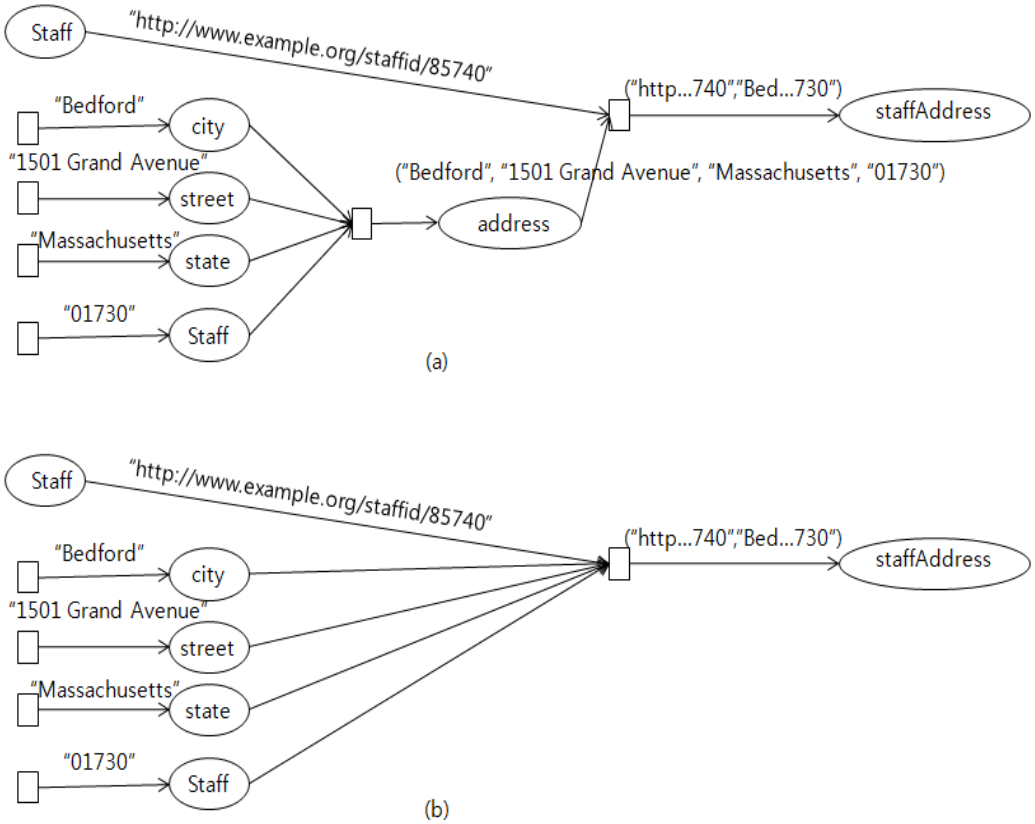


Fig. 7. Two Possible CPN Models for Fig. 6: (a) Exact Translation of Fig. 6 and (b) Abbreviated Version.

4.4 Reification

The following reification description can be transformed into the CPN shown in Fig. 8:

```
<rdf:Description rdf:about=”David”>
<rdf:believesIn rdf:resource=”#Statement123”/>
</rdf:Description>

<rdf:Statement rdf:about=”Statement123”>
  <rdf:subject>Bob</rdf:subject>
  <rdf:predicate>owner</rdf:predicate>
  <rdf:object>xxx</rdf:object>
```

