<Workshop>Reasoning about RDF Elements

Reasoning about RDF Elements

Vilas Wuwongse[1], Chutiporn Anutariya[1] and Ekawit Nantajeewarawat[2]
Email: vw@cs.ait.ac.th, ca@cs.ait.ac.th and ekawit@siit.tu.ac.th

[1]Computer Science & Information Management Program
School of Advanced Technologies
Asian Institute of Technology
Pathumtani 12120, Thailand

[2]Department of Information Technology
Sirindhorn International Institute of Technology
Thammasat University
Pathumtani 12120, Thailand

Abstract

This paper proposes a theoretical framework for reasoning about RDF elements by employment of the theory of Declarative Programs. With reasoning capability, rules can be defined in order to explicitly and formally state semantic relationships between RDF elements, which will enable inference or derivation of new RDF elements from existing ones. Hence, in addition to computation by pattern matching, the proposed framework can formulate and handle complicated computations with RDF elements. Application of this framework to Web-based resource discovery problems is presented with an example.

Keywords: RDF elements, declarative programs, resource discovery problems.

1 Introduction

The recent expansion in the use of the Internet and the World Wide Web as information distribution media has made available huge resources in electronic form, so that today's Web resembles a gigantic library of books, pictures, songs, movies, weather forecasts, horoscopes, yellow pages, etc. The development of an effective information retrieval technique has become one of the major issues in view of the fact that almost all presently available search engines merely employ techniques which simply match words or sentences in documents. Users are often annoyed at receiving several thousands hits after just typing in a few keywords. Moreover, these search results still do not present users with enough information to determine their relevance, so that, in turn, in most cases further exploration is required.

A solution, proposed to date for coping with these limitations on the Web, provides sufficient descriptions, known as *metadata*. A search by means of metadata is expected to be more efficient and more accurate, since users can precisely formulate more specific queries which will yield more precise answers. For example, a user may want to find any items authored by John Smith, i.e., only items authored by John Smith rather than all items containing the words John Smith.

Relying on the concepts of metadata, the *Resource Description Framework (RDF)* [5], a collaborative design effort under the auspices of the W3C (World Wide Web Consortium), has been developed, in order to provide a unified framework for processing metadata. RDF metadata can be adopted in a wide range of application areas including resource discovery, digital library and electronic commerce.

This paper attempts to develop a framework and technique for reasoning about RDF elements. With this capability, one can define rules, which describe semantic relationships between RDF elements, and will be able to derive new RDF elements from existing ones even though they have not previously explicitly stated. In addition, application of the proposed framework to resource discovery problems allows to incorporate reasoning or deductive capabilities into the retrieval process.

The proposed framework employs *Declarative Program* (DP) Theory [1, 2, 3] - a generalization of the conventional logic program theory, in which terms are generalized into any *data objects* and substitutions into *specializations*. In particular, it has been carefully defined with generality and applicability to data structures of any domains, each of which is characterized by a mathematical structure, called a *specialization system*.

Section 2 introduces the DP theory, Section 3 develops a specialization system for RDF elements and RDF Declarative Programs, Section 4 presents an application of RDF Declarative Programs to Web-based Resource Discovery Problems with an example and Section 5 draws conclusion.

2 Declarative Program Theory

Certain fundamental definitions of declarative program theory [3] will be recalled.

2.1. Specialization Systems

A *specialization system* is an abstract structure derived from the generalization of *substitutions* in the conventional logic programs, and defined in terms of certain very simple axioms:

Definition 1 (Specialization System)

A *specialization system* is a four-tuple $\Gamma = < \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$ of three sets $\mathcal{A}$, $\mathcal{G}$, $\mathcal{S}$ and a mapping μ from $\mathcal{S}$ to the set of all partial mappings on $\mathcal{A}$, i.e., μ: $\mathcal{S} \rightarrow$ *partial_map*($\mathcal{A}$), that satisfies the requirements:

1. $\forall s_1, s_2 \in \mathcal{S}\ \exists s \in \mathcal{S}: \mu(s) = \mu(s_1) \circ \mu(s_2)$,
2. $\exists s \in \mathcal{S}\ \forall a \in \mathcal{A}: \mu(s)(a) = a$,
3. $\mathcal{G} \subset \mathcal{A}$,

where $\mu(s_1) \circ \mu(s_2)$ is the composite mapping of the partial mappings $\mu(s_1)$ and $\mu(s_2)$. The elements of $\mathcal{A}$, $\mathcal{G}$ and $\mathcal{S}$ are called *objects*, *ground objects*, and *specializations*, respectively, the set $\mathcal{G}$ the domain, and the specializations, which satisfy the second requirement, *identity specializations*. □

Intuitively, Conditions 1 to 3 mean that

1. for all specializations s_1 and s_2, there exists a specialization s such that the corresponding partial mapping of s is the composition of the two mappings corresponding to s_1 and s_2,
2. there is a specialization that does not change any objects, and
3. ground objects are objects.

For $\theta \in \mathcal{S}$, postfix notation allows to represent $\mu(\theta)(a)$ as $a\theta$. If b exists such that $\mu(\theta)(a) = b$, θ is said to be applicable to a, and a specialized to b by θ. A specialization $\theta \in \mathcal{S}$ is applicable to a subset B of A if θ is applicable to any element of B; $B\theta$ is then defined by $B\theta = \{b\theta \mid b \in B\}$. Henceforth, any condition that includes $a\theta$ or $B\theta$ will also include the condition that θ is applicable to a or B.

Definition 2 (Mapping yielding Sets of all Ground Objects)

A mapping *rep*: $\mathcal{A} \rightarrow$ *powerset*($\mathcal{G}$) is defined by $rep(a) = \{g \mid g = a\theta \in \mathcal{G}, \theta \in \mathcal{S}\}$. □

2.2. Declarative Programs

Declarative programs and other related concepts can now be defined in terms of a specialization system $\Gamma = < \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$.

