Applying Model Checking Techniques to Temporal Queries over WorldWideWeb

OYANO, Jun

1 Introduction

Through the evolution of WWW, many useful services prevailed over Internet. On the other hand, demands for new services have increased rapidly, however, it is infeasible to create new services from scratch because of time and economical reasons. To cope with the problems, eXtensible Markup Language (XML) has been proposed as general framework for exchanging and reusing existent data effectively. In addition, a new paradigm called "Semantic Web"[6] is expected to give useful semantics to XML documents. Semantic Web is constructed over Resource Description Framework (RDF) which annotates relationships among XML documents. The RDF document itself is an XML document, but it describes data about data, so we call it as Meta-data. The Semantic Web paradigm introduces some new methods for exploring the WWW space by exploiting the Meta-data, not only with the simple string pattern matching.

The RDF adopts $S-P-O$ triple as its model theory[7] , and RDF's meaning is given by Knowledge Interchange Format (KIF)[8]. KIF has an expressive power as the First Order Logic (FOL) and it can be processed by using existent proof machines because KIF has LISP-like grammar. Because KIF has a restriction against the inclusion of the Russell class, it is impossible to express temporal properties, i.e. "eventually", "always", "until", and so on.

Let us think about an e-Learning system over Internet. Suppose that a certain beginner learner has a plan to study subject A finally. A requires the knowledge of B, and B also requires C. If the learner has no experience of these subjects, now he/she should learn subject C, not A, of course. In other words, the e-Learning system should be inclusive of query methods by which learners can find the "next step" in order to achieve their goals. Here we consider the necessity of more expressive logical formula, which can describe actions generated by documents.

The FOL which has a transition relation represented by two variables, called FO^2, can involve expressive ability of CTL, PDL, mu-calculus and other temporal logics[5]. This means that by introducing a certain transition relation into RDF model with FOL semantics, you can treat RDF documents space as models of temporal logics, therefore, it will be possible to ask temporal queries over WWW and to check temporal assertions about WWW services.

But it is known that compositions of many processes cause "state explosion" problem. In addition, the decision problem of rich temporal logics such as CTL is known as PSPACE-complete[5] and looks infeasible. Nevertheless many optimizations are pre-

sented not only algorithms and also reduction of model spaces.

Exploiting abstraction of models reduces state spaces by ignoring uninteresting transitions and states of the system.

The rest of the paper starts with some preliminaries defining mu-calculus, its interpretations and the relationship with the naive checking algorithm. Section 3 introduces two interpretations for $S - P - O$ triple of the RDF Model Theory, discusses possibilities of regarding WWW as the temporal database. The compositions of processes and reducing their state spaces occupy in Section 4. The paper finishes with conclusions and future works in Section 5.

2 Preliminaries

Because computer systems play an important role in our infrastructure, the importance of verifying the correctness of system has been pointed out. Many activities for proving the required properties of concurrent systems have been tried and succeeded, especially in Hardware, cf. VLSI, Protocol Design, Bus architecture, and so on. In this section, we introduce the formal system and describe the required properties of systems.

2.1 Temporal Logics

To verify these systems, we should describe their assertions using an appropriate formal logic which can cope with various changes conditioned chronologically. However, by using the general FOL, we cannot express time varying property.

Suppose that the proposition $X = 0$ meaning a variable X has value 0 in a certain time. In general, since the value of X depends on system's situation, we cannot determine its truth value of $X = 0$ without other, especially time, informations. To cope with this problem, we can introduce a new proposition $X_t = 0$, however, it requires infinite propositional variables in one sentence, and it is impossible to handle it in the general FOL framework.

Temporal Logics were originally developed by philosophers for investigating the notion of time in natural language arguments. They are also useful for expressing temporal properties of concurrent systems because they can describe variations of the system's conditions along the element of time progress without introducing time explicitly.

We can get a rich logic system through introducing the state transition relation and new variables expressing the relation into the FOL. FO^2 is a logic system which restricts that the number of transition variables are up to two. The express power of FO^2 is powerful, but too strong sometime. This means that it is too expensive in general for its decidability. We can use $FO^2(TC)$ or $FO^2(LFP)$ instead of the general FOL. $FO^2(TC)$ and $FO^2(LFP)$ (or simply FP^2) are fragments of FO^2, and have Transitive Closure Operator (TC) and Least Fixpoint Operator(LFP) respectively. These logics have enough expressive power, particularly FP^2 has the same ability as Modal mu-calculus[5] and involves modal logics and other useful temporal logics – Computational Tree Logic (CTL), Linear Time Logic (LTL), Propositional Dynamic Logic (PDL) and so on.

For simplicity's sake, we adopt Modal mu-calculus, which is equal to FP^2, as the canonical logic in this paper.