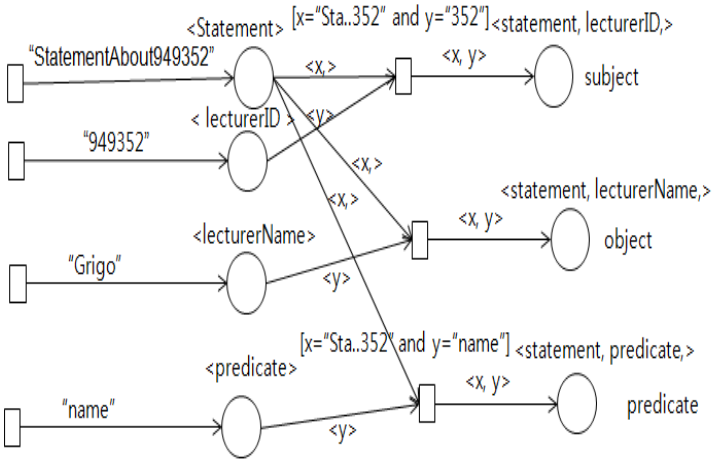


Fig. 8. A Sample CPN Model of Reification

5. Simulation

Drawing an inference is easy with a CPN model. The objective is to extract answers for RDF queries in a CPN. The following query retrieves all associate professors excluding inherited ones:

^AssociateProfessor

With CPNs, we can find answers for these types of queries by firing all the source transitions directly connected to the place "AssociateProfessor." For example, we can find the answer to this query by firing the transition t1 of the CPN shown in Fig. 4.

The following query retrieves all staff members including inherited ones:

staffMember

By firing all the transitions connected to the incoming arcs of the place staffMember, we can find the answer to this query. In the case of Fig. 4, we fire t1, t4, and t10.

We can also easily find the answer to the following query by performing simulations with the CPN:

Select X, Y

From {X} isTaughtBy {Y}

Given the query, we fire all the transitions connected to the incoming arcs of the place isTaughtBy. In the case of Fig. 4, we fire t1, t4, t2, and t9. Firing these transitions in sequence puts the token <"DisMath", "David"> in the isTaughtBy place. That means "DisMath" isTaughtBy "David".

Given the following query

Select N

from course{X}.isTaughtBy{Y}, {C} name {N} ,

where Y= "David" and X= "C",

we add an out transition to the isTaughtBy place. The transition t15 in Fig. 9 is such a transition. We then attach a guard [x="David"] to the transition. The variable z in Fig. 9 is the answer to the query.

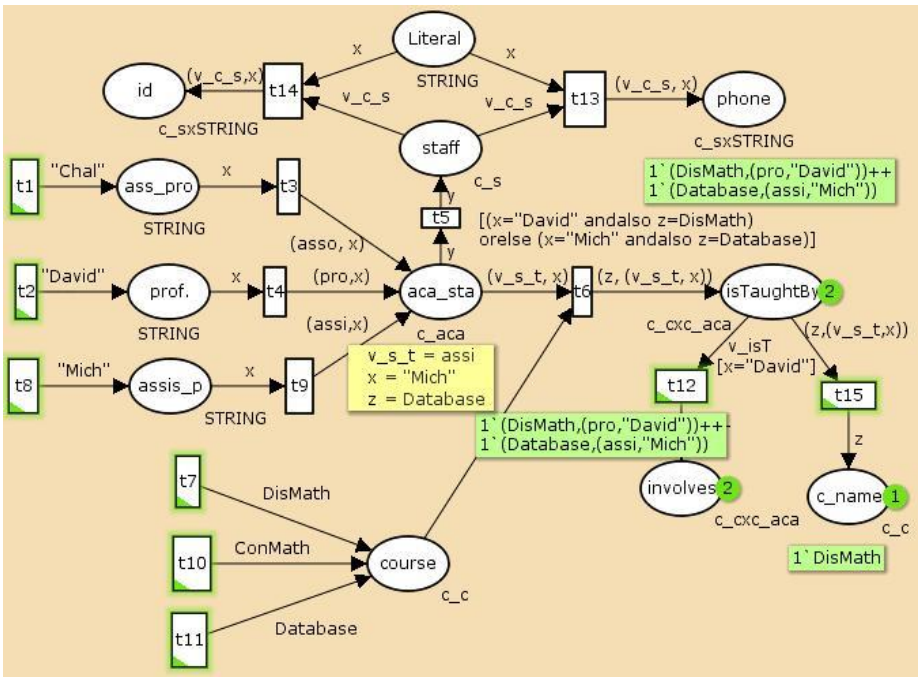


Fig. 9. A CPN representing the RDF model in Fig. 1 (constructed by CPNTools) and answers to some sample queries.

