XML DECLARATIVE DESCRIPTION WITH FIRST-ORDER LOGICAL CONSTRAINTS

CHUTIPORN ANUTARIYA
Computer Science Program, Shinawatra University, Pathumthani 12160, Thailand

VILAS WUWONGSE
Computer Science & Information Management Program, Asian Institute of Technology, Pathumthani 12120, Thailand

KIYOSHI AKAMA
Center for Information and Multimedia Studies, Hokkaido University, Sapporo 060, Japan

The expressive power of XML Declarative Description (XDD), a unified XML-based representation language for the Knowledge Grid, is enhanced by a well-defined mechanism for modeling arbitrary XML first-order logical constraints (FLCs)—a special kind of constraints comprising XML expressions and logical symbols. The resulting knowledge representation can uniformly express explicit and implicit information, ontologies, axioms as well as integrity, structural and FLCs. It facilitates direct use of ordinary XML elements as its basic language component and semantic units, and formally defines XML clauses for modeling advanced complex statements. It achieves sound, efficient, and flexible computation or inference by means of the Equivalent Transformation (ET) paradigm—a new computational model based on semantic preserving transformations. Basic ET computational rules for reasoning with XDD descriptions with FLCs are also presented. Due to its well-founded mechanism and expressiveness, employment of the proposed representation and computation framework to model a knowledge grid and its services not only enables direct representation of knowledge bases described by such emerging Semantic Web ontology languages as RDF(S) and OWL, but also offers additional descriptive facilities by allowing expression of and reasoning with rules, relationships, and constraints. Moreover, in order to provide machine-interpretable descriptions of knowledge grid services, standard service description languages, e.g., WSDL, UDDI, OWL-S and WSMO are employed and extended with facilities to define additional service relationships, constraints, and composition rules.

Key words: knowledge representation, knowledge grid, XML declarative description, XML first-order logical constraints, equivalent transformation rules.

1. INTRODUCTION

The central concern to the development of an infrastructure with supports of the sharing and coordinated use of diverse resources in a dynamic and global-wide distributed environment has led to an emerging research area on grid computing technologies (Foster et al. 2002; Zhuge 2004). In this new large-scale computing environment, where resources continue to be decentralized and distributed, the focus has been made on dynamic assemble of shared resources and services despite the diversity while providing uniform grid service interfaces.

Knowledge Grid and Grid Intelligence (KGGI) (Cannataro and Talia 2003; Zhuge 2002, 2004) is one of the key research fields in the grid computing which addresses well-established mechanisms for global knowledge sharing and management as well as intelligent means for grid resource interchange, flexible collaboration and integration of grid services. A knowledge grid can be conceived as a knowledge resource which provides a set of services. These services can be computation (problem-solving) or information provision, and can return a result: Yes/No or Exist/No-exist.

Semantic Web ontology languages, such as RDF(S) (Lassila and Swick 1999; Brickley and Guha 2003), DAML+OIL (Hendler and McGuinness 2000) and OWL (Bechhofer et al. 2004), can be employed to represent knowledge in the Knowledge Grid. However, their expressiveness is limited in that definition of arbitrary constraints, axioms, and rules, which requires capabilities beyond their predefined primitive modeling constructs, is not supported. Hence, significant extension is demanded with well-defined semantics and sufficient
expressive power to model complex relationships, axioms and rules as well as FLCs and negation.

To accomplish this goal, this paper proposes the theory of XML Declarative Description (XDD) with First-Order Logical Constraints (FLCs) (Wuwongse et al. 2001, 2003; Anutariya et al. 2003) as a language for the Knowledge Grid. It provides a fundamental architecture which allows uniform representation, discovery, integration, and interchange of knowledge in the Knowledge Grid environment. In essence, it employs emerging standard technologies on metadata, ontologies, and the Semantic Web (Berners and Lee 1999) as a foundation for providing machine-comprehensible descriptions of each component. It is an XML-based knowledge representation with precise, well-defined declarative semantics and a flexible computational mechanism. It provides in a single formalism a simple, yet expressive mechanism for succinct and uniform representation of explicit and implicit information, application axioms, rules, and constraints.

XDD’s basic modeling elements are ordinary XML elements, which can readily be used to represent explicit complex entities and their relationships in a real application domain. Moreover, it extends this capability of ordinary XML elements by additionally allowing representation of implicit complex entities as well as their classes, relationships, rules, and constraints in terms of XML expressions—a generalization of XML elements with variables—and XML clauses. An XDD description is formulated as a set of ordinary XML elements, XML expressions with variables, and XML clauses. Its declarative semantics is formally defined as a set of ordinary XML elements which are directly described by or derivable from the description itself.

This paper enhances the expressive power of ordinary XDD descriptions (Wuwongse et al. 2001, 2003) by generalizing the concept of ordinary XML constraints into referential XML constraints. Inclusion of such constraints allows an XDD description to refer to other XDD descriptions in order to represent/enforce (higher-order) complex relations and restrictions. The meaning of a given referential constraint is formally defined on the basis of the meaning of its referred descriptions. Appropriate definition of negative constraints, formalized as a kind of referential constraints, yields a well-defined mechanism for formulation of complex statements which can declaratively describe negative information or negation.

Moreover, in order to enable straightforward formulation of arbitrarily complex XML first-order logic formulae and thus to enhance the expressive power of XDD, this paper further extends it with a support of XML FLCs—special kind of XML constraints which comprise XML expressions and logical symbols: \(\neg\), \(\land\), \(\lor\), \(\Rightarrow\), \(\forall\) and \(\exists\).

Due to its well-defined mechanism and expressiveness, employment of XDD to model the Knowledge Grid not only permits direct representation of knowledge bases described by existing prominent Web ontology languages, but also offers extensive features to such languages by enabling expression of and reasoning with rules, relationships, and constraints that involve those ontological elements/statements. In particular, ontology definitions and their instances are directly represented in XDD as XML elements (or ground XML unit clauses) while the axiomatic semantics of each modeling primitive, their relationships, application rules, and constraints are formalized as XML clauses. Queries about both explicit and implicit information are expressible in the same manner.

For the sake of self containment, Section 2 recalls the fundamental concepts of normal XDD descriptions. Section 3 presents their extension to represent referential constraints and negation, followed by a well-defined formalization for the expression of and reasoning with arbitrary first-order logical formulae in Section 4. Section 5 presents their computation by means of Equivalent Transformation paradigm, Section 6 outlines an XDD approach to modeling the Knowledge Grid, Section 7 discusses related work, and Section 8 draws conclusions together with future research direction.
2. NORMAL XDD DESCRIPTIONS: AN INFORMAL REVIEW

This section informally reviews fundamental definitions of XDD theory by the means of simple explanations and examples since its theoretical details as well as the formal semantics have already been presented in (Wuwongse et al. 2001, 2003).

Basically, an XDD description is defined as a set of ordinary XML elements, XML expressions with variables, and their relationships represented in terms of XML clauses. An XML element (or a ground XML expression) denotes a semantic unit and is considered to be a surrogate of an information item in a real-world domain. An XML expression with variables represents implicit information or a set of semantic units. An XML clause models a rule, conditional relationship, integrity constraint, or an ontological axiom. The precise and formal semantics of an XDD description is defined as a set of ordinary XML elements themselves without employment of other formalisms. These elements are surrogates of real, tangible or intangible objects, or their relationships in the domain of interest, modeled by the description.

Six disjoint classes of XML variables with different syntactical usage and specialization characteristics are defined by Table 1. The data structure and specialization behavior of XML expressions are characterized by a mathematical abstraction, called XML specialization system (Wuwongse et al. 2001, 2003) which is a quadruple \( \langle A_X, G_X, S_X, \mu_X \rangle \), where

- \( A_X \) is the set of all XML expressions,
- \( G_X \) is the subset of \( A_X \) comprising all ground XML expressions,
- \( S_X \) is the set of XML specializations,
- \( \mu_X \) is the specialization operator which determines, for each XML specialization \( \theta \) in \( S_X \), the change of an XML expression in \( A_X \) caused by \( \theta \). Unless confusion may arise, the application of a specialization \( \theta \) to an XML expression \( a \) by the operator \( \mu_X \), formally denoted by \( \mu_X(\theta)(a) \), will be simply represented as \( a^\theta \).