2.2 Modal mu-Calculus

Modal mu-Calculus (mu-calculus), introduced by Kozen[2], is a powerful language. Intuitively, it has operators of general propositional logic ($\wedge, \vee, \neg$), and Box operator($[\cdot]$), Diamond operator($\langle\cdot\rangle$) for expressing the passage of time. Moreover, for a certain proposition Φ with a propositional variable Z and its equation $Z = \Phi(Z)$, we can express the least root ($\mu Z.\Phi(Z)$) and the greatest root ($\nu Z.\Phi(Z)$) with fixpoint operator μ, ν respectively. Using these fixpoint operators. we can mention infinite time progress such as 'eventually', 'always', etc.

With the mu-calculus, we can treat infinite states of transition systems, but the investigation of such systems will be nondeterministic and therefore they cannot be handled automatically. This phenomenon is not suitable for our propose: an application for the temporal database system over WWW. In this paper, we adopt the finite transition systems as our models.

2.2.1 mu-calculus: Syntax

A formula Φ of mu-calculus is constructed from a set of propositional constants $\mathcal{P}$ and its elements $P, Q, \ldots$, a set of propositional variables Vars and its elements $X, Y, Z, \ldots$, a set of labels $\mathcal{L}$ and its elements $a, b, \ldots$ as the following rule:

$$\Phi ::= P \mid Z \mid \neg\Phi_1 \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid [K]\Phi_1 \mid \langle K\rangle\Phi \mid \nu Z.\Phi_1 \mid \mu Z.\Phi_1$$

Note that K is a subset of $\mathcal{L}(K \subseteq \mathcal{L})$, all occurrences of Z within Φ_1 of $\sigma Z.\Phi_1$ ($\sigma = \mu$ or ν) fall under an even number of negations in Φ for its syntactically monotone. Besides, we employ abbreviations such as $\mathbf{tt} \stackrel{\text{def}}{=} \nu Z.Z$, $\mathbf{ff} \stackrel{\text{def}}{=} \neg\mathbf{tt}$ properly. Readers can find details in other textbooks[1, 3].

Intuitively, interpretations of $\vee, \wedge, \neg$ are in the usual way, $[a]\Phi$ means that "Φ holds in all states reachable in one step by making an a-transition". Similarly, the meaning of $\langle a\rangle\Phi$ is: "it is possible to make an a-transition to a state where Φ holds". Roughly, μ, ν operators corresponds to 'eventually' and 'always' respectively.

2.2.2 Relationships with other Logics

As we mentioned in the previous section, temporal queries for Semantic Web can be expressed by the FOL a binary relation R within its vocabulary. For example, we can express the assertion $\Box\Diamond q$ in the modal logic using R as follows:

$$\forall y(R(x, y) \rightarrow \exists z(R(y. z) \wedge q(z)))$$

The FO^2 is the fragment of FOL with R, which has only two individual variables. The expressive power of mu-calculus is equal to a fragment of FO^2, that is FP^2, and involves with other useful logics.

$\langle a^*\rangle\Phi$ in PDL which means "on some a-path" can be translated as $\mu Z.\Phi \vee \langle a\rangle Z$ in mu-calculus. Similarly, $[a^*]\Phi$ that means "on all a-path" also can be translated into $\nu Z.\Phi \wedge [a]Z$.

The mu-calculus can express assertions in the CTL. which is richer than the PDL. For example, "on all paths, eventually Φ" and "on some path, Φ_1 until Φ_2" can be translated into mu-calculus formulae easily with a notice that $A \Rightarrow B$ means "A can be translated into B".

$$\mathbf{AF}\ \Phi \Rightarrow \mu Z.\Phi \vee \mathbf{AX}\ Z \Rightarrow \mu Z.\Phi \vee [-]Z$$
$$\mathbf{E}[\Phi_1 \mathbf{U} \Phi_2] \Rightarrow \mu Z.\Phi_2 \vee (\Phi_1 \wedge \mathbf{EX}Z) \Rightarrow \mu Z.\Phi_2 \vee (\Phi_1 \wedge \langle - \rangle Z)$$

2.2.3 mu-calculus: Semantics

Let $\mathcal{L}$ be a set of labels, $\mathcal{S}$ a set of states, and $\xrightarrow{a}$ a binary relation over $\mathcal{S}$ and $a \in \mathcal{L}$.