6. Ontology for Indoor Location-Based Service

Consider an indoor location-based service (ILBS) provided by a small, imaginary museum exhibit area shown in Fig. 10. In the area, there are 7 booths (Booth1 to 7), 2 vending machines (one for hot coffee (d2) and the other for soft drinks (d1)), and 12 chairs (c1 to c12). One of the main parts of the ILBS is the moving object database (MODB) in which the spatio-temporal information of moving objects is recorded. The ILBS translates questions such as "Who is visiting Booth 1?" into a query of the MODB and displays the answer to the query on the user interface of the ILBS. The MODB records the information in a relational database.

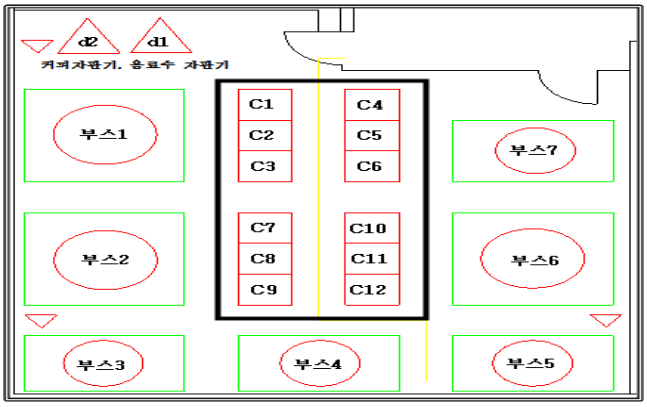


Fig. 10. Museum Exhibition Room

The database has tables such as Point, Node, Booth, and so on as shown in Fig. 11. A node is a conjunction of two paths, a path consists of links, and a link is identified by two adjacent nodes. A booth has many attributes such as Boundary_ID, Exhibit_ID, and so on. In this example, a moving object is a visitor to the exhibit area. In the process of translating a user's question into a query, the ILBS sometimes refers to the ontology to resolve any ambiguities in the terms. For example, if the question is "Who is taking a rest?" then the ILBS interprets it into "Who is at the location of chairs?" referring to the ontology. After that, it is translated into a query for the MODB.

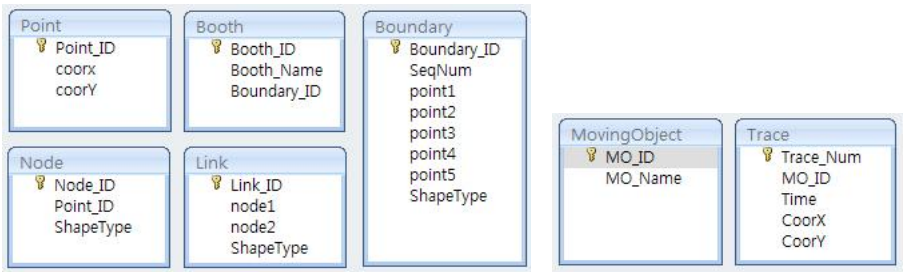
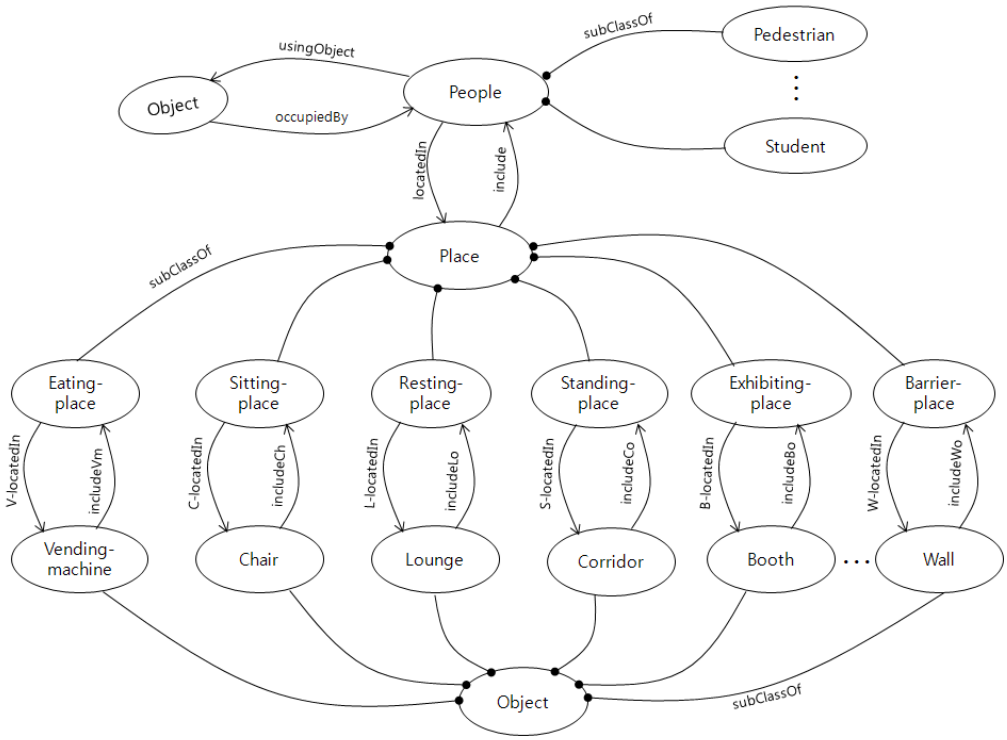