An XML clause \( C \) formally has the form

\[ H \leftarrow B_1, \ldots, B_n, \]

where \( n \geq 0, H \) is an XML expression, and \( B_i \) is an XML expression or an XML constraint. \( H \) is called the head and \( \{B_1, \ldots, B_n\} \) the body of the clause. When its body is the empty set, \( C \) will be referred to as an XML unit clause and the symbol \( \leftarrow \) is often omitted; hence, an XML element or document can be mapped directly onto a ground XML unit clause.

An XML constraint—useful for defining a restriction on XML expressions or their components—is a formula of the form \( \langle \phi, a_1, \ldots, a_n \rangle \), where \( n > 0, \phi \) is a constraint predicate and \( a_i \) is an XML expression. The satisfaction (truth or falsity) of a ground constraint is

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Prefix</th>
<th>Specialization Into</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-variable</td>
<td>$N:$</td>
<td>An element type or attribute name</td>
</tr>
<tr>
<td>S-variable</td>
<td>$S:$</td>
<td>A string</td>
</tr>
<tr>
<td>P-variable</td>
<td>$P:$</td>
<td>A sequence of zero or more attribute-value pairs</td>
</tr>
<tr>
<td>E-variable</td>
<td>$E:$</td>
<td>A sequence of zero or more XML expressions</td>
</tr>
<tr>
<td>E1-variable</td>
<td>$E1:$</td>
<td>An XML expression</td>
</tr>
<tr>
<td>I-variable</td>
<td>$I:$</td>
<td>Parts of XML expressions</td>
</tr>
</tbody>
</table>
predetermined. When the context is clear, some constraints will be represented using in-fix notation. For instance, \([S:year > 2002]\) and \([S:price := S:regFee*0.25]\) represent the constraints \((GT, S:salary, 10000)\) and \((Mul, S:regFee, 0.25, S:price)\), respectively.

Speaking intuitively, the meaning of a given XDD description \(P\), denoted by \(\mathcal{M}(P)\), is defined as a set of all XML elements which are directly described by and are derivable from the unit and the non-unit XML clauses in \(P\), respectively, i.e.,:

- Given an XML unit clause \((H \leftarrow)\) in \(P\), for \(\theta \in S_X:\n  H\theta \in \mathcal{M}(P)\) if \(H\theta\) is a ground XML expression.
- Given an XML non-unit clause \((H \leftarrow B_1, \ldots, B_i, B_{i+1}, \ldots, B_n)\) in \(P\), assuming without loss of generality that \(B_1, \ldots, B_i\) are XML expressions in \(A_X\) and \(B_{i+1}, \ldots,B_n\) XML constraints, for \(\theta \in S_X:\n  H\theta \in \mathcal{M}(P)\) if \(H\theta\) is a ground XML expression, \(B_1\theta, \ldots, B_i\theta \in \mathcal{M}(P)\), and \(B_{i+1}\theta, \ldots,B_n\theta\) are true constraints.

Further explanations of XDD descriptions and their declarative semantics will be given from a practical point of view by the means of examples.

**Example 1.** The XDD description \(KB\) of Figure 1 presents an example of modeling a simple ontological knowledge base application. It comprises of a fragment of domain-specific ontology definitions \(D_1–D_4\), three ontology instances \(E_1–E_3\) together with XML clauses \(C_1\) and \(C_2\). The elements \(D_1–D_4\) represent ontology definitions expressed in OWL language and describe that \(InterConference\) and \(LocalConference\) are subclasses of \(Conference\). The elements \(E_1–E_3\) are ontology instances describing an international conference \(INTELLCOMM2004\), organized in 2004 by AIT—a university located in Thailand.

The clause \(C_1\) formalizes the axiomatic semantics of an OWL construct \(owl: subClassOf\) by specifying that if \(X\) is an instance of \(ClassB\) and \(ClassB\) is a \(subClassOf\) \(ClassA\), then \(X\) is also an instance of \(ClassA\). The clause \(C_2\) represents an example of an application rule which is inexpressible by existing ontology languages. It asserts that for a conference held in Thailand, in the year 2002 or later, the editors of its proceedings are those who serve as the PC-chair of the conference and the price of the proceedings is equal to 25% of the conference’s registration fee. The meaning of \(KB\) is the set of XML elements which are directly described by the unit clauses, i.e., the elements \(D_1–D_4\) and \(E_1–E_3\) together with those which are deducible from the description, i.e., the elements \(E_4\) and \(E_5\) of Figure 2.

### 3. REFERENTIAL XDD DESCRIPTIONS

Here, XDD descriptions will be extended with the ability to represent referential XML constraints (Anutariya, Wuwongse, and Akama 2003), which generalize the concept of ordinary XML constraints by inclusion of a facility for referring to other XDD descriptions, allowing representation of (higher-order) relations or restrictions. The meaning of a given referential constraint is determined based on the meanings of its referred descriptions. Thus, by inclusion of the referential constraints in the clause’s body, an XML clause can additionally represent/enforce complex relations/restrictions, the truth or falsity of which is evaluated according to the meaning of particular XDD descriptions.
**a. Domain-specific ontology definitions and instances.**

\[ D_1: \text{<owl:Class rdf:id="Conference"/>} \]
\[ D_2: \text{<owl:Class rdf:id="University"/>} \]
\[ D_3: \text{<owl:Class rdf:id="InterConference"/>} \]
\[ D_4: \text{<owl:Class rdf:id="LocalConference"/>} \]
\[ E_1: \text{<c:University rdf:about="#ait"/>} \]
\[ E_2: \text{<c:University rdf:about="#cit"/>} \]
\[ E_3: \text{<c:InterConference rdf:about="#intelcomm2004"/>} \]

**b. An ontological axiom and an application rule.**

% The Conference-expression contained in the body of C₂ selects from the knowledge base only those conferences held in Thailand. The constraint [$$\text{year} >= 2002$$] restricts that the conference year (represented by the variable $$\text{year}$$) must be 2002 or later. The second and third constraints respectively specify that the variable $$\text{price}$$ is equal to 25% of the conference registration fee (represented by $$\text{regFee}$$), and $$\text{procName}$$ is the concatenation of the strings "Proceedings" of " and the conference name (represented by $$\text{confName}$$).

% The head of C₂ then derives Proceedings-element with the name $$\text{procName}$$, the price $$\text{price}$$, and the editors $$\text{pcChairList}$$ (i.e., the sequence of those who serve as the conference PC chair).

**FIGURE 1.** $\text{KB} = \{D_1, D_2, D_3, D_4, E_1, E_2, E_3, C_1, C_2\}$: an XDD description modeling a simple ontological knowledge base application.
The definitions of XML constraints, XML clauses, and XDD descriptions are redefined:

**Definition 1.** **XML constraints, XML clauses, and XDD descriptions** are defined inductively by:

1. An *XML constraint* is an \((n + m + 1)\)-tuple \(\langle \phi, a_1, \ldots, a_n, Q_1, \ldots, Q_m \rangle\), where
   - \(n + m \geq 1\),
   - \(\phi\) is a mapping from \(G_1 \times \cdots \times G_n \times K_1 \times \cdots \times K_m\) to \(\{\text{true}, \text{false}\}\), where \(G_i\) is \(G_{\mathcal{X}}\), and \(K_j\) is \(2^{G_{\mathcal{X}}}\),
   - \(a_i\) is an XML expression in \(G_{\mathcal{X}}\),
   - \(Q_j\) is an XDD description.
2. An *XML clause* is a formula \((H \leftarrow B_1, \ldots, B_n)\), where \(n \geq 0\), \(H\) is an XML expression, and \(B_i\) is either an XML expression or an XML constraint.
3. An *XDD description* is a set of XML clauses.

