A3-D4

Personalization for the Semantic Web II

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	IST506779/Hannover/A3-D4/D/PU/b1
Responsible editors:	Nicola Henze
Reviewers:	Matteo Baldoni, Lilia Cheniti-Belcadhi
Contributing participants:	Hannover, Heraklion, Goettingen, Munich, Vienna, Turin, Warsaw
Contributing workpackages:	A3
Contractual date of deliverable:	31 August 2005
Actual submission date:	29 August 2005

Abstract
This report provides an overview of the achievements of working group A3 for bringing personalization functionality to the Semantic Web. It continues the work started in the deliverable A3-D1 which gave an overview on personalization know-how of partners in A3. In the deliverable at hand, we report on our achievements on reasoning for personalization in the Semantic Web (see Section 2), and particular personalization methods (see Section 3). Prototypes and applications will be discussed in deliverable A3-D6 (month 24).

Keyword List
semantic web, reasoning, personalization

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

Personalization for the Semantic Web II

Grigoris Antoniou[1], **Matteo Baldoni**[2], **Cristina Baroglio**[2], **Robert Baumgartner**[3], **François Bry**[4], **Nicola Henze**[5], **Wolfgang May**[6], **Viviana Patti**[2], **Slawomir T. Wierzchon**[7]

[1] Institute of Computer Science – FORTH
Heraklion Crete, Greece
`antoniou@ics.forth.gr`
[2] Dipartimento di Informatica — Università degli Studi di Torino
C.so Svizzera, 185 — I-10149 Torino (Italy)
{`baldoni,baroglio,patti`}`@di.unito.it`
[3] Institut für Informationssysteme
Technische Universität Wien, Austria
`baumgart@dbai.tuwien.ac.at`
[4] Research Center L3S &
ISI- Knowledge-Based Systems, University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
`henze@l3s.de`
[5] Institut für Informatik
Ludiwg-Maximilians-Universität München, Germany
`francois.bry@ifi.lmu.de`
[6] Institut für Informatik
Universität Göttingen, Germany
`may@informatik.uni-goettingen.de`
[7] Institute of Computer Science – Polish Academy of Sciences
Ordona 21, 01-237 Warsaw, Poland
`stw@ipipan.waw.pl`

29 August 2005

Abstract
This report provides an overview of the achievements of working group A3 for bringing personalization functionality to the Semantic Web. It continues the work started in the deliverable A3-D1 which gave an overview on personalization know-how of partners in A3. In the deliverable at hand, we report on our achievements on reasoning for personalization in the Semantic Web (see Section 2), and particular personalization methods (see Section 3). Prototypes and applications will be discussed in deliverable A3-D6 (month 24).

Keyword List
semantic web, reasoning, personalization

Contents

1 Executive Summary **1**

2 Reasoning Methods for Personalization in the Semantic Web **1**
2.1 Overview on “Reasoning Methods for Personalization in the Semantic Web” . . . 2
2.2 Personalization: From the World Wide Web to the Semantic Web 2

3 Personalization Techniques **3**
3.1 Planning and Reasoning about Actions 3
3.2 Adaptive Hypermedia Approach to Personalization 5
3.3 Rule-based User Modeling: Application to e-Learning domain 6

4 Conclusion **6**

5 Acknowledgment **7**

6 Appendix **9**

1 Executive Summary

The work in working group A3 is centered around three axes: In the first axis, we research foundations for personalization and adaptation in the Semantic Web, and in particular aim at logical frameworks for describing and characterizing appropriate personalization functionality. This axis is therefore called **Adaptive Functionality**. The second axis is on deploying personalization functionality in systems and prototypes – the **Testbeds**-axis. In the third axis, we develop a personalized information system for the Semantic Web: a **personalized Web portal** for the REWERSE network.

This report belongs to the Adaptive Functionality axis and provides an overview of the achievements of working group A3 on researching personalization functionality for the Semantic Web. The good integration between the researchers of group A3 has been successfully continued and is also visible by observing the authors of papers attached to this deliverable.

The present report continues the work started in deliverable A3-D1, in which we gave an overview on personalization know-how of partners in A3, and in REWERSE. In this deliverable, we particularly report about:

1. **Reasoning Methods for Personalization in the Semantic Web:**
   (a) how personalization based on reasoning can be realized in the Semantic Web
   (b) how personalization techniques for the World Wide Web and the Semantic Web can be characterized
2. **Techniques for Personalization:**
   (a) how personalization can be realized by employing **planning and reasoning about actions**
   (b) how personalization based on **adaptive hypermedia** can be realized and generalized to achieve re-usable, encapsulated personalization functionality for the Semantic Web
   (c) how rule-based user modeling for the Semantic Web can be achieved

2 Reasoning Methods for Personalization in the Semantic Web

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 vision, a particular attention should be devoted to *personalization*: By bringing the user's needs into the center of interaction processes, personalized Web systems overcome the one-size-fits-all paradigm and provide individually optimized access to Web data and information.