Fig. 11. Tables of the MODB

A sample ontology for the imagined exhibit area is shown in Fig. 12. Three classes "people," "place," and "object" are used for the ontology for the museum's LBS. The class "object" has various subclasses such as "vending-machine," "chair," "lounge," "corridor," and so on. C1, C2, …, Cn are the instances of "chair," and their type is "chair." The object property C-locatedIn relates the class "chair" to the class "sitting-place."



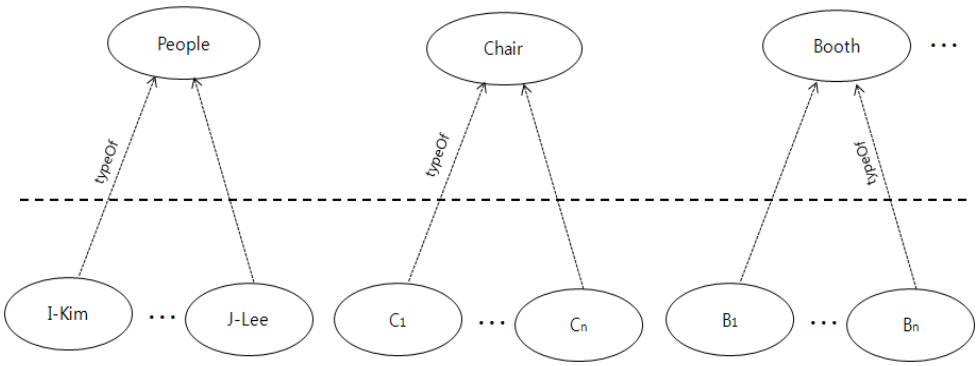


Fig. 12. The ontology and RDF for the exhibit area' LBS

As shown in Fig. 13, the domain of "c-loacatedIn" is "chair," and its range is "sitting-place." It can indicate that people can sit on a chair; a person sitting on a chair is taking a break and not moving; a booth is a place where an exhibit is displayed and can be viewed by people; a booth is something a person cannot walk through, and so forth.

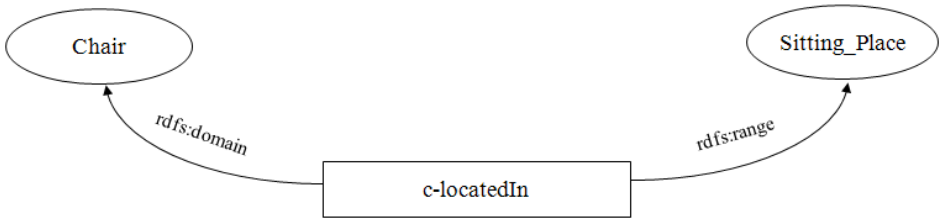


Fig. 13 Object Property c-locatedIn Related to Two Classes

Fig. 14 shows a sample OWL stating that the object property c-loactedIn has the "chair" as its domain and the "sitting-place" as its range and that includeCh has an inverse relationship with c-locatedIn.

```
<owl:ObjectProperty rdf:ID="c-locatedIn">
   <rdfs:domain rdf:resource="#Chair"/>
   <rdfs:range rdf:resource="#Sitting_Place"/>
      <owl:inverseOf rdf:resource="#includeCh"/>
</owl:ObjectProperty>
```

Fig. 14. Object Property c-locatedIn written in OWL

Fig. 15 shows the ontology for the LBS (in TopBraid, an ontology editing tool).

