

Automated Reasoning on the Web

Slim Abdennadher¹, José Júlio Alves Alferes², Grigoris Antoniou³, Uwe Aßmann⁴, Rolf Backofen⁵, Cristina Baroglio⁶, Piero A. Bonatti⁷, François Bry⁸, Włodzimierz Drabent⁹, Norbert Eisinger⁸, Norbert E. Fuchs¹⁰, Tim Geisler¹¹, Nicola Henze¹², Jan Małuszyński⁴, Massimo Marchiori¹³, Alberto Martelli¹⁴, Sara Carro Martínez¹⁵, Wolfgang May¹⁶, Hans Jürgen Ohlbach⁸, Sebastian Schaffert⁸, Michael Schröder¹⁷, Klaus U. Schulz⁸, Uta Schwertel⁸, and Gerd Wagner¹⁸

¹ *German University in Cairo (Egypt)*, ² *New University of Lisbon (Portugal)*, ³ *Institute of Computer Science, Foundation for Research and Technology – Hellas (Greece)*, ⁴ *University of Linköping (Sweden)*, ⁵ *University of Jena (Germany)*, ⁶ *University of Turin (Italy)*, ⁷ *University of Naples (Italy)*, ⁸ *University of Munich (Germany)*, ⁹ *Polish Academy of Sciences (Poland)*, ¹⁰ *University of Zurich (Switzerland)*, ¹¹ *webXcerpt (Germany)*, ¹² *University of Hannover (Germany)*, ¹³ *University of Venice (Italy)*, ¹⁴ *University of Turin (Italy)*, ¹⁵ *Telefónica Investigación y Desarrollo (Spain)*, ¹⁶ *University of Göttingen (Germany)*, ¹⁷ *University of Dresden (Germany)*, ¹⁸ *University of Eindhoven (The Netherlands)*

Abstract

Automated reasoning is becoming an essential issue in many Web systems and applications, especially in emerging Semantic Web applications. This article first discusses reasons for this evolution. Then, it presents research issues currently investigated towards automated reasoning on the Web and it introduces into selected applications demonstrating the practical impact of the approach. Finally, it introduces a research endeavor called REVERSE (cf. <http://reverse.net>) recently launched by the authors of this article which is concerned with developing automated reasoning methods and tools for the Web as well as demonstrator applications.

“For the semantic web to function, computers must have access to [...] sets of inference rules that they can use to conduct automated reasoning.” Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web, Scientific American, May 2001

1 Introduction

Tim Berners-Lee’s seminal vision of the Web, i.e. a platform for exchanging between computers documents related to each other by hypertext links, is immensely successful not only in research or for office work but also for all kinds of other purposes. Information of any kind is usually better published on the Web than on any other medium because, on the Web, information systems can be accessed from everywhere at any time. This makes updates immediately visible everywhere and thus makes the Web an ideal platform for information systems. Furthermore, the Web is a very flexible framework for data modeling because its premier data modeling languages, HTML and XML, offer (possibly nested) record-like data structures, the so-called HTML or XML “documents”, without requiring compliance to any data schema. Furthermore, with XML tags can be freely chosen making self-explaining data items easy to specify. In addition, the Web does support the coexistence of very heterogeneous data models, which greatly contributes to its success. Thus, the Web is convenient for exchanging all kinds of data and it is intensively used for that purpose.

The Web is no longer limited to a human-centered use for down-loading and displaying texts, pictures, and other contents. Instead, automated data processing on the Web for purposes such as data exchange and eCommerce has become a central issue for Web systems and applications.

Nonetheless, today’s Web is mostly used in a highly interactive manner. A high human involvement is needed for properly selecting and combining Web data. This is, of course, the case as soon as the data of concern are included in or described by non-trivial texts. In such cases, (more or less advanced) forms of processing of (more or less natural) languages are indispensable. A high human involvement is also needed on the Web with non-textual data, e.g. Web documents similar to database records.

Even conceptually simple tasks on non-textual Web data often require a non-negligible amount of “human data processing”. Consider for example the planning of a train trip using Web-based railways timetables: a non-negligible amount of “human data processing” is needed e.g. for properly interpreting train categories (e.g. ICE is the German counterpart to TGV in France), fares (e.g. “BahnCard 25” in Germany and train colors in France), or even locations (where exactly is “Westbahnhof” in Vienna? Is Lille in France or in Belgium?) or temporal indications (Is a day a legal holiday in a country? Does “Fasching” in Vienna refer to the same period of time as “mardi gras” in Lille?). If a task requires to consult several Web-based information systems, as it is the case for planning a train trip from Vienna to Lille, then several (implicit or explicit) data schemas, naming conventions, and price systems have to be combined. All information needed is undoubtedly available on the Web but processing it properly requires a large amount of “human data processing”. Automation of such data processing turns out to be in most cases rather difficult, if at all possible.

The Semantic Web is a vision how such “human data processing”, i.e. such common forms of reasoning on the Web could be automated without referring to full-fledged natural language processing or to advanced reasoning methods. The Semantic Web vision is about contents published on the Web being labeled with “semantic annotations” and these annotations being used for an automated retrieval and composition of Web contents. E.g., in the previous example, “ICE” and “TGV” could be both annotated as “fast trains” thus making it possible to easily relate each notion to the other, socio-cultural notions such as “Fasching” and “mardi gras” could be related to each other as well as to certain days, and the train station “Westbahnhof” could be related to a location on Vienna’s city map.

Keeping with the tolerance to heterogeneity which has been an essential factor in the success of the traditional Web, the Semantic Web vision does not impose any particular form for semantic annotations. In fact, very different formalisms have already emerged like RDF (cf. <http://www.w3.org/RDF/>) – a variation on the Entity-Relationship database schema language –, OWL (cf. <http://www.w3.org/TR/owl-features/>), an ontology language stemming from description logics, and thesauri initially developed for information retrieval and/or natural language processing. The “semantic annotation languages” in use on the emerging Semantic Web provide extremely different ways of relating concepts with each other: hierarchies, graphs, etc.

Also specific methods for reasoning on these concepts and relations are currently being developed. Reasoning on the Web, however, cannot be confined to RDF, ontology reasoning, or reasoning on data of any specific kind. E.g. in the previous example, one not only needs to infer properties from train categories but also to draw conclusions from temporal and location data train timetables refer to. Automating finding a convenient train connection from Vienna to Lille surely involves inferring the time necessary for commuting in Paris between “Gare de l’Est” and “Gare du Nord”. This can rather easily be done using data available on the Web such as city maps – and we, human Web users, actually do it. Such forms of reasoning might well go beyond RDF or ontology reasoning. Thus, reducing the amount of “human data processing” on the Web calls for automated reasoning not only on semantic annotations (such as RDF or OWL data items), but also on Web contents of all kinds, i.e. primarily XML documents.