An XML constraint \(\langle \phi, a_1, \ldots, a_n, Q_1, \ldots, Q_m \rangle\) is called a *simple XML constraint* if it comprises solely of XML expressions (i.e., \(m = 0\)), and called a *referential XML constraint* if it refers to some XDD descriptions (i.e., \(m \geq 1\)). Moreover, it is a *ground XML constraint* if it comprises only of ground XML expressions and XDD descriptions. A clause \(C\) is a *ground clause* if it comprises only of ground XML expressions and ground XML constraints. The head of \(C\) is denoted by \(\text{head}(C)\) and the set of all XML expressions and XML constraints in the body of \(C\) by \(\text{object}(C) \cup \text{con}(C)\), respectively. Let \(\text{body}(C) = \text{object}(C) \cup \text{con}(C)\). Given a specialization \(\theta \in \mathcal{S}_X\), application of \(\theta\) to a clause \((H \leftarrow B_1, \ldots, B_n)\) yields the clause \((H\theta \leftarrow B_1\theta, \ldots, B_n\theta)\), and to a constraint \(\langle \phi, a_1, \ldots, a_n, Q_1, \ldots, Q_m \rangle\) the constraint \(\langle \phi, a_1\theta, \ldots, a_n\theta, Q_1, \ldots, Q_m \rangle\).

Employing the formalized concepts, a *negative constraint*—one of the most important referential XML constraints—will be given for the representation of negative information or negation.

**Definition 2.** A *negative constraint* has the form

\[ \langle f_{\text{not}}, a, Q \rangle, \]
where

- $f_{not}$ is a mapping from $G_X \times 2^{G_X}$ to \{true, false\} such that
  - $f_{not}(g, G) = true$ if $g \notin G$,
  - $f_{not}(g, G) = false$ if $g \in G$,
- $a$ is an XML expression,
- $Q$ an XDD description.

A constraint \langle $f_{not}$, $a$, $Q$ \rangle restricts that the expression $a$ must not be true with respect to the description $Q$, i.e., $a$ must not be included in the meaning of $Q$. In the sequel, such a constraint will be simply represented as $[a \notin \mathcal{M}(Q)]$. This formulation of negation is natural because it permits specification of the negative information and its respective evaluation context.

**Definition 3.** Given an XML expression $a \in A_X$, let $rep(a)$ denote the subset of $G_X$ which comprises of all ground instances of $a$, that is:

$$rep(a) = \{g \in G_X \mid g = a\theta, \theta \in S_X\}.$$ 

For any XML expression $a$, define a **non-rep$a$** constraint as an XML constraint of the form

\langle non-rep$a$, $x$ \rangle,

where

- **non-rep$a$** is a mapping from $G_X$ to \{true, false\} such that for $g \in G_X$,
  - **non-rep$a$(g) = true** if $g \notin rep(a)$,
  - **non-rep$a$(g) = false** if $g \in rep(a)$.
- $x$ is an XML expression.

Speaking intuitively, a ground constraint \langle **non-rep$a$**, $x$ \rangle ensures that the XML expression $x$ must not be specialized into a ground instance of $a$. Hence, it can be simply represented as $[x \notin rep(a)]$.

**Example 2.** By referring to the description $KB$ of Figure 1, the following clause $C_{neg}$ derives those universities which never organize a conference.

\[
\begin{align*}
C_{neg}: \quad & <UniversityNoConference \ rdf:about=\$S:uni> \\
& \$S:uName \\
& </UniversityNoConference> \\
\langle & <c:University \ rdf:about=\$S:uni> \\
\langle & <c:name>$S:uName$</c:name> \\
& $E:uniProperties \\
& </c:University> , \\
& \langle <c:Conference \ rdf:about=\$S:conf> \\
& <c:organizer \ rdf:resource=\$S:uni/> \\
& $E:confProperties \\
& </c:Conference> \notin \mathcal{M}(KB) ]
\end{align*}
\]

It simply specifies that a university $SS:uni$ is a UniversityNoConference, if there exists no conference the organizer of which is that university. The first XML expression in $C_{neg}$’s body finds a University-element from the knowledge base and the followed
negative constraint specifies that a conference whose organizer property refers to that university must not exist with respect to the description $KB$. $C_{\text{neg}}$’s head then derives that such a university is a UniversityNoConference.

**Definition 4.** The **declarative semantics** of an XDD description $P$, denoted by $\mathcal{M}(P)$, is defined inductively by:

1. Given the meanings $\mathcal{M}(Q_1), \ldots, \mathcal{M}(Q_m)$ of XDD descriptions $Q_1, \ldots, Q_m$, a ground constraint $\langle \phi, g_1, \ldots, g_n, Q_1, \ldots, Q_m \rangle$ is a true XML constraint if $\phi(g_1, \ldots, g_n, \mathcal{M}(Q_1), \ldots, \mathcal{M}(Q_m)) = \text{true}$. Define the set $T\text{con}$ as the set of all true ground XML constraints, i.e.,:

   $$T\text{con} = \{ \langle \phi, g_1, \ldots, g_n, Q_1, \ldots, Q_m \rangle | g_i \in G_X, Q_i \text{ is an XDD description}, \phi(g_1, \ldots, g_n, M(Q_1), \ldots, M(Q_m)) = \text{true} \}.$$

2. The meaning $\mathcal{M}(P)$ of the description $P$ is the set of ground XML expressions defined by:

   $$\mathcal{M}(P) = \bigcup_{n=1}^{\infty} [T_P]^n(\emptyset),$$

   where
   - $T_P^1(\emptyset) = T_P(\emptyset)$,
   - $[T_P]^n(\emptyset) = T_P([T_P]^{n-1}(\emptyset))$ for each $n > 1$,
   - the mapping $T_P : 2^{G_X} \rightarrow 2^{G_X}$ is defined by:

   $$T_P(G) = \{ \text{head}(C\theta) | C \in P, \theta \in S_X, C\theta \text{ is a ground clause}, \text{object}(C) \subset G, \text{con}(C) \subset T\text{con} \}.$$

   From its definition, one can yield that the meaning of a description is defined based on the meanings of its referred descriptions, and thus XDD descriptions with referential constraints must be **stratified**. Speaking intuitively, the meaning of $P$, i.e., $\mathcal{M}(P)$, is a set of all ground XML expressions which are directly described by and are derivable from the unit and the non-unit XML clauses in $P$.

**4. XDD DESCRIPTIONS WITH FIRST-ORDER LOGICAL CONSTRAINTS**

This section further extends the definition of referential XDD descriptions by a mechanism to express arbitrary first-order formulae in the body of an XML clause. Such an expression is represented as a **FLCs**—a special kind of constraints which is composed of XML expressions and logical symbols, consisting of $\neg$, $\land$, $\lor$, $\Rightarrow$, $\forall$ and $\exists$. This helps to
enhance the expressive power of XDD by allowing representation of the rules, constraints, and queries more naturally.

**Definition 5.** An **FLC** is inductively defined as follows:

1. An XML expression is an FLC.
2. An XML constraint is an FLC.
3. If $F$ and $G$ are FLCs, then so are $\neg F$, $F \land G$, $F \lor G$ and $F \Rightarrow G$.
4. If $F$ is an FLC and $x$ is an XML variable, then $(\forall x \ F)$ and $(\exists x \ F)$ are FLCs.

**Definition 6.** An **XDD description with FLCs** is formalized as a set of extended XML clauses with FLCs. Each clause has the form $H \leftarrow F$, where the head $H$ is an XML expression and the (possibly absent) body $F$ an FLC.

Employing those transformation techniques developed in the first-order logic theory (Lloyd 1987), extended XDD descriptions with FLCs can be straightforwardly transformed into equivalent referential XDD descriptions (with negative constraints) in a finite number of steps. Therefore, the meaning of a given description with FLCs is directly defined by the means of its equivalent description. Such transformation rules will be discussed in more details in the next section.

