

1 Introduction

The World Wide Web (WWW) contains a vast amount of information valuable for planning and design tasks in engineering. Electronic product catalogues and online repositories, for instance, provide component descriptions which can support the human engineer as well as semi-automatic synthesis programs in finding appropriate parts. Engineering design involves usage of domain specific technical knowledge together with creative problem solving skills to select properly functioning components that comply with a set of requirements like performance goals, physical or business constraints like cost limits. Intelligent agents perform specific tasks on behalf of users and incorporate reasoning or planning to solve problems. This paper analyzes how semi-structured Web data and meta-knowledge, especially constraints, can be fused and transformed into a form which can be applied by intelligent agents in order to support further automated knowledge processing.

Web resources related to a domain may differ in syntax, structure and semantic content. Therefore, we may adopt the concept of a *mediator* which originates from database systems research. A mediator integrates applications and heterogeneous information sources in such a way that they keep their autonomy and has to ensure that information requests can be transformed, distributed and matched by information offers which are stored as concrete resources (cf. [Wie92]). More recently, in the area of Web-based information systems, additional features were introduced to cope with the demands of open system architectures. For instance, common interchange formats, document type definitions (DTDs), and shared meta-level knowledge and data, like standardised conceptual models, called *ontologies*, have been identified as central issues for the construction of distributed information systems which take their input from Web resources. The eXtensible Markup Language (XML) has emerged as a new standard for exchanging and representing data and structure on the Web. For a wide range of applications, XML allows users to define the structure of documents by means of Document Type Definitions. A DTD is essentially a formal grammar which restricts the structure of valid documents. Semantically similar product categorisations may coincide or differ in various structural aspects, such as equivalence, overlap, and inheritance of attributes or arithmetic measurement of values. An ontology is a formal explicit specification of an agreed standardised vocabulary, a precise definition of the basic terms and the relationships that can exist between them [Gru93]. Hence, the use of terms from an agreed ontology in product descriptions is suggested as a means to circumvent misunderstandings and support mediation. However, in many cases, the mediation task may require not only to match structures and semantic contents of documents, but more elaborate reasoning as well.

Among the many data models that can be used to enable reasoning and construction with the semi-structured information contained in Web resources and associated meta-information and knowledge, it is believed that graphs are the most adequate ones. Graph theory provides algebraic means, for example projections and equivalence, to model knowledge transformation and sharing. Most important, labelled directed acyclic graphs offer a formal data model for subsets of the standard XML for exchanging Web data and representing meta-data, and for the design of query languages as XML-QL (cf. [D⁺98]).

Facilities to specify declarative constraints which apply to the structure and content of Web resources and interacting constraints and, furtheron, means for the automated processing of constraints are essential for the ef-

efficient automatization of complex tasks. The recent W3C recommendation XML Schemas (cf. [Fal01]) allows to specify more constraints than in a DTD. They have data types that, for example, allow to restrict the range of a value to a certain range, but not to compare the values of two pieces of data. Moreover, a description standard does not implement any constraint processing at all; like DTD validation, constraint processing is entirely an application issue. However, it is a standard practice to combine knowledge representation, logical reasoning and constraint solving in "constraint logics". In order to combine the advantages of an efficient graph-based handling of knowledge representation with the expressiveness and reasoning facilities of constraint logic, this paper analyses a logic-based representation framework which allows for efficient graph operations without losing the reasoning and constraint solving capabilities of constraint logic. It is based on graph-based data models for XML (cf. [GMW99]) and exploits a convenient formal correspondence between mathematical semantic structures (DAGs), syntactic term encoding, and logical specifications. This correspondence is extensively exploited in the SEAMLESS framework (cf. [Eus00]) which allows to specify and solve complex synthesis problems as is required for the mediator tasks described above. A graph term algebra and canonical forms for classes of equivalent objects which enable efficient graph operations within a logic-based framework are employed to implement the functionality.

The remainder of this paper is organised as follows. Section 2 presents an architecture for knowledge mediation on the Web. Section 3 describes the graph-based knowledge representation language. Section 4 shows how to formalise ontologies and constraints attached to Web meta data by constrained feature graphs and clauses. Section 5 sketches the synthesis of rules that link ontological descriptions and DTDs, and describes the mediator shell. Finally, we compare this framework to other approaches and summarize our work.

2 An Architecture for Knowledge Mediation

The WWW can be considered as a market place where services and products are freely offered and requested. The general idea of a *mediator system* is to integrate applications and heterogeneous information sources in such a way that they keep their autonomy (cf. [Wie92]). A mediator has to ensure that semantic knowledge requests from users or applications can be transformed, distributed and matched by information offers which are stored as concrete resources, having resolved semantic ambiguities. Matching service descriptions have to be transformed back into high-level information, suitable for further processing. The KRAFT project has developed an agent-based framework for distributed query processing [PHG+99]. All information processing units are realised as interacting software agents, using a subset of the Knowledge Query and Manipulation Language (KQML) performatives [FLM97]. The *facilitator* maintains a data base ("yellow pages") of all registered service providers and is able to map service types to appropriate providers. A provider first has to export the type of service it wishes to advertise via the facilitator. Information resources are coupled to the mediator, using *wrapper* modules. Information is exchanged in the KRAFT Constraint Interchange Format (CIF).