Definition 3 (Declarative Program)

Let X be a subset of $\mathcal{A}$. A *definite clause* on X is a formula of the form:

$$H \leftarrow B_1, B_2, ..., B_n$$

where $H, B_1, B_2, ..., B_n$ are *objects* in X, often referred to as *atoms*. H is called the *head* and $(B_1, B_2, ..., B_n)$ the *body* of the *definite clause*. The set of all *definite clauses* on X is denoted by *Dclause*(X). A *declarative program* on X is a (possibly infinite) set of *definite clauses* on X. A *declarative program* on $\mathcal{A}$ is also called a *declarative program* on Γ. □

A definite clause will often be called simply a clause. Let C be a definite clause. The head of C will be denoted by *head*(C), and the set of all atoms in the body of C by *body*(C). A clause C is called a *unit clause* if *body*(C) is empty.

Let C be a definite clause $C = (H \leftarrow B_1, B_2, ..., B_n)$. When $\theta \in \mathcal{S}$ is applicable to $H, B_1, B_2, ..., B_n$, a definite clause $C\theta = (H\theta \leftarrow B_1\theta, B_2\theta, ..., B_n\theta)$ is obtained. A definite clause C' is an instance of C iff there is a specialization θ such that $C' = C\theta$. A definite clause C is a ground definite clause iff $C \in Dclause(\mathcal{G})$. A ground instance of a clause C is a ground clause which is an instance of C. Let P be a declarative program on

Γ. Denote the set of all ground instances of definite clauses in P by *Gclause(P)*.

2.3. Semantics of Declarative Programs

The mapping T_P: $2^{\mathcal{G}} \rightarrow 2^{\mathcal{G}}$, defined for a program P on Γ, will be used to define the declarative semantics of a program in Definition 5.

Definition 4 (Mapping T_P)

Let $\Gamma = < \mathcal{A}, \mathcal{G}, \mathcal{S}, \mu >$ and $I \subset \mathcal{G}$. A mapping T_P: $2^{\mathcal{G}} \rightarrow 2^{\mathcal{G}}$ is defined by

$T_P(I) = \{head(C) \mid C \in Gclause(P), body(C) \subset I\}$. □

By means of the mapping T_P, define declarative semantics of a program P:

Definition 5 (Declarative Semantics of a Program)

Let P be a program on Γ. The declarative semantics of P, $M(P)$, is defined by

$M(P) = \bigcup_{n=1}^{\infty} [T_P]^n(\varnothing)$, where ∅ is the empty set. □

3 RDF Declarative Programs

3.1. Basic Patterns of RDF Elements

In the RDF Model and Syntax Specification proposed by the W3C, RDF elements are defined either in Serialization or Abbreviated Syntax, both of which are based on XML [4]. Serialization Syntax can express the full capabilities of the RDF data model, while Abbreviated Syntax includes additional grammar, in order to provide a more compact form for representation of a subset of the data model. Only Serialization Syntax will be considered in this paper.

It is assumed here that the namespace name for the RDF schema has been declared and abbreviated as `RDF`. Examples of RDF tag names are `RDF:RDF`, `RDF:Description`, `RDF:Seq`, `RDF:Bag` and `RDF:Alt`.

RDF elements have the form `<RDF:RDF> content </RDF:RDF>`, where `content` is an *RDF term* or a sequence of RDF terms. By convention, an RDF term assumes one of the forms:

(a) *empty form*

$< t\ b_1="c_1"\ \ldots\ b_n="c_n"\ />$

(b) *simple form*

$<t\ b_1="c_1"\ \ldots\ b_n="c_n">\ c_{n+1}\ </t>$

(c) *nested form*

$<t\ b_1="c_1"\ \ldots\ b_n="c_n">a_1\ a_2\ \ldots\ a_m\ </t>$,

where t is a tag type, the b_i are all distinct attribute names, the c_i are constants and the a_i are ground RDF terms. In order to represent implicit information contained in an RDF term, an RDF term used here may carry variables. The formal definition of RDF terms (with variables) will be given in the next sub-section.

Note that RDF terms of the empty form $<t\ b_1="c_1"\ \ldots\ b_n="c_n"/>$ can be equivalently represented in the simple form $<t\ b_1="c_1"\ \ldots\ b_n="c_n">\ </t>$. In the sequel, all empty terms will be written in the simple form.

3.2. Specialization System for RDF Terms

This sub-section defines a specialization system for RDF terms, $\Gamma_R = <\mathcal{A}_R, \mathcal{G}_R, \mathcal{S}_R, \mu_R>$, which will be used in the definition of the specialization system for RDF elements in Sub-section 3.3.

In the sequel, let T be a set of *tag types*, B *attribute names* (or *qualifiers*), C *constants*, *TVAR tag-variables* (or *T-variables*), *BVAR attribute-variables* (or *B-variables*), *CVAR constant-variables* (or *C-variables*), and *PVAR attribute-value-pair-variables* (or *P-variables*). Assume that:

1. B contains an element `CONTENT`.
2. C contains ε, which denotes the empty string or white spaces.
3. No element in T begins with "`TVAR:`" and every element in *TVAR* begins with "`TVAR:`".
4. No element in B begins with "`BVAR:`" and every element in *BVAR* begins with "`BVAR:`".
5. No element in C begins with "`CVAR:`" and every element in *CVAR* begins with "`CVAR:`".
6. Every element in *PVAR* begins with "`PVAR:`".

Definition 6 (RDF term)

An *RDF term* is a formula of the form:

$$< t\ P_1 \ldots P_n > </ t >,$$

where

- t is a tag type in T or a tag variable in *TVAR*,
- $n \geq 0$ and P_i is a P-variable in *PVAR* or P_i takes the form b=c, where
  - $b \in BVAR$ and $c \in C \cup CVAR \cup \mathcal{A}_R$, or
  - $b \in B - \{\text{CONTENT}\}$ and $c \in C \cup CVAR$, or
  - $b = \text{CONTENT}$ and $c \in C \cup CVAR \cup \mathcal{A}_R$.