2.1 Overview on "Reasoning Methods for Personalization in the Semantic Web"

Members of the working groups I1, I2, I4, I5, and A3 have worked together to discuss recent approaches for using rules and rule-languages in the logic layer of the Semantic Web, and for supporting approaches to personalization in the Semantic Web. Special attention has been given to the important aspects of evolution, updates and events, and their consequences for personalization and reasoning. Approaches to personalization via reasoning about actions has exemplified for different scenarios, and Web Service-based architectures for personalization have been discussed.

More details can be found in

- *Grigoris Antoniou, Matteo Baldoni, Cristina Baroglio, Robert Baumgartner, Francois Bry, Thomas Eiter, Nicola Henze, Marcus Herzog, Wolfgang May, Viviana Patti, Sebastian Schaffert, Roman Schindlauer, Hans Tompits: Reasoning Methods for Personalization on the Semantic Web. Annals of Mathematics, Computing & Teleinformatics (AMCT), Vol.2, No., 1, pp 1-24.*
- *M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. Artificial Intelligence Review, 22(1):3-39, 2004.*

See Appendix.

2.2 Personalization: From the World Wide Web to the Semantic Web

Approaches to personalization have been investigated since the early days of the World Wide Web. Two major "schools of thoughts" can be distinguished in the WWW-case: Approaches for personalization which work on the so-called Web graph (these are the Web mining-based approaches to personalization), and those approaches which work on reasonably small but well-defined subgraphs of the Web (these are adaptive hypermedia approaches to personalization). In order to establish personalization on the Semantic Web, it is essential to study personalization techniques on the World Wide Web, learn from research findings and use the possibilities of machine-processable semantic descriptions of Web resources provided in the Semantic Web (including languages as well as architectures and other technologies) to overcome problems and gaps reported in the WWW-case. The investigations of members of working group A3 resulted in an article which contains a thorough analysis of personalization techniques in the World Wide Web, discusses approaches for generalizing these techniques to the Semantic Web, and outlines alternative approaches for the Semantic Web-case.

More details can be found in Matteo Baldoni, Cristina Baroglio, Nicola Henze: Personalization for the Semantic Web. Reasoning Web, First International Summer School, LNCS Tutorials vol. 3564, pp. 183-211, Springer, 2005. See Appendix.

3 Personalization Techniques

3.1 Planning and Reasoning about Actions

Another line of research has investigated the use of reasoning techniques in order to obtain forms of personalization. The study has involved two application areas: that of educational systems and the emerging area of web services. In both cases, the idea that we explored is to base adaptation on the *reasoning capabilities* of a *rational agent*, built by means of a declarative language.

For what concerns the former application domain, the focus was put on the possible uses of three different reasoning techniques, namely *planning*, *temporal projection*, and *temporal explanation*, which have been developed for allowing software agents to build action plans and to verify whether some properties of interest hold after the application of given sequences of actions. In both cases actions are –usually– not executed in the real world but their execution is simulated "in the mind" of the system, which has to foresee their effects (or their enabling causes) in order to build solutions. A group of agents, called *reasoners*, works on a dynamic domain description, where the basic actions that can be executed are of the kind "attend course X" and where also complex professional expertise can be described. Effects and conditions of actions (courses) are essentially given in terms of a set of abstract *competences*, which are connected by causal relationships. The set of all the possible competences and of their relations defines an *ontology*. This multi-level description of the domain bears along many advantages. On the one hand, the high modularity that this approach to knowledge description manifests allows course descriptions as well as expertise descriptions to be added, deleted or modified, without affecting the system behavior. On the other hand, working at the level of competences is close to human intuition and enables the application of both goal-directed reasoning processes and explanation mechanisms.

The reasoning process that supports the *definition* of a *study plan*, aimed at reaching a certain learning goal, either computes over the effects of attending courses (given in terms of competence acquisition, credit gaining, and the like) or over those conditions that make the attendance of a course reasonable from the educator point of view. The logic approach also enables the *validation* of student-given study plans with respect to some learning goal of interest to the student himself.

The same mechanisms can be used for composing, in an automatic and goal-driven way, learning objects that are represented according to the SCORM standard of representation. This kind of descriptions, in fact, account also for a semantic annotation in terms of preconditions and learning objectives, that allows the interpretation of learning objects as actions.