With the vast amount of information and hidden knowledge in Web resources, standards as XML and DTDs for structuring, typing and se-

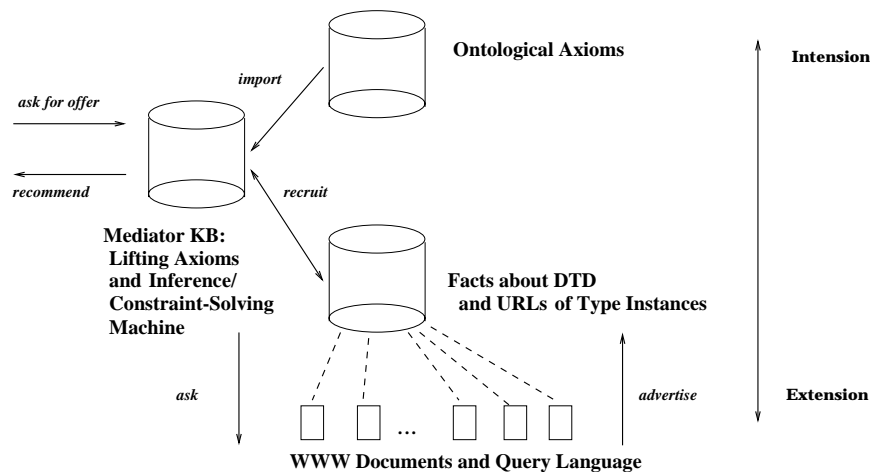


Figure 1: An Architecture for Mediation

semantic conceptual abstractions as shared ontologies have become evident. Using current data models for WWW information, there seems to be no way of declaring basic types and to constrain admissible values in the same formalism. The presented work establishes means that enable integration, representation and reasoning using Web standards, semantic concepts, and domain-specific knowledge like constraints. Mediation tasks as query processing and integration of heterogeneous information sources can be handled more effectively. Problem solving tools can re-use integrated and transformed knowledge for the automated solution of complex tasks. Knowledge integration and transformation is achieved by adopting the SEAMLESS multi-layer architecture (cf. [Eus00]). DTDs and ontology are incorporated into the distributed architecture through an intermediate context-dependent mediator knowledge base that in SEAMLESS terms provides a viewpoint. Figure 1 demonstrates the resulting architecture. The mediator architecture thus employs three essential technologies which will be shortly examined: "XML", "Shared Ontology" and "Viewpoints".

Maintenance of XML Documents The eXtensible Markup Language (XML) is a machine readable format for structured information representation on the Web and data exchange across networks and between heterogeneous information systems. An XML document primarily consists of a strictly nested hierarchy of tagged elements with a single root. Each tagged element has a sequence of zero or more attribute/value pairs, and a sequence of zero or more subelements. Access to semi-structured XML Web data is realised through a number of query languages as XML-QL (cf. [D+98]) or Lore (cf. [GMW99]). The XML-QL example in Figure 2 selects in the document at `www.modem_company.com` all modems that support "Video-Conferencing" and generates a price list.

```
WHERE <pc_modem_card> <type>video_conferencing</type></pc_modem_card>
<price><value>$value</value></price>
in www.modem_company.com CONSTRUCT $value
```

Figure 2: XML-QL Query

Standard vocabularies are summarised in Document Type Definitions (DTDs). A DTD is essentially a formal grammar which declares tags and their structural relations and restricts the structure of valid documents. For example, a simplified document type description for a PC-Modem card is sketched in Figure 3.

```
<!ELEMENT pc_modem_card(type,speed,price)>
<!ATTLIST pc_modem_card model CDATA>
<!ELEMENT type(#PCDATA)>
<!ELEMENT speed(speed_unit,speed_value)>
<!ELEMENT speed_value(#PCDATA)>
<!ELEMENT speed_unit(#PCDATA)>
<!ELEMENT price(value,unit,measure)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT measure(#PCDATA)>
<!ELEMENT unit(#PCDATA)>
```

Figure 3: XML Document Type Definition

The *facilitator* knowledge base encompasses descriptions of all registered service providers, which are consistent with a given set of DTDs and found at specific URLs. This knowledge is stored as facts about DTDs and associated lists of the locations (URLs, Uniform resource locators) of individual services which are consistent with the type definitions.

Shared Axiomatic Ontology Different applications can associate different meanings with similar or identical terms. Whereas humans can (usually) distinguish between different interpretations of either the information content or transaction intent, agents will only be able to act meaningfully if the content exchanged between them, and the services they provide, carry sufficient explicit information. In order to enable consistent behaviours amongst the (virtual) participants in a virtual market and to allow complex interactions such as mediation, greater levels of semantic content have to be made explicit and represented in a computer readable form. Shared ontologies have been identified as central issue for the development of open distributed environments.

An ontology is a formal explicit specification of an agreed standardised vocabulary, a precise definition of the basic terms and the relationships that can exist between them [Gru93]. An ontology may specify knowledge and information on various abstraction levels. The OntoBroker [DEMS99] uses formal ontologies to extract, reason, and generate meta-data in the WWW. Various organisations develop international standards for business processes, models or product descriptions to enable the integration of heterogeneous applications and human-users for specific business processes across the supply and production chain. For example, RosettaNet, a consortium of manufacturers, distributors, resellers, end users, and solution providers, devises a framework for business-to-business electronic data interchange over the Internet. Within the RosettaNet framework, a technical standard for a universal laptop has been developed (cf. [Ros98]). Ontologies which standardise certain behavioural aspects of business transactions are also being developed.

Ontologies in this extended SEAMLESS KRAFT architecture shall exist as axiomatisations of generic domain models. Axioms identify a basic

set of abstract properties, characteristics and associations for the semantic description of services, products or generic processes. For example, generic domain models may exist in the form of formal specifications for devices and components, using constraint logic programs.

Mediation by Viewpoints From the user's perspective, an ontology may provide a meta-level description on the underlying data. In this sense, ontologies act as generic semantic types, that is, they are intensional conceptual descriptions. A concrete information resource, for example a concrete document, may implement an abstract type with certain changes in structure, syntax and symbol names. In concrete inference processes about problem-related facts which need to be extracted from Web documents the system has to rely on knowledge about the related DTDs.

Applying the SEAMLESS architecture, a mediator knowledge-base can be constructed as a viewpoint. Generally, a viewpoint is an intermediate, loosely coupled component that provides the assumptions under which meta-level axioms become true in a particular context. It describes how to "lift" facts and data from the consulted local repositories to a more abstract level so that they can be compared and integrated with others from different sources. A viewpoint may contain additional domain knowledge that supports the integration task.

The viewpoint concept is applied to set-up mediator knowledge-bases that combine a shared ontology and DTDs. Lifting rules shall make causal relationship and applicable constraints among DTDs and a given shared ontology explicit. Using viewpoints supports easy and flexible construction of the mediator knowledge-base in the highly dynamic WWW context. On the basis of a shared ontology, users or software agents can pose semantic queries to the mediator knowledge-base without having to consider the location, the format or context-specific information of the underlying data. The mediator uses knowledge-based reasoning and embeds constraint solving to generate precise data requests. Individual records will then be retrieved using wrappers. Wrappers transform queries from the logical representation format to the existing XML source format, for example XML-QL, using the parsing rules and facilitator knowledge about locations and distribute them in a bi-directional way.