How should languages and systems for reasoning on the Web be conceived? This is an open question which is almost impossible to fully answer because of the lack of experience. However, hypotheses can already yet be made. Hypotheses sustaining the work of the authors of this article in their joint work towards languages and systems for reasoning on the Web are given below in Section 2. Since reasoning on the Web has to refer to Web contents, querying the Web and reasoning on the Web have to be brought together. This issue is addressed in Section 3. Most Web contents are not static but are regularly updated. In updating Web contents, reasoning on Web contents is generally involved as it is the case with information systems of all kinds. Deductive approaches to Web evolution and reactivity are addressed in Section 4. Information systems offering evolving data are almost always compelled to fulfill possibly complex and therefore mostly declarative high-level specifications referred to in the following as “policies”. This of course also holds of Web information systems. Declarative approaches to specifying and reasoning on “policies” are discussed in Section

5. The next four sections address three “ancillary” issues: typing, composition, using a “controlled language” in reasoning on the Web, and syntax. For programs of any kind, typing and composition are of great importance. These issues are especially interesting to investigate in the context of reasoning programs because they themselves involve reasoning methods. Typing and composition are addressed in Section 6 and Section 7, respectively. The promises of a controlled language on the Web are discussed in Section 8. Much of the Web’s success goes back to its data modeling languages HTML and XML, and to tools for these languages. It is surely not rash to assume that defining syntax and tools for Web rule (or reasoning) languages will play a similarly important role with the Semantic Web. These issues are discussed in Section 9. Applications demonstrating the impact that a new technology already has or is likely to soon have are essential in introducing a technology. Section 10 introduces to three classes of important applications of the Semantic Web: event, temporal, and geographical data; adaptive information systems; and the emerging Bioinformatics Semantic Web. This article is included in Section 11 with a presentation of the research project REVERSE (cf. <http://reverse.net>) recently launched by the authors of this article and other researchers.

2 Working Hypotheses

The following working hypotheses have been formulated by the authors of this article and fellow researchers engaged in a joint research endeavor towards languages and systems for reasoning on the Web. These working hypotheses might well not be necessarily true in all possible contexts. Nonetheless, they are convenient for sustaining the afore-mentioned research endeavor.

Working hypothesis 1: As the current Web is heterogeneous in data formats and semantics, the future Web will be heterogeneous in its reasoning forms. Nonetheless, a minimal collection of inter-operable Web reasoning languages is desirable.

Working hypothesis 2: The (heterogeneous) forms of reasoning relevant on the Web should refer to all layers of the so-called Semantic Web Tower (cf. <http://www.w3.org/RDF/Metalog/docs/sw-easy>), i.e. to XML data, meta-data such as XML document schemas, RDF data and schemas, to ontologies (e.g. DAML+OIL or OWL), vocabularies, and thesauri, etc. thus avoiding an explosion of similar yet different reasoning languages used for similar purposes in different contexts.

Working hypothesis 3: Reasoning programs should be first-class Web data that can be retrieved from the Web, rendered like standard Web documents, and processed (on the server and/or client sides). Also they are to be processable by reasoning programs, too (meta-level reasoning).

Working hypothesis 4: Web data, meta-data, and rules can be incorrect and/or incomplete in both their formats and their semantics and common-sense forms of reasoning are necessary for advanced Web applications.

3 Querying

A language for the (Semantic) Web bringing together querying and reasoning on the Web must build upon a notion of “elementary query”. A first issue in developing such a language is therefore the choice of a query paradigm and of an inference paradigm that fit well together. Let us consider the state-of-the-art Web query languages: XPath (cf. <http://www.w3.org/TR/xpath>), XQuery (cf. <http://www.w3.org/TR/xquery/>), and XSLT (cf. <http://www.w3.org/Style/XSL/>).

Using XPath, one can describe a navigation through (tree-shaped) HTML or XML documents using constructs inspired from regular expressions. These constructs give rise to express navigations through HTML and XML documents. E.g., the XPath expression

```
/desc::section[child::figure]/parent::chapter
```

retrieves from a document all elements labeled `chapter` that encompass elements labeled `section`, that themselves can occur at any depth and contain at least one subelement labeled `figure`. Such “navigational queries” might become rather complicated, especially when so-called “reverse axes”

such as `parent` are used. In contrast, XPath expressions involving only so called “forward axes” such as `descendant`, `child`, and `following-sibling` are much easier to understand for a programmer, and to evaluate for a language processor. Note that XPath expressions involving only forward axes are very close to term queries, or atoms, as used in logic programming, automated reasoning, and SQL [33, 2].

XQuery and XSLT build upon XPath in different although quite similar manners by providing programmers with language constructs for re-arranging data selected by using XPath into new documents. Such re-arrangements are commonly called “transformations”. XQuery and XSLT use patterns similar to the atoms of logic, logic programming, and automated reasoning for expressing how the data selected using XPath expressions are re-arranged into new data items.

“Elementary queries” of a (Semantic) Web query language fulfilling the following requirements would be easier to combine with an inference mechanism than queries à la XPath:

- **Term queries.** A query should have a term or atom-like structure. Such queries can be seen as aforms, answers to such queries as “form fillings”. They can be called “positional” in contrast to the “navigational” queries of XPath, XQuery, and XSLT.
- **Incompleteness.** Queries should allow for incomplete specifications of data searched for. Incomplete queries in both breadth, i.e. concerning siblings, and depth, i.e. concerning children, should be possible.
- **Referential transparency.** The meaning of every expression in a query, especially of a variable, should be the same wherever it appears. I.e., destructive assignments should be prohibited, variables should be like in functional or logic programming.
- **Multiple variable bindings.** Like in logic, logic programming, SQL, XPath, XQuery, and XSLT, queries with several answers should be possible, each answer binding the query variables differently.