The labeled transition system $\mathcal{T}$ with $\mathcal{L}$ is given as $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} | a \in \mathcal{L}\})$. Then, the model of mu-calculus $\mathcal{M}$ is defined as a pair of a transition system $\mathcal{T}$ and a valuation function $\mathcal{V} : \mathrm{Var} \to 2^{\mathcal{S}}$ as follows:

$$\mathcal{M} \stackrel{\text{def}}{=} (\mathcal{T}, \mathcal{V})$$

The semantics of mu-calculus is given by the correspondence between the formula Φ and a set of states in which the formula Φ holds with respect to the model $\mathcal{M}$ as follows:

$$\|\Phi\|_{\mathcal{M}} \stackrel{\text{def}}{=} \{s \in \mathcal{S} \mid \mathcal{M}, s \models \Phi\}$$

From now on, we omit the notation $\mathcal{M}, \mathcal{T}, \mathcal{V}$ if it is clear from its context. Let us fix the model $\mathcal{M}$ and take care of only the valuation $\mathcal{V}$. The set $\|\Phi\|_{\mathcal{V}}$ is defined recursively as follows:

$$\|Z\|_{\mathcal{V}} = \mathcal{V}(Z) \qquad \|\neg\Phi\|_{\mathcal{V}} = \mathcal{S} - \|\Phi\|_{\mathcal{V}}$$
$$\|\Phi_1 \wedge \Phi_2\|_{\mathcal{V}} = \|\Phi_1\|_{\mathcal{V}} \cap \|\Phi_2\|_{\mathcal{V}} \qquad \|\Phi_1 \vee \Phi_2\|_{\mathcal{V}} = \|\Phi_1\|_{\mathcal{V}} \cup \|\Phi_2\|_{\mathcal{V}}$$
$$\|[K]\Phi\|_{\mathcal{V}} = \{s \in \mathcal{S} \mid \forall s' \in \mathcal{S}. \forall a \in K. \ s \xrightarrow{a} s' \Rightarrow s' \in \|\Phi\|_{\mathcal{V}}\}$$
$$\|\langle K \rangle \Phi\|_{\mathcal{V}} = \{s \in \mathcal{S} \mid \exists a \in K. \exists s' \in \mathcal{S}. \ s \xrightarrow{a} s' \wedge s' \in \|\Phi\|_{\mathcal{V}}\}$$
$$\|\nu Z.\Phi\|_{\mathcal{V}} = \bigcup\{s \subseteq \mathcal{S} \mid \|\Phi\|_{\mathcal{V}[Z:=S]} \supseteq S\} \qquad \|\mu Z.\Phi\|_{\mathcal{V}} = \bigcap\{S \subseteq \mathcal{S} \mid \|\Phi\|_{\mathcal{V}[Z:=S]} \subseteq S\}$$

Notice that we regard $\|\Phi\|_{\mathcal{V}}$ as a semantic function from a formula to a set of states which hold that formula, and it is monotonic with respect to set inclusion relation, i.e. $S \subseteq S' \Rightarrow \|\Phi\|_{\mathcal{V}[X:=S]} \subseteq \|\Phi\|_{\mathcal{V}[X:=S']}$ where $\mathcal{V}[Q := W]$ is a new valuation that is the same as $\mathcal{V}$ except that $\mathcal{V}[Q := W](Q) = W$.

This is enough to ensure the existence of the fixpoints:

$$\|\mu Z.\Phi\|_{\mathcal{V}} = \bigcup_{\alpha \in \mathbf{Ord}} \|\mu^\alpha Z.\Phi\|_{\mathcal{V}} \qquad \|\nu Z.\Phi\|_{\mathcal{V}} = \bigcap_{\alpha \in \mathbf{Ord}} \|\nu^\alpha Z.\Phi\|_{\mathcal{V}}$$

where **Ord** is the class of all ordinals.

Furthermore, it is also continuous under the premise that the model $\mathcal{M}$ is finite in this monograph. Hence the least and the greatest fixpoints can be computed by the iterative evaluation, and their iteration will stop within finite time.

$$\|\mu^0 Z.\Phi\|_{\mathcal{V}} = \emptyset \qquad \|\nu^0 Z.\Phi\|_{\mathcal{V}} = \mathcal{S}$$
$$\|\sigma^{\alpha+1} Z.\Phi\|_{\mathcal{V}} = \|\Phi\|_{\mathcal{V}[Z:=\|\sigma^\alpha Z.\Phi\|_{\mathcal{V}}]}$$
$$\|\mu^\omega Z.\Phi\|_{\mathcal{V}} = \bigcup_{\alpha < \omega} \|\mu^\alpha Z.\Phi\|_{\mathcal{V}} \qquad \|\nu^\omega Z.\Phi\|_{\mathcal{V}} = \bigcap_{\alpha < \omega} \|\nu^\alpha Z.\Phi\|_{\mathcal{V}}$$

Directly from previous definitions of the semantics of mu-formulae, we can write

down the naive algorithm for computing its meaning.