The order of the P_i, called *attributes* is immaterial. Duplicate attributes are not differentiated[1]. Moreover, attribute names other than CONTENT can not appear more than once in the same tag. □

Note that:

1. Terms taking the form of:

   $<t\ P_1 \ldots P_{i-1}$ CONTENT="c" $P_{i+1} \ldots P_n> </t>$, where $c \in C$, and

   $<t\ P_1 \ldots P_{i-1}$ CONTENT=a $P_{i+1} \ldots P_n> </t>$, where $a \in \mathcal{A}_R$,

   are normally written as:

   $<t\ P_1 \ldots P_{i-1}\ P_{i+1} \ldots P_n>\ c\ </t>$, and

   $<t\ P_1 \ldots P_{i-1}\ P_{i+1} \ldots P_n>\ a\ </t>$,

   respectively. RDF terms taking the first form are called *simple terms* and those in the second form are *nested terms*. In addition, simple terms with no attributes or those taking the form

   $< t > </t >$

   are said to be in *atomic form* and called *atomic terms*.

2. The distinctions between T-variables, B-variables, C-variables and P-variables are:
   - T-variables are those that start with the prefix "TVAR:" and can only be specialized to RDF terms in $\mathcal{A}_R$,
   - B-variables are those that start with the prefix "BVAR:" and can only be specialized to attribute names in B,
   - C-variables are those that start with the prefix "CVAR:" and can only be specialized to constants in C, and
   - P-variables are those that start with the prefix "PVAR:" and can only be specialized to sets of P-variables in 2^{PVAR} or pairs of attribute and value.

Definition 7 ($\mathcal{A}_R$, the set of RDF terms)

$\mathcal{A}_R$ is the set of all *RDF terms* constructed from the elements in T, B, C, *TVAR*, *BVAR* and *PVAR*. □

Definition 8 (RDF tree)

An RDF tree is an equivalent graphical representation of an RDF term

$$<t\ P_1 \ldots P_n> </t>,$$

where t is shown as an ellipse node and the P_i as immediate subtrees of the node t denoted by sub_i (see Figure 1-a); if

1. P_i is a P-variable, the subtree representing P_i will consist of a single circle node representing P_i (see Figure 1-b),
2. P_i takes the form b=c,
   2.1. if c is a constant or a C-variable, the subtree representing P_i has as its root a circle node representing b, connected to a child rectangle node representing c (see Figure 1-c),
   2.2. if c is an RDF term, the subtree representing P_i has as its root a circle node representing b, connected to an RDF tree representing the term c (see Figure 1-d).

Denote the RDF tree of the term a by *tree*(a). □

By the definition of RDF trees, a particular RDF tree, representing the term a, is considered to be a 3-leveled tree, which contains

- at the first level, an ellipse node, representing the tag type or T-variable of the term a,
- at the second level, circle nodes representing attribute names, B-variables and P-variables of a,

[1] Note that two attributes $b = v_1$ and $b = v_2$, where $v_1 \neq v_2$ are not considered to be duplicate.

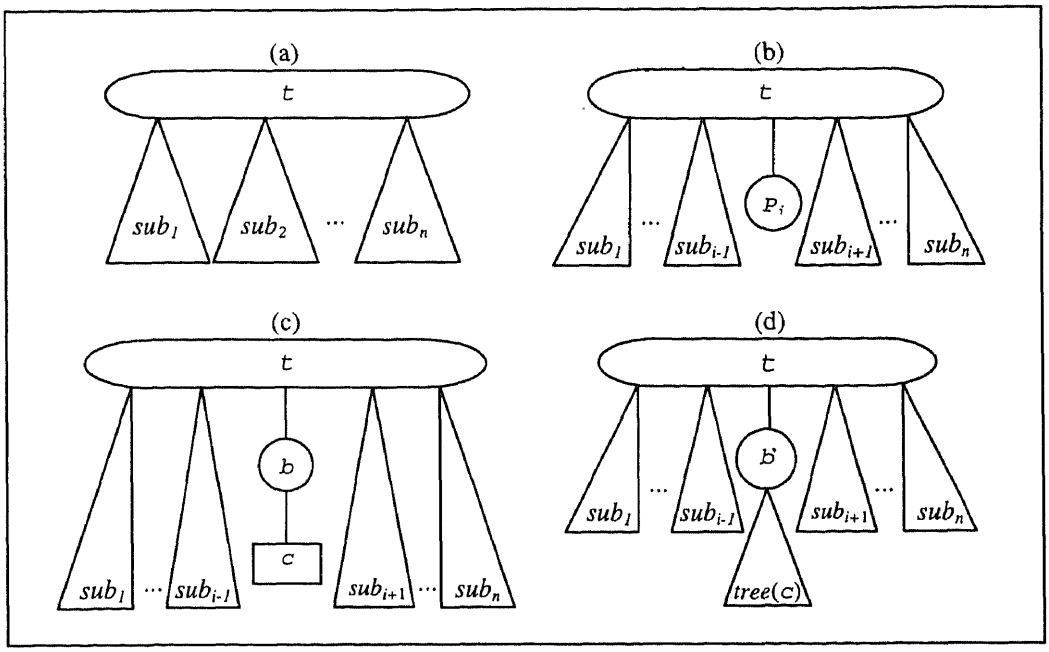

Figure 1. The RDF tree.

- at the third level, rectangle nodes and immediate subtrees, which, respectively, represent atomic values, C-variables and other terms nested in *a*; in particular, these rectangle nodes and subtrees provide corresponding values for the attribute names and *B*-variables at the second level.

Definition 9 ($\mathcal{G}_R$, the set of ground RDF terms)

$\mathcal{G}_R$ is that subset of $\mathcal{A}_R$ which consists of all ground (variable-free) terms in $\mathcal{A}_R$. □

In other words, $\mathcal{G}_R$ is the set of ground RDF terms, assuming one of the forms:

<t b_1="c_1" ... b_n="c_n" CONTENT="c_{n+1}"> </t>
or
<t b_1="c_1"... b_n="c_n" CONTENT=g_1...CONTENT=g_m> </t>,

where

- t is a tag type in *T*,
- the b_i are all distinct attribute names in *B* – {CONTENT},
- the c_i are constants in *C*, and
- the g_i are RDF terms in $\mathcal{G}_R$.