A rather simple, although somehow “incomplete” inference à la Prolog seems to be appropriate for a (Semantic) Web query language. Indeed, such an inference both, is sufficient for many applications, and gives rise to implementing more sophisticated forms of reasoning. Arguably, inference for reasoning on the Web should support the following:

- **Views.** Inference à la Prolog is closely related to SQL views, a well established concept in databases and information systems.
- **Non-monotonic negation.** This kind of negation is best suited for implementing information systems.
- **Compositional semantics.** A (structurally) recursive definition of the semantics of a “query program” in terms of the semantics of its parts, i.e. a Tarski-style model theory, would provide with a rather natural declarative semantics.

A further issue in developing a language bringing together querying the Web and reasoning is whether a classical-style or object-oriented logic such as F-Logic [34, 35] and as used in the Web query language XPathLog [41] by the LoPiX system [40] should be retained for both queries and inference. Considering that XML proper has no notion of object and/or inheritance, it is tempting to conclude that a classical-style logic might well be more appropriate on the Web than an object-oriented logic. Note, however, that some XML-related formalisms such as XML Schema (cf. <http://www.w3.org/XML/Schema>) do have a weak notion of inheritance: According to XML Schema attribute and/or attribute values are inherited from ancestor elements and overriding is possible.

First investigations with a prototype query language called Xcerpt [16, 17, 11, 10] (cf. <http://xcerpt.org>) suggest that the requirements above can be nicely combined into a practical although formally defined language.

4 Evolution and Reactivity

Besides the realization (i.e. technologies and languages) and the use (i.e. querying) of the Semantic Web, its maintenance and evolution are important issues. The Semantic Web is a “living organism”, combining a multitude of autonomous data sources/knowledge repositories, each of which evolves in time and must be able to react to both external and internal events. In case that the data or behavior of one source changes, there may be various adaptations and reactions of other sources.

This dynamic character of the Semantic Web requires (declarative) languages and mechanisms for specifying its maintenance and evolution.

To cope with such a reactive behavior, and to be able to reason about it, Semantic Web resources must be equipped with some form of “reaction” (or “event-condition-action”) rules specifying in presence of which events, and under what conditions, which actions are to be triggered. In this context, events may either be local events, such as the occurrence of an update on data, as well as external events due to incoming messages and communication with other web resources, and the conditions should consist of queries as specified in the Section above. Actions may be requests for local updates on the data of the web resource, as well as remote actions (or transactions) to be communicated to other resources.

The existence of these actions brings us to another crucial issue for the study of evolution in the web: that of how to update web resources. These resources contain data facts described in some Web data modelling language, and mechanisms and languages for updating these facts are required. If data can be queried by a logic programming-like or SQL-like language, such as the language Xcerpt just mentioned (cf. <http://xcerpt.org>), it seems reasonable that updates to data are based in concepts developed in the context of logic programming and database querying too. In fact, a rule-based language for describing updates and transactions on data, such as Transaction Logic [13], together with mechanisms for specifying changes in semi-structured data, seem adequate as a starting point for an update counterpart of a query language with the characteristics of Xcerpt.

But updates to Web resources should not be confined to updates of its data facts. Resources in a Semantics Web with reasoning capabilities may have, besides data facts, knowledge derivation rules (be it simply views, or more elaborate rules such as those, referred in the following section, describing policies), as well as reaction rules of the kind just described. A language for specifying updates in the Web should take into consideration the addition, deletion and changes of these rules as well. Language constructs for changing rule-based knowledge similar to those found in updates of logic programs (e.g. [1, 25]) seem appropriate for the Semantic Web, too. When updating rules, be it on the Web or in any other knowledge base, inconsistencies between new and already present data are bound to appear, and mechanisms for dealing with such inconsistencies are required. Such mechanisms may be inspired from methods developed for updates of logical theories and for belief revision.

Besides the issue of coping with local updates of resources, in a highly distributed knowledge base such as the (Semantic) Web, propagation of updates between resources is also a crucial issue. While distributed databases have well-defined connections between participating sub-databases, the Web consists of connected autonomous resources with imprecisely defined connections. Modifications of resources belonging to the Semantic Web – especially, when Adaptive Web Systems are concerned – also require modifications and adaptations on related resources. The forms update propagation might take depend on the relationship and communication modes between the resources. Two update propagation strategies can be distinguished: a *push strategy* in case a source knows that other sources depend on its knowledge and a *pull strategy* in other cases. With the push strategy, whenever a site changes its data, it informs the sites that depend on that data. With the pull strategy, each site is responsible of regularly checking whether other sites it depends upon have been modified. These aspects of update propagation in the web require further investigation, where classical knowledge representation and management research must be combined with web-specific issues, such as those of searching and detecting useful knowledge and changes.

5 Policies

For a long time, logic programming and rule-based formalisms have been considered appealing policy specification languages. This is witnessed by a large body of literature (cf. [12]). In their simplest form, security policies are meant to pose constraints on a system’s behavior (e.g. *file F cannot be accessed by user U*), but they can also serve to specify more complex behaviors including decisions (e.g. *what should be asked from user U before granting access to service S?*) and explanations such as suggesting how to get permission to obtain the desired service). Complex behavior is essential in an open context such as the Semantic Web where the clients or users of a service are often occasional and do not know much about how to interact with the service. More recently, the notion of policy has been generalized to include other specifications of behavior and decisions, including “business rules” in all their forms (integrity, derivation, and reaction rules), that can be naturally represented through rule-oriented representation techniques [29]. In the emerging

area of service-oriented computing, “policy” is sometimes used to refer to the orchestration of elementary and compound services. In this broad sense, policies specify the interplay (dialogs, negotiations, etc.) between different entities and actors for the purpose of delivering services while enforcing some desired application constraints and client requirements.

Some of the advantages of representing policies by rules are as follows [24]: Writing rules is usually faster and cheaper than writing imperative or object-oriented code; rules are more concise and easier to understand, share and maintain, especially in a global open environment such as the web, where self-documenting specifications are one of the current approaches to enabling interoperability.

In a similar perspective, a single declarative (*semantic*) policy specification can be used in several ways, for example not only to enforce a security policy, but also to enable negotiations and explanations as mentioned above [12]. The connection to the Semantic Web vision is clear: a knowledge-based definition of a policy can be re-used in a variety of ways that need not be figured out in advance thereby achieving a level of flexibility such as those required by modern interoperability scenarios.