**Example 3.** Consider a simple query which finds the names of the universities that for all conferences they have organized in the year 2000 or later, the registration fees of which are less than 350. Using FLCs, the query can be simply expressed as the description $P = \{C_p\}$. The clause $C_p$ specifies that for each university $S:uni$, retrieve its name ($S:uName$), if for every international conference held in the year 2000 or later and organized by such a university, its registration fee ($S:registrationFee$) is less than 350.

$$C_p: \quad \langle \text{SelectedUniversity} \rangle \\
\quad \langle \text{name} \rangle \langle S:uName \rangle \langle / \text{name} \rangle \\
\quad \langle / \text{SelectedUniversity} \rangle \\
\quad \langle c:University \rangle \quad \langle \text{rdf:about} \rangle \langle S:uni \rangle \\
\quad \langle / \text{c:University} \rangle \\
\quad \langle c:name \rangle \langle S:uName \rangle \langle / \text{name} \rangle \\
\quad \langle / \text{c:name} \rangle \\
\quad \langle S:uniProperties \rangle \\
\quad \langle / \text{S:uniProperties} \rangle \\
\quad \land \forall S:conf \forall S:year \forall S:regFee \forall S:confProperties \\
\quad \quad ((\langle c:InterConference \rangle \langle \text{rdf:about} \rangle \langle S:conf \rangle \\
\quad \quad \langle c:year \rangle \langle S:year \rangle \langle / \text{c:year} \rangle \\
\quad \quad \langle c:organizer \rangle \langle \text{rdf:resource} \rangle \langle S:uni \rangle \\
\quad \quad \langle c:registrationFee \rangle \langle S:regFee \rangle \\n\quad \quad \langle / \text{c:registrationFee} \rangle \\
\quad \quad \langle / \text{c:InterConference} \rangle \\
\quad \quad \land \quad \langle S:year \rangle \geq 2000 \\
\quad \quad \Rightarrow \quad \langle S:regFee \rangle < 350 \rangle \\
\quad \quad \langle / \text{S:uniProperties} \rangle \\
\quad \quad \langle / \text{c:InterConference} \rangle \\
\quad \quad \land}$$

5. **COMPUTATIONAL FRAMEWORK**

Since XDD focuses on information/knowledge representation, in order to provide a concise and expressive language with precise and well-defined semantics, its underlying representation scheme is separated from its computational mechanism. It achieves efficient manipulation of and reasoning with XDD descriptions by employment of Equivalent Transformation (ET) (Akama, Shimitsu, and Miyamoto 1998) computational paradigm.
5.1. Equivalent Transformation

*ET* (Akama et al. 1998) is a new, flexible, and efficient computational model which solves a given problem by simplifying it through repetitive application of (semantically) equivalent transformation rules. Let $P$ be an XDD description which models a particular application or knowledge base and $Q_1$ is a problem or query. The meaning of $(P \cup Q_1)$, i.e., $M(P \cup Q_1)$, yields the set of XML elements which represent the solutions to the formulated problem. This new computation paradigm applies *ET rules* (procedural rewriting rules) in order to successively transform $(P \cup Q_1)$ into $(P \cup Q_2)$, $(P \cup Q_3)$, etc., while maintaining the conditions $M(P \cup Q_1) = M(P \cup Q_2) = M(P \cup Q_3) = \ldots$, until a desirable description $P \cup Q_n$, which is a simpler but equivalent description from which the answers to the given problem can be drawn easily and directly, is obtained (cf. Figure 3). More precisely, $P \cup Q_1$ is successively transformed until it becomes the description $P \cup Q_n$, where $Q_n$ contains solely XML unit clauses which readily represent the answers to the problem described by $Q_1$. Furthermore, since only ET rules are applied, the meanings of the descriptions are maintained and the correctness of the computation is always guaranteed. The unfolding transformation—the fundamental computational mechanism in logic programming—presents an example of ET rules. There exist many other ET rules which reflect or exploit specific application domain knowledge and data structure, and thus could lead to more efficient computation.

In general, ET rules can be classified into two types: *general* and *application-specific*. *General rules* are devised based on the general knowledge, such as those rules which define axiomatic semantics of each ontology modeling construct as well as rules for handling negation, set constraints and FLCs, while *application-specific rules* implement specific domain knowledge such as business rules and application policies. An ET rule has the form

$$\text{Head}, \{\text{Condition}\} \xrightarrow{\text{ET}} \{\text{Execution}_1\}, \text{Body}_1;$$
$$\text{ET} \quad \{\text{Execution}_2\}, \text{Body}_2;$$
$$\quad \ldots$$
$$\text{ET} \quad \{\text{Execution}_n\}, \text{Body}_n;$$
It specifies backward-chaining-like computation and reads: if the pattern of an expression specified by the Head of a rule matches with the target expression and the Condition is satisfied, then the \( n \) bodies fire simultaneously, i.e., the built-in or user-defined operations specified in each \( \text{Execution}_i \) are performed, and the expressions specified in the Body\(_i\) replace the target object.

An ET program consists of a sequence of ET rules, where each rule is associated with an appropriate rule priority. When there exist many applicable ET rules, the one with the highest priority will be selected for execution. If there is more than one rule with the highest priority, any one among such rules will be selected arbitrarily. Because every ET rule is semantics-preserving, all rules in the program are independent, and execution of any applicable rules will not impact the computational correctness. Therefore, rule priority is defined solely for controlling rule selection strategy of the program, i.e., for efficiency improvement, while still preserving the program’s soundness property.

5.2. XML Equivalent Transformation (XET)

Founded on the XDD theory and the ET paradigm, XML Equivalent Transformation (XET) engine (Anutariya et al. 2002a; Anutariya, Wuwongse, and Wattanapailin 2002b) has been developed for materializing the equivalent transformation of XDD descriptions, hence allowing more insight manipulation of and reasoning with XML expressions without a necessity for data conversion. Basically, an XET program comprises of a set of XET rules—ET rules for computing with XDD descriptions and encoded in XML format—and a set of XML elements/documents regarded as the program’s data or facts. It takes as its input a query formalized as an XDD description and then computes the query’s answers by applying the program’s rules and facts. Moreover, in order to ensure XML well-formedness property, the six disjoint classes of XML variables (cf. Table 1) are expressed in XET with the prefix Nvar\(_x\), Svar\(_x\), Pvar\(_x\), Evar\(_x\), E1var\(_x\), and Ivar\(_x\), respectively.

Figure 4 shows an XET program KB.xet together with the program’s facts Facts.xet implementing a fragment of the example ontological knowledge base application KB of Example 1, where every XET rule has an equal priority.

5.3. ET Rules for Negative Constraints

This section defines three general ET rules for computation with negative constraints. The proofs that they are semantic-preserving transformations are given in the Appendix. Throughout this section, let \( Q, Q' \) and \( Q'' \) be XDD descriptions, and \( a \) and \( b \) be XML expressions. Moreover, assume without loss of generality that \texttt{xet:NotRule} is a reserved tag name not occurring in \( Q \).

First, a transformation, which defines the starting point for processing negative constraints, is presented. It merges the expression \( a \) of a given negative constraint \( [a \notin M(Q)] \) and its referred description \( Q \) into a new description \( Q' = Q \cup \{<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \leftarrow a\} \), and then transforms the given constraint into \( [<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \notin M(Q')] \). Thus, closely interrelated computation with the expression \( a \) and the referred description \( Q \) becomes possible by equivalent transformation of the description \( Q' \).