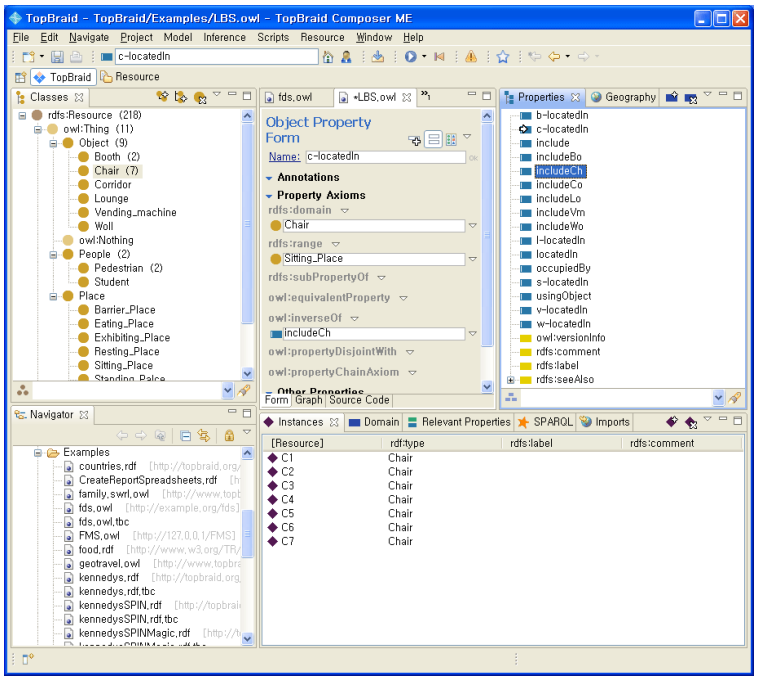


Fig. 15. Ontology for the LBS (in TopBraid)

As discussed in Section 3, the ontology can be transformed into the Petri net shown in Fig. 16. All places (e.g., staff, visitors, chairs, and booths) have only two input transitions in the figure, but in the real world, they would have many input transitions. For example, if there are 10 chairs, then the place chair has 10 input transitions. An object property in an ontology is represented as exactly one place in the Petri net. There are many object properties in the ontology, and there should be that many places representing properties; however, in Fig. 16, only one property is represented, and others are omitted.

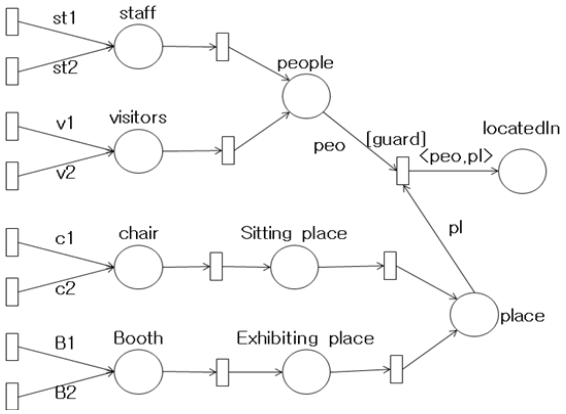


Fig. 16. Petri Net Representation of the Ontology in Fig. 12.

A colored Petri net shown in Fig. 16 can be represented in the matrix in Table 3 after unfolded [1]. In the matrix, a row corresponds to a transition, and a column corresponds to a place in a Petri net. Entry 1 at M[ij], where M, i, and j represent the matrix, the i-th

row, and the j-th column, respectively, shows that there is an arc from the i-th transition to the j-th place. On the other hand, Entry -1 at M[ij] shows that there is an arc from the j-th place to the i-th transition.