More details can be found in:

- *M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. Artificial Intelligence Review, 22(1):3-39, 2004.*

- *M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting SCORM courseware. In L. Aroyo and C. Tasso, editors, AH 2004: Workshop Proceedings, Part I, International Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web, pages 4-13, Eindhoven, The Netherlands, August 2004. Technische Universiteit Eindhoven.*

Reasoning techniques can also be used for automatically retrieving and composing web services in order to accomplish (complex) tasks of interest, respecting the constraints and the preferences of the user. In this case, the reasoning process works on a declarative description of the communication policies that are followed by the services, which are to be included in their documentation. The approach that was proposed is set in the Semantic Web field of research and inherits from research in the field of multi-agent systems. By taking the abstraction of web services as software agents, that communicate by following predefined, public and sharable interaction protocols, the possible benefits, provided by a declarative description of their communicative behavior, in terms of *personalization* of the service selection and composition have been studied. The approach models the interaction protocols provided by web services by a set of logic clauses representing policies, thus at high (not at network) level. A description by policies is definitely richer than the usual service profile description consisting in the input and output, precondition and effect properties usually taken into account for the matchmaking. Moreover having a logic specification of the protocol, it is possible to reason about the effects of engaging specific conversations, and, on this basis, to perform many tasks in an automatic way; in particular, selection and composition. Actually, the approach that has been proposed can be considered as a *second step* in the matchmaking process, which narrows a set of already selected services and performs a *customization* of the interaction with them. Indeed, the problem that this proposal faces can intuitively be described as looking for a an answer to the question "Is it possible to make a deal with this service respecting the user's goals?". Given a representation of the service in terms of logic-based interaction policies and a representation of the customer needs as abstract goals, expressed by a logic formula, logic programming reasoning techniques are used for understanding if the constraints of the customer fit in with the policy of the service.

More specifically, we have presented an approach to web service selection and composition that is based on reasoning about conversation protocols, within the framework of an agent language, DyLOG, based on a modal logic of action and beliefs. The approach extends with communication the proposal to model rational agents in [1]. Since the interested was on reasoning about the local mental state's dynamics, this approach differs from other logic-based approaches to communication in multi-agent systems, as the one taken in [12], where communicative actions affect the global state of a system, Actually, the target of these latter approaches is to prove global properties of the overall multi-agent system execution. The focus on the internal specification of interaction protocols for planning dialogue's moves is closer to [11], where negotiation protocols, expressed by sets of dialogue constraints, are included in the agent program and used for triggering dialogues that achieve goals. However such an approach does not support plan extraction and it cannot exploit information about the others, that instead is supplied by nested beliefs.

More details can be found in

- *M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about interaction protocols for web service composition. In M. Bravetti and G. Zavattaro, editors, Proc. of 1st Int. Workshop on Web Services and Formal Methods, WS-FM 2004, volume 105 of Electronic Notes in Theoretical Computer Science, pages 21-36. Elsevier Science Direct, 2004.*

- *M. Baldoni, C. Baroglio, L. Giordano, A. Martelli, and V. Patti. Reasoning about communicating agents in the semantic web. In F. Bry, H. Henze, and J. Maluszynski, editors, Proc. of the 1st International Workshop on Principle and Practice of Semantic Web Reasoning, PPSWR 2003, volume 2901 of LNCS, pages 84-98, Mumbai, India, December 2003. Springer.*

3.2 Adaptive Hypermedia Approach to Personalization

In the area of adaptive hypermedia, research has been carried out to understand how personalization and adaptation strategies can be successfully applied in hypertext systems and hypertext like environments. It has been stated that in the area of adaptive hypermedia and of adaptive web–based systems, the focus of developed systems has been so far on *closed world* settings. This means that these systems work on a fixed set of resources which are normally known to the system designers at design time (see the discussion on closed corpus adaptive hypermedia [3]). This observation also relates to the fact that the issue of authoring adaptive hypermedia systems is still one of the most important research questions in this area, see e. g. [2]. A generalization of adaptive hypermedia to an *Adaptive Web* [4] depends therefore on a solution of the closed corpus problem in adaptive hypermedia. Within the Personal Reader project (www.personal-reader.de), we propose an architecture for applying *some* of the techniques developed in adaptive hypermedia to an open corpus. A modular framework of components / services - for providing the user interface, for mediating between user requests and available personalization services, for user modeling, for providing personal recommendations and context information, et cetera, is the core of the Personal Reader framework [10].