function eval$(\Phi, \mathcal{V})$
  if $\Phi = p$ **then return** $\{s | p \in L(s)\}$;
  if $\Phi = Q$ **then return** $\mathcal{V}(Q)$;
  if $\Phi = \Phi_1 \wedge \Phi_2$ **then return** eval$(\Phi_1, \mathcal{V}) \cap$ eval$(\Phi_2, \mathcal{V})$;
  if $\Phi = \Phi_1 \vee \Phi_2$ **then return** eval$(\Phi_1, \mathcal{V}) \cup$ eval$(\Phi_2, \mathcal{V})$;
  if $\Phi = \langle a \rangle \Phi'$ **then return** $\{s | \exists t[s \xrightarrow{a} t \wedge t \in \text{eval}(\Phi', \mathcal{V})]\}$;
  if $\Phi = [a]\Phi'$ **then return** $\{s | \forall t[s \xrightarrow{a} t \Rightarrow t \in \text{eval}(\Phi', \mathcal{V})]\}$;
  if $\Phi = \sigma Q.\Phi(Q)$ **then**
    if $\sigma = \nu$ **then** $Q_{\text{val}} := \mathcal{S}$; **else** $Q_{\text{val}} := \emptyset$;
    repeat
      $Q_{\text{old}} := Q_{\text{val}}$;
      $Q_{\text{val}} := \text{eval}(\Phi, \mathcal{V}[Q := Q_{\text{val}}])$;
    until $Q_{\text{val}} = Q_{\text{old}}$;
    return Q_{val};
  end if
end function

Figure 1: the naive algorithm

If you want to prove that "bad thing", expressed by Φ, will never happen in the system $\mathcal{M}$, you can check the emptiness of $\|\Phi\|_{\mathcal{M}}$. Similarly, you can claim that a certain state s has "good property", expressed by Ψ, by checking $s \in \|\Psi\|_{\mathcal{M}}$. We call this procedure as *model checking*[1].

The model checking using such the naive algorithm is called as *global* because it explores all states of the model. Because this algorithm corresponds with its semantics directly, it is intuitive. However, on verifying concurrent systems or component devices, the number of states increases too rapidly, and such phenomenon is called "state explosion" problem. To cope with the state explosion, the strategy that invests only a certain target state and its surrounding states is called *local* model checking. Since the local model checking needs only a part of the entire system, it is useful for application like an agent program which checks appropriate conditions for some particular users. In the worst case, even local model checking algorithm costs as same as the global model checking.

The global checking strategy enjoys advantages in cases we want to compute "all states which hold some properties" or to prove "there is no state falls in any bad situations".

Both global/local model checking techniques have advantages in practice, though we treat only the global strategy for the simplicity's sake. The naive global model checking costs $O(n^k)$ computations, where n is the number of states in the transition system and k is the depth of nesting of the nesting fixpoints for reasoning FO^2, CTL* and other useful temporal formulas. It tells us the naive algorithm infeasible. Emerson and Lei[9] improved the algorithm using the notion of alternation depth. Their algorithm requires $O(n^d)$ where d is alternation depth. Besides Long, Browne, Clark, Jha, and Marrero[10] developed the algorithm needs only $O(n^{d/2})$ iterations. In practice, many interesting properties of system can be expressed that in the mu-calculus formula whose alternation depth is at most 2, moreover, there is the linear algorithm[11] with the restriction that formula's alternation depth to 1. We can also use many devices of the model checking techniques and useful tools.

For the model space, it is known that Ordered Binary Decision Diagram (OBDD) makes the amount of states decrease dramatically[1]. It exploits the binary decision tree as the characteristic function of state instead of handling states themselves explicitly, and this technique is called implicit or *symbolic* model checking. It is known that the OBDD is effective especially in hardware verification because of its somewhat regularity. But if you cannot enjoy such regularity of the system, the number of decision trees will be same as the number of explicit states space in the worst case.

3 Semantic Web as Database

As the number of Internet users is exploding in the currency of WWW, requests to improve its facilities are also increasing at the same time. The Semantic Web is the representation of data on the WWW and it enables us to make queries semantically. We'll offer an idea for giving semantics of transitions to the RDF, which permits us to explore WWW using temporal formulae.

3.1 RDF Model Theory

The interpretation of Semantic Web is based on the RDF Model Theory. The model of the RDF is represented by the $S-P-O$ triple, and the Knowledge Interchange Format (KIF) defines the RDF vocabularies. S, P and O mean "Subject", "Predicate", and "Object" respectively. This triple can be described as $S \xrightarrow{P} O$ by the Graph theoretical notation, and also as $(S,O) \in P \Leftrightarrow$ "$P(S,O)$ is true" by the Predicate Logic.