The above vision challenges the existing rule-based technologies, e.g. logic programming. In particular, a policy language specification should address the following issues:

- **Software interoperability.** Some existing systems (such as Hermes and Impact) adopt wrappers as a uniform method for the interaction of a rule base with arbitrary software packages. Wrappers are being replaced by more flexible, service oriented approaches based on standards such as WSDL (cf. <http://www.w3.org/TR/wsdl>), UDDI (cf. <http://www.uddi.org/>), WSCL (cf. <http://www.w3.org/TR/2002/NOTE-wscl10-20020314/>) that aim at reducing the effort required to integrate a new piece of software into a heterogeneous environment. There is a need for a better understanding of how such a shift affects syntax and semantics of policy languages.
- **Compatibility with existing standards.** Besides the service description standards mentioned above, there are rule standardization initiatives such as RuleML (cf. <http://www.ruleml.org/>). The design of a policy language may suggest extensions to such standards. Besides this, policies may refer to document bases and resources encoded in XML-based standards; to facilitate parsing and querying of such documents, a policy language should integrate suitable mechanisms as discussed above.
- **Towards a uniform policy model.** A policy model should accommodate the common aspects of constraint enforcement, decision making, and explanations in the various incarnations discussed above (security, business rules, etc.), maybe specifying a small set of predicates with standard intended meanings. This might involve the specification of a policy ontology. However, it is not yet clear whether the different notions of policies are uniform enough to allow for such a unifying model.
- **Transaction model.** Policies induce negotiations and other forms of interactions with users or clients. Such interactions are typically state-dependent. Suitable constructs are required for modeling such interactions. Such constructs must have suitable semantics and be appropriately implemented so as to capture transaction dynamics.
- **Heterogeneous representations.** This is the typical Semantic Web issue. It requires policy languages to be handled by ontology-aware engines and to make it possible to include meta-information in policy specifications (e.g. in order to handle appropriately the contents of those certificate fields whose encoding is not specified by the X.509 standard). Related technical issues range from policy translation to context-dependent reasoning and integrations of logic programming and description logics [30].
- **Policy interoperability.** Questions such as: “Do the policies governing two (sub)systems allow for the successful interaction of the two systems?”, “How long can negotiations last?”, etc., need suitable tools to be answered, especially because the open nature of the application scenarios make answering such questions in general a hard task.

6 Typing

In the context of Web applications, typing in a broad sense can be considered from three complementary viewpoints:

- In the sense of type systems giving rise to static type checking of programs, with a particular focus on the emerging rule languages of the Semantic Web.
- In the sense of database schemas as well as of XML DTD's (cf. <http://www.w3.org/TR/REC-xml>), XML Schemas (cf. <http://www.w3.org/XML/Schema>), or Relax NG [49].
- In the sense of ontologies or of a set of RDF specifications.

An interesting research goal is to seek for a notion of type unifying these three complementary views and to use it for type checking of the emerging Web rule languages. There are strong indications that the experience of logic programming will play an important role in the development of these rule languages. Therefore, it is promising to investigate whether and how the existing approaches to typing of logic programs are applicable to Web rules. For example, the descriptive approach to typing of logic programs (cf. [23] and the references therein) is usually based on regular tree grammars or, equivalently, tree automata [21]. DTDs and XML schemas are variations of tree grammars, too. The investigations along these lines should be focused on the declarative XML query and transformation languages developed for the Semantic Web. They will hopefully result in techniques for (static or dynamic) type checking, type inference, and type-based debugging of rule programs. The role of types for optimization of reasoning processes and for Web query optimization are promising research issues.

A further aspect of the research on types in the context of the Semantic Web is related to the issue of integration of the emerging rule layer of the Semantic Web with the RDF(S) (cf. <http://www.w3.org/RDF/>) layer and the ontology layer where the Web Ontology language OWL (cf. <http://www.w3.org/TR/owl-features/>) may be seen as the emerging standard. RDF(S) and OWL are designed for the description of application domains as hierarchies of classes. The Web rules of an application will refer to objects of the application domain. The classes/subclasses of objects defined by the relevant ontology can be used as types/subtypes for typing the rules. For the rules based on Horn clauses and ontologies based on Description Logics (such as OWL) this is related to the important question how to integrate Horn clauses with Description Logics [22, 39, 31]. For example, query answering in the system described in [22] includes dynamic type checking with respect to the structural description in the Description Logic \mathcal{AL} . An interesting question is whether static typing techniques would also be possible for types defined by ontologies. Adaptation for that purpose of the descriptive typing techniques for logic programs would be based on checking inclusion of classes specified in Description Logics.

7 Composition

On the future Semantic Web, applications will be very heterogeneous. They will not only be built from many different forms of components, Web services, and systems, but will also refer to many ontologies in different ontology languages. Essentially, an application will have to deal with the combination of a multitude of components, ontologies, and ontology languages. Mastering this heterogeneity is a major task for the Semantic Web technology and requires excellent reuse techniques, in particular, from component-based and composition-based software engineering.

A software technology of primary interest in this context is the so-called “Invasive Software Composition” [4], which unifies view-based component development, generic component templates, and aspect-oriented development. With this technology, views on software components, expanders of generic components on different abstraction levels, and aspect weavers can be realized easily. The key to success is that a software component is no longer treated as a “black box”, but is opened up as a “gray box” with a specific interface for composition. Then, composition operators can parameterize components, merge views on components, and weave aspects of components by transforming the program fragments that the composition interface exhibited. First experience shows that gray box technologies increase software component reuse, simplify software models, and increase interoperability of components.

Furthermore, if a meta-model for the component language is given, the gray box composition technology can be made *generic* in terms of the underlying component language [26]. This meta-model can be exchanged, and the composition operations will nevertheless work for the new language [3]. This implies that the gray box composition technology can be transferred to the languages of the Semantic Web, leading to component and composition frameworks for languages like XML, RDF (cf. <http://www.w3.org/RDF/>), Datalog, RuleML (cf. <http://www.ruleml.org>), or OWL (cf. <http://www.w3.org/TR/owl-features/>). These frameworks are the subject of research

described elsewhere in this article. These frameworks will facilitate composition of applications on the Semantic Web, arriving at a new level of re-usability for components and services [3].

8 Controlled English

“A truly semantic web is more likely to be based on natural language processing than on annotations in any artificial language.” John F. Sowa. CG Mailing List, 19 October 2003

Languages for the Semantic Web

- should be formal to be computer-processable, moreover they should be based on logic to allow automatic reasoning,
- should be readable even by people not familiar with formal notations.

At first sight, these requirements seem to exclude each other, however the development of Attempto Controlled English (ACE, cf. <http://www.ifi.unizh.ch/attempto/>) proves that they can be effectively reconciled.