The following notion of an expandable term will be used in the definition of the specialization mapping, ν_R (Definition 11).

Definition 10 (Expandable term)

Let the EXP attribute be the attribute-value pair written as EXP="here". Define an *expandable term* as a term *t* in $\mathcal{A}_R$ which contains the EXP attribute. Then, let $\mathcal{H}_R$ be the subset of $\mathcal{A}_R$ that consists of all expandable terms in $\mathcal{A}_R$. □

Definition 11 (ν_R, a specialization mapping)

Let $\mathcal{L}_R$ be $(BVAR \times (B - \{\text{CONTENT}\})) \cup (BVAR \times \{\text{CONTENT}\}) \cup (CVAR \times C) \cup (PVAR \times (BVAR \times CVAR)) \cup (PVAR \times (\{\text{CONTENT}\} \times \mathcal{A}_R)) \cup (PVAR \times 2^{PVAR}) \cup (TVAR \times T) \cup (TVAR \times \mathcal{H}_R) \cup (TVAR \times \{\varepsilon\})$, where 2^{PVAR} is the power set of *PVAR*. The specialization mapping ν_R: $\mathcal{L}_R \rightarrow partial\_map(\mathcal{A}_R)$ is defined as follows:

1. Attribute Name Restriction

- When $e = (bvar, b) \in BVAR \times (B - \{\text{CONTENT}\})$,

if the values corresponding to *bvar*, for all occurrences of *bvar* in *a*, are constants or C-variables, then

$\nu_R(e)(a) = a' \in \mathcal{A}_R$,

where a' is obtained by replacing all occurrences of *bvar* in *a* by *b*;

otherwise $\nu_R(e)$ is not applicable to *a*.

- When $e = (bvar, b) \in BVAR \times \{\texttt{CONTENT}\}$,

  $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

  where a' is obtained by replacing all occurrences of *bvar* in *a* by *b*.

2. *Constant Restriction*

   - When $e = (cvar, c) \in CVAR \times C$,

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *cvar* in *a* by "*c*".

3. *Attribute-Value-Pair Restriction*

   - When $e = (pvar, (bvar, cvar)) \in PVAR \times (BVAR \times CVAR)$,

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *pvar* in *a* by the pair *bvar=cvar*.

   - When $e = (pvar, (b, a'')) \in PVAR \times (\{\texttt{CONTENT}\} \times \mathcal{A}_R)$,

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *pvar* in *a* by the pair $b=a''$.

4. *P-variable Expansion*

   - When $e = (pvar, \{pvar_1, \ldots, pvar_n\}) \in PVAR \times 2^{PVAR}$,

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *pvar* in *a* by the sequence of $pvar_1$, $pvar_2$, ..., and $pvar_n$ separated by whitespaces.

5. *Tag Type Restriction*

   - When $e = (tvar, t) \in TVAR \times T$,

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *tvar* in *a* by *t*.

6. *Term Restriction*

   - When $e = (tvar, a'') \in TVAR \times \mathcal{A}_R$,

     if all *tvar* terms[2] nested in *a* (in any level) only occur in atomic form, then

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing all occurrences of *tvar* terms, written as "<*tvar*> </*tvar*>", in *a* by the term a'';

     otherwise $\nu_R(e)$ is not applicable to *a*.

   - When $e = (tvar, h) \in TVAR \times \mathcal{H}_R$,

     if no *tvar* term nested in *a* (in any level) occurs in atomic form, then

     $\nu_R(e)(a) = a' \in \mathcal{A}_R$,

     where a' is obtained by replacing each *tvar* term nested in *a* by *h* and the EXP attribute in *h* by attributes of that *tvar* term;

     otherwise $\nu_R(e)$ is not applicable to *a*.

   - When $e = (tvar, \varepsilon) \in TVAR \times \{\varepsilon\}$,

     if all *tvar* terms nested in *a* (in any level) take the form <*tvar*> CONTENT </*tvar*>, where $\texttt{CONTENT} \in C \cup \mathcal{A}_R$, then

[2] *tvar terms* have the form <*tvar attr*$_1$ ... *attr*$_n$> </*tvar*>, where $n \geq 0$.

$\nu_R(e)(a) = a' \in \mathcal{A}_R$,

where a' is obtained by removing all occurrences of <*tvar*> and </*tvar*> from a;

otherwise $\nu_R(e)$ is not applicable to a.

□

Definition 12 ($\mathcal{S}_R$, the set of specializations, and the mapping μ_R)

Let $\mathcal{S}_R = (\mathcal{E}_R)^*$, the set of all sequences on $\mathcal{E}_R$. Based on ν_R, the mapping μ_R: $\mathcal{S}_R \rightarrow partial\_map(\mathcal{A}_R)$ is:

$\mu_R(\lambda)(a) = a$, where λ denotes the null sequence,

$\mu_R(e \cdot s)(a) = \mu_R(s)(\nu_R(e)(a))$, where $e \in \mathcal{E}_R$, $s \in \mathcal{S}_R$ and $a \in \mathcal{A}_R$.

Note that $\mu_R(s)(a)$ is defined only if all elements in s are successively applicable to a. □

Definition 13 (Specialization system for RDF terms)

The specialization system for RDF terms is $\Gamma_R = <\mathcal{A}_R, \mathcal{G}_R, \mathcal{S}_R, \mu_R>$. □

Proposition 1 (Axiomatic requirements of the specialization system for RDF terms)

The specialization system for RDF terms in Definition 13 satisfies the three requirements of specialization systems, i.e.,:

1. $\forall s_1, s_2 \in \mathcal{S}_R, \exists\, s \in \mathcal{S}_R$: $\mu_R(s) = \mu_R(s_1) \circ \mu_R(s_2)$,
2. $\exists\, s \in \mathcal{S}_R, \forall a \in \mathcal{A}_R$: $\mu_R(s)(a) = a$,
3. $\mathcal{G}_R \subset \mathcal{A}_R$. □

Proof

1. In order to verify that Γ_R satisfies the first requirement, let $s_1 = e_1\, e_2 \ldots e_n$ and $s_2 = e_1'\, e_2' \ldots e_m'$. The definition of μ_R shows that there exists $s = e_1' e_2' \ldots e_m' e_1\, e_2 \ldots e_n$ such that $\mu_R(s) = \mu_R(s_1) \circ \mu_R(s_2)$.
2. Obviously, $\mu_R(\lambda)(a) = a$, for each $a \in \mathcal{A}_R$, where λ is the null sequence.
3. Definition 9 has already states that $\mathcal{G}_R$ is a subset of $\mathcal{A}_R$. ■

3.3. Specialization System for RDF Elements

The specialization system for *RDF elements* will be constructed by means of the specialization system for *RDF* terms, defined in Sub-section 3.2.