Table 3. Matrix Representation of the Ontology in Fig. 12

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	...	[27]	...	[m]
[0]	1	-1	0	0	0	0	0	0	0	0	0	...	0	...	0
[1]	1	0	-1	0	0	0	0	0	0	0	0	...	0	...	0
[2]	1	0	0	-1	0	0	0	0	0	0	0	...	0	...	0
[3]	0	1	0	0	-1	0	0	0	0	0	0	...	0	...	0
[4]	0	1	0	0	0	-1	0	0	0	0	0	...	0	...	0
[5]	0	1	0	0	0	0	-1	0	0	0	0	...	0	...	0
[6]	0	0	1	0	0	0	0	-1	0	0	0	...	0	...	0
[7]	0	0	1	0	0	0	0	0	-1	0	0	...	0	...	0
[8]	0	0	0	1	0	0	0	0	0	-1	0	...	0	...	0
[9]	0	0	0	1	0	0	0	0	0	0	-1	...	0	...	0
...	...	...	...	...	...	....	...	...	...	...	....	...	...	...	...
...	...	...	...	...	...	....	...	...	...	...	....	...	...	...	...
[n]	0	0	0	0	0	0	0	0	0	0	0	...	1	...	-1

Given a matrix M that represents the ontology, we can draw both forward and backward inferences. For example, if "sitting place" is given, the forward inference returns "place," whereas the backward inference returns "c1," …, and "cn." which are the leaf nodes of the ontology. In the database, only the leaf nodes are used as the entries of tables. That is, an inner node such as "sitting place" does not appear anywhere in the database. Therefore, a query such as "Who are sitting at the moment?" cannot be handled by the database, unless it understands that a chair is something on which a person can sit.

The number of arguments of the backward inference is 1, a place, p_j. The backward inference, as its first step, finds all entries of M[i, j] = 1 on the j-th column of M. Let E be a set of i-th rows of M satisfying M[i, j] = 1. Then "E is empty" implies that "p_j is a leaf node." Thus, if E is empty, then it prints out the identity of p_j. Otherwise, for each i in E, it finds all the entries of -1 on the i-th row. Let A be a set of places q satisfying M[i, q] = -1. It then recursively calls itself with q as its argument for each q in A. The process of the backward inference is shown in Table 4.

Table 4. The Proposed Backward Inference Algorithm

```
Algorithm backward(int j)
1. Let E be a set of i so that M[i, j] = 1;
2. If E is empty, then print out the identity of the place p_j and exit( );
3. (for each i in E) {
4.    Let A be a set of places q so that M[i, q] = -1 ;
5.    ( for each q in A) { recursively call backward(q); }
6.}
```

The forward inference takes one or more place ids as arguments. It puts exactly one token at each of the argument places and no token at any other places. We call this placement of tokens the initial marking. It then returns all the places that have a token

in any reachable marking. For this purpose, the inference module performs the process described in Table 5. For each argument place corresponding to the j-th column of the matrix, the inference engine adds one more row (e.g., the k-th row) to the matrix and sets M[k, j] = 1. For a row of M, if there is no entry -1 in the row, the transition corresponding to the row is enabled. Then the forward inference engine finds all the enabled transitions. For each row r_i corresponding to the enabled transition, it finds all entry 1s in the row. Let M[i, j_1], …, M[i, j_q] be such entries, that is, let M[i, j_1]=1, …, M[i, j_q]=1. Then it outputs the identities corresponding to places j_1, …, j_q as part of the answer and replaces all the entries -1 in columns j_1, …, j_q. It goes back to the step finding enabled transitions. The inference process stops if there is no enabled transition.

Table 5. The Proposed Forward Inference Algorithm

Algorigm forward(int[ ] q)
// q is the set of column indices mapped onto the ambiguous words.
1. Initialization; (Let E be empty, remove all the source nodes)
2. (for each j in q) {append a row of 0 to M and let j-th entry of the row be 1;}
3. Find all enabled transitions and add them to E;
4. If E is not empty {
5. pick up any transition from E and let's assume it be the transition t_i, or i-th row;
6. (for each j such that M[i, j]=1) {
7. Print the identity of the place j;
8. Replace all -1 in the column j with 0;
9. }
10. goto 3
11.}

The proposed inference algorithms are now implemented and used by our MODB. For example, the MODB interprets "Find places to sit" as "Find all available chairs" by referring to the sample ontology (Fig. 17). The inference module is invoked with "sitting place" as its argument in this case.

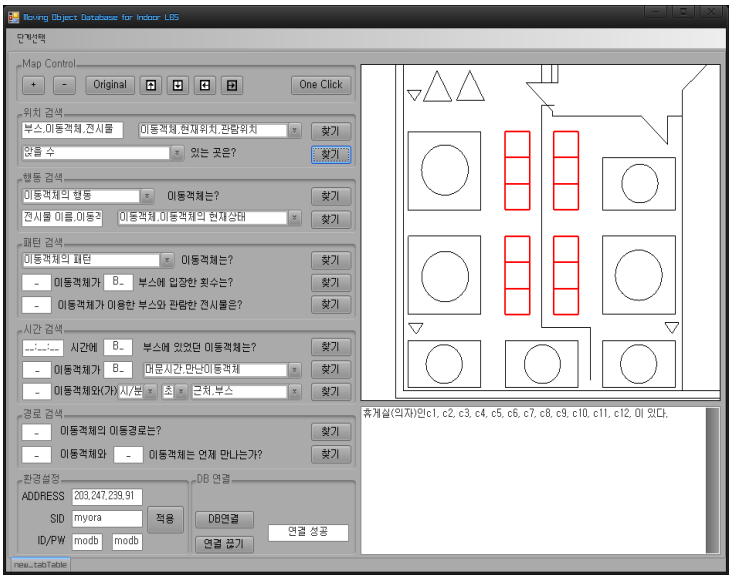


Fig. 17. The Proposed MODB that interprets "Find places to sit" as "Find all available chairs" by referring to the Sample Ontology

7. Conclusion

This paper proposes a method that can transform RDF models into CPN models. An RDF model consists of an RDFS layer and an RDF layer. Therefore, constructing an RDF model is one thing and drawing an inference with the model is another. On the other hand, the RDFS and RDF layers are mixed in a CPN. Therefore, we can extract answers to RDF queries by performing simulations with CPN models. In this regard, we constructed a CPN model for an RDF model and performed simulations to generate answers to sample RDF queries by using CPNTools.

To test the practicality of the proposed method, we introduced an ontology for a small, imaginary exhibit area in a museum. For the location-based service, we built a prototype Indoor Moving Objects Database (IMODB) system for the exhibit area. The results indicate that the IMODB can successfully resolve ambiguities in queries by referring to the proposed ontology. IMODB is an essential ingredient for Indoor Location Based Service (ILBS). Among the many applications of ILBS are Navigation, Point of Interesting service, U-Healthcare, Social Networking Service, Robotics, Mobile Broadcasting Service, and so on.

With a CPN model of an RDF model, we can discover a wide range of RDF model characteristics by analyzing the CPN. Future research should enhance the generalizability of the proposed method so that it can deal with OWL ontologies. Further, mathematical analysis methods for the CPN model should be explored so that answers to RDF queries can be extracted more efficiently. For example, the CPN model of OWL ontology may have the following characteristics: it has source transitions and sink transitions, out-degree of a place is one, out-degree of a transition is also one, and so on. Making use of these characteristics, we will figure out some useful necessary and sufficient conditions for a certain proposition to be an answer of a query.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0006942) and by 'Industry, Academy and Research Institute Co-development of Techniques' funded by the Small and Medium Business Administration (R00046281). Joo's work was supported by 'Industry, Academy and Research Institute Co-development of Techniques' funded by the Small and Medium Business Administration (K-2011-A0013-00002). Lee's work was supported by 'Development of Global Culture and Tourism IPTV Broadcasting Station' Project through the Industrial Infrastructure Program for Fundamental Technologies funded by the Ministry of Knowledge Economy (10037393).

References

[1] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol. 77, no. 4, **(1989)** April, pp. 541-580.
[2] http://wiki.daimi.au.dk/cpntools/cpntools.wiki
[3] Grioris Antoniou, Frank van Harmelen, *A Semantic Web Primer*, The MIT Press, **(2004)**.
[4] Frank Manola, Eric Miller, "RDF Primer," W3C Recommendation 10 Def. 2004, http://www.w3.org/TR/REC-rdf-syntax/, **(2004)**.
[5] Ora Lassila, Ralph R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification", http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/, **(1999)** February 22.
[6] Dan Brickley, R.V. Guha, "Resource Description Framework (RDF) Schema Specification 1.0", http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, **(2000)** March 27.

[7] Graham Klyne, Jeremy J. Carroll, "RDF Concepts and Abstract Syntax", http://www.w3.org/TR/rdf-concepts/, **(2004)** February 10.
[8] Richard Fikes and Deborah L. McGuinness, "An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL", http://www.daml.org/2001/03/axiomatic-semantics.html, **(2001)** October.
[9] C.A. Petri, "Kommunikation mit Automaten," Bonn:Institut fur Instrumentelle Mathematik, Schriften des IIM Nr. 3, 1962. Also English translation, "Communication with Automata," New York:Griffiss Air Force Base, Tech. Rep. RADC-TR-65-377, vol. 1, Suppl. 1, **(1966)**.
[10] K. Jensen, "*An Introduction to the Practical Use of Coloured Petri Nets*," Lecture Notes in Computer Science vol. 1492, Springer-Verlag **(1998)**, pp. 237-292.
[11] Kurt Jensen, "An Introduction to the Theoretical Aspects of Coloured Petri Nets," Lecture Notes in Computer Science vol. 803, Springer-Verlag **(1994)**, pp. 230-272.
[12] K. Jensen, L.M. Kristensen, L. Wells: *Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems*. International Journal on Software Tools for Technology Transfer, 9 **(2007)**, Springer Verlag, 213-254.
[13] Dong-Her Shih, Hsiu-Sen Chiang, Binshan Lin, "A Generalized Associative Petri net for Reasoning," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 9, **(2007)** September, pp. 1241-1251.
[14] Dragan Gasevic, "Petri Nets on the Semantic Web Guidelines and Infrastructure", ComSIS, vol. 1, no.2, **(2004)** November, pp.127-151.
[15] G. Zhang, F. Meng, C. Jiang, J. Pang, "Using Petri Net to Reason with Rule and OWL," Proceedings of the sixth IEEE International Conference on Computer and Information Technology (CIT'06), Seoul Korea, **(2006)** September 20-22, CIT 2006:42.

Authors

Jaegeol Yim received the M.S. and Ph.D. degrees in Computer Science from the University of Illinois at Chicago, in 1987 and 1990, respectively. He is a Professor in the Department of Computer Science at Dongguk University at Gyeongju Korea. His current research interests include Petri net theory and its applications, Location Based Service, AI systems, and multimedia systems. He has published more than 50 journal papers, 100 conference papers (mostly written in Korean Language), and several undergraduate textbooks.

Jaehun Joo is a professor of Department of Information Management at Dongguk University (Gyeongju) in Korea and Editor-in-Chief, Journal of Korea E-commerce Research Academy. Also he was a Visiting Professor of Department of Management at University of Nebraska-Lincoln. He received his Ph. D. from Busan National University. His areas of research interest are electronic commerce, business ecosystems, collective intelligence, semantic web, and knowledge management. He published many papers in Int. J. Human-Computer Studies, Journal of Sustainable Tourism, Information Systems Management, International Journal of Industrial Engineering, Expert Systems with Applications, Journal of Computer Information Systems, Decision Support Systems, etc.

Gyeyoung Lee received the M.S. degree in Computer Science from the Dongguk University in 1982 and the Ph.D. degree in Computer Engineering from the Dankook University in Korea in 1992, respectively. He is a Professor in the Department of Computer Science at Dongguk University at Gyeongju, Korea. His current research interests include Petri net theory, AI systems and Speech processing systems.