**ET Rule 1. Merging of a Negative Object**

\[
\begin{align*}
[a \notin M(Q)] & \xrightarrow{\text{ET}} \{Q' = Q \cup \{<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \leftarrow a\}\} \\ \ \ \ [<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \notin M(Q')].
\end{align*}
\]
Next, let a referred description $Q'$ allow equivalent transformation into $Q'' \cup \{ <\text{xet:NotRule}>b </\text{xet:NotRule}> \leftarrow \}$. Hence, the negative constraint $[<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q')]$ can be replaced by $[a \notin rep(b)]$ and $[<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q'')]$, where $[a \notin rep(b)]$ ensures that $a$ must not be specialized into a ground instance of $b$ (by Definition 3). This process is materialized by:

**ET Rule 2. Lifting Transformation**

\[
[<\text{xet:NotRule}>a\not<\text{xet:NotRule}> \notin \mathcal{M}(Q)]
\]
\[
\{ Q' = Q'' \cup \{ <\text{xet:NotRule}>b</\text{xet:NotRule}> \leftarrow \} \} 
\]

\[\xrightarrow{ET} \quad [a \notin rep(b)], \]

\[ [<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q')]. \]
If the referred description $Q'$ of a constraint $[<\text{et:NotRule}>a</\text{et:NotRule}> \notin \mathcal{M}(Q')]$ cannot be further transformed equivalently into a description $Q'' \cup \{<\text{et:NotRule}>b</\text{et:NotRule}> \leftrightarrow \}$, such a constraint can be eliminated from a clause. This is implemented by:

**ET Rule 3. Elimination of Negative Constraint**

$$[<\text{et:NotRule}>a</\text{et:NotRule}> \notin \mathcal{M}(Q')]$$

\{ $Q'$ cannot be transformed into $Q'' \cup \{<\text{et:NotRule}>b</\text{et:NotRule}> \leftrightarrow \}$ \}

$$\xrightarrow{\text{ET}} \langle \text{true} \rangle.$$  

5.4. ET Rules for FLCs

Basic ET rules for computing with FLCs, based on certain transformation techniques developed in the first-order logic theory (Lloyd 1987) are presented. Since the proofs that these rules are semantic-preserving transformations are already discussed by Lloyd (1987), they are omitted in this paper. Throughout this section, let $F, G$ be FLCs, $x_i$ XML variables, $a_i$ XML expressions, and $Q_i$ XDD descriptions.

**ET Rule 4. Negation of a constraint**

$$\neg(\phi, a_1, \ldots, a_n, Q_1, \ldots, Q_m) \xrightarrow{\text{ET}} (\phi', a_1, \ldots, a_n, Q_1, \ldots, Q_m),$$

where the constraint mapping $\phi'$ is the negated of $\phi$, i.e., a ground constraint $(\phi', g_1, \ldots, g_n, Q_1, \ldots, Q_m)$ is true (respectively, false), if $(\phi, g_1, \ldots, g_n, Q_1, \ldots, Q_m)$ is false (respectively, true).

**ET Rule 5. Double negation**

$$\neg\neg F \xrightarrow{\text{ET}} F.$$

**ET Rule 6. Conjunction**

$$F \land G \xrightarrow{\text{ET}} F, G.$$

**ET Rule 7. Negation of a conjunction**

$$\neg(F \land G) \xrightarrow{\text{ET}} \neg F.$$  

$$\xrightarrow{\text{ET}} \neg G.$$

**ET Rule 8. Disjunction**

$$F \lor G \xrightarrow{\text{ET}} F.$$  

$$\xrightarrow{\text{ET}} G.$$

**ET Rule 9. Negation of a disjunction**

$$\neg(F \lor G) \xrightarrow{\text{ET}} \neg F, \neg G.$$
ET Rule 10. Implication

\[(F \Rightarrow G) \xrightarrow{ET} \neg F.\]

\[\xrightarrow{ET} G.\]

ET Rule 11. Negation of an implication

\[\neg (F \Rightarrow G) \xrightarrow{ET} F, \neg G.\]

ET Rule 12. Universal quantifier

\[\forall x_1, \ldots, \forall x_n F \xrightarrow{ET} \neg \exists x_1 \ldots \exists x_n \neg F.\]

ET Rule 13. Negation of universal quantifiers

\[\neg \forall x_1, \ldots, \forall x_n F \xrightarrow{ET} \exists x_1 \ldots \exists x_n \neg F.\]

ET Rule 14. Existential quantifier

\[\exists x_1 \ldots \exists x_n F \xrightarrow{ET} F.\]

ET Rule 15. Negation of an FLC

Negation of an FLC is transformed into a negative constraint which refers to a newly generated description \(Q\):

\[\neg F \xrightarrow{ET} \{Q = \{G \leftarrow F\}[G \notin \mathcal{M}(Q)]\},\]

where \(G\) is a new XML expression not appearing in the description and constructed using all the free variables in \(F\).

Example 4. Based on the presented general ET rules for negative constraints and FLCs, and the XET program \(\text{KB.xet}\) and facts \(\text{Facts.xet}\) (Figure 4) which implements an example application-specific rules and facts, the evaluation of the query \(P\) of Example 3 is demonstrated.

Step 1. Transform \((F \land G)\) into \((F, G)\) and \((\forall x_1, \ldots, \forall x_n F)\) into \((\neg \exists x_1, \ldots, \exists x_n \neg F)\) by ET Rules 6 and 12, respectively:

\[P = \{ C_P; \quad \text{<SelectedUniversity>}
\quad \text{name} <$S$:uName</name>
\quad \text{</SelectedUniversity>}
\quad \leftarrow \quad \text{<c:University rdf:about=}$S$:uni>
\quad \text{<c:name>$S$:uName</c:name>}
\quad \text{$SE$:uniProperties}
\quad \text{</c:University>}.\]
Step 2. By ET Rule 15, transform the negation in $C_p$’s body into a negative constraint referring to a new generated description $Q$:

$$P = \{ C_p: <\text{SelectedUniversity}> \\
\quad <\text{name}>\$uName</\text{name}> \\
\quad <\text{name}>\$uName</\text{name}> \\
\quad <\text{name}>\$uName</\text{name}> \\
\quad <\text{University}> \\
\quad <\text{Negation1}>\$uName</\text{Negation1}> \notin \mathcal{M}(Q) \}$$

$$Q = \{ C_q: <\text{Negation1}>\$uName</\text{Negation1}> \\
\quad \exists \$conf \exists \$year \exists \$regFee \exists \$confProperties \\
\quad \neg ((( <\text{InterConference} rdf:about=\$conf> \\
\quad <\text{c:year} \$year <\text{c:year}> \\
\quad <\text{c:organizer} rdf:resource=\$uni/> \\
\quad <\text{c:registrationFee} \$regFee</\text{registrationFee}> \\
\quad <\text{E:confProperties} \\
\quad <\text{c:InterConference}> \\
\quad \wedge ([\$year >= 2000]) \\
\quad \Rightarrow ([\$regFee < 350]) \}$$

Step 3. Transform $\exists x_1, \ldots, \exists x_n \ F$ in $C_q$’s body into $F$ by ET Rule 14:

$$Q = \{ C_q: <\text{Negation1}>\$uName</\text{Negation1}> \\
\quad \sim \neg ((( <\text{InterConference} rdf:about=\$conf> \\
\quad <\text{c:year} \$year <\text{c:year}> \\
\quad <\text{c:organizer} rdf:resource=\$uni/> \\
\quad <\text{c:registrationFee} \$regFee</\text{registrationFee}> \\
\quad <\text{E:confProperties} \\
\quad <\text{c:InterConference}> \\
\quad \wedge ([\$year >= 2000]) \\
\quad \Rightarrow ([\$regFee < 350]) \}$$

Step 4. Transform $\neg (F \Rightarrow G)$ in $C_q$’s body into $F$, $\neg G$ by ET Rule 11:

$$Q = \{ C_q: <\text{Negation1}>\$uName</\text{Negation1}> \\
\quad \neg ((<\text{InterConference} rdf:about=\$conf> \\
\quad <\text{c:year} \$year <\text{c:year}> \\
\quad <\text{c:organizer} rdf:resource=\$uni/> \\
\quad <\text{c:registrationFee} \$regFee</\text{registrationFee}> \\
\quad <\text{E:confProperties} \\
\quad <\text{c:InterConference}> \\
\quad \wedge ([\$year >= 2000]), \\
\quad \neg ([\$regFee < 350]) \}$$
Step 5. In \( C_q \)'s body, transform \((F \land G)\) into \((F, G)\) by ET Rule 6 and \(\neg [S:\text{regFee} < 350]\) into \([S:\text{regFee} >= 350]\) by ET Rule 4:

\[
Q = \{ \begin{align*}
\text{Step 6. Transform the negative constraint in } C_p \text{'s body (cf. Step 2) by ET Rule 1:}
\end{align*}
\]

\[
P = \{ \begin{align*}
\text{Step 7. Unfolding the } \text{xet:Negation1-expression in } C_{qq} \text{'s body with the clause } C_q \text{ transforms } Q' \text{ into:}
\end{align*}
\]

\[
Q' = \{ \begin{align*}
\end{align*}
\]
Step 8. Because the `c:InterConference`-expression in `C_{qq}`'s body can be matched successfully with the seventh fact of `KB.xet` (Lines 22–35), `Q'` is transformed into:

\[
Q' = \{ C_{qq} : \text{<xet:NotRule>}
\text{<xet:Negation1> \#ait</xet:Negation1>}
\text{<xet:NotRule>}
\text{\{ [2400 >= 2000] },
\text{[300 >= 350] \}}
\]