For example, assume that P is "*ex:MemberOf*", S "*yanpi@cuc.ac.jp*", O "*CUC*"(1). The corresponding $S-P-O$ triple can be expressed as follows for each semantics:

$$yanpi@cuc.ac.jp \xrightarrow{ex:MemberOf} CUC \textbf{ iff } (yanpi@cuc.ac.jp, CUC) \in ex:MemberOf$$

The KIF[8] is a wide spread format especially in the fields of the Artificial Intelligence or the Knowledge Engineering. Its expressiveness is equal to the FOL and it is capable of coordinating with the existing theorem provers because of its LISP compatible format. The KIF syntax adopts the free syntax for simplification, and it means that objects, constants, functions and relation symbols have no clear distinction in the syntactical meanings. This corresponds with the fact that each of S, P, O has ability to hold its URI. Hence, it is possible to write it down self-applicable in the RDF-notation, such as $P \xrightarrow{P} O$, because this means $(P,O) \in P$, the Russell class(2). To avoid this type of confusion, the KIF adopts the notion of extensions, the predicate is restricted to be unfolded only once. This device suppresses the RDF's semantics to be equal to those of the FOL.

3.2 Interpretations of RDF Model Theory

Notice that the $S-P-O$ triple can be interpreted in two ways:

1. $(S,O) \in P$
2. $S \xrightarrow{P} O$

(1) Chiba University of Commerce

(2) Moreover, we can omit O using the empty-tag: that is $P \in P$

When we interpret the $S-P-O$ as the FOL, let us read $(S,O) \in P$ as "S's attribute P is O", i.e. $S.P = O$ in the Tuple Logic. Besides, $\mathcal{M}$ identified with some relation $\mathcal{R}$ in Relational Database (RDB) and the state S is as the tuple t of $\mathcal{R}$. Using this trick, we can identify the Tuple Logic with $S-P-O$ triple, and we can regard the model-checking over its model $\mathcal{M}$ as the RDB's queries for relation $\mathcal{R}$.

First, when O is omitted because of an empty tag of RDF, it'll be as follows:

$$S.P \Rightarrow \mathcal{M}, S \models P \ \ (\text{ that means } \ S \in \| P \|_{\mathcal{M}})$$

Next, when O is specified as P, we have the proposition $P = O$ and the appropriate correspondence:

$$S.P = O \Longrightarrow \mathcal{M}, S \models P = O \ \ (\text{ that means } \ S \in \| P_{=O} \|_{\mathcal{M}})$$

Moreover "P is less than 100", for example, is expressed as

$$S.P < 100 \Longrightarrow S \models_{\mathcal{M}} P < 100$$

In the RDB framework, results of queries using formula Φ are same as the set of tuples which make logical formula Φ true, i.e. $\{t \in \mathcal{R} \mid \Phi^*(t)\}$ where $\Phi^*(t)$ is a tuple formula which is corresponding with Φ, therefore we can identify the query in the RDB by the Tuple Logic with the model checking over the corresponding model:

$$\{t \in \mathcal{R} \mid \Phi^*(t)\} \Longrightarrow \{S \in \mathcal{S} \mid S \models_{\mathcal{M}} \Phi\}$$

On the other hand, $S-P-O$ is read as "S has a P transition to O" in the Graph Theoretical way. This means that the $S-P-O$ triple can be interpreted as an appropriate transition system (Kripke-Structure), and we can introduce temporal logics, such as mu-calculus, into the RDF models.

3.3 Extracting Courses

In this section, we extract the $S-P-O$ triples as the labeled transition system from RDF documents using certain tags, and treat the set of $S-P-O$ triples as the Kripke-Structure $\mathcal{T}$.

Let us define special tags which specify conditions of transitions. Notice that distributed documents are able only to "assert" its requirements whether the documents can be acceptable or not. After some agreement between each of documents, appropriate transitions will occur.

We introduce two special tags, "`<pre_state>`" and "`<post_state>`". The form of these tags is a pair of state and label, and therefore "`state:label`" for the simplification of our discussion.

The transition relations will be defined by pattern matching between `<pre_state>` of s' and s of `<post_state>`. We also introduce a binary operator "$\star$" to merge the assertion elements of each states. The general form of the pattern which asserts the transition condition between state A and B is as follows:

$$A \star \text{preB} \ : \ \text{labelA} \star \text{labelB} \ : \ \text{postA} \star \ B$$

where preB stands for `<pre_state>` of state B, postA is for `<post_state>` of A, and $*$ is the representation for any states and any labels. The pattern matching rule with the operator $\star$ is specified as follows:

$$s \star * \rightarrow s, \quad * \star s \rightarrow s \quad * \star * \rightarrow *$$
$$s \star s \rightarrow s, \quad s \star s'(s \neq s\prime) \rightarrow \bot$$

If the transition has $\bot$ as the value of its elements, the transition itself will be canceled. According to this rule, for example, we'll get the result B:cuc:E ($B \xrightarrow{cuc} E$) by the pattern matching, " $B \star B$: cuc $\star * :* \star E$ " where <post_state> of B is "*: cuc ", and E's <pre_state> is "$B : *$".