More details can be found in

- *Stefan Decker, Michael Sintek, Andreas Billig, Nicola Henze, Peter Dolog, Wolfgang Nejdl, Andreas Harth, Andreas Leicher, Susanne Busse, Jörn Guy Süß, Zoltan Mikols, Jose-Luis Ambite, Matthew Weathers, Gustaf Neumann, Uwe Zdun: Triple - and RDF Rule Language with Context and Use Cases. W3C Workshop on Rule Languages for Interoperability, 27-28. April 2005, Washington, D.C., USA.*
- *Fabian Abel, Robert Baumgartner, Adrian Brooks, Christian Enzi, Georg Gottlob, Nicola Henze, Marcus Herzog, Matthias Kriesell, Wolfgang Nejdl, Kai Tomaschewski: The Personal Publication Reader. Semantic Web Challenge, 4th International Semantic Web Conference, November 6-10 2005, Galway, Ireland.*
- *Nicola Henze: Personal Readers: Personalized Learning Object Readers for the Semantic Web. 12th International Conference on Artificial Intelligence in Education, AIED'05, 18-22 July 2005, Amsterdam, The Netherlands.*
- *Robert Baumgartner, Nicola Henze, and Marcus Herzog: The Personal Publication Reader: Illustrating Web Data Extraction, Personalization and Reasoning for the Semantic Web. European Semantic Web Conference ESWC 2005, Heraklion, Greece, May 29 - June 1 2005.*
- *Robert Baumgartner, Christian Enzi, Nicola Henze, Marc Herrlich, Marcus Herzog, Matthias Kriesell, and Kai Tomaschewski: Semantic Web enabled Information Systems: Personalized Views on Web Data. International Ubiquitous Web Systems and Intelligence Workshop (UWSI 2005), Co-located with ICCSA 2005, Suntec Singapore, 9-12 May 2005.*
- *Nicola Henze: A Definition of Adaptive Educational Hypermedia Systems. Article in the forthcoming Encyclopedia of Multimedia by Springer.*

3.3 Rule-based User Modeling: Application to e-Learning domain

User modeling is concerned with gathering, discovering, deducing, and providing knowledge about users to supply user centered adaptation components with information needed in adaptation decisions. In our work, we have studied approaches to rule-based user modeling particularly in the e-learning domain.

In [9], we have proposed an approach for dynamically generating personalized hypertext relations powered by reasoning mechanisms over distributed RDF annotations. We have shown an example set of reasoning rules that decide for personalized relations to example pages given some page. Several ontologies have been used which correspond to the components of an adaptive hypermedia system: a domain ontology (describing the document space, the relations of documents, and concepts covered in the domain of this document space), a user ontology (describing learner characteristics), and an observation ontology (modeling different possible interactions of a user with the hypertext). For generating hypertext structures, a presentation ontology has been introduced.

Based on this general approach for characterizing e-Learning with ontological descriptions and reasoning approaches, we have concentrated on approaches to rule-based learning modeling [9, 8], and rule-based methods for learner assessment [5, 6, 7].

More details can be found in

- *Nicola Henze, Peter Dolog, and Wolfgang Nejdl: Reasoning and Ontologies for Personalized E-Learning. Educational Technology & Society, 2004, Vol. 7, Issue 4.*
- *Lilia Cheniti-Belcadhi, Rafik Braham, Nicola Henze, and Wolfgang Nejdl: A Generic Framework for Assessment in Adaptive Educational Hypermedia. Proceedings of the IADIS WWW / Internet 2004 Conference, October 2004, Madrid, Spain.*
- *Peter Dolog and Michael Schäfer. Learner modeling on the semantic web. In Proc. of PerSWeb 2005 Workshop: Workshop on Personalization on the Semantic Web: 10th International Conference, UM 2005, Edinburgh, Scotland, UK, July 2005.*
- *Lilia Cheniti-Belcadhi, Nicola Henze, Rafik Braham: Towards a Service Based Architecture for Assessment. Proceedings of the Thirteenth GI-Workshop on Adaptation and User Modeling in interactive Systems (ABIS 05), October 2005, Saarbrücken, Germany.*
- *Lilia Cheniti-Belcadhi, Nicola Henze, Rafik Braham: An Assessment Framework for eLearning in the Semantic Web. Proceedings of the Twelfth GI- Workshop on Adaptation and User Modeling in interactive Systems (ABIS 04), October 2004, Berlin, Germany.*

4 Conclusion

This report summarizes the achievements of working group A3 on transforming and applying personalization techniques for the Semantic Web. Working group A3 has investigated on reasoning methods for personalization in the Semantic Web (together with working groups I1, I2, I4, I5), and provided an in-depths analysis on personalization techniques in the World Wide Web and the Semantic Web.

Additionally, research on specific personalization techniques has yield to insights on how to transform existing personalization techniques to the Semantic Web, or to adopt new techniques for the Semantic Web. In particular, A3 has been focusing on personalization techniques based on reasoning about actions, transforming and adopting adaptive hypermedia techniques, and rule-based user/learner modeling.

5 Acknowledgment

We would like to thank Lilia Cheniti-Belcadhi and Peter Dolog for their discussions and contributions to learner modeling.
This research has been co-funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Program project REWERSE number 506779 (cf. `http://rewerse.net`).

References

[1] Baldoni, M., Giordano, L., Martelli, A., and Patti, V. Programming Rational Agents in a Modal Action Logic. *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation 8*, 5 (2004), 597–635.

[2] Bra, P. D., Aerts, A., Smits, D., and Stash, N. AHA! Version 2.0: More Adaptation Flexibility for Authors. In *Proceedings of the AACE ELearn'2002 conference* (Oct. 2002).

[3] Brusilovsky, P. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction 11* (2001), 87–110.

[4] Brusilovsky, P., and Maybury, M. *The Adaptive Web.* Communications of the ACM, 2002.

[5] Cheniti-Belcadhi, L., Henze, N., and Braham, R. An Assessment Framework for E-Learning in the Semantic Web. In *Proceedings of the Twelfth GI- Workshop on Adaptation and User Modeling in interactive Systems (ABIS 04)* (Berlin, Germany, October 2004).

[6] Cheniti-Belcadhi, L., Henze, N., and Braham, R. A Framework for dynamic Assessment for Adaptive Content Presentation in Educational Hypermedia. In *Proceedings of the IADIS WWW / Internet 2004 Conference* (Madrid, Spain, October 2004).

[7] Cheniti-Belcadhi, L., Henze, N., and Braham, R. Towards a service based architecture for assessment. In *Proceedings of the Thirteenth GI- Workshop on Adaptation and User Modeling in interactive Systems (ABIS 05)* (Saarbrücken, Germany, October 2005).

[8] Dolog, P., and Schäfer, M. Learner modeling on the semantic web. In *nternational Workshop on Personalization on the Semantic Web PersWeb'05* (Edinburgh, UK, July 2005).

[9] Henze, N., Dolog, P., and Nejdl, W. Reasoning and Ontologies for Personalized e-Learning. *Educational Technology & Society 7*, 4 (2004).

[10] Henze, N., and Kriesell, M. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *1st International Workshop on Engineering the Adaptive Web (EAW 2004)* (Eindhoven, The Netherlands, 2004).

[11] Sadri, F., Toni, F., and Torroni, P. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences. In *Proc. of ATAL'01* (Seattle, WA, 2001).

[12] Shapiro, S., Lespance, Y., and Levesque, H. J. Specifying communicative multi-agent systems. In *Agents and Multi-Agent Systems - Formalisms, Methodologies, and Applications* (1998), vol. 1441 of *LNAI*, Springer-Verlag, pp. 1–14.

6 Appendix

The appendix includes two overview papers on personalization for the Semantic Web which have been published in Journals or are included into books:

- Grigoris Antoniou, Matteo Baldoni, Cristina Baroglio, Robert Baumgartner, Francois Bry, Thomas Eiter, Nicola Henze, Marcus Herzog, Wolfgang May, Viviana Patti, Sebastian Schaffert, Roman Schindlauer, Hans Tompits: Reasoning Methods for Personalization on the Semantic Web. Annals of Mathematics, Computing & Telefinformatics (AMCT), Vol.2, No., 1, pp 1-24.
- Matteo Baldoni, Cristina Baroglio, Nicola Henze: Personalization for the Semantic Web. Reasoning Web, Springer LNCS, 2005.

Further, it contains one selected paper for each personalization technique described in section 3:

Planning and Reasoning about Actions: M. Baldoni, C. Baroglio, and V. Patti: "Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions." Artificial Intelligence Review, 22(1):3-39, 2004.

Immune-based Recommendation Technique: Slawomir Wierzchon: "Artificial immune system approach to adaptation" To be published as a book chapter by the Institute of System Research of Polish Academy of Sciences.

Adaptive Hypermedia Approach to Personalization: Fabian Abel, Robert Baumgartner, Adrian Brooks, Christian Enzi, Georg Gottlob, Nicola Henze, Marcus Herzog, Matthias Kriesell, Wolfgang Nejdl, Kai Tomaschewski: The Personal Publication Reader. Semantic Web Challenge, 4th International Semantic Web Conference, November 6-10 2005, Galway, Ireland. *Rated among the top-5 submissions for the Semantic Web Challenge*

Rule-based User Modeling: Nicola Henze, Peter Dolog, and Wolfgang Nejdl: Reasoning and Ontologies for Personalized E-Learning. Educational Technology & Society, 2004, Vol. 7, Issue 4.

Reasoning Methods for Personalization on the Semantic Web

Grigoris Antoniou
Institute of Computer Science – FORTH
Heraklion Crete, Greece
antoniou@ics.forth.gr

Matteo Baldoni, Cristina Baroglio, Viviana Patti
Dipartimento di Informatica
Università degli Studi di Torino, Torino, Italy
{baldoni,baroglio,patti }@di.unito.it

Robert Baumgartner, Thomas Eiter, Marcus Herzog, Roman Schindlauer, Hans Tompits
Institut für Informationssysteme
Technische Universität Wien, Austria
{baumgart,herzog}@dbai.tuwien.ac.at, {eiter,roman,tompits}@kr.tuwien.ac.at

François Bry, Sebastian Schaffert
Institut für Informatik
Ludiwg-Maximilians-Universität München, Germany
{francois.bry, sebastian.schaffert}@ifi.lmu.de

Nicola Henze
Information Systems Institute – Semantic Web Group,
University of Hannover, Germany
henze@kbs.uni-hannover.de

Wolfgang May
Institut für Informatik
Universität Göttingen, Germany
may@informatik.uni-goettingen.de

***Abstract*— The Semantic Web vision of a next generation Web, in which machines are enabled to understand the meaning of information in order to better interoperate 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 vision, a particular attention should be devoted to *personalization*: By bringing the user's needs into the center of interaction processes, personalized Web systems overcome the one-size-fits-all paradigm and provide individually optimized access to Web data and information. In this paper, we provide an overview of recent trends for establishing personalization on the Semantic Web: Based on a discussion on reasoning with rule- and query languages for the Semantic Web, we outline an architecture for service-based personalization, and show results in personalizing Web applications.**

***Index Terms*— semantic web, personalization, reasoning for the semantic web, rule languages, query languages, web data extraction**

I. Introduction

The aim of the Semantic Web initiative [27] is to advance the state of the current Web through the use of semantics. More specifically, it proposes to use *semantic annotations* to describe the meaning of certain parts of Web information. For example, the Web site of a hotel could be suitably annotated to distinguish between hotel name, location, category, number of rooms, available services etc. Such meta-data could facilitate the automated processing of the information on the Web site, thus making it accessible to machines and not primarily to human users, as it is the case today.

However, the question arises as to how the semantic annotations of different Web sites can be combined, if everyone uses terminologies of their own. The solution lies in the organization of vocabularies in so-called *ontologies*. References to such shared vocabularies allow

interoperability between different Web resources and applications. For example, a geographic ontology could be used to determine that Crete is a Greek island and Heraklion a city on Crete. Such information would be crucial to establish a connection between a requester looking for accommodation on a Greek island, and a hotel advertisement specifying Heraklion as the hotel location.

At the writing time of this paper, there are recommendations by the World Wide Web Consortium for the lower layers of the Semantic Web tower, including the ontology layer of the Semantic Web. The logic layer, residing on top of the semantic languages and ontology languages, is still to shape.

In this paper, we take a certain perspective on reasoning, rule- and query languages for the Semantic Web: We investigate the required expressiveness of reasoning languages for the Semantic Web which foster personalized Web applications. After a brief introduction to the Semantic Web (Section II, we introduce rule languages for the Semantic Web, with particular notion to nonmonotonic rules (Section III). Aspects of evolution, updates and events are discussed in the subsequent section, exemplified by an event-condition-action approach. Reasoning about actions for implementing personalization is described in Section V.

We then turn attention to mechanisms and applications for maintaining effective reasoning and rule-based approaches: For querying and transforming semantic descriptions, we discuss the *Xcerpt* language (Section VI). An approach to automatically generate semantic descriptions by Web data extraction is provided by the *Lixto Suite* (Section VII). The last section finally goes practical and describes the *Personal Reader Framework* for personalization services on the Semantic Web, which integrates ideas from the previous sections. We outline the service-based architecture of the Personal Reader framework, and describe first example Readers for two application domains.

II. Reasoning and the Semantic Web: State of the Art

The development of the Semantic Web proceeds in steps, each step building a layer on top of another. At the bottom layer we find *XML*, a language that lets one write structured Web documents with a user-defined vocabulary. XML is particularly suitable for sending documents across the Web, thus supporting syntactic interoperability. *RDF* [22] is the basic Semantic Web language for writing simple statements about Web objects (called resources and identified uniquely by a URI, a Universal Resource Identifier). Statements are triples composed of a binary predicate linking together two resources; they are logically represented as logical facts $P(x,y)$. *RDF Schema* [30] provides a simple language for writing ontologies. Objects sharing similar characteristics are put together to form *classes*. Examples for classes are hotels, airlines, employees, rooms, excursions etc. Individuals belonging to a class are often referred to as instances of that class. Binary *properties* (such as *works_for*) are used to establish connections between classes. The application of predicates can be restricted through the use of *domain and range restriction*s. For example, we can restrict the property *works_for* to apply only to employees (domain restriction), and to have as value only companies (range restriction).

Classes can be put together in hierarchies through the *subclass relationship*: a class C is a subclass of a class D if every instance of C is also an instance of D. For example, the class of island destinations is a subclass of all destinations: every instance of an island destination (e.g. Crete) is also a destination. The hierarchical organization of classes is important due to the notion of *inheritance*: once a class C has been declared a subclass of D, every known instance of C is *automatically* classified also as instance of D. This has far-reaching implications for matching customer preferences to service offerings. For example, a customer may wish to make holidays on an indonesian island. On the other hand, the hotel Noosa Beach advertises its location to be Bali. It is not necessary (nor is it realistic) for the hotel to add information that it is located in Indonesia and on an island; instead, this information is inferred by the ontology automatically.

But there is a need for more powerful ontology languages that expand RDF Schema and allow the representations of more complex relationships between Web objects. For example, cardinality constraints (every course must be taught by at least one lecturer) and special properties of predicates (e.g. transitivity, symmetry etc.). Ontology languages, such as *OWL* [40], are built on the top of RDF and RDF Schema. For an easy yet comprehensive introduction to the Semantic Web see [5].

So far, reasoning on the Semantic Web is mostly reasoning about knowledge expressed in a particular ontology. This is possible because ontology languages are *formal languages*, which, for example, allow us to reason about:

- *Class membership*: If x is an instance of class C, and C is a subclass of D, then we can infer that x is an instance of D.
- *Equivalence of classes*: If class A is equivalent to class B, and B is equivalent to class C, then we can infer that A is equivalent to C.
- *Consistency*: If we have declared that classes C and D are disjoint, and x is an instance of both C

and D, then there is an error.

- *Classification*: If we have declared that certain property-value pairs are sufficient conditions for membership in class A, then if an individual x satisfies such conditions, we can conclude that x must be an instance of A.

Derivations such as the preceding can be made *mechanically* instead of being made by hand. Such reasoning support is important because it allows one to:

- check the consistency of an ontology and the knowledge,
- check for unintended clashes between classes,
- automatically classify instances of classes.

Automated reasoning support allows one to check many more classes than could be checked manually. Checks like the preceding ones are valuable for designing large ontologies, where multiple authors are involved, and for integrating and sharing ontologies from various sources.

A. Introducing Rules

At present, the highest layer that has reached sufficient maturity is the ontology layer in the form of the description logic-based language OWL [40]. The next step in the development of the Semantic Web will be the logic and proof layers, and rule systems appear to lie in the mainstream of such activities. Moreover, rule systems can also be utilized in ontology languages. So, in general rule systems can play a twofold role in the Semantic Web initiative:

(a) they can serve as extensions of, or alternatives to, description logic-based ontology languages; and
(b) they can be used to develop declarative systems on top of (using) ontologies.

Reasons why rule systems are expected to play a key role in the further development of the Semantic Web include the following:

- Seen as subsets of predicate logic, monotonic rule systems (Horn logic) and description logics are orthogonal; thus they provide additional expressive power to ontology languages.
- Efficient reasoning support exists to support rule languages.
- Rules are well known in practice, and are reasonably well integrated in mainstream information technology, such as knowledge bases, etc.

As an exemplary application, rules can be a natural means for expressing personalization information. For example, the following rules says that "If E is an exercise related to concept C and person X has read the material on C, then E can be presented to X.

$$exercise(E, C), hasRead(X, C) \rightarrow presentExercise(E, X)$$

A more thorough discussion of personalization rules is found in Section VIII-B.

Possible interactions between description logics and monotonic rule systems were studied in [55]. Based on that work and on previous work on hybrid reasoning [69] it appears that the best one can do at present is to take the intersection of the expressive power of Horn logic and description logics; one way to view this intersection is the Horn-definable subset of OWL.

One interesting research thread deals with the exchange of rule sets between applications, making use of Semantic Web languages. Works in this direction include the RuleML initiative [87], based on the XML and RDF languages, and SWRL [61], a recent proposal based on OWL.

A few implementations of rule systems, tailored to reasoning on the Web, exist yet. The most important systems are Mandarax [73] and Triple [91].

III. Nonmonotonic Rules for the Semantic Web

Apart from the classical rules that lead to monotonic logical systems, recently researchers started to study systems capable of handling *conflicts among rules*. Generally speaking, the main sources of such conflicts are:

- Default inheritance within ontologies.
- Ontology merging, where knowledge from different sources is combined.
- Rules with exceptions as a natural representation of business rules.
- Reasoning with incomplete information.

Defeasible reasoning [4] is a simple rule-based approach to reasoning with incomplete and inconsistent information. It can represent facts, rules, and priorities among rules. The main advantage of this approach is the combination of two desirable features: enhanced representational capabilities allowing one to reason with incomplete and contradictory information, coupled with low computational complexity compared to mainstream nonmonotonic reasoning. The main features of this approach are:

- Defeasible logics are rule-based, without disjunction.
- Rules may support conflicting conclusions.
- The logics are skeptical in the sense that conflicting rules do not fire. Thus consistency is preserved.
- Priorities on rules may be used to resolve some conflicts among rules.
- The logics take a pragmatic view and have low computational complexity.

Recent system implementations, capable of reasoning with monotonic rules, nonmonotonic rules, priorities, RDF data and RDF Schema ontologies, are DR- DEVICE [14] and the system in [3].

Answer Set Programs are nonmonotonic logic programs based on the Answer Set Semantics by Gelfond and Lifschitz [49], which use extended logic programs for reasoning and problem solving by considering possible alternative scenarios. Apart from expressing knowledge by facts and disjunctive rules in a declarative way, ASP is capable of handling incomplete information and default knowledge. Furthermore, user preferences and desires can be accommodated using constructs for expressing priorities and weak constraints (i.e., constraints that can be violated at a penalty). Several very efficient implementations of ASP reasoners exist, e.g., smodels [81] and DLV [67], the latter providing frontends for preferences extensions as well as brave and cautious reasoning. These systems offer gradually expressiveness complexity in alignment with the (lower) complexity of syntactic fragments.

With respect to an application in the domain of the Semantic Web, the advantages of ASP are:

- High expressiveness.
- Declarative semantics.
- Model generation in addition to inference

Model generation enables a problem-solving paradigm in which the solutions of a problem instance are declaratively encoded by the models of the logic program.

Using ASP in the context of the Semantic Web has first been proposed by [60]. A recent extension of ASP programs provides an interface to description logic knowledge bases (such as OWL ontologies) [44], [45]; such extended programs, so-called *dl-programs*, may contain queries to the ontology. This formalism allows a flow of knowledge from the ontology to the logic program and back, exploiting the possibilities of handling terminological knowledge in a nonmonotonic application. A prototype implementation via Web-interface is available.

IV. Evolution, Updates and Events

Personalization of the Web heavily depends on dynamic aspects: it is not given a priori, but it is *adaptive* – i.e., *evolving*, and *reacting* upon *events*, e.g., inputs of the user. Furthermore, personalization is often implemented via *reactive* behavior – i.e., by (personalized) rules that specify what to do in a given situation.

In [74], we have discussed generic query (see also above section), update, and event languages for the Semantic Web. Evolution of the Web is a twofold aspect: on today's Web, evolution means mainly evolution of individual Web sites that are updated locally. In contrast, considering the Web and the Semantic Web as a "living organism" that *consists* of autonomous data sources, but that will *show* a global "behavior" leads to a notion of evolution of the Web as *cooperative evolution* of its individual resources. Personalization aspects deal primarily with local evolution (e.g., that a portal site adapts to evolving profiles of its registered users) and local reactivity (reacting on a user's interaction). But, in the "background", the personalization also potentially affects the global communication of the node (e.g., to gather special information that a user requests, or to react on remote events that are relevant to some of its users), and, the more "intelligent" such Web nodes get, the more they need global communication to deal with the requirements of being personalized. Even more, there can be data exchange about personalization aspects (user profiles) between personalized nodes (although, here also non-technical issues, what a node is entitled to tell another about a user, come into play).

In the same way as proposed in [74], for languages for *evolution and reactivity* in the Semantic Web, we recommend to follow a modular approach. The first step is to provide local personalization of a node that is –at the beginning– part of the conventional Web (see e.g., [57]). The next steps then (i) extend the results to local personalization of the Semantic Web (i.e., semantics-based personalization), (ii) enhance personalization to a "semantic" service (i.e., an ontology for personalization), and (iii) then apply Semantic Web reasoning on the personalization level. In addition to the global language aspects sketched above, the internal mechanisms for evolution of the local personalization base, e.g., as evolution of a logic program, are to be considered [2], [42].

When considering evolution of and events on the Web, two aspects must be taken into account: there are "local" updates that apply to a given Web site only; and there are changes in distributed scenarios that must be propagated from one Web site to another. This means, that in addition to local update languages there must be a declarative, semantic framework for generically handling and *communicating* changes (in general, not as explicit updates, but as changes of a situation, described wrt. a combined ontology of the application and of generic events).

During the development of (generic) languages for evolution and reactivity, personalized nodes will seamlessly be integrated with the application scenarios to be developed. In course, reactive functionality will be employed for implementing personalization and adaptivity (as shown below, by integrating suitable sublanguages for (atomic) events and actions into the generic languages). Analogously, personalization and adaptivity