ACE is a controlled subset of standard English that allows users to express technical texts, e.g. specifications, precisely, and in the terms of the respective application domain. ACE texts are computer-processable and can be unambiguously translated into first-order logic. ACE appears perfectly natural, but being a controlled subset of English it is in fact a formal language with the semantics of the underlying first-order logic representation. The Attempto system and Attempto Controlled English are intended for users who want to use formal methods, but may not be familiar with them. Thus the Attempto system has been designed in a way that allows users to work solely on the level of ACE without having to take recourse to its internal logic representation.

ACE uses an application-specific vocabulary and is defined by a small set of construction and interpretation rules that constrain its syntax and semantics. These rules are implemented by the Attempto Parsing Engine (APE) that translates a multi-sentential ACE text into a coherent logic representation.

Its logic underpinnings allow us to reason in ACE. To support automatic reasoning in ACE we have developed the Attempto Reasoner RACE (Reasoning in ACE) [27]. RACE proves that one ACE text is the logical consequence of another one, and gives a justification of the proof in ACE. Variations of the basic proof procedure permit query answering and consistency checking. With the help of auxiliary first-order axioms and evaluable Prolog predicates we can perform complex deductions on ACE texts containing plurals and numbers.

For the Semantic Web applications described in this article, both ACE and RACE will be extended, for instance

- ACE will be enhanced by language constructs to represent mathematical data structures and operations on these structures,
- RACE will be extended to deal with advanced queries, such as 'why', 'why not', 'what if', 'under which conditions does ... occur', and to provide explanations to answers to these queries.

Furthermore, the inter-operability of both ACE and RACE with languages and formalisms under development, as described elsewhere in this article, will be investigated.

ACE is optimally suited to express rules, e.g. rules defining business policies. To execute/simulate ACE rules we will provide both forward and backward reasoning environments that allow users to execute ACE rules in batch and in interactive mode.

More information on ACE and RACE as well as demo versions of the Attempto system can be found at <http://www.ifi.unizh.ch/attempto/>.

9 Rule Markup

A Web rule markup language has several purposes. It may serve as a lingua franca to exchange rules between different systems and tools. It may be used: to express derivation rules for enriching Web ontologies by adding definitions of derived concepts or for defining data access permissions; to describe and publish the reactive behavior of a system in the form of reaction rules; or to provide

a complete XML-based specification of a software agent. Further uses may arise in novel Web applications.

Given the great diversity of rule concepts and existing rule languages, it is vital for rules to play a role on the Web to include a large set of important concepts and languages in an integrative Web rule language framework. Such a framework will consist of several overlapping sublanguages that share a common meta-model. The development of this rule meta-model is a difficult conceptualization and integration problem.

Thus, an interesting and challenging research issue is to develop an integrative rule markup language framework for Web reasoning languages. Preliminary work on this problem suggests that there are a number of issues related to the characteristics of the Web, such as using uniform resource identifiers (URIs) for various purposes or integrating distributed remote knowledge sources, which are typically not considered in more traditional reasoning languages.

In trying to identify design principles for rule markup languages, it is surely promising to investigate rule modeling languages in combination with information modeling languages such as Object-Role Modeling (ORM) and the Unified Modeling Language (UML). A promising approach would be to follow the model-driven software engineering approach (MDA) proposed by the Object Management Group (OMG), where an initial semi-formal (“computation-independent”) business/domain model is first refined and transformed into a (“platform-independent”) logical design model, corresponding to an executable specification, and then into a (“platform-specific”) efficient implementation model.

For the purposes of the above-mentioned research goals, one needs to make certain extensions to the information modeling language(s) used in order to be able to account for the concrete types of rule considered. The resulting rule modeling language will hopefully help in unifying the syntax and markup of the Web reasoning languages under development, as described elsewhere in this article. It is expected that an iterative process based on mutual interactions between the developers of such reasoning languages and the promoters of unifying syntax and markup will be needed. Case studies, based on requirements analysis and domain modeling, will be important means to facilitate this process.

10 Selected Applications

10.1 Reasoning with Geotemporal Information, Geospatial Information, and Events

The seminal paper where John Alan Robinson introduced the resolution principle [47] gave the area of automated reasoning (AR) such a boost that many AR researchers were convinced that sooner or later the activity of proving theorems in mathematics could be automated. Quite soon, however, it became clear that a purely logical language with purely syntactical inference rules is too weak to do any serious mathematics. The next step was the introduction of *theory resolution* [48]. Theory resolution is a schema for combining purely logical reasoning with special algorithms dedicated to reasoning in special theories. The theories which up till now have been investigated with respect to theory resolution are basic mathematical theories: equality, orderings, semigroup theory, etc. This made automated reasoning more useful for mathematics, but it is still in a rather primitive state, compared to the abilities of human mathematicians.

The current situation of automated reasoning for Web applications can be compared to the situation of automated reasoning for mathematics after the invention of the resolution principle. The knowledge representation, inference and query mechanisms for Web applications incorporate very general concepts, logical operators, relations and functions in an ontology language like OWL (cf. <http://www.w3.org/TR/owl-features/>), the tree structure of XML documents in query languages, etc. Some systems, for example XML-schema (cf. <http://www.w3.org/XML/Schema>) have a notion of datatype, but these are not yet theories in the sense that one could reason with and on the structures defined by datatypes.

As in automated reasoning for mathematics, the next step is to introduce theory reasoning into Web reasoning. This requires two measures: (1) general mechanisms for incorporating theories into the inference procedures, and (2) special theories have to be adapted to the general theory reasoning mechanism. A general mechanism for incorporating theories into ontology languages which are based on description logics is the “concrete domain” approach of Baader and Hanschke [5]. A general mechanism of a similar kind still has to be developed for deductive Web query

languages (cf. Section 3), policy languages (cf. Section 5), and other deductive languages to be used on the Semantic Web.

A further issue is, of course, the choice and development of concrete theories to be incorporated into Web reasoning. They must be useful for practical and widespread applications as well as general enough. The theory of geotemporal reasoning and the theory of geospatial reasoning matches these requirements very well because locations and time play essential roles on today's Web and will most likely become even more important with the advent of mobile computing and Semantic Web applications.

With "geotemporal information" we mean any kind of temporal information that is based on some established human calendar system plus real-life temporal reference points such as events ("last Sunday", "begin of easter holidays", "world cup finals", "next elections", etc.) seasons ("early spring", "late summer" etc.) as well as epochs in history ("middle ages", "Roman empire", "second world war", etc.). The kind of temporal information to be addressed includes time points, intervals, durations and periodicities ("every second year"). We speak of "geo"temporal information since the focus is not on the temporal behavior of microsystems, algorithmic processes, etc.