Definition 14 ($\mathcal{A}$, the set of RDF elements)

Let $\mathcal{A}$ be the set of all formulae in form of <RDF:RDF> $t_1\ t_2 \ldots t_n$ </RDF:RDF> where t_i is an RDF term in $\mathcal{A}_R$. Elements in $\mathcal{A}$ are called *RDF elements*. □

Similar to RDF terms, RDF elements can also be represented equivalently by RDF trees. Denote the RDF tree of the element a in $\mathcal{A}$ by *tree*(a).

Definition 15 ($\mathcal{G}$, the set of ground RDF elements)

$\mathcal{G}$ is that subset of $\mathcal{A}$ which consists of all ground objects in $\mathcal{A}$. □

Definition 16 ($\mathcal{S}$, the set of specializations, and the mapping μ)

Let $\mathcal{S} = \mathcal{S}_R$. The mapping μ: $\mathcal{S} \rightarrow partial\_map(\mathcal{A})$ is defined by:

$\mu(s)$(<RDF:RDF> $t_1\ t_2 \ldots t_n$ </RDF:RDF>)
= <RDF:RDF> $\mu_R(s)(t_1)\ \mu_R(s)(t_2) \ldots \mu_R(s)(t_n)$ <RDF:RDF>,

where $s \in \mathcal{S}$ and $t_i \in \mathcal{A}_R$. □

Definition 17 (Specialization system for RDF elements)

The specialization system for RDF elements is $\Gamma = <\mathcal{A}, \mathcal{G}, \mathcal{S}, \mu>$. Elements of $\mathcal{A}$, $\mathcal{G}$ and $\mathcal{S}$ are called *atoms*, *ground atoms* and *specializations*, respectively. □

Proposition 2 (Axiomatic requirements of the specialization system for RDF elements)

The specialization system for the *RDF* elements in Definition 17 satisfies the three requirements of specialization systems, i.e.,:

1. $\forall s_1, s_2 \in \mathcal{S}\ \exists\, s \in \mathcal{S}$: $\mu(s) = \mu(s_1) \circ \mu(s_2)$,
2. $\exists\, s \in \mathcal{S}\ \forall a \in \mathcal{A}$: $\mu(s)(a) = a$,
3. $\mathcal{G} \subset \mathcal{A}$. □

Proof Obvious from the definition of Γ_R. ■

3.4. RDF Declarative Program

After the specialization system for RDF elements is defined (cf. Definition 17), the definitions of RDF definite clause, RDF declarative program and the declarative semantics of an RDF declarative program are obtained directly from Definitions 3, 4 and 5, respectively.

$P = \{\ c_1 : a_1 \leftarrow .$
$\quad c_2 : a_2 \leftarrow .$
$\quad c_3 : a_3' \leftarrow a_3.$
$\quad c_4 : a_4' \leftarrow a_4.\ \},$

where

a_1 =
```
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.example.com/resource1/">
    <DC:Title>Browsing the Internet</DC:Title>
    <DC:Creator>Scott Ring</DC:Creator>
    <DC:Subject>Internet</DC:Subject>
    <DC:Subject>WWW</DC:Subject>
    <DC:Subject>HTML</DC:Subject>
  </RDF:Description>
</RDF:RDF>,
```

a_2 =
```
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.example.com/resource2/">
    <DC:Title>Searching on the Web</DC:Title>
    <DC:Creator>John Smith</DC:Creator>
    <DC:Subject>Internet</DC:Subject>
    <DC:Subject>WWW</DC:Subject>
    <DC:Subject>Resource Discovery</DC:Subject>
  </RDF:Description>
</RDF:RDF>,
```

a_3' =
```
<RDF:RDF>
  <TVAR:A>
    <DC:Creator>
      <RDF:Description>
        <BIB:Name>John Smith</BIB:Name>
        <BIB:Email>john@ait.ac.th</BIB:Email>
        <BIB:Affiliation>Asian Institute of Technology</BIB:Affiliation>
      </RDF:Description>
    </DC:Creator>
  </TVAR:A>
</RDF:RDF>,
```

a_3 =
```
<RDF:RDF>
  <TVAR:A>
    <DC:Creator>John Smith</DC:Creator>
  </TVAR:A>
</RDF:RDF>,
```

a_4' =
```
<RDF:RDF>
  <TVAR:B>AIT</TVAR:B>
</RDF:RDF>,
```

a_4 =
```
<RDF:RDF>
  <TVAR:B>Asian Institute of Technology</TVAR:B>
</RDF:RDF>.
```

Figure 2. Declarative Program *P*.

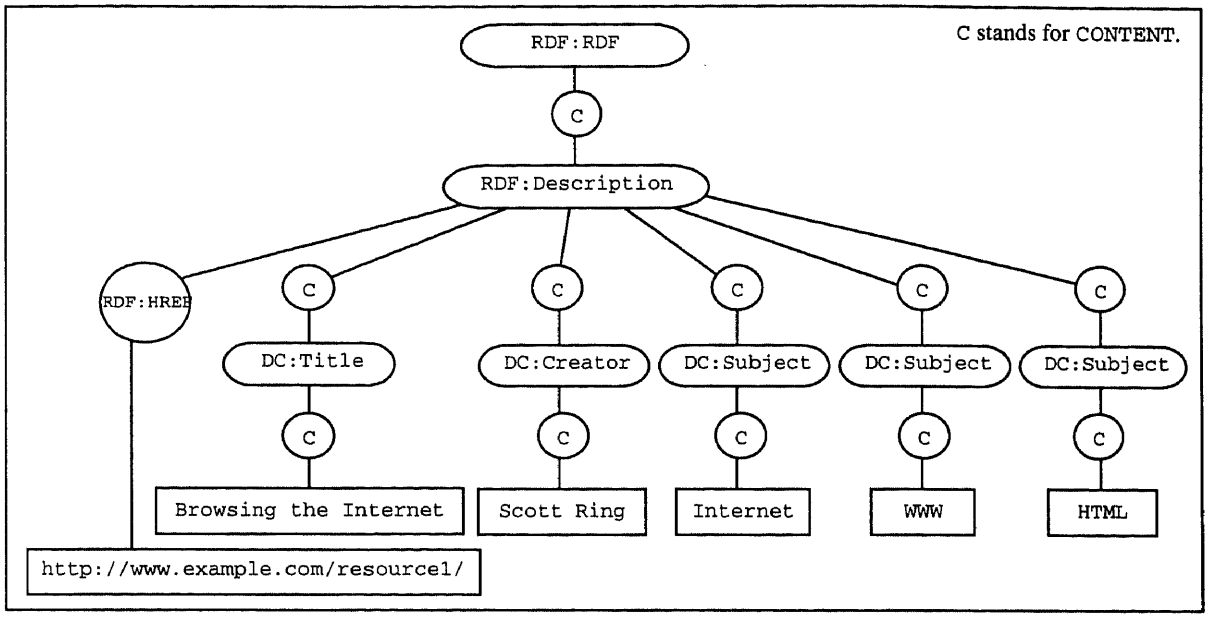

Figure 3. *tree*(a_1), the RDF tree of atom a_1.

4 Application to Web-based Resource Discovery Problems: An Example

A Web resource described by *RDF* metadata can be represented directly as a ground *RDF* element in $\mathcal{G}$. In addition to these representations, rules can be used to explicitly define (possibly complex) relationships between *RDF* elements. A rule is simply written as an *RDF* definite clause *C*, where *head*(*C*) is true, if all atoms in *body*(*C*) are true, whence a collection of Web resources can be specified by an RDF declarative program *P* which comprises unit clauses, representing selected Web resources, and non-unit clauses, representing rules. The declarative meaning of *P* then yields the set of ground atoms each of which represents one Web resource in the specified collection. A query, which expresses a user's information need (possibly implicitly) is represented by an atom in $\mathcal{A}$, and the result of this query will be the set of Web resources which can be specialized from the query and is contained in the declarative meaning of *P*.

Based on these modeling concepts, a resource discovery problem is formulated as an intersection problem [2] between the set of RDF elements that describe Web resources and the set of RDF elements that a query represents.

Example: Let a declarative program *P* on Γ, which specifies a collection of Web resources, be defined in Figure 2. Unit clauses c_1 and c_2 represent RDF elements describing Web resources, non-unit clause c_3 defines a rule which provides additional bibliographic information of John Smith and non-unit clause c_4 gives the abbreviated name of Asian Institute of Technology. In addition to these rules, which are intended to be simple for illustration, complex rules can also be defined when one has to deal with more complicated relationships between RDF elements. RDF trees representing those atoms in *P* are depicted in Figures 3 - 6.

A query which finds **only** documents whose authors' affiliations are AIT and contain information about Internet can be formulated as an atom $q \in \mathcal{A}$, i.e.,:

```
q = <RDF:RDF>
      <RDF:Description PVAR:C>
        <DC:Creator>
          <RDF:Description PVAR:D>
            <BIB:Affiliation>
              AIT
            </BIB:Affiliation>
          </RDF:Description>
        </DC:Creator>
        <DC:Subject>
          Internet
        </DC:Creator>
      </RDF:Description>
    </RDF:RDF>.
```

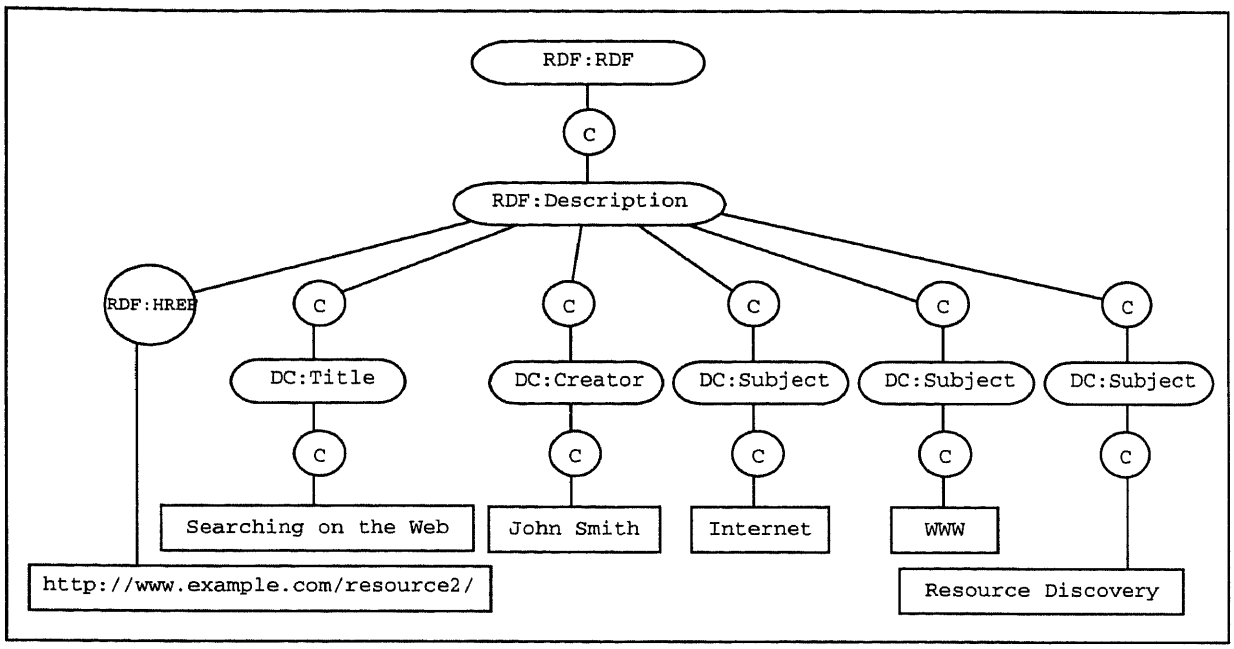

Figure 4. *tree*(a_2), the RDF tree of atom a_2.

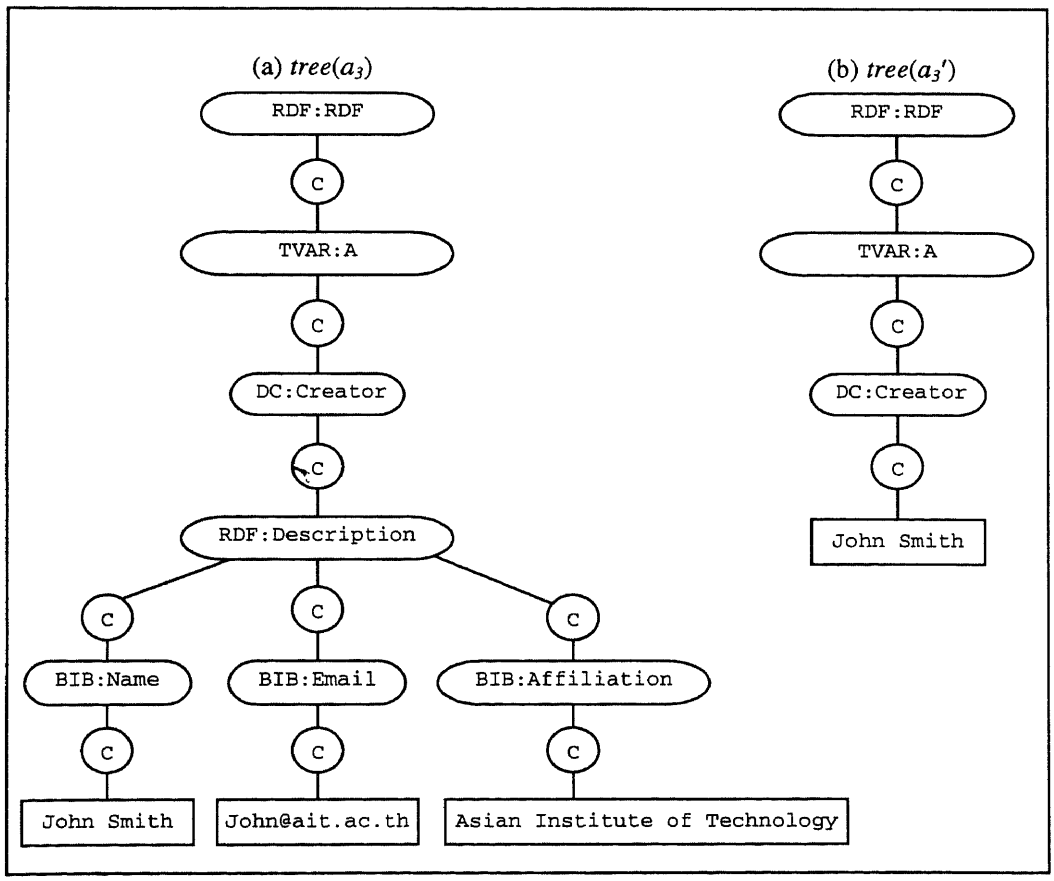

Figure 5. *tree*(a_3) and *tree*(a_3'), the RDF trees of atoms a_3 and a_3'.

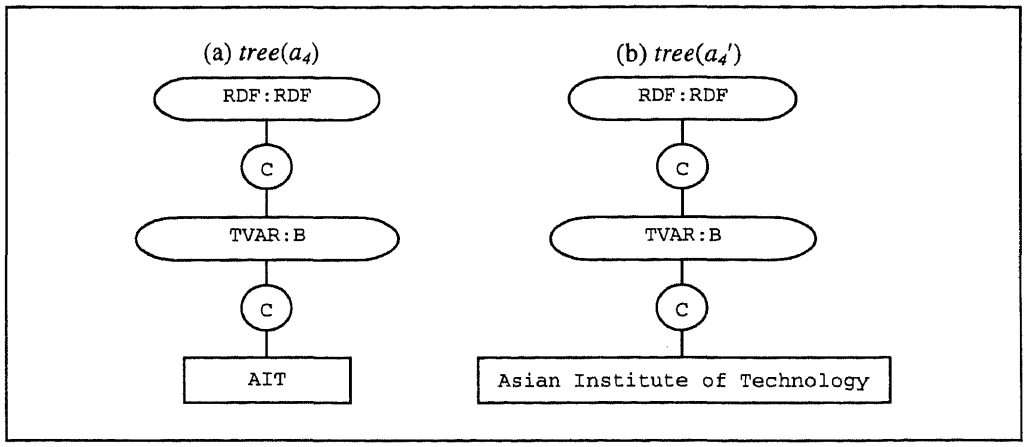

Figure 6. *tree*(a_4) and *tree*(a_4'), the RDF trees of atoms a_4 and a_4'.

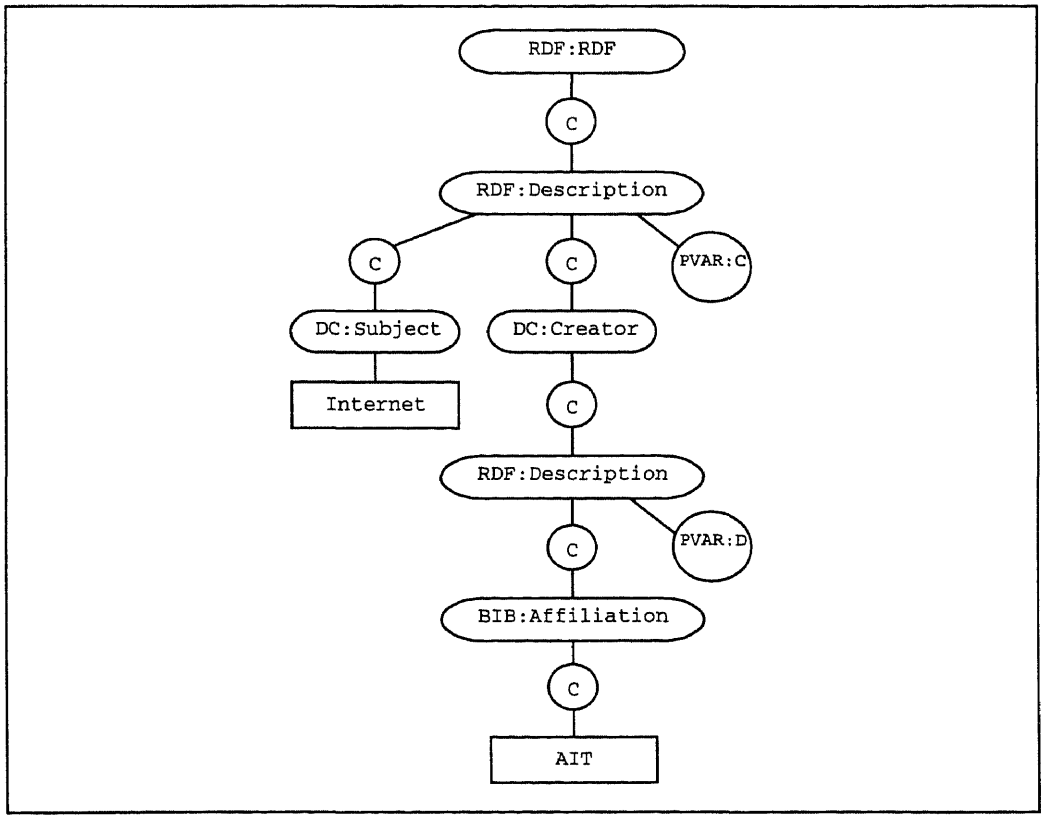

Figure 7. Query atom q and *tree*(q), the RDF tree of atom q.

Figure 7 illustrates the RDF tree of the query atom q. In processing such a query, first, define atoms a_2' and $a_2'' \in \mathcal{G}$ as in Figure 8. Recalling the definition of T_P (cf. Definition 4), one obtains:

$T_P^1(\varnothing) = \{ a_1, a_2 \}$,
$T_P^2(\varnothing) = \{ a_1, a_2, a_2' \}$,
$T_P^3(\varnothing) = \{ a_1, a_2, a_2', a_2'' \}$, and
$T_P^4(\varnothing) = T_P^3(\varnothing)$.

a_2' =
```
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.example.com/resource2/">
    <DC:Title>Searching on the Web</DC:Title>
    <DC:Creator>
      <RDF:Description>
        <BIB:Name>John Smith</BIB:Name>
        <BIB:Email>john@ait.ac.th</BIB:Email>
        <BIB:Affiliation>Asian Institute of Technology</BIB:Affiliation>
      </RDF:Description>
    </DC:Creator>
    <DC:Subject>Internet</DC:Subject>
    <DC:Subject>WWW</DC:Subject>
    <DC:Subject>Resource Discovery</DC:Subject>
  </RDF:Description>
</RDF:RDF>
```

a_2'' =
```
<RDF:RDF>
  <RDF:Description RDF:HREF="http://www.example.com/resource2/">
    <DC:Title>Searching on the Web</DC:Title>
    <DC:Creator>
      <RDF:Description>
        <BIB:Name>John Smith</BIB:Name>
        <BIB:Email>john@ait.ac.th</BIB:Email>
        <BIB:Affiliation>AIT</BIB:Affiliation>
      </RDF:Description>
    </DC:Creator>
    <DC:Subject>Internet</DC:Subject>
    <DC:Subject>WWW</DC:Subject>
    <DC:Subject>Resource Discovery</DC:Subject>
  </RDF:Description>
</RDF:RDF>
```

Figure 8. Atoms a_2' and a_2''.

Hence, the declarative meaning of P (cf. Definition 5), $M(P)$, includes atoms a_1, a_2, a_2' and a_2''. Since query atom q can be specialized to atom a_2'', i.e., there exists $\theta \in \mathcal{S}$ such that $q\theta = a_2''$, the resource whose URI is `http://www.example.com/resource2/` will be retrieved.

5 Conclusions

The RDF is a kind of knowledge/information representation which has no inference mechanism; hence computation with its elements is very limited. Apart from computation by pattern matching, other means of computation is difficult to devise under the present RDF form. This paper has proposed and developed a theoretical foundation upon which reasoning with RDF elements can be carried out. Consequently, complicated computations with RDF elements become possible. Complex queries, like those of SQL in databases, about contents of the resource described by RDF elements can be formulated and handled. This is part of the future research work.

References

1. Akama, K., 1993. Declarative Semantics of Logic Programs on Parameterized Representation Systems. *Advances in Software Science and Technology*, 5:45-63.

2. Akama, K., Kawaguchi, Y. and Miyamoto, E., 1998. Solving intersection problems using equivalent transformation (in Japanese). *Journal of the Japanese Society of Artificial Intelligence* (submitted).

3. Akama, K. and Wuwongse, V., 1997. Equivalent Transformation for Constrained Declarative Programs on String Domains. *Technical Report, Computer Science and Information Management Program, Asian Institute of Technology, Bangkok, Thailand.*

4. Bray, T., Paoli, J. and Sperberg-McQueen, C.M., "Extensible Markup Language (XML) 1.0", Feb 1998. http://www.w3.org/TR/REC-xml/.

5. Lassila, O. and Swick, R.R., "Resource Description Framework (RDF) Model and Syntax", 1998. http://www.w3.org/TR/WD-rdf-syntax/