After the execution of the merge operation over all the documents, we obtain the transition system; $\mathcal{T} = \{s \xrightarrow{a} s' \mid s, s' \in \mathcal{S}, a \in \mathcal{L}\}$. We adopt $\mathcal{T}$ as the Kripke-Structure extracted from RDF documents.

For instance, from a certain RDF document (Figure 2), we'll acquire a transition system as Figure 3.

Now, we are going to try to make temporal queries over the Kripke-Structure that we obtained in the previous manner. Suppose that each state has its own predicate corresponding to itself; this means that we identify the state S as its predicate S. Semantically, $\| \text{ S } \|_{\mathcal{M}} = \{S\}$, therefore, the predicate S is true only in the state S.

We also premise that (A within 1) is the predicate which expresses "states which are reachable from the state A within n steps". Notice that we can define the semantics of (A within n) inductively as $\|\texttt{(A within 0)}\|_{\mathcal{M}} = \{\text{A}\}$ and

$$\|\texttt{(A within } n\texttt{)}\|_{\mathcal{M}} = \|\texttt{(A within } n-1\texttt{)}\|_{\mathcal{M}} \cup \{s' | s \rightarrow s', s \in \|\texttt{(A within } n-1\texttt{)}\|_{\mathcal{M}}\}$$

```
<curriculum>
 <subject>
   <course_data uri="A"
     course="C Programming"
     university="CUC" year="2003"/>
   <conditions>
     <pre_state>
       <cond state= "No_Such_State"
         trans= "No_Such_Trans" />
     </pre_state>
   </conditions>
 </subject>
```

Figure 2: RDF-like Document

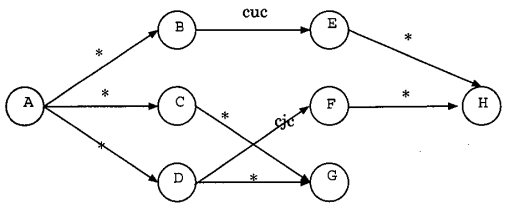

Figure 3: An Extracted Transition System

```
trans.setWorkingVal();
{ Vector oldSet;
  resultSet=new Vector();
  int index=0;
  do {
    index++;
    oldSet=
       (Vector)resultSet.clone();
    trans.setVecOfAtom("X",resultSet);
    resultSet=trans.evalUnion(
    trans.eval("G"),
    trans.evalExistTrans("-",
          trans.evalVal("X")));
  } while
    (!trans.isEqual(resultSet,oldSet));
}
printResult(resultSet,"mu X. G or <->X");
```

Figure 4: A Fragment of the Toy Program

```
mu(1) X. G or <->X: G
mu(2) X. G or <->X: G C D
mu(3) X. G or <->X: G A C D
mu(4) X. G or <->X: G A C D
mu X. G or <->X: G A C D

A within 1: A B C D

(A within 1) and (mu X. G or <->X): A C D

{(t.uri,t.university,t.course,t.year)|
  t |= (A within 1) and (mu X. G or <->X)}:
  uri | university | course | year
  A | CUC | Entrance Examinatin    | 2003
  C | CUC | Ethics of Information  | 2003
  D | CJC | Foundation of Computer | 2002
```

Figure 5: Result of Evaluation

with slight extensions of our semantics.

Let us think about a query with the next formula.

$$(\text{A within 1}) \wedge (\mu X.\text{G} \vee \langle - \rangle X)$$

The states obtained through the evaluation of the above formula would say, "The path that leads to the state G and is within 1 step from the state A", that means "The classes which are recommended to take in the next step if you would like to take the class called G finally" informally.

Here, we present the sample code in Figure 4 for computing this set with the naive algorithm in Java. It is a simple toy program using the SAX parser, and its total lines are up to 700 lines, therefore, it is feasible without the considerations of its efficiency. Intuitively, `trans` expresses $\mathcal{T}$, `evalUnion(A,B)` does $A \cup B$, `evalExistTrans("-","X")` does $\langle - \rangle X$. Using this program, we get the state C and D which fulfill the formula `"(A within 1) and (mu X. G or <->X)"`, and then we can investigate the contents of their attributes, "{uri,university,course,year}".

Consequently, by reading the appropriate $S-P-O$ triples of RDF documents as the transition system, we can enjoy the temporal formulae as the Database Query Language. We see the possibility of applying of the temporal formulae to the practical database systems.