With "geospatial information" we mean any kind of spatial information that is based on an established system of geographic coordinates and on real-life locations such as countries, cities, places, rivers, etc. The kind of spatial information that we address includes geographic positions, distances, directions, and relations such as neighborhood, inclusion, etc.

Temporal notions refer quite often to events. The analysis of the expression "after the Olympic games in Rome", for example, depends on a detailed modeling of events. Therefore, as a further research area, the classification and modeling of events and event types has to be investigated.

A basic "algebraic" time ontology which is geared to Web applications is currently being developed. Algebraic means that basic objects and operations must be identified. A basic object is for example a particular year x or a particular summer y . Operations might be "next year" or "next summer" or "the third week in summer y ", etc. Based on this, a specification language for specifying definable notions is under definition [18]. For example, "weekend" could be defined as the union of Saturday and Sunday (in a particular week), the seasons could be defined as periodic intervals starting at 21 March each year. The specification language includes the representation and processing of fuzzy notions like "around noon". Fuzzy notions can be included by using fuzzy intervals as temporal intervals. This has impacts on many components of the system. For example Allen's interval relations must be redefined to yield fuzzy values [45].

The area of geospatial reasoning is much wider than geotemporal reasoning. Therefore, a first step will be devoted to investigate which geographic notions are relevant for Web applications, how to classify them and how to relate them with mathematical concepts. For example, "London" can clearly be represented as a polygon, whereas "southern London" is a fuzzy notion which may need to be represented as a two-dimensional fuzzy set, obtained by imposing a "southern" fuzzy operator to the London polygon. "The next bus stop" could be represented as a function mapping bus stops (coordinates) to bus stops, etc. One needs to identify a core set of basic notions which can be represented mathematically and preferably with data structures from Geographic Information Systems (GIS) so as to use existing knowledge. Basic notions need to be distinguished from definable notions, for which a definition mechanism is needed.

In order to support the geotemporal and geospatial reasoning, a generalized thesaurus for events embedded in a hierarchy is currently being designed. The systematics should be compatible with other classification schemes such as the news subject scheme of the International Press Telecommunication Council IPTC (cf. <http://www.iptc.org/pages/index.php>).

10.2 Personalized Information Systems

The Semantic Web vision of a next generation Web, in which machines are enabled to understand the meaning of information in order to better inter-operate and better support humans in carrying out their tasks, is very appealing and fosters the imagination of smarter applications that can retrieve, process and present information in enhanced ways. In this framework, a particular attention should be devoted to "Adaptive Information Systems", i.e. information producing systems that can autonomously inter-operate – either with humans or with other systems –, tailoring their processing and its outcome to specific requests. The generic term "requests" is used here because one can actually think of different forms of adaptation, and of different approaches to achieve them. One form of adaptation, prominent in the literature of web-based applications, focuses on the personalization of *information presentation*, which is changed according to the user's age, culture, skills,

and past choices. The information about the user is stored in a *user model*, which in some cases is dynamically changed along time. User models are often used also for personalizing *reading sequences* in a hyperspace of information items; applications in this area draw considerably from the research field known as *Adaptive Hypermedia* [14].

Another example is given by *recommendation systems*, used to support users in solving tasks. In this case, rather than a user model, systems must encompass the ability of foreseeing the consequences of the proposed actions [9].

More recently, the studies on *Web Services* [19] raised the need of other kinds of adaptation, related to the *use* of a service. For instance, in some applications it is important to personalize the execution of a service to a user's specific requests or to properly choose a set of services that are to be combined so as to accomplish a more complex task.

Last but not least, adaptation and personalization are fundamental issues in the perspective of moving towards open systems that can use documents and other resources that are not restricted to belong to a local set of items but evolve according to an open and dynamic perspective of the Web.

In order to obtain adaptation it is necessary to represent information about resources: static information, such as a description with respect to some ontology, but also dynamic information, such as the description of an activity, in ways that can be reasoned about. Part of the requested information about application domains is "manifested" in the ontology layer of the so-called Semantic Web Tower (cf. <http://www.w3.org/RDF/Metalog/docs/sw-easy>). Arguably, the natural place for personalization in the Semantic Web Tower is the logic layer: reusable personalization functionality expressed via rules [32]. Personalization functionality will require to reason about domain knowledge (ontologies), about semantically enriched resource descriptions, and also about user models.

The first steps of the research outlined above will focus on the logical characterization of the adaptive functionality and will aim at defining a set of scenarios, mostly in the fields of eLearning and adaptive hypermedia systems, in which the role of personalization and adaptation is prominent. It is also planned to investigate the emerging area of Web Services where tasks such as personalized search based on querying and rule mechanisms could be accomplished.

10.3 The Bioinformatics Semantic Web

With recent technological advances, biology has changed dramatically to a data-driven science. Projects like the human genome project, which resulted in a publicly accessible database containing the whole human DNA, are only single examples of an explosion in biological information online accessible. There are hundreds of databases and bioinformatics tools online with ten thousands of 3D structures of proteins, with hundred thousands of protein sequences and with millions of literature abstracts [15]. The current bottleneck upon which future progress in biology depends is the coherent integration of all these public, online resources. Thus, bioinformatics is an ideal field for testing Semantic Web technologies for three reasons: First, Web-based systems and Web databases have been applied very early in bioinformatics [15], second the dramatic increase of data produced in the field calls for novel processing methods, third, the high heterogeneity of bioinformatics data require semantic-based integration methods.

Scenario. Consider the following scenario: a biologist obtains a novel DNA sequence nothing is known about. He or she wants to run an alignment, but has specific requirements for the alignment. These requirements are captured as rules and constraints, which are taken into account by the online accessible Semantic Web enabled sequence comparison service.

The researcher found a number of significantly similar sequences in yeast for which there is gene expression data available. The scientist requests from the Semantic Web enabled gene expression database and tool expression data for the relevant genes. He or she defines rules, which capture which expression profiles are interesting, e.g. all genes which are, as it is called, highly expressed at the beginning and end of the experiment are of interest.