Step 9. By evaluating the constraint `[300 >= 350]` in `C_{qq}`'s body, which is false, the clause `C_{qq}` is removed, and hence `Q'` is transformed into:

\[
Q' = \{
\}
\]

Step 10. Because `Q' = \{ \}`, ET Rule 3 is applied and the negative constraint in `C_p`'s body (cf. Step 6) removed:

\[
P' = \{ C_p : \text{<SelectedUniversity>}
\text{<name>$SS:uName</name>}
\text{</SelectedUniversity>}
\\leftarrow \text{<c:University rdf:about=$SS:uni>}
\text{<c:name>$SS:uName</c:name>}
\text{$E:uniProperties}
\text{</c:University>}
\}
\]

Step 11. Unfold the `c:University`-expression in `C_p`'s body with the fifth and sixth facts of `KB.xet` (Lines 14–17 and Lines 18–21):

\[
P = \{ C_{p1} : \text{<SelectedUniversity>}
\text{<name>AIT</name>}
\text{</SelectedUniversity> \leftarrow}
\}
\]

\[
P = \{ C_{p2} : \text{<SelectedUniversity>}
\text{<name>CIT</name>}
\text{</SelectedUniversity> \leftarrow}
\}
\]

This example has shown that the query formalized as an XDD description `P` can be equivalently transformed into a simpler description consisting of two unit clauses, which presents the query's answers:

\[
\text{<SelectedUniversity>}
\text{<name>AIT</name>}
\text{</SelectedUniversity>}, \text{ and}
\text{<SelectedUniversity>}
\text{<name>CIT</name>}
\text{</SelectedUniversity>}
\]

6. KNOWLEDGE GRID REPRESENTATION FRAMEWORK

In Knowledge Grid environment, knowledge can be distributed across the nodes in a grid. This knowledge can be shared and accessed in a transparent manner enabling one to conceive it as a single global-wide knowledge repository. Hence, a well-established infrastructure with supports for uniform knowledge representation, discovery, interchange-ability,
and integratability is demanded. In addition, a knowledge grid can provide a set of services such as decision making, problem solving, information providing, or knowledge retrieval, and can return a result: Yes/No, Exist/No-exist. However, due to their diversity in terms of, for instance, architecture, availability, capabilities, security, cost, and QoS, a well-founded mechanism which enables a service to be defined as an aggregation of several related ones is required.

This section proposes an XDD approach of dealing with these requirements of the Knowledge Grid. Figure 5 illustrates an overview of the proposed framework. The next sub-sections then formalize the approach in more details.

6.1. Domain Modeling

Ontologies play an important role in providing an ability to model, represent and exchange formal conceptualization as well as information of particular domains in a precise, machine-understandable form. Standard, predefined domain-specific ontologies can be shared and reused. However, two communicating knowledge grid services—either in the same or different domains—may employ different ontologies, hence an ontology-mapping ability is required.

Recently developed ontology markup languages, such as RDF(S), and OWL, only provide facilities for describing concept- and property-hierarchies, some particular axioms and constraints, and ontology instances. However, they still lack support for representing arbitrary axioms, rules, and constraints—an essential feature required by many applications.

A description of domain-specific knowledge encoded in an ontology language, such as RDF and OWL, becomes immediately an XDD description comprising solely of ground XML unit clauses. Moreover, XML non-unit clauses can be employed to define the axiomatic semantics of each ontology modeling primitive which includes a certain notion of implication.

Besides sole employment of the predefined modeling constructs, XDD goes beyond the expressiveness of a particular ontology language by yielding facilities for modeling arbitrary rules, axioms, constraints, and queries in terms of XML non-unit clauses.
6.2. Knowledge Grid Service Description

Before a knowledge grid can offer a service to others, its properties, capabilities, and interfaces need to be advertised in the service registries which can be publicly accessed by the other knowledge grids. These service registries are also knowledge grid services themselves, which maintain service ontologies as well as knowledge about available services in the Knowledge Grid.

Recent industrial and standard service description languages can be used for knowledge grid discovery, execution, and composition. Examples include UDDI (UDDI 2000), which defines interface of a public service registry, WSDL (Christensen et al. 2001), which describes protocol-independent functional description of services, OWL-S (Burstein et al. 2004)—an OWL-based ontology markup language for describing Web services—and WSMO (Roman, Lausen, and Keller 2004)—an ontology for describing Semantic Web services, the syntax of which is defined by WSML (de Bruijn et al. 2004) and includes human-readable, XML and RDF syntaxes.

These descriptions, represented by ServiceProfile-element in OWL-S, WebService-element in WSMO, or businessEntity and businessService in UDDI, are directly modeled as XML unit clauses, while their additional relationships, such as preconditions, postconditions, constraints, and the service semantics are expressed by appropriate XML clauses. A search for a particular knowledge grid service is expressed as an XML clause; its head specifies the pattern of the returned result and its body describes properties and capabilities of the desired knowledge grid. Such a clause will be evaluated on service registries and will return as its reply a list of qualified knowledge grid services.

Furthermore, XDD enables integration of knowledge grid services by allowing a service to be an aggregation of the existing ones. A definition of a service composition can be represented by an XML clause with its head specifying the composition result and its body the composition rules and constraints. In other words, a composition rule has the form:

\[ H \leftarrow B_1, \ldots, B_n, \beta_1, \ldots, \beta_m \]

where

- \( H \) models the service resulting from integrating services from \( n \) knowledge grids,
- the \( B_i \) represents the aggregating \( n \) services, and
- the \( \beta_j \) describes composition rules and constraints.

Similarly, queries (or service requests) are formalized as XML clauses. Execution of a given query by XET reasoning engine is performed by transforming the query clause successively and equivalently using the implemented ET rules and the formalized knowledge in the Knowledge Grid.

Since service discovery, composition, negotiation, execution, and monitoring in Semantic Web or in Knowledge Grid environment are themselves complex and difficult problems (Keller 2004), they become active research area. This section merely outlines the XDD approach, while further detailed research and investigation on this dynamic aspect of services by means of XDD as well as current prominent technologies are still ongoing. To summarize, Table 2 and Figure 6 depict the role of XDD as a foundation for knowledge grid modeling by employment of open, most recent, technologies—OWL, OWL-S, WSMO, UDDI and WSDL. In the figure, each number in the parentheses indicates a certain modeling component described by Table 2. Note that since the proposed framework is general and not restricted to particular languages, besides using the suggested ones, other technologies can also be employed.
TABLE 2. Knowledge Grid Modeling Components

<table>
<thead>
<tr>
<th>Modeling Components</th>
<th>Represented By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Knowledge Grid service descriptions</td>
<td>XDD using OWL-S, UDDI, WSDL or WSMO</td>
<td>Registry information for service advertisement and discovery, Declarative interfaces for knowledge grid execution.</td>
</tr>
<tr>
<td>2. Service composition rules</td>
<td>XDD or XDD using WSMO</td>
<td>Modeling of new services as aggregation of several existing ones.</td>
</tr>
<tr>
<td>3. Ontology-based conceptual model</td>
<td>XDD using OWL or WSMO</td>
<td>Modeling of a domain conceptual model comprising ontology definitions and instances.</td>
</tr>
<tr>
<td>4. Ontological axioms</td>
<td>XDD</td>
<td>Definitions of the axiomatic semantics of each ontology modeling primitive.</td>
</tr>
<tr>
<td>5. Domain-specific rules</td>
<td>XDD or XDD using WSMO</td>
<td>Modeling of application-specific rules &amp; logic, axioms, constraints and queries.</td>
</tr>
</tbody>
</table>

FIGURE 6. Knowledge Grid modeling language layer.

7. RELATED WORK

Important, current approaches to XML-based knowledge representation languages which can be employed to model the Knowledge Grid can be classified into: rule-based, ontology-based and their integration.
With an effort to extend XML’s capabilities by rule-expression and reasoning mechanisms, there arise several XML rule languages developed for various applications and purposes, such as BRML (Grosof et al. 1999), RFML (Boley 2001) and XXML (Lee and Sohn 2002). Most, if not all, of these existing rule-based approaches are mere XML encoding of particular logic program theories, which exhibit difficulty in direct representation and manipulation of arbitrary Web resources encoded in XML. In other words, these approaches demand XML data and their schemas to be translated into particular predefined representations corresponding to the employed original rule language formalisms.

Major, recent ontology-based approaches such as RDF(S) (Lassila and Swick 1999; Brickley and Guha 2003), DAML+OIL (Hendler and McGuinness 2000) and OWL (Bechhofer et al. 2004) are theoretically founded on Description Logic (DL). They offer expressive facilities for modeling ontology definitions by means of their modeling constructs such as subClassOf, subPropertyOf and inverseOf. However, they still lack capabilities for definition of arbitrary axioms beyond those predefined constructs. Moreover, their logic foundation on Description Logic merely concentrates on concept (or class) expressions and subsumption without a concern on property subsumption—an important language feature expressed by means of the subPropertyOf construct. In order to deal with such a limitation, Qu (2004) proposes Predicate-Ordered Sort-Ordered Logic (POSOL) by extending order-sorted logic (Beierle and Hedtstuck 1992) with the partial order on predicates (or property names). Its employment to model the Knowledge Grid is also discussed. Although POSOL can formally describe the semantics of OWL, a complete algorithm for mapping complex OWL constructs into POSOL sentences as well as a set of inference rules to reason with them are still underway.

An attempt to combine the two main approaches: rule-based and ontology-based has resulted in the development of Description Logic Programs (DLPs) (Grosof et al. 2003) which is defined as their expressive intersection. An initial study has been carried out on equivalence mapping of a fragment of DAML+OIL DL onto corresponding Horn rules, while further study on their expressive integration/union for a more complete interoperability between rules and ontologies is part of its future work. Semantic Web Rule Language (SWRL) (Horrocks et al. 2004) is another proposal for rule extensions of OWL DL and OWL Lite. However, similar to many existing XML rule-based approaches, SWRL simply provides mere XML encoding of Horn-style rules, where rules are not succinctly represented in a straightforward manner and hence make them hardly readable and difficult to be interpreted. One of the ultimate goals of DLPs and SWRL is to enable rules to have access to ontological definitions and conversely to enable ontology definitions to be supplemented by rules. As demonstrated in several examples, this goal can be readily achieved by the framework developed in this paper.

Web Service Modeling Ontology (WSMO) (Roman et al. 2004) is a recent proposal which develops an ontological infrastructure for describing Semantic Web services and for handling with service integration problem. Web Service Modeling Language (WSML) (de Bruijn et al. 2004) is proposed as the language for WSMO. It provides also a logical language for defining logical statements in WSMO, which is basically a first-order language with certain F-logic (Kifer et al. 1995) modeling constructs. There are a number of language variants in WSML with different degrees of logical expressiveness and thus leading to different computational complexity: WSML-Core, -Flight, -DL, -Rule and -Full. Their underlying logical formalisms range from the intersection of DL and Horn-logic to full first-order logic with non-monotonic extensions.

Zhuge (2002) presents a different approach to knowledge grid modeling with a focus on sharing and managing globally distributed knowledge resources. The approach models a world wide knowledge grid (WWKG) as a three-dimensional knowledge space:
(knowledge-category, knowledge level, location) identifying knowledge contents and their locations and expressed by its proposed simple XML-based language. Zhuge (2002) also develops a knowledge grid operation language KGOL, providing a set of basic operation statements for insertion, update and access of knowledge at a particular location. Nevertheless, such an approach mainly deals with knowledge management functionalities, and does not address the semantic foundation as well as computation and reasoning mechanisms of its developed knowledge representation. To fill this need, XDD and XET can readily be employed to serve as its foundation, where a knowledge grid represented by its proposed XML knowledge representation becomes immediately an XML unit clause in XDD with well-defined declarative semantics, and hence allowing an application of inference mechanisms by ET paradigm.

8. CONCLUSIONS

The paper has proposed the theory of XDD with FLCs which readily provides additional descriptive facilities on top of existing ontology-based representation languages. It allows ontology definitions and instances to be directly represented and their meanings formally determined. Moreover, it augments such languages’ representation power by the abilities to define rules, relationships and axioms in terms of XML clauses with FLCs in a more natural and straightforward manner. Thus, complex business rules, policies, constraints and queries, which are inexpressible by such languages, become uniformly representable and computable under a single formalism. Besides direct representation of XML-based ontology languages, any ordinary well-formed XML elements can be exploited in XDD as basic modeling elements and semantic units, and hence making it possible to reason with arbitrary XML applications without a necessity to translate or re-annotate them using particular ontology languages.

XDD achieves sound, efficient and flexible computation by means of the ET framework. It computes by successive transformation of a given XDD description, which represents a problem’s specification, into an equivalent but simpler one, which readily and directly yields the answers to the problem. XET reasoning engine which provides a programming and execution environment for transforming XDD descriptions under the ET framework has been developed and used in a number of prototype applications. For more information, please refer to http://sourceforge.net/projects/xet and http://kr.cs.ait.ac.th/xet.

Founded on the proposed representation and computation mechanisms as well as a service-oriented architecture, the paper has also developed a framework for Knowledge Grid modeling. In particular, it models a knowledge grid as a knowledge resource which provides a set of services and can return a result. A description of domain-specific knowledge maintained by a particular knowledge grid is represented as an XDD description. It comprises an ontology-based conceptual model, ontological axioms and domain-specific rules expressed uniformly as XML clauses. In addition, knowledge grid service descriptions, represented by such standard service description languages as OWL-S, WSMO, UDDI and WSDL, are directly expressed as XML (unit) clauses, while their additional relationships such as pre-conditions, post-conditions, constraints and compositions are formulated by appropriate XML clauses. Similarly, queries (or service requests) are formalized as XML clauses. Execution of a given query by XET reasoning engine is performed by transforming the query clause successively and equivalently using the implemented ET rules and the formalized knowledge in the Knowledge Grid.

Further research includes enhancement of the proposed framework by employment of OGSA architecture and Globus Toolkit (Foster et al. 2002) for the definition of uniform knowledge grid service semantics and their implementation. Development of more efficient
ET rules specific for XML data structure as well as incorporation of general built-in ET rules into the XET engine for dealing with FLCs are ongoing. Implementing the tableau method as a set of appropriate ET rules for efficiency improvement is also one of the authors’ research interests and is under investigation.

REFERENCES


This appendix proves that ET Rules 1–3 for computation with negative constraints, given in Section 5.3, are semantic-preserving transformations. Throughout the appendix, let \( P, Q, Q', \) and \( Q'' \) be XDD descriptions, \( C \) and \( C' \) be XML clauses, and \( a \) and \( b \) be XML expressions. Moreover, assume without loss of generality that \texttt{xet:NotRule} is a reserved XML tag name not occurring in \( C, P, \) or \( Q \).

A. Proof of ET Rule 1

Proposition 1. Let \( Q \) and \( Q' = Q \cup \{ <\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \leftarrow a \} \) be XDD descriptions, and \( a \) an XML expression. Then,
\[
<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \theta \in \mathcal{M}(Q) \iff a \theta \in \mathcal{M}(Q).
\]

Proof. For \( \theta \in \mathcal{S}_X \),
\[
<\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \theta \in \mathcal{M}(Q) \\
\iff <\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \theta \in \mathcal{M}(Q \cup \{ <\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \leftarrow a \}) \\
\iff <\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \theta \in \mathcal{M}(Q) \cup \{ <\texttt{xet:NotRule}>\theta \in \mathcal{M}(Q) \cap \text{rep}(a) \} \\
\iff <\texttt{xet:NotRule}>a</\texttt{xet:NotRule}> \theta \in \{ <\texttt{xet:NotRule}>\theta \in \mathcal{M}(Q) \cap \text{rep}(a) \} \\
\text{\textit{if} xet:NotRule is a reserved tag name not occurring in C, P, or Q} \\
\iff a \theta \in \{ g \mid g \in \mathcal{M}(Q) \cap \text{rep}(a) \} \\
\iff a \theta \in \mathcal{M}(Q) \cap \text{rep}(a) \\
\iff a \theta \in \mathcal{M}(Q) \quad \text{\textit{if} a \theta \in \mathcal{M}(Q) then a \theta \in G_X \text{ and } a \theta \in \text{rep}(a)}
\]
Therefore, $<\text{xet:NotRule}>a</\text{xet:NotRule}>\theta \in \mathcal{M}(Q')$ and $a\theta \in \mathcal{M}(Q)$ are equivalent. ■

Proposition 2. $<\text{xet:NotRule}>a</\text{xet:NotRule}>\theta \in \mathcal{G}_X \Leftrightarrow a\theta \in \mathcal{G}_X$.

Proof. Since $\text{xet:NotRule}$ is a variable-free tag name, $<\text{xet:NotRule}>a</\text{xet:NotRule}>\theta$ is a ground XML expression in $\mathcal{G}_X$ if and only if $a\theta$ is a ground XML expression in $\mathcal{G}_X$.

Therefore, $<\text{xet:NotRule}>a</\text{xet:NotRule}>\theta \in \mathcal{G}_X$ and $a\theta \in \mathcal{G}_X$ are equivalent. ■

Theorem 1. Let

- $Q = Q \cup \{<\text{xet:NotRule}>a</\text{xet:NotRule}> \leftarrow a\}$,
- $C$: $(H \leftarrow B_1, \ldots, B_n [a \notin \mathcal{M}(Q)], B_{i+1}, \ldots, B_n)$,
- $C'$: $(H \leftarrow B_1, \ldots, B_n [<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q)], B_{i+1}, \ldots, B_n)$.

Then, $\mathcal{M}(P \cup \{C\}) = \mathcal{M}(P \cup \{C'\})$, and $P \cup \{C\}$ can be transformed into $P \cup \{C'\}$.

Proof. Given that $Q' \in Tcon$

$\Leftrightarrow a\theta \in \mathcal{G}_X - \mathcal{M}(Q)$ \quad // by Definition 2 and Definition 4
$\Leftrightarrow <\text{xet:NotRule}>a</\text{xet:NotRule}>\theta \in \mathcal{G}_X - \mathcal{M}(Q)$ \quad // by Propositions 1 and 2
$\Leftrightarrow [<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q)] \in Tcon$ \quad // by definition

Therefore, $\mathcal{M}(P \cup \{C\}) = \mathcal{M}(P \cup \{C'\})$, and $P \cup \{C\}$ can be equivalently transformed into $P \cup \{C'\}$. ■

From Theorem 1, one can readily yield ET Rule 1 (Merging of a Negative Object):

$[a \notin \mathcal{M}(Q)] \xrightarrow{\text{ET}} [Q' = Q \cup \{<\text{xet:NotRule}>a</\text{xet:NotRule}> \leftarrow a\}]$
$[<\text{xet:NotRule}>a</\text{xet:NotRule}> \notin \mathcal{M}(Q)]$.

B. Proof of ET Rule 2

Proposition 3. Let XDD description $Q'$ be transformed equivalently into XDD description $Q'' \cup \{<\text{xet:NotRule}>b</\text{xet:NotRule}> \leftarrow \}$. Then,

$\mathcal{M}(Q') = M(Q') \cup rep(<\text{xet:NotRule}>b</\text{xet:NotRule}>)$.

Proof. Given that $Q'$ can be transformed equivalently into XDD description $Q'' \cup \{<\text{xet:NotRule}>b</\text{xet:NotRule}> \leftarrow \}$, by definition of Equivalent Transformation:

$\mathcal{M}(Q') = \mathcal{M}(Q' \cup \{<\text{xet:NotRule}>b</\text{xet:NotRule}> \leftarrow \})$
$= \mathcal{M}(Q') \cup rep(<\text{xet:NotRule}>b</\text{xet:NotRule}>)$.
Theorem 2. Let

- \( Q = Q^{\prime} \cup \{ <\text{xet:NotRule}> b </\text{xet:NotRule}> \leftarrow \} \),
- \( C: (H \leftarrow B_{i_1}, \ldots, B_{i_2}, [ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ], B_{i_3+1}, \ldots, B_{i_2}) \),
- \( C^{\prime}: (H \leftarrow B_{i_1}, \ldots, B_{i_2}, [ a \not\in \text{rep}(b) ], [ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ], B_{i_3+1}, \ldots, B_{i_2}) \).

Then, \( \mathcal{M}(P \cup \{ C \}) = \mathcal{M}(P \cup \{ C^{\prime} \}) \), and \( P \cup \{ C \} \) can be transformed into \( P \cup \{ C^{\prime} \} \).

Proof. Therefore, \( \mathcal{M}(P \cup \{ C \}) = \mathcal{M}(P \cup \{ C^{\prime} \}) \), and \( P \cup \{ C \} \) can be equivalently transformed into \( P \cup \{ C^{\prime} \} \).

From Theorem 2, one can readily yield ET Rule 2 (Lifting Transformation):

\[
\begin{align*}
[ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ] \\
\{ Q = Q^{\prime} \cup \{ <\text{xet:NotRule}> b </\text{xet:NotRule}> \leftarrow \} \} \\
\xrightarrow{ET} [ a \not\in \text{rep}(b) ], \\
[ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ].
\end{align*}
\]

C. Proof of ET Rule 3

Theorem 3. Let

- \( Q^{\prime} \) be a description which cannot be transformed equivalently into a description \( Q^{\prime\prime} \cup \{ <\text{xet:NotRule}> b </\text{xet:NotRule}> \leftarrow \} \),
- \( C: (H \leftarrow B_{i_1}, \ldots, B_{i_2}, [ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ], B_{i_3+1}, \ldots, B_{i_2}) \),
- \( C^{\prime}: (H \leftarrow B_{i_1}, \ldots, B_{i_2}, B_{i_3+1}, \ldots, B_{i_2}) \).

Then, \( \mathcal{M}(P \cup \{ C \}) = \mathcal{M}(P \cup \{ C^{\prime} \}) \), and \( P \cup \{ C \} \) can be transformed into \( P \cup \{ C^{\prime} \} \).

Proof. It is obvious that if the referred description \( Q^{\prime} \) cannot be transformed equivalently into a description \( Q^{\prime\prime} \cup \{ <\text{xet:NotRule}> b </\text{xet:NotRule}> \leftarrow \} \), then \( [ <\text{xet:NotRule}> a </\text{xet:NotRule}> \not\in \mathcal{M}(Q^{\prime}) ] \) is evaluated to be a true negative.
constraint in $T_{con}$. Therefore, such a constraint can be eliminated from the clause. That is, $\mathcal{M}(P \cup \{C\}) = \mathcal{M}(P \cup \{C'\})$, and $P \cup \{C\}$ can be equivalently transformed into $P \cup \{C'\}$. ■

From Theorem 3, one can readily yield ET Rule 3 (Elimination of Negative Constraint):

\[
\begin{align*}
\alpha \notin \mathcal{M}(Q) \\
\{Q \text{ cannot be transformed into } Q' \cup \{b \leftarrow \} \} \\
\end{align*}
\]

\[\xrightarrow{ET} \langle true \rangle.\]