4 Practical Applications

In the previous example, we used an e-Learning material which is written according to the RDF-like grammar. The learner can take classes only along the paths specified by the course, and it seems quite unnatural. Here, let us read "$A \rightarrow B$" in Figure 3 as "it is required to execute A before B", not as "you should execute B just after A". This means that the learners have possibilities of taking other courses, but it is necessary to keep the ordering specified by the course.

Suppose a user agent "$U(X)$" and a course agent "**Course**". The notation $U(X)$ means that the user U has the set X of "credits". And also $credit(t)$ is a function which takes the credit as its argument, and then it returns a set of names of classes one can take using that credit. We can define the user agent using the pi-calculus[4] like notation as follows.

$$U(X) \stackrel{\text{def}}{=} \Sigma_{c_t \in \cup_{x \in X} credit(x)} c_t\langle x \rangle.\overline{x}(x_{\text{new}}).U(X \cup \{x_{\text{new}}\})$$

This definition tells you that the user can take only classes corresponding the ticket are held by the user already, and he receive a new credit as the certification of the class.

As the premise of this paper, a particular course is consisted of other classes. Each of them receives the ticket and it issues a new credit at the end of that class. For example, the class B in the previous example takes the ticket generated from the A's credit x through the communication line c_A and returns the credit relevant to B (i.e t_B) through the line which is generated from the ticket x.

$$B \stackrel{\text{def}}{=} c_A(x).\overline{x}\langle t_B \rangle.B$$

The course is an aggregation constructed from these processes:

$$\textbf{Course} \stackrel{\text{def}}{=} (A \mid B \mid C \mid D \mid E \mid F \mid G)$$

Finally, we obtain the entire system from the composition of the user agent $U(X)$ and **Course**.

$$U(X) \mid \textbf{Course}$$

The set of transitions of X with respect to $U(X)$ will be as follows: $\{t_A\} \xrightarrow{B} \{t_A, t_B\}$, $\{t_A\} \xrightarrow{C} \{t_A, t_C\}$, $\{t_A\} \xrightarrow{D} \{t_A, t_D\}$, $\{t_A, t_B\} \xrightarrow{C} \{t_A, t_B, t_C\}$, $\{t_A, t_B\} \xrightarrow{D} \{t_A, t_B, t_D\}$, $\{t_A, t_C\} \xrightarrow{B} \{t_A, t_B, t_C\}, \ldots, \{t_A, t_B, t_E, t_D, t_F, t_H\}, \ldots$ where $\xrightarrow{B}$ expresses the communication with B informally(3).

The verification of **Course** is a series of checks. For all X of $U(X)$ and $\forall c \in X$, for example, we should check $\exists c'.c' \longrightarrow c \wedge c' \in X$. In other words, if a learner can take a class, she should have the credit as the premise of the class.

4.1 Exploitation of Preorder

Practically, the scenario that only one user can utilize the course is also unnatural, hence more than one user $(U_{i \in I})$ share the course in general.

$$\Pi_{i \in I} U_i(X) \mid \textbf{Course}$$

Notice that either the cardinality of the index I of this user group is infinite, or is even finite, the state explosion problem will occur easily. To cope with this problem, we should introduce some devices.

First, we introduce the notion of the preorder simulation. The preorder simulation H, with respect to the model $\mathcal{M}$ and $\mathcal{M}'$, is such a relation that if $H(s, s')$ and $s \rightarrow s_1$ in the transition relation $\mathcal{T}$ of the model $\mathcal{M}$ then $\exists s' \rightarrow s'_1$ in $\mathcal{T}'$ of $\mathcal{M}'$ and $H(s_1, s'_1)$. If we can have a relation as H, we'll say $\mathcal{M}$ and $\mathcal{M}'$ are in the preorder relation, and we describe it as $\mathcal{M} \preceq \mathcal{M}'$.

For the ACTL* formulae which is restricted form of CTL*, we know that if $\mathcal{M} \preceq \mathcal{M}'$ then $(\mathcal{M}' \models \phi \Rightarrow \mathcal{M} \models \phi)$[1].

Here, we consider a special agent **Almighty** which has the ability to take every action, **Almighty** $\stackrel{\text{def}}{=} l.$**Almighty** where $(l \in \mathcal{L} \cup \{\tau\})$. For some $i \in I$ we can check the preorder between the following components:

$$\Pi_{i \in I} U_i(X) \mid \textbf{Course} \;\preceq\; U_i(X) \mid \textbf{Almighty} \mid \textbf{Course}$$

Exploiting this preorder, we can check an interesting formulae such as , "If you offer a request, you can get the credit eventually" – **AG**(Request $\rightarrow$ **AF**Credit) – and "Always eventually the course will be enabled" – **AG**(**AF** CouseEnabled) – are confirmed by checking over $U_i(X)$ | **Almighty** | **Course** instead of $\Pi_{i \in I} U_i(X)$ | **Course**. Unfortunately, in the example presented here, after the requested ticket of **Almighty**, **Almighty** can continue on idling and keeping the connection of some course in **Course** forever, like **Almighty** $\xrightarrow{c_B}$ **Almighty** $\xrightarrow{\tau}\xrightarrow{\tau} \ldots$. This means the course will not be released for other user agents. With this information, the designer can improve the specification of **Course**.