The genes are part of a larger process and the researcher is interested in their gene products. A query to the protein database SWISSPROT determines these. Do these proteins interact with each other? To answer this question a Semantic Web service is queried, which computationally determines protein interactions. A user-defined rule formulating what constitutes a protein domain interaction, is applied on the fly to SCOP, the structural classification of proteins, and PDB, a large protein structure database. The rule-based sequence similarity tool mentioned above is used

to determine whether the scientist's proteins of interest are similar to any interacting proteins computed from SCOP and the PDB.

Finally, the scientist wishes to relate the protein interaction network to metabolic pathways.¹ As all the tools used refer to the same ontologies and terminology defined through the gene ontology, the researcher can easily investigate a mapping from the interaction network to a relevant metabolic pathway obtained from a Semantic Web enabled pathway server.

During the above information foraging, the scientist constantly used literature databases to read relevant articles. Despite the tremendous growth of 8000 articles a week, the biologist still manages to quickly find the relevant articles as he or she uses an ontology-based search facility, which guides the search, automatically specializing querying, where too many hits are obtained, and generalizing, where too few articles can be found.

Rules for systems integration and modeling. One approach to tackle the challenges in bioinformatics in general and the scenario above in particular, will rest upon the use of rules, reasoning and ontologies. Broadly, these can be applied in two contexts:

1. Systems integration: Rules for mediation, including consistency, and to formulate complex queries.
2. Modeling: Rules to model biological systems.

Currently, much interaction with online data sources is done manually through HTML pages [15]. However, many online Bioinformatics tools already provide XML output, so that more complex querying and interaction is possible. These could be simple queries to a single XML document retrieving some of its attributes (e.g. given an XML document representing a protein structure, query the protein document for a domain of that protein). The above structures could be complemented by similar sequences for which the structure has been predicted using rules and constraints.

Queries can be complex, involving different sources and ontologies. E.g. use a rule that specifies if a search for literature in the medical literature database PubMed retrieves too many (or too few) results, then find keywords in gene ontology and specialize (or generalize) them and re-issue the query to PubMed. As another example for complex queries, consider the Biomolecular interaction network (BIND), which already provides some output in XML format. The problem of defining interactions is very complex, and enhanced expressivity would be required to define the different classes of interactions. Interactions can then be deduced from several sources, such as PDB, metabolic/regulative pathways or networks, etc. Rules can be used again to model these networks and to query them.

The problems above have already begun to be addressed, in particular regarding transparent access to bioinformatics databases [38] and integration protein annotations [43, 44, 42]. In depth experience in merging bioinformatics ontologies [37], integration of an ontology with gene expression data [7, 8], and consistent annotation of proteins [43, 44, 42] have already been gathered. Relevant work concerning protein structure prediction with constraints [6, 36, 46] and concerning the ability to flexibly query data in the form of networks [20] and in the light of mismatches [28] between concepts have already been carried out.

11 REVERSE

The research issues described above are investigated in REVERSE, a research "Network of Excellence" of the 6th Framework Programme of the EU Commission. REVERSE stands for "REasoning on the WEb with Rules and SEMantics". Its objective is to strengthen Europe in the area of reasoning languages for Web systems and applications, especially Semantic Web systems and applications aiming at enriching the current Web with reasoning capabilities as described above. REVERSE will also develop university education and training as well as technology transfer and awareness activities so as to spread excellence within its research field in Europe. REVERSE involves 27 European research and industry organizations and about 100 computer science researchers and professionals. REVERSE is co-ordinated by François Bry of the University of Munich (co-ordinator), Jan Małuszyński of the University of Linköping (deputy co-ordinator), and Hans Jürgen Ohlbach of the University of Munich (deputy co-ordinator). Most authors of this article are members of REVERSE's steering committee. The EU Commission supports REVERSE with more

¹A metabolic pathway is an abstract representation of the exchange of substances in biochemical reactions and of the proteins and other molecules involved in the reaction.

than 5 Millions Euro over 4 years. REVERSE has started on 1st March 2004. Its activities can be followed at <http://reverse.net>.

Acknowledgement

This research has been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REVERSE number 506779 (cf. <http://reverse.net>).

References

- [1] J. J. Alferes, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. LUPS: A language for updating logic programs. *Artificial Intelligence*, 138(1–2):87–116, 2002.
- [2] ANSI. *Database Language SQL (Structured Query Language)*, 1999. ISO/IEC 9075:1999.
- [3] Uwe Aßmann. Composing Frameworks and Components for Families of Semantic Web Applications. In François Bry, Nicola Henze, and Jan Małuszyński, editors, *Proceedings of the International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 2901 in LNCS, pages 1–15. Springer-Verlag, 2003.
- [4] Uwe Aßmann. *Invasive Software Composition*. Springer-Verlag, February 2003.
- [5] Franz Baader and Philipp Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. In John Mylopoulos and Ray Reiter, editors, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 452–457. Morgan Kaufmann, 1991.
- [6] Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of Constraint Programming Techniques for Structure Prediction of Lattice Proteins with Extended Alphabets. *Journal of Bioinformatics*, 15(3):234–242, 1999.
- [7] Liviu Badea. Functional Discrimination of Gene Expression Patterns in Terms of the Gene Ontology. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2003.
- [8] Liviu Badea and Doina Tilivea. Integrating Biological Process Modelling With Gene Expression Data and Ontologies for Functional Genomics (Position Paper). In *Proceedings of the International Workshop on Computational Methods in Systems Biology*, University of Trento, 2003. Springer-Verlag.
- [9] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning About Self and Others: Communicating Agents in a Modal Action Logic. In C. Blundo and C. Laneve, editors, *Proceedings of the 8th Italian Conference on Theoretical Computer Science (ICTCS)*, volume 2841 of LNCS. Springer, 2003.
- [10] Sacha Berger, François Bry, and Sebastian Schaffert. A Visual Language for Web Querying and Reasoning. In François Bry, Nicola Henze, and Jan Małuszyński, editors, *Proceedings of the International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 2901 in LNCS, pages 99–112. Springer-Verlag, 2003.
- [11] Sacha Berger, François Bry, Sebastian Schaffert, and Christoph Wieser. Xcerpt and visXcerpt: From Pattern-Based to Visual Querying of XML and Semistructured Data. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, Berlin, Germany, 2003.
- [12] Piero A. Bonatti and Pierangela Samarati. *Logics for Emerging Applications of Databases*, chapter Logics for Authorization and Security, pages 277–323. Springer-Verlag, 2003.
- [13] A. J. Bonner and M. Kifer. Transaction Logic Programming. In D. S. Warren, editor, *Proceedings of the 10th International Conference on Logic Programming (ICLP)*, pages 257–279. MIT Press, 1993.
- [14] Peter Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.
- [15] François Bry and Peer Kröger. A Computational Biology Database Digest: Data, Data Analysis, and Data Management. *Distributed and Parallel Databases*, 13(1):7–42, 2002.
- [16] François Bry and Sebastian Schaffert. A Gentle Introduction into Xcerpt, a Rule-based Query and Transformation Language for XML. In *Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, June 2002. (Invited Paper).

- [17] François Bry and Sebastian Schaffert. Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In *Proceedings of the International Conference on Logic Programming (ICLP)*, LNCS 2401. Springer-Verlag, 2002.
- [18] François Bry, Bernhard Lorenz, Hans Jürgen Ohlbach, and Stephanie Spranger. On Reasoning on Time and Location on the Web. In François Bry, Nicola Henze, and Jan Małuszyński, editors, *Proceedings of the International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 2901 in LNCS, pages 69–83. Springer-Verlag, 2003.
- [19] J. Bryson, D. Martin, S. McIlraith, and L. A. Stein. Agent-Based Composite Services in DAML-S: The Behavior-Oriented Design of an Intelligent Semantic Web, 2002.
- [20] Nathalie Chabrier and François Fages. Symbolic Model Checking of Biochemical Networks. In *Proceedings of the 1st International Workshop on Computational Methods in Systems Biology (CMSB)*, LNCS, Riverto, Italy, March 2003. Springer-Verlag.
- [21] H. Common, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree Automata Techniques and Applications. <http://www.grappa.univ-lille3.fr/tata/>, 1999.
- [22] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *Intelligent Information Systems*, 10(3):227–252, 1998.
- [23] W. Drabent, J. Małuszyński, and P. Pietrzak. Using parametric set constraints for locating errors in CLP programs. *Theory and Practice of Logic Programming*, 2(4-5):549–610, 2002.
- [24] S. Staab (ed.). Where are the Rules? *IEEE Intelligent Systems*, 18(5):76–83, 2003.
- [25] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A Framework for Declarative Update Specifications in Logic Programs. In B. Nebel, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 649–654. Morgan Kaufmann, 2001.
- [26] Ira R. Forman and Scott H. Danforth. *Putting Metaclasses to Work: a New Dimension in Object-Oriented Programming*. Addison-Wesley Longman, Redwood City, CA, 1999.
- [27] Norbert E. Fuchs and Uta Schwertel. Reasoning in Attempto Controlled English. In François Bry, Nicola Henze, and Jan Małuszyński, editors, *Proceedings of the Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, number 2901 in LNCS, pages 174–188. Springer-Verlag, 2003.
- [28] David Gilbert and Michael Schroeder. FURY: Fuzzy Unification And Resolution Based on Edit Distance. In *Proceedings of BIBE2000 - IEEE International Symposium on Bio-Informatics and Biomedical Engineering*. IEEE Press, 2000.
- [29] Benjamin N. Grosf. Representing e-Commerce Rules Via Situated Courteous Logic Programs in RuleML. *Electronic Commerce Research and Applications*, 3(1), 2004.
- [30] Benjamin N. Grosf, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs With Description Logic. In *Proceedings of the World Wide Web Conference (WWW)*, pages 48–57, 2003.
- [31] B.N. Grosf, I. Horrocks, R.Volz, and S.Decker. Description Logic Programs: Combining Logic Programs with Description Logic. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 48–57. ACM Press, 2003. <http://www2003.org/cdrom/papers/refereed/p117/p117-grosf.html>.
- [32] Nicola Henze and Wolfgang Nejdl. Logically Characterizing Adaptive Educational Hypermedia Systems. In *Proceedings of the International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH)*, Budapest, Hungary, 2003.
- [33] C. J. Hirsch and J. L. Hirsch. *SQL: The Structured Query Language*. Tab Books, ; ACM CR 8812-0907, 1988.
- [34] Michael Kifer and Georg Lausen. F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Schema. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 134–146, 1989.
- [35] Michael Kifer, Georg Lausen, and James Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, July 1995.
- [36] L. Krippahl and P. Barahona. PSICO: Solving Protein Structures with Constraint Programming and Optimisation. *Constraints*, 7(3/4):317–331, July/October 2002.

- [37] P. Lambrix and A. Edberg. Evaluation of Ontology Merging Tools in Bioinformatics. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 589–600, 2003.
- [38] P. Lambrix and V. Jakoniene. Towards Transparent Access to Multiple Biological Databanks. In *Proceedings of the 1st Asia-Pacific Bioinformatics Conference*, pages 53–60, Adelaide, Australia, 2003.
- [39] A. Levy and M-C. Rousset. CARIN: A Representation Language Combining Horn rules and Description Logics. *Artificial Intelligence*, 104(1-2):165–209, 1998.
- [40] Wolfgang May. LoPiX: A System for XML Data Integration and Manipulation. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2001.
- [41] Wolfgang May. XPath-Logic and XPathLog: A Logic-Programming Style XML Data Manipulation Language. *Theory and Practice of Logic Programming*, 4(3):239–287, 2004.
- [42] S. Möller, E. V. Kriventseva, and R. Apweiler. A Collection of Well Characterised Integral Membrane Proteins. *Bioinformatics*, 16(12):1159–1160, 2000.
- [43] S. Möller, U. Leser, W. Fleischmann, and R. Apweiler. EDITtoTrEMBL: a Distributed Approach to High-Quality Automated Protein Sequence Annotation. *Bioinformatics*, 15(3):219–227, 1999.
- [44] S. Möller, M. Schroeder, and R. Apweiler. Conflict-Resolution for the Automated Annotation of Transmembrane Proteins. *Comput. Chem.*, 26(1):41–46, 2001.
- [45] Hans Jürgen Ohlbach. Fuzzy Time Intervals and Relations – The FuTIRe Library. Technical Report PMS-2004-4, Institute for Informatics, University of Munich, 2004. <http://www.pms.informatik.uni-muenchen.de/mitarbeiter/ohlbach/systems/FuTIRe>.
- [46] P.N. Palma, L. Krippahl, J.E. Wampler, and J.J.G. Moura. BiGGER: A New (Soft) docking Algorithm for Predicting Protein Interactions. *Proteins: Structure, Function, and Genetics*, 39:372–384, 2000.
- [47] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.
- [48] Mark E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, 1(4):333–356, 1985.
- [49] Eric van der Vlist. *Relax NG*. O’Reilly, 2003.