3 Graphs for Structured Web Knowledge Bases

The representation of the various domain-specific Web knowledge resources as XML documents, DTDs, local and shared ontologies in a form suitable for knowledge-based reasoning requires transformations on different levels. In [Eus00], it has been pointed out that graphs are an adequate means to transform data and information from heterogeneous sources on different abstraction levels into re-usable knowledge. SEAMLESS exploits a convenient formal correspondence between mathematical semantic structures (DAGs) or feature graphs, syntactic term encoding and logical specifications. It uses graphterms for the encoding of feature graphs as well as object graphs, and term rewriting for the implementation of operations. Having derived canonical term representations for classes of isomorphic DAGs allows to perform matching operations in an effective way [Eus97].

Graphs capture the structure of a wide range of data and knowledge modelling means as for example entity relationship diagrams. They are perhaps the most common natural means for capturing abstract structure

of composed systems and to facilitate construction tasks. A complex site in the World-Wide Web has a natural abstraction as a graph where node parameters may be bound to page locations (URLs) and links describe structural and semantic relationships between parts of the hyper-document. For instance, a software or hardware system may be modularised as a graph or configuration of "components" or "parts" and "connectors" [Gar95]. The structure of a C-Web that is the properties associated to classes of documents and data sets and the various relations linking these classes together, is modelled by a labelled directed graph [Ch00]. Generally, when using graphs for knowledge representation, nodes represent data values, object identifiers or the domain of values and edges the relationships between them.

XML Graphs Most important, labelled directed graphs offer a formal data model for subsets of standards for exchanging or modelling Web data and meta-data like XML and RDF [LS99], and for the design of query languages. Query languages like XML-QL (cf. [D⁺98]) and data management systems like Lore (cf. [GMW99]) use labelled directed graphs to model (semi)-structured data and regular path-expressions to navigate through graphs. In the remainder of this paper, a slight variant of the graph data models [D⁺98], [GMW99]) shall be used for the representation of XML documents.

Definition 1 *A XML graph consists of:*

1. *A labelled directed acyclic graph G , in which each node is represented by a unique string called element identifier;*
2. *G 's inner nodes are labelled with sets of attribute-value pairs, where each attribute-name is a string and each atomic-value has an atomic type drawn from integer, real, string or ID, IDREF or IDREFS;*
3. *G 's leaves are labelled with values (strings);*
4. *G 's edges are either labelled with element tag identifiers, where each label is a string, or, attributes of type IDREF and IDREFS which point from the referring element to the referenced element;*
5. *G has a distinguished node called the root.*

Using the data model in definition 1, concrete XML documents can be mapped rather straightforward into syntactic graphs where certain nodes of the graph are annotated with semantic values of the constituent the node represents. For example, a pc modem card description which conforms to the DTD in Figure 3 could be modelled by the XML-graph in Figure 4.

An important abstraction mechanism for building knowledge bases is building a hierarchy of classes or concepts and attributes that characterise them, based on the content of their knowledge objects. The classes are ordered, according to subsumption relationships. Hierarchical orderings can be exploited to build more efficient retrieval and matching procedures. In this paper, feature graphs will be employed to abstract from sets of XML documents, to represent some simple DTDs and conceptual knowledge in a uniform manner. In order to enable the representation of attributes in the XML sense, the common definition of feature graphs is slightly modified. Sets of attribute-variable pairs may be assigned to inner nodes, whereas, usually, inner nodes of feature graphs are considered to be variables.

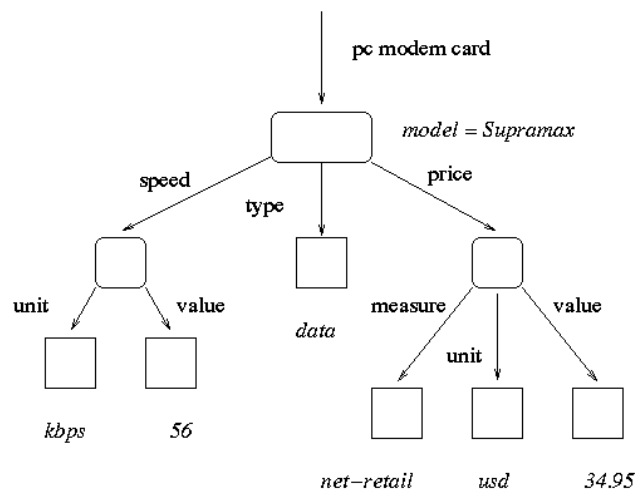


Figure 4: XML Graph PC Modem Card

Definition 2 Suppose that there are given a finite set of variables, a set of features, and a set of constants. A feature graph is a finite, rooted, connected and directed graph whose edges are labeled with feature symbols so that the labels of the edges are pairwise distinct. Every inner node of the feature graph is a variable or assigned a set of attribute-variable pairs. Every terminal node is either a constant or a variable.

Obviously, feature graphs can be constructed from XML graphs by replacing XML value parameters with variables.

XML Graphterm Encodings (Serial) Herbrand terms, named *graphterms*, were devised as part of the author's previous work. Graphterms provide means for the encoding of finite, rooted and labelled DAGs and abstraction [Eus97]. In this context, features or edge labels are seen as binary operators that must be interpreted as functional relations. From function symbols, constants and variables, graphterms are recursively constructed. Let f be a function symbol, $Nodelabel$ be either a constant, variable or an expression. A *graphterm* is a recursively constructed term $f(Nodelabel, Graphterm\ list)$, where *Graphterm list* is either the empty list $[],$ or denotes a list of graphterms. For each labelled rooted DAG, there is a graphterm representation. To ensure that graphterms can be interpreted as rooted, directed, acyclic graphs in a one-to-one way, further axioms constrain the set of valid graphterms (cf. [Eus97]). A *graphterm abstraction* is a graphterm that is constructed from a ground graphterm by substituting nodelabels with variables, based on a mapping from the set of nodelabels into a finite set of variables. Graphterm abstractions may be considered as encodings of feature graphs.