4.2 Exploitation of Abstraction

In many cases, the question whether the users can attain their goals or not lead us to

(3) $U(\{X\}) \xrightarrow{B} U(\{X \cup \{t_B\}\}) \stackrel{\text{def}}{=} U(\{X\}) \xrightarrow{c(t)} \cdot \xrightarrow{\bar{t}\langle t_B \rangle} U(\{X \cup \{t_B\}\})$

the reachability problem over some model $\mathcal{M}$.

Here let us define $\mathcal{M} \preccurlyeq \mathcal{M}'$ as if $s, s' \in \mathcal{M}'$ and $s \rightarrow \ldots \rightarrow s'$ then $s, s' \in \mathcal{M}$ and $s \rightarrow \ldots \rightarrow s'$. Moreover, $\mathcal{M} \stackrel{\text{def}}{=} \mathcal{M}_1 \ominus \mathcal{M}_2$ also can be defined like $\mathcal{T} \stackrel{\text{def}}{=} \mathcal{T}_1 \cup \mathcal{T}_2$, that is $\mathcal{S} \stackrel{\text{def}}{=} \mathcal{S}_1 \cup \mathcal{S}_2$ and $\rightarrow \stackrel{\text{def}}{=} \rightarrow_1 \cup \rightarrow_2$.

It is clear that if $\mathcal{M}' \preccurlyeq \mathcal{M}''$ then $\mathcal{M} \ominus \mathcal{M}' \preccurlyeq \mathcal{M} \oplus \mathcal{M}''$. The queries over details of the course $\mathcal{M}'$ are often useless; it is enough to search over the abstracted model $\mathcal{M}''$ which indicates the entrances and exits. If the user needs more information, he should issue questions in more detailed domain.

5 Conclusion and Future Works

In this paper, we presented a simple idea of exploiting the temporal formulae adapted to the framework of Semantic Web. It is enabled by applying the model checking techniques over the model which is extracted from the RDF documents. The $S - P - O$ triples are regarded as the predicates of the FOL and as the Kripke Structure.

The use of powerful temporal formulae, which we suggest, is the method through which users can enjoy Semantic Web search in maximum. And also for system designers as well it was proven to be a useful method for the guarantee of some services over WWW.

Moreover, we believe that we've presented some useful model checking techniques for coping with the state explosion problem, however, these are ad-hoc techniques as yet.

In the future work, we'll analyze typical cases of practical e-Learning and e-Commerce systems, and we are going to develop a more general, effective and useful framework and its tools.

Acknowledgment

I gratefully thank to Prof. Kyoko Kondo for her valuable comments on my paper. Moreover, she checked and corrected my manuscript written in poor English.

References

[1] E.M. Clarke Jr., O. Grumberg, and D. A. Peled. Model Checking. MIT Press,1999

[2] D.Kozen. Results on the propositional mu-calculus. Theoretical Computer Science 27.

[3] C. Stirling. Modal and Temporal Properties of Processes. e Springer,2001

[4] R. Milner. communicating and mobile systems: the π-calculus. CAMBRIDGE UNIVERSITY PRESS, 1999

[5] M.Y. Vardi. Why Is Modal Logic So Robustly Decidable? Descriptive Complexity and Finite Models, N. Immerson and Ph. Kolaitis, eds. American Mathematical Society, 1997

[6] http://www.w3.org/2001/sw/

[7] http://www.w3.org/TR/rdf-mt/

[8] P. Hayes and C. Menzel. A Semantics for the Knowledge Interchange Format, Proceedings of 2001 Workshop on the IEEE Standard Upper Ontology,2001

[9] E.A. Emerson and C.L. Lei. Efficient model checking in fragments of the proposi-

tional mu-calculus. Proceedings of the 1st Annual Symposium on Logic in Computer Science. IEEE Computer Society Press, 1986

[10] D. Long, A. Browne, E. Clarke, S. Jha, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. Workshop on Computer-Aided Verification, LNCS 818. Springer, 1994

[11] R. Cleaveland and B. Steffen. A Linear-Time Model-Checking Algorithm for Alternation-Free Modal Mu-Calculus. Proceedings of Computer Aided Verification, 1991