Building a Semantic-Web Portal Using Frame-based System *

Ching-Long Yeh, Jia-Yang Chen and Chih-Hsien Lin
Department of Computer Science and Engineering
Tatung University
40 Chungshan N. Rd. 3rd Sec.
Taipei, 104, Taiwan
chingyeh@cse.ttu.edu.tw

Abstract

In this paper we report employing frame-based system as the basis to build up a ontology-based portal. The basic idea is to transform the ontology schema and instances collected from various digital library sites using the semantics of the frame-based system. The service programs written using the frame-based reasoning formalism thus can make use the knowledge.

1 Introduction

A digital library portal provides services based on an integrated view for user to access contents from various digital library sites [24, 14]. A semantic web portal is an appropriate tool to build up the integrated service layer for digital libraries [15][30]. The kernel of a typical semantic web portal consists of a repository of facts created based on the schema of ontologies and an inference engine [5]. In front of the portal are various service programs, including discovery, tailoring and social interaction, implemented based on the inference engine. In the other end are content provision interfaces that collect content from various sources. In a previous work [30] we have designed such a portal that provides a seamless interface for user to access the services provides by various digital libraries. Frame-based representation is a natural choice to carry out work of Semantic Web [20]. In this paper, we describe an implementation of the portal using a frame-based system, flex [27].

The inference engine of flex is based on the logic programming system, Prolog [7], and supports frame representation. Furthermore, it supports forward and backward chaining inferences, which enables us to develop programs at conceptual level. Using Flex, we can access persistent databases using standard database connection interfaces. In addition to local GUI, it has utilities to make programs interact with outside world, such as the connection with web server, TCP/IP and agent libraries.

The content provision component of the portal we design collects contents in RDF format from various digital library sites. The RDF documents are then converted into instances of flex

*This research was supported by the Taiwan National Science Council under Contract No. NSC92-2422-H-036-322

representation and stored in the appropriate locations in the repository. The repository is a directory structure on the basis of the schema of ontology. The service programs in the front end are written using the forward and backward chaining representation.

In the next section, we describe the components of the portal architecture. In Section 3, we describe the conversion of RDF to the language of flex. In Section 4, we show the implementation result. Finally conclusions are made.

2 System Description

In this section, we first give an overview of the architecture and then describe the frame representation we use to implement the portal.

2.1 Overview of System Architecture

The architecture of the ontology-based portal is summarized as shown in Figure 1. The central part of the portal is a knowledge base consisting of ontology and knowledge warehouse. The information provision component consists programs for annotation and metadata wrapping. In the front end, the discovery, use, tailoring and social interaction functions are implemented based on an inference engine.

The inference engine of flex interprets its own representation language, Knowledge System Language (KSL). Thus we need a translation mechanism to represent both the ontology and knowledge warehouse in Figure 1 using the semantics of KSL. The resulting representations are then stored in their respective repositories as illustrated in Figure 2. The translation mechanism and the repository organization are described in Section **??**. The service programs, for

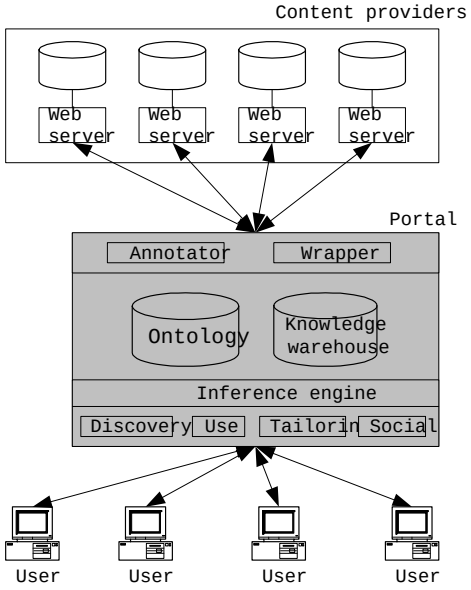

Figure 1: A high-level view of the components of the ontology-based portal.

example, conceptual search and semantic navigation, are written based on the forward and backward chaining mechanism. In the remainder of this section we give an introduction to the frame representation and inference rules in flex.

2.2 Frame-Based Reasoning in flex

flex supports frame-based reasoning with inheritance, rule-based programming and datadriven procedures fully integrated within a logic programming environment, and contains its own English-like Knowledge Specification Language (KSL) [27].

A frame is a complex a data structure that is used to model the objects in the real world. Each frame has its own name, its parent frame names, a number of attributes and the respective default values. For example, the frame `feline`

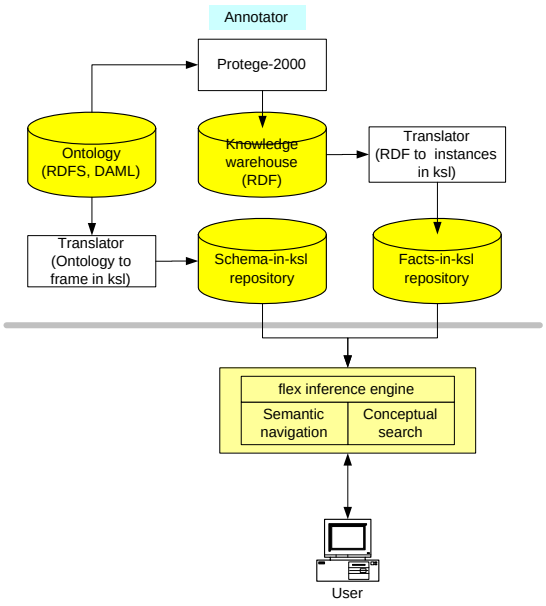

Figure 2: The kernel of the ontology-based portal.

in Figure 3, inherits the attributes from both of its parent frames, `mammal` and `carnivore`. Two instances, `sylvester` and `sammy` are defined to have the default attributes and values of cat. Instead of inheriting default values, an instance can have its specific values ovrerriding the ones from its ancestors. The inheritance relationship of frame and instance thus forms a class hierarchy. The inference engine then operates based on such a hierarchy of the domain. Individual frame or group of frames can be attached procedures to control the consistency of data and knowledge when requests are made to update, access, or create an instance or slot to which they are attached. For example, the frame `employee` in Figure 4 is attached with a `launch` procedure, `new_employee`. When an instance of new male employee is created an action is performed to collect the personal details. In addition to control the creation of new instance, a procedure can be attached to individual slot and is activated when accessing or updating the the values of the slot. The

```
frame feline is a mammal, carnivore
   default legs are 4.
frame cat is a feline
   default habitat is house and
   default meal is kit_e_kat.
instance sylvester is a kind of cat.
instance sammy is an instance of cat.
```

Figure 3: A example of frames and instance in flex.

```
frame employee
   default sex is male .
launch new_employee
   when Person is a new employee
   and sex of Person is male
   then male_enrolment_questions(Person) .
instance dave is an employee .
```

Figure 4: Example of attached procedure in flex.

frame and instance representations in flex are used to encode the classes and instances in RDF, as is clarified in Section 3.

Production rules in `if-then` format are used to specify the forward chaining rules for data-driven programming. Also backward chaining rule that expands current goal into a list of sub-goals is effective in goal-driven programming. Both types of reasoning are used to implement the control parts of the services programs as is described in Section 4. The questions and answers mechanism of flex along with the web interface package are used to build browser-based interface.

3 Transformation of Ontology into Frame Representation

As shown in Figure 2, the ontology and RDF instances collected from content sources must be converted into the representation in order to be interpreted by flex inference engine. In this section, we

```
<rdf:RDF>
<rdfs:Class rdf:about="&person;AcademicStaff">
 <rdfs:subClassOf rdf:resource="&person;Employee"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;AdministrativeStaff">
 <rdfs:subClassOf rdf:resource="&person;Employee"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;Employee">
 <rdfs:subClassOf rdf:resource="&rdfs;Resource"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;Lecturer">
 <rdfs:subClassOf rdf:resource="&person;AcademicStaff"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;Researcher">
 <rdfs:subClassOf rdf:resource="&person;AcademicStaff"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;Secretary">
 <rdfs:subClassOf rdf:resource="&person;AdministrativeStaff"/>
</rdfs:Class>
<rdfs:Class rdf:about="&person;TechnicalStaff">
 <rdfs:subClassOf rdf:resource="&person;AdministrativeStaff"/>
</rdfs:Class>
</rdf:RDF>
```

Figure 5: Fragment of a person ontology in RDFS.

first describe the transformation of ontology into the frame representation in flex. Then we describe the transformation of RDF instances into the instances in flex.

3.1 Transformation of Ontology Schema into KSL

Ontology can range from simple taxonomies to frames with complex value restrictions [9]. In this paper we focus on the basic frame-based ontology language, RDFS, as developed in W3C [8] and the further elaborated one, DAML+OIL [17].

RDF is a language for describing relationships among resources. RDFS is an extension of RDF that is used to describe the vocabularies used in RDF documents. It defines several types of resources: `rdfs:Resource`, `rdfs:Class`, `rdfs:Literal`, `rdfs:Datatype`, `rdf:XMLLiteral`, and `rdf:Property`. On the other hand, it defines a number of relationships: `rdfs:range`, `rdf:domain`, `rdf:type`, `rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:label`, and `rdfs:comment`. The classes and relationships among them in an ontology are defined using the above definitions and RDF terms. For example, a person ontology used in the semantic web portal, KA2 [26], is shown in Figure 5. Note that the entity reference, `&person;`, is the shorthand of the URI used to qualify the values in the ontology. The transforming RDFS ontology into flex is that each class along with its parent class information (`subClassOf`) is represented as a `frame` of flex. For example, the ontology in Figure 5 is represented as the following frames in flex.

```
frame 'AcademicStaff' is a 'Employee'.
frame 'AdministrativeStaff' is a 'Employee'.
frame 'Employee' is a 'Resource'.
frame 'Lecturer' is a 'AcademicStaff'.
frame 'Researcher' is a 'AcademicStaff'.
frame 'Secretary' is a 'AdministrativeStaff'.
frame 'TechnicalStaff' is a'AdministrativeStaff'.
```

The properties of each class are then attached to the corresponding frame definition. For example, the `Employee` class in Figure 5 has the following frame definition.

```
frame 'AcademicStaff' is a 'Employee'
 default affiliation is a 'Organization' and
 default  worksAtProject is a 'Project' and
 default  headOf is a 'Project' and
 default  headOfGroup is a 'ResearchGroup'.
```

DAML+OIL is an extension to RDFS with additional expressiveness. In addition to definitions of classes and relationships, a DAML+OIL ontology also has constraints on classes and relationships. For example, the `Female` class shown below has `Animal` as its parent class and has a constraint on its members that excludes instances of male.

```
<daml:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <daml:disjointWith rdf:resource="#Male"/>
</daml:Class>
```

For ontology of this kind, we model the classes and relationships using the frame semantics in flex. Furthermore we use attached procedures to model the constraint part of classes and relationship. For example, we use the following attached procedure to ensure the consistency of newly created instances of `Female`. Other types of attached procedures are used to guard the access or updating of slot values of classes and relationships.

```
launch sexCheckOfFemale
   when Female is a new instance of 'Female'
```

```
then check that Female is not a 'Male' and
write('A new female: ') and
write(Female) and nl.
```

3.2 Transformation of Instances into KSL

Instances based on ontology schema may be created by using edting tools such as Protege-2000 [31]. Instances based on standard metadata format such as DC [19] may be imported from other sites. Or they may be wrapped from structured data sources such as relational database. Same instances can be represented in multiple RDF formats. For example, the fact that `Asia` is a `continent` is represented in two different syntactic forms [16].

```
<continent rdf:ID="Asia"/>

<rdf:Description rdf:ID="Asia">
  <rdf:type>
   <rdfs:Class rdf:about="#continent"/>
  </rdf:type>
</rdf:Description>
```

Though in different syntactic forms, they are all converted into `instance` form in flex. For example, the following RDF instance about the information of a lecture is converted into an `instance` in KSL.

```
<ka2new:Lecturer
   rdf:about="&ka2new;ka2new_00025"
   ka2new:address="ttu 602"
   ka2new:email="hhchen@cse.ttu.edu.tw"
   ka2new:firstname="hh"
   ka2new:lastname="chen"
   ka2new:name="hhchen"
   ka2new:phone="3295"
   rdfs:label="ka2new_00025"/>

instance ka2new_00025 is a 'Lecturer'
 label is 'ka2new_00025' and
 phone is '3295' and
 name is 'hhchen' and
 lastname is 'chen' and
 firstname is 'hh' and
 email is 'hhchen@cse.
```

4 Implementation

In this section we describe an implementation of the system shown in Figure 2. We describe the automatic transformation of RDF documents into flex instances using the DCG formalism in Prolog and the repository. Then we describe the structure of repository for the storage of the resulting instances. Finally we show the discovery service programs.

4.1 Transformation of RDF into flex Instances Using DCG

The transformation of ontology schema and instances into frame representation is an essential, as described in Section 3, to make the RDF-based knowledge source workable in the frame-based system. An ontology may be written using different languages, such as DRFS and DAML+OIL, that increases the complexity to deal with the features in respective languages. Furthermore, this kind of transformation is not a repeated task. Thus in this paper we transform ontology into the frame representation in flex by hand. The other kind of transformation, from RDF instances to flex instances, is performed repeatedly once new RDF instances are imported from other sites. Thus we develop a translator to make the process automatic.

The translator consists of two parts: analysis and generation. The analysis takes an RDF instance as input and constructs its structure. The generation part produces the corresponding form in flex. We use the DCG (definite clause grammar) in Prolog to implement the translator [7]. The DCG implements context-free grammars. A grammar stated in DCG is executed directly by the Prolog inference engine, that makes it a syntax analyzer. Semantic interpretation routines can be interleaved within the right-hand side of DCG rules to handle the generation tasks.

An input RDF document is segmented into lexical entries before it is input to the parser. The parser is constructed by representing straightforwardly the formal grammar described in the RDF specification [21] as DCG rules. For example, the rule of `Description` element in the formal grammar of RDF

and the corresponding DCG representation are shown as below.

```
 description::='<rdf:Description'
  idAboutAttr? bagIdAttr? propAttr* '/>'
  | '<rdf:Description' idAboutAttr?
    bagIdAttr? propAttr* '>'
    propertyElt* '</rdf:Description>'
  | typedNode


description(_) -->
 halfSTG('Description',NS),
 (idAboutAttr(IdAboutAttr);[]),
 (bagIdAttr(BagIdAttr);[]),
  propAttrStar(IdAboutAttr),
  (['/>']
   ;
   ['>'],
   propertyEltStar(IdAboutAttr),
   fullETG('Description',NS))
   ;
   typedNode.
```

The generation part consists of semantic actions inserted as appropriately in the right-hand side of DCG rules. The semantic actions are used to interpret the syntactic structures found in the right-hand side of a DCG rule during the course of parsing the input. For example, a semantic action, `getAllTriples(Obj)`, is inserted in the following DCG rule at the point where a container or a description is found.

```
obj(Obj) -->
 container(Obj)
    ;
 description(_),{getAllTriples(Obj)}.
```

The resulting flex instances are stored in a repository according to the classes they belong to. The repository is a directory structure that is built on the basis of the class hierarchy found in ontologies. The root of the directory corresponds to the root class in the ontology, for example. A directory from the root directory is created for each class in the ontology. Each directory in the repository contains a file that is used to store the transformed instances of that class. Thus when an RDF instance is successfully transformed into its flex counterpart, the result is appended to the file in the corresponding directory.

4.2 Discovery Service Programs

At present, we have implemented two discovery service programs for used to access the content stored in the repository: conceptual search and semantic navigation. The conceptual search program accepts the form, $class[A_1 : V_1, A_2 : V_2, \cdots]$, as the terms of search, where $class$ is the name of class to be searched, and $A_i : V_i$'s are the attribute-value pairs used to constrain the space of the class. The above form can be used in a flexible way: only the class name is required, and others are optional. User can specify class name with certain attributes, or more specifically, by adding values of the attributes. A browser-based GUI for the conceptual search service can be found at the URL, `http://swportal.cse.ttu.edu.tw/concept.htm`.

In addition to the conceptual search, we also provide a navigation interface to give user a whole view of the content of the repository according to the concept hierarchy of ontology. User can use the navigation interface in an way similar to directory expansion to obtain the target concepts she is interested in. The URL of semantic navigation service is:

5 Conclusions

In this paper we build a semantic web portal using a frame-based system. The system consists of a knowledge repository, service programs and an inference engine. The knowledge repository, a directory-based structure, stores the instances of knowledge transformed from RDF instances. The service programs, currently conceptual search and semantic navigation, are written using the reasoning formalisms of the inference engine. We have tested the functions of the portal by using some ontologies obtained from the

KA2 project[26]. Also we have created a metadata class under the root class of our ontology to populate popular metadata formats, such as DC[19] and RSS[10], used in digital libraries. At the moment, the service programs work well. However, we need to gather more data from other digital library sites to see how it performs when the knowledge repository scales up. Currently we do not deal with the namespaces found in RDF document. We will extend the naming convention in current system to fit the namespaces. We are developing other service programs of the portal such as personalization and social interaction.

References

[1] National Digital Archives Program, http://www.ndap.org.tw/.

[2] Open Directory Project. http://dmoz.org/.

[3] SW Annotation and Authoring. http://annotation.semanticweb.org/.

[4] Anupriya Ankolenkar, et al. DAML-S: web service description for the Semantic Web. In *The Semantic Web - ISWC 2002*. Springer. 2002.

[5] R. Benjamins, D. Fensel, S. Decker, and Gomez-Perez. KA2: building ontologies for the Internet: a mid term report. *International Journal of Human Computer Studies*. pp. 687-712. 1999.

[6] D. Box et al. Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. W3C.

[7] I. Bratko. *Prolog Programming for Artificial Intelligence, 3rd ed.*. Addison-Wesley. 2001.

[8] D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft 12 November 2002. W3C.

[9] D. L. McGuinness. Ontologies come of age. In Dieter Fensel, Jim Hendler, Henry Lieberman, and Wolfgang Wahlster, editors. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. MIT Press, 2002.

[10] RDF Site Summary (RSS) 1.0. http://web.resource.org/rss/1.0/.

[11] R. Scott Cost et al. ITtalks: a case study in the Semantic Web and DAML+OIL. *IEEE Intelligent Systems Special Issue*. 2002.

[12] S.Staab et al. AI for the web - ontology-based community web portals. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence*, Austin, Texas, USA. 2000.

[13] S.Staab et al. Semantic community web portals. *WWW9*. Amsterdam. 2000.

[14] D. Fulker and G. Janee. Components of an NSDL architecture: technical scope and functional model. *JCDL '02*. ACM. 2002.

[15] A. Gupta, B. Ludascher and R. W. Moore. Ontology services for curriculum development in NSDL. *JCDL '02*. ACM. 2002.

[16] F. van Harmelen, P. F. Patel-Schneider and I. Horrocks (eds.). Reference description of the DAML+OIL (March 2001) ontology markup language. http://www.daml.org/2001/03/reference.

[17] I. Horrocks, F. van Harmelen and P. Patel-Schneider. DAML+OIL. DAML Program. 2001.

[18] Amzi! inc. *Building Expert Systems in Prolog*. Springer-Verlag. 1989.

[19] S. Kokkelink and R. Schwanzl. Expressing Qualified Dublin Core in RDF / XML. Dublin Core Metadata Initiative. 2002. http://dublincore.org/documents/dcq-rdf-xml/.

[20] O. Lassila and D. L. McGuinness. The role of frame-based representation on the Semantic Web. KSL Techical Report, KSL-01-02. Stanford University. 2002.

[21] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation 22 February 1999.

[22] K. R. McKeown. *Text Generation*. Cambridge University Press. 1985.

[23] N. F. Noy and D. L. McGuinness. Ontology development 101: a guide to creating your first ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. 2001.

[24] A. Powell and L. Lyon. The DNER technical architecture: scoping the information environment. 2001. http://www.ukoln.ac.uk/distributed-systems/jisc-ie/arch/dner-arch.html.

[25] R. Studer, V. R. Benjamins and D. Fensel. Knowledge engineering: principles and methods. *Data Knowledge Engineering*. 1998.

[26] Y. Sure. KA2 - Knowledge Acquisition Community Ontology. http://ontobroker.semanticweb.org/ontos/ka2.html.

[27] P. Vasey. *flex Expery System Toolkit, version 1.2*. Logic Programming Associates Ltd. London, England. 1989.

[28] C. L. Yeh and C. G. Chen. Design and implementation of an ontology-based web portal. *Proceedings of the First Workshop of Digital Archive Technology*. Taipei, Taiwan. 2002.

[29] C. L. Yeh and R. F. Lin. Design and implementation of an RDF triple store. *Proceedings of the First Workshop of Digital Archive Technology*. Taipei, Taiwan. 2002.

[30] C. L. Yeh. Development of an Ontology-Based Portal for Digital Archive Services. *Proceedings of the International Conference on Digital Archive Technologies (ICDAT2002)*, Academia Sinica, Nankang, Taipei, Taiwan. 2002.

[31] The Protege Project. http://protege.stanford.edu/.

[32] Carl Lagoze, et al. The open archives initiative protocol for metadata harvesting. OAI. Jun. 2002.