To define that two graphterms are equivalent, it is recalled that graphterms are an encoding of graphs. Two graphterms are equivalent if the graphs associated with them are identical. Isomorphism defines a further equivalence relation on graphterms. For each DAG, there exist a combinatorial number of graphterms that represent the given graph. One major result concerning our graphterm encoding is that there exist canonical forms

for sets of equivalent graphterms. Canonical forms that represent identical graphs are to be constructed from any representant by ordering subgraphs in a recursive manner, using the graphterm ordering defined in [Eus97]. Canonical representatives for classes of isomorphic DAGs are constructed from canonical ground graphterms by abstraction, based on an isomorphic mapping from the set of nodelabels into a finite set of variables.

Canonical forms can lead to exponential reductions in the size of the knowledge base and the complexity of retrieval within a logic-based framework. Let T_1, T_2 be a canonical ground graphterms, C be the abstraction graph, constructed from T_1 , $var(C)$ denote the set variables. T_1, T_2 denote identical graphs, if $T_1 = T_2$. The graph associated with T_1 is isomorphic to the graph associated with T_2 , iff C *subsumes* T_2 , i.e. if and only if there exists a term substitution σ such that $\sigma(C) = T_2$.

The first step in setting-up the Web knowledge-base consists in constructing the graphterms from extracted Web resources stored in different persistent information resources to map it onto structures on which logical and algebraic operations apply. XML graphterms define specific syntactic graphterms from existing Web resources. Each node label can be associated with a set of properties by an attribute list. Figure 5 shows a graphterm encoding of the XML data, visualized in Figure 4.

```
pc_modem_card([model("Supramax"),
[type([], data),
speed([], [speed_value([], 56), speed_unit([], kbps)]),
price([], [value([], 34.95), unit([], usd), measure([], net_retail)])]).
```

Figure 5: Graphterm Encoding of XML data

A XML document is translated into a graphterm representation $f(\text{AttributeList}, \text{ListofSubgraphs})$, where *ListofSubgraphs* could be the empty list, as follows:

- Algorithm 1**
1. *The element tag identifier of the document root element is the outmost functor f of the representing graphterm.*
 2. *The list of attribute-name/atomic-value pairs belonging to the element are mapped into AttributeList.*
 3. *While the set of child elements is non-empty, for each child element the graphterm encoding is generated and successively inserted into the list of subelement representations ListofSubgraphs. If a child element contains an attribute of type IDREFS that points to an element, then a copy of the referenced element is generated and inserted into the list of subelement term encodings.*

XML Graphterm Operations Various operations on graphterms that enrich the built-in facilities of a logic programming language and that implement operations on graphs were devised and implemented (cf. [Eus97]). These functions can be used to query, select, construct and transform XML graphterms and feature graphs. The term encoding of XML data and meta-data facilitates efficient reasoning in the same formalism.

Conjunctions of sequences of features, i.e. path expressions of the form $feature.feature_1.feature_2 \dots, G$, which may include wildcards, can be used to identify parts of XML elements and attributes. For example, let

XT denote the XML term in Figure 5. Interpreting $select_subgraph(XT, pc_modem_card.type)$ returns $type(data, [])$. Interpreting $replace_nodevalue(XT, pc_modem_card.price.value, unknown)$, transforms XT into a XML term where the price value has been changed to $unknown$. From XML graphterms, graph abstractions are constructed by replacing node and attribute values with disjoint variables in a one-to-one way. Figure 6 shows the result of applying an abstraction to the XML graphterm in Figure 5.

```
pc_modem_card([model(Name)],
  [type([], Data),
   speed([], [speed_value([], Value), speed_unit([], Unit)]),
   price([], [value([], Price), unit([], Currency), measure([], M)])]).
```

Figure 6: XML Graphterm Abstraction

Feature graphs constructed from a canonical XML graphterm by an abstraction encode structural properties of a set of document instances. They may be considered as the encoding of a basic DTD, where element identifiers are wrapped into feature symbols and element attribute symbols into attributes of the feature graph.

This representation induces efficient procedures for validating XML documents by term subsumption. Let G be a XML graphterm, C be a feature graphterm, $var(C)$ denote the set of leaf variables, $const(G)$ denote the set of element values. The XML graph G is an instance of the feature graph C , C *subsumes* G , if there exists a substitution $\sigma : var(C) \rightarrow const(G)$ such that $\sigma(C) = G$, supposing canonical term representations. In other words, for this restricted forms of document structure description that are represented by feature graphs, the simple built-in term subsumption facility of a logic programming language can be used to verify conformance of a XML document to a structure description. The proposed representation format allows to state arbitrary knowledge about XML documents or structure abstractions by atoms, i.e. predicates that are applied to graphterms. Figure 7 shows an example.

```
category(
  pc_modem_card([model(Name)],
    [type([], Data),
     speed([], [speed_value([], Value), speed_unit([], Unit)]),
     price([], [value([], Price), unit([], Currency), measure([], M)])]),
  product_description).
```

Figure 7: Knowledge about Web Data

Thus, Web knowledge can be easily integrated into any knowledge base.

4 Specifying Ontologies and Constraints

Having introduced models to formalise XML data and meta knowledge, this section aims at formal models for the specification of ontologies and constraint knowledge attached to Web meta-data. An ontology consists of domain concepts and further background knowledge. Feature graphs provide

a means for the representation of domain concepts. Semantic information on the domain of individual type variables provides valuable knowledge for subsequent effective information inferencing. Suppose that there are information resources that contain information about airline flights. It is only reasonable that the start time of an interconnecting flight is always later than the end time of the first leg. However, neither current approaches to ontological engineering as for example taxonomies or frame logic, nor W3C recommendations provide means to express and process constraints over different information sources.

Feature graphs have an isomorphic declarative logic representation. While it is a standard practice to combine logical reasoning and constraint solving in "constraint logics", in the specific case of reasoning with graph-structured knowledge the straightforward translation approach will suffer from the generally inefficient handling of this kind of inferences in general-purpose logic. In order to combine the advantages of the more efficient term-based handling of feature graphs used in SEAMLESS with the expressiveness of constraint logic, a combined framework which allows for efficient graph operations without losing the constraint solving capabilities is presented.

Definition 3

- A constraint satisfaction problem (CSP) is given by
 - a finite set of variables $\{X_1, \dots, X_n\}$;
 - a function $X_i :: D_i$ which maps every variable X_i to a finite set D_i , its domain;
 - a finite set of constraints which restrict the values that the variables can simultaneously take. For example, an arithmetic constraint is an expression of the form $X_1 R X_2$ where R is a relation symbol $\{=, \neq, >, \geq, <, \leq\}$ and X_1, X_2 are either variables or finite domain constants.
- A solution to a constraint satisfaction problem is an assignment of a value from its domain to every variable, in such a way that each constraint is satisfied.

Exploiting the logical correspondences among graphs and logical specifications, it is proposed to model semantic constraints as constraint satisfaction problems that augment the declarative feature graph specifications. Domain concepts are represented as constrained feature graphs. Further background knowledge is represented by constrained feature clauses. These are clauses which embed graphterms that are interpreted as feature graphs.

Definition 4

- A constrained feature graph is an expression

$$\forall X, X_1, \dots, X_n : c(X, Gterm) \wedge X :: D \wedge X_1 :: D_1 \wedge \dots \wedge X_n :: D_n$$

- A constrained feature clause is a constrained feature graph or an expression

$$\forall X, X_1, \dots, X_n : c(X, Gterm) \wedge X :: D \wedge X_1 :: D_1 \wedge \dots \wedge X_n :: D_n \wedge Constraints \wedge P_1 \wedge \dots \wedge P_m$$

where

- "Gterm" refers to a XML feature graphterm, "var(GTerm)" $\supset X_1, \dots, X_n$, "Constraints" refers to a finite conjunction of constraints, and P_1, \dots, P_n denote atoms. An atom is a predicate applied to constants, variables or terms, especially graphterms.
- the sets of variables are disjoint:
 $\{X, X_1, \dots, X_n\} \cap \text{var}(P_i)_{i=1, \dots, m} = \emptyset$. Joins are made explicit by replacing shared variables with explicit equality; thus $p(X, Y) \leftarrow q(X, Z)$ would be replaced with $p(X, Y) \leftarrow q(X, Z) \wedge Y = Z$.
- similarly, value assignments are made explicit; thus $p(X, a)$ is replaced by $p(X, Y) \leftarrow Y = a$.

Figure 8 illustrates a representation of a domain concept modem, using constrained feature graphs.

$$\begin{aligned}
 \forall \text{Name}, \dots : \text{modem}(\text{Name}, [\\
 & \text{interface}(S, []), \\
 & \text{type}(T, []), \\
 & \text{speed}(_ [\text{speed_value}(Sp, []), \text{speed_unit}(Kbps, [])], \\
 & \text{linux_compatibility}(\text{model}(\text{ModelL}), [\text{status}(CStatus, [])]), \\
 & \text{in_stock}(\text{model}(\text{ModelS}), [\text{status}(SStatus, [])]), \\
 & \text{price}(_ [\text{value}(V, []), \text{unit}(U, []), \text{measure}(M, [])])) \wedge \\
 & S :: [\text{external}, \text{internal}] \wedge \\
 & T :: [\text{data}, \text{fax}, \text{voice}, \text{video_conferencing}] \wedge \\
 & Sp :: [28.8, 33.6, 56] \wedge KB = kbps \wedge \\
 & SStatus :: [\text{yes}, \text{no}, \text{limited}] \wedge \\
 & CStatus :: [\text{yes}, \text{no}, \text{limited}] \wedge \\
 & V > 0 \wedge U = \text{euro} \wedge M = \text{gross_retail_price}.
 \end{aligned}$$

Figure 8: Formal Representation of a Domain Concept Modem

As each mediation starts with a user request formalised as query, first the procedure is sketched which retrieves those ontological concepts which are consistent with the query. A query is consistent with a concept description, if and only if the canonical graphterms of query and the ontological specifications match, i.e. share the same term structure, and the union of both constraint sets is satisfiable.

5 Knowledge-based Mediation

The goal of this paper is to design a mediator that enables users or knowledge-based applications an integrated semantic access to XML encoded Web resources content and structure via ontologies, exploiting meta-knowledge, especially constraints and applying automated reasoning and constraint solving.

Consider the following scenario. A broker wants to deliver communication devices that match consumer requirements at all times. The broker has relationships with various parts manufacturers all over the world. Producers can advertise their product information to the facilitator by providing machine processable meta descriptions of multiple Web resources. Meta descriptions are submitted in the form of the DTDs that are used to structure the parts catalogue, such as price or weight, and the locations of information resources. The consumers formulate their requirements using a convenient Web interface at the broker site. Requirements are goals containing

complex combinations of conjunction of atoms, graph-path expressions and additional constraints on the occurring variables. Queries are expressed using terms from the shared ontology, without having to know about the location or content of context-dependent dynamically changing information resources. Figure 9 illustrates how a typical goal combines constraints of different types:

$$\begin{aligned}
&\Leftarrow \exists Model, Price : modem(model(Model), _) \\
&\wedge modem.speed \rightarrow max \\
&\wedge modem.linux_compatibility.status = ok \\
&\wedge modem.deliverable.status = ok \\
&\wedge Price = modem.price.value \wedge Price < 40 \wedge \\
&\quad modem.price.unit = euro.
\end{aligned}$$

Figure 9: Example Query with Constraints

One of the problems in satisfying this requirement is that the syntactic terms used by producers and consumers don't match. For example, even if a remote parts manufacturer has a required component in stock, the broker can't deliver immediately, because the shipping time has to be taken into account. Producers and consumers may use different measures, e.g. "prices before tax" versus "prices after tax", and units, e.g. "dollar" versus "euro", to calculate prices. Supplementary background knowledge, that is not provided by the producer, is needed to ensure that a specific modem is compatible with the Linux operating system.

The fundamental problem which needs to be solved is a knowledge mediation problem. That is, there is a goal object, the user request, expressed against the terms of a shared ontology, that has to be answered with semantic meaningful information. Information has to be constructed by exploiting additional knowledge, composing and transforming information pieces from various Web resources.

An ontology in this framework consists of domain concepts that are encoded by a set of rules built of feature graphs, containing a set of variables with attached constraints and stored in a knowledge base. The facilitator knowledge base stores meta information of Web resources, based on DTDs and locations of resources, by non-instantiated facts using feature graphs. When integrating ontologies with meta information that describe various Web resources, DTDs and parts of the domain concepts may equal and differ in various syntactic, symbolic and structural aspects such as renaming of symbols, overlap or set containment of attributes and features, and measurement of units, although they share some common semantics or an application objective. Knowledge sharing among parts of the recursively constructed domain concepts and DTDs is operationalised by mediation rules that are expressed by constrained feature clauses. Following examples show how knowledge sharing can be expressed.

1. **Specialisation:**

$$\begin{aligned}
modem(_, _) \leftarrow modem.type = internal \wedge \\
pc_modem_card(_, _)
\end{aligned}$$

2. **Overlapping domain values:**

$$\begin{aligned}
speed_measure(S_1, []) \leftarrow speed_measure(S_2, []) \wedge \\
S_1 = S_2 \wedge S_1 \in \{28.8, 33.6, 56\} \cap \{33.6, 56\}
\end{aligned}$$

3. Equivalence modulo **measurement of units**:

$$\begin{aligned}
 & price(-, [value(GP, []), unit(GU, []), measure(GM, [])]) \\
 \leftarrow & GM = retail_price \wedge GU = euro \\
 & \wedge LM = net_retail_price \wedge LU = gbp \\
 & \wedge GP = LP * ((1 + VAT/100) * Exchange) \\
 & \wedge vat(VAT, []) \\
 & \wedge exchange_rate(-, [target(GU, []), source(LU, []), \\
 & \quad rate(Exchange, [])], \\
 & \wedge price(-, [value(LP, []), unit(LU, []), measure(LM, [])])
 \end{aligned}$$

Parts of the recursively structured domain concepts have to be associated with concrete context-specific information stored in various XML resources. Starting from some given initial set of knowledge sharing axioms, mediation rules can be synthesised. The general idea of the synthesis process is to derive a conjunction of premises and constraints iteratively by finding equivalent DTDs for parts of the concept and fusing the constraints. The synthesis process terminates when the domain concept is decomposed into a set of subconcepts so that all subconcepts are associated with DTDs. Figure 10 visualises a synthesis step.

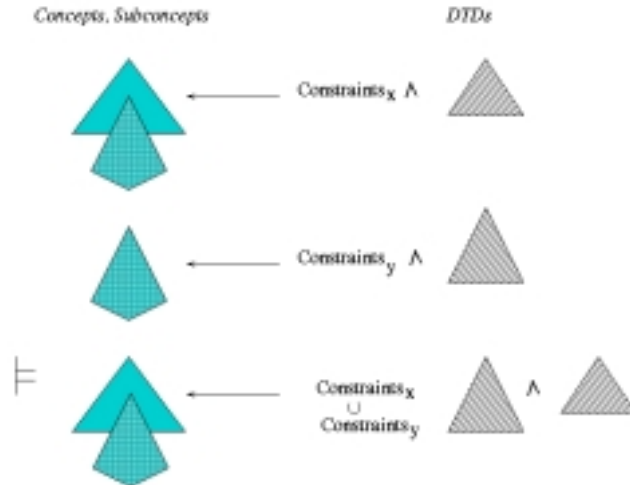


Figure 10: Generating Rules by Substitution of Equivalent Subgraphs

The synthesis process is guided by the following logical rules.

$$\begin{aligned}
 & DomainConcept \leftarrow Constraints_x \wedge DTD_x \\
 & Subconcept \leftarrow Constraints_y \wedge DTD_y \\
 & Subconcept \subset DomainConcept \\
 \vdash & \\
 & DomainConcept \leftarrow Constraints_x \cup Constraints_y \wedge \\
 & \quad DTD_x \wedge DTD_y
 \end{aligned}$$

For example, the generated clause in Figure 11 couples the context-specific description of a "PC modem card" (cf. Figure 5) and further information resources with the shared domain model introduced by the ontology.

A mediator shell is realised as a viewpoint that uses various methods for knowledge-based and graph-based reasoning of the SEAMLESS system, and the knowledge-base with shared ontologies, the context-dependent mediation rules, the facilitator facts about locations and further design knowledge. Using the proposed representation formats, constrained feature graphs and

```

modem(model(GName), [
  interface(GS, []), type(GT, []),
  speed(⊖, [speed_measure(Sp, []), unit(Kbps, [])]),
  linux_compatibility(model(ModelL), [status(CStatus, [])]),
  in_stock(model(ModelS), [status(SStatus, [])]),
  price(⊖, [value(GP, []), unit(GU, []), measure(LM, [])]) ←
  GS = internal ∧
  GSp :: [33.6, 56] ∧ GT :: [data, fax, voice] ∧
  GName = ModelL ∧ GName = ModelS ∧
  SStatus :: [ok, no, limited] ∧ CStatus :: [ok, no, limited] ∧
  GU = euro ∧ GM = gross_retail_price ∧
  LU = usd ∧ LM = net_retail_price ∧
  GName = LName ∧
  GSp = LSp ∧
  GP is LP * ((1 + VAT/100) * Exchange) ∧
  vat(⊖, [country(Country, []), vat_rate(VAT, [])]),
  exchange_rate(⊖, [target(GU, []), source(LU, []),
    rate(Exchange, [])]),
  linux_compatibility(ModelL, [status(CStatus, [])]),
  pc_modem_card(model(LName),
    [type(T, []),
    speed(⊖, [speed_value(LSp, []), speed_unit(Kbps, [])]),
    price(⊖, [value(LP, []), unit(LU, []), measure(LM, [])])]).

```

Figure 11: Mediator Lifting Rule

constrained feature clauses, we construct the knowledge base which can be considered to be a specific constraint logic program. Graph selection and application of the built-in reasoning services – such as matching, unification and constraint services – of a constraint-logic engine can then be used to efficiently automate basic mediation steps. Applying the proposed framework, the mediation task is achieved by the following steps in Algorithm 2:

- Algorithm 2**
1. Transforming the user request into a logical query by searching concept descriptions from the ontology knowledge base that match the graph path expressions and satisfy the attached constraints.
 2. Goal processing using the lifting rules and applying the built-in facilities of the constraint-logic programming language deducts a set of facts that need to be proven.
 3. Locating suitable external resources, using the facilitator knowledge base.
 4. Wrapping the retrieved XML information into a ground graphterm representation and storing them in a temporary knowledge base;
 5. Integrating returned values into the CSP and testing satisfiability. Whenever a solution to the CSP resulting from a query and a candidate concept description can be computed, the corresponding XML document values are regarded as valid answers. Satisfiability causes composition of the single results. Non-satisfiability causes backtracking until all possible information resources are searched.

For example, transforming the query in Figure 9 in Step 1 of the above algorithm yields the goal with attached constraints in Figure 12:

$$\begin{aligned} \Leftarrow \exists Name, \dots : & \text{modem}(Name, [\\ & \text{interface}(S, []), \\ & \text{type}(T, []), \\ & \text{speed}(_ , [\text{speed_value}(Sp, []), \text{speed_unit}(Kbps, [])]), \\ & \text{linux_compatibility}(\text{model}(ModelL), [\text{status}(CStatus, [])]), \\ & \text{in_stock}(\text{model}(ModelS), [\text{status}(SStatus, [])]), \\ & \text{price}(_ , [\text{value}(V, []), \text{unit}(U, []), \text{measure}(M, [])]) \wedge \\ & Sp \rightarrow \text{max} \wedge CStatus = \text{ok} \wedge SStatus = \text{ok} \wedge \\ & V < 40 \wedge U = \text{euro}. \end{aligned}$$

Figure 12: Transformed Query

Automated processing of the goal with respect to the mediator knowledge base in Figure 11 deduces information requests to multiple information resources. For example, the subgoal

$$\Leftarrow \text{exchange_rate}(_ , [\text{target}(\text{euro}, []), \text{source}(\text{usd}, []), \text{rate}(\text{Exchange}, [])])$$

and further information requests "linux-compatibility" and "in-stock" are generated. In step 3, the facilitator locates suitable external Web resources that could answer this information request. The retrieved XML information for the exchange rate 0.94 for euro against "usd" is wrapped into the graphterm format. Finally, the information about the "pc-modem-card" described in Figure 5 is accessed, satisfies the stated constraints and yields as model identifier $X = \text{"Supramax"}$.

6 Related Work

With the advent of expressive metadata standards, eg, XML or RDF, which allow to encode semantic annotations to data items, there is an increasing interest in methods for gathering data and information from the Internet for the purpose of automatically distilling knowledge that can be re-used by high-level applications. In the case of XML, such methods can exploit the well-known properties of labelled DAGs, a mathematical concept which provides a cognitively adequate and natural approach to represent static structure of XML Web knowledge bases and meta information, to model knowledge sharing, and to automate query-processing, transformations and constructions (cf. [D⁺98], [GMW99]). In this paper, graph models for Web data and metadata are an intermediate step on the way towards a representation that supports efficient automated reasoning and constraint solving.

Generally, the proposed mediator framework is influenced by approaches from the area of semi-structured data models in which ontologies modelled in declarative languages describe the domain of discourse as in HERMES [AE95], declarative languages are used for the specification of mediators as in TSIMMIS [], and automatization of mediation tasks is achieved by automated reasoning as in KOMET [CJKS97]. Within the Context Interchange framework [GMS95], ontologies not only exist as abstract conceptualization of particular domains: so-called "contexts" augment the ontologies by providing the assumptions pertaining the specific local situation. A clausal representation for contexts and ontologies is adopted. Based on this foundation, automated query answering is achieved by an abductive framework. ONTOBROKER [ES00] uses ontologies that are formalised in Frame Logic to derive DTDs which define the structure of documents intended to be

used during automatic brokerage processes. Ontologies are used to define a common conceptual structure which establishes a basis for the mutual understanding between the agents participating in this process. While most mediator projects in this area confine themselves to the solution of the "matching problem", the KRAFT (Knowledge Re-use and Fusion Transformation) architecture [PHG⁺99] focuses on the exchange and re-use of constraint knowledge held in distributed data sources, using a non-executable interchange format.

Intelligent mediator agents, also named matchmaker agents are emerging in e-commerce and e-business (cf. [GTM99]). Mediator agents are helping customers and commercial sites finding matching interests. In [FW99], a paradigm for content-focused matchmaking based on constraint solving is presented. Matchmaker agents interact with so-called "customers". The matchmakers knowledge base and the customers needs are both modelled as a network of constraints. Customer can refine their constraints in an iterative way. Matchmaking agents are conceived as constraint solvers that suggest tentative solutions that are consistent with the stated constraints until a satisfying solution is created. [Kel95] presents an agent-based architecture for smart catalogues and brokering that supports cross-search of multiple heterogeneous SQL-based product data bases. Personallogic is a commercial tool that enables consumers to narrow down their product search by allowing them to specify constraints on product features¹.

7 Conclusions

In this paper, we proposed a new mediator framework that provides both human users and automated applications an integrated straightforward access to composed information contained in multiple XML documents via ontologies. Employing meta-knowledge, especially constraints, supports the selection of the most appropriate choice. We conceive mediators as flexible components in multi-agent networks that establish currently missing links between a lot of interesting information (which resides on the Web), meta-knowledge, and the applications (which have no direct access to the Web data) or users, using wrappers and facilitators. Capturing arithmetic, set-theoretic or Boolean constraints as meta-knowledge is essential for incorporating information resources into systems for automated planning, synthesis, and design processes.

According to our knowledge, the proposed framework is the first approach that combines the automated extraction of knowledge from XML (meta)data, the formal representation thereof, and its augmentation through application and situation related constraint knowledge in a common constraint logic formalism. Exploiting the built-in reasoning and constraint-solving facilities of a constraint-logic programming language, several tasks can be efficiently automated.

The main contributions of this paper comprise: (1) A transformation procedure that translates XML documents into logical term structures. In other words, knowledge can be automatically extracted from XML Web sites. As a side effect, a DTD can be generated automatically from existing XML graphterms by a simple abstraction procedure applied to canonical XML graphterms. For some cases, validating a XML document against a DTD is reduced to verifying term subsumption. (2) A uniform constraint logic formalism for representing syntactic patterns, structure and meta-data

¹<http://www.personallogic.com>

of XML Web resources and meta-knowledge. Constrained feature graphs and constrained feature clauses enable constraints referring to the structure or content of Web resources and, most important, interacting constraints can be directly assigned to the XML feature graph term variables. Specific operations for retrieval, transformation, reasoning, constraint rewriting and solving that might otherwise have been performed by either combinatorial search or explicit specialized algorithms can be performed in a more efficient way by mechanisms that take advantage of the canonical graph term encoding and using the built-in algorithms for unification or subsumption of a logic programming language. (3) Lifting rules that provide the assumptions under which ontological axioms become true in a particular context and take into account constraints can be semi-automatically synthesized from atomic relationships among Web data and ontologies. (4) The mediation algorithms exploit the inherently recursive structure of the representation to cope with complex queries and documents by automatic decomposition of the original problem into subgoals that can be solved with reasonable computational effort. The underlying constraint logic proof procedure assembles the overall result automatically provided that solutions for all subproblems exist.

Future work will focus on the loose integration of concrete common ontologies and Web resources as knowledge bases into the SEAMLESS synthesis and learning system [Eus00]. The graph-based methodology itself can be extended to other semi-structured information resources.

Acknowledgements Part of this material is based on work supported by the KRAFT project in Aberdeen, Scotland; funded by the ESPRC and BT.

References

- [AE95] S. Adah and R. Emery A uniform framework for integrating knowledge in heterogeneous knowledge systems. In *Proceedings 11th International Conference on Data Engineering*, pages 513–521. Philadelphia, 1995.
- [Ch00] V. Christophides. Community Webs (C-Webs): Technological Assessment and System Architecture. Research report, C-WEB IST-1999-13479.
- [CGL99] D. Calvanese G. Giacomo and M. Lenzerini. Representing and reasoning on XML documents: A description logic approach. *Journal of Logic and Computation*, 9(3):295–318, 1999.
- [CJKS97] J. Calmet, S. Jekutsch, P. Kullmann, and J. Schü. Komet - a system for the integration of heterogeneous information sources. In *Proc. ISMIS'97*, pages 318–327. Springer Verlag, 1997.
- [DEMS99] S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In R. Meersman et al, editor, *Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8)*. Rotorua, New Zealand, Kluwer Academic Publishers, Boston, 1999.
- [D⁺98] A. Detsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for XML, 1998. "http://www.w3.org/TR/NOTE-xml-ql".
- [ES00] M. Erdmann and R. Studer. How to structure and access XML documents with ontologies. *Data and Knowledge Engineering, Special Issue on Intelligent Information Integration*, 36(3):317–335, 2001.

- [Eus97] J. Eusterbrock. Canonical term representations of isomorphic transitive DAGs for efficient knowledge-based reasoning. In *Proceedings of the International KRUSE Symposium, Knowledge Retrieval, Use and Storage for Efficiency*, pages 235–249, 1997.
- [Eus00] J. Eusterbrock. Composing reusable synthesis methods through graph-based viewpoints. In S. Hölldobler, editor, *Intellectics and Computational Logic, Papers in Honor of W. Bibel*, pages 143–158. Kluwer academic publishers b.v., 2000.
- [Fal01] D.C. Fallside (editor). XML Schema Part 0: Primer. *W3C Recommendation*, 2001; Available at: "http://www.w3.org/TR/xmlschema-1/".
- [FLM97] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In Jeffrey Bradshaw, editor, *Software Agents*. AAAI/MIT Press, Cambridge, 1997.
- [FW99] E. Freuder, and R. Wallace. Suggestion Strategies for Constraint-Based Matchmaker Agents. In *Principles and Practise of Constraint Programming (CP'98)*, Fourth International Conference, Pisa, *Lecture Notes in Computer Science, Vol. 1520*, pages 192–204. Springer, 1998.
- [Gar95] D. Garlan. Research directions in software architecture. *ACM Computing Surveys*, 27(2):257–260, 1995.
- [GMS95] C.H. Goh, S.E. Madnick, and M.D. Siegel. Ontologies, contexts, and mediation: Representing and reasoning about semantic conflicts in heterogeneous and autonomous systems. Technical report, Sloan School of Management Working Paper 3848, 1995.
- [GMW99] R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the Lore data model and query language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB'99)*, pages 25–30. Philadelphia, 1999.
- [GPQ⁺97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom. The TSIMMIS approach to mediation: Data models and Languages. *Journal of Intelligent Information Systems*, 1997.
- [GTM99] R. Glushko, J. Tenenbaum, and B. Meltzer. An XML framework for Agent-based E-commerce. *Communications of the ACM*, 42(3):106–114, 1999.
- [Gru93] T.R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [GPM95] F. Gudermann, A. Puder, and S. Markwitz. Service trading using conceptual structures. In *3rd International Conference on Conceptual Structures*. Springer Verlag, 1995.
- [Ros98] Inc. RosettaNet. RosettaNet laptop technical specification. Final Draft for Member Vote, 1998.
- [Kel95] A.M. Keller. Smart catalogs and virtual catalogs, 1995.
- [LS99] O. Lassila and R.R. Swick. Resource Description Framework(RDF) Model and Syntax Specification, *W3C Recommendation*, 1999; Available at: "http://www.w3.org/TR/1999/REC-rdf-syntax-199902."
- [MGM98] A.G. Moukas R.H. Guttman and P. Maes. Agent-mediated electronic commerce: A survey. *Knowledge Engineering Review*, 13(2):147–159, 1998.
- [PHG⁺99] A. Preece, K. Hui, A. Gray, P. Marti, T. Bench-Capon, D. Jones, and Z. Cui. The KRAFT architecture for knowledge fusion and transformation. In *Research and Development in Intelligent Systems XVI (Proc. Expert Systems 99)*, pages 23–38, 1999.
- [Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, pages 38–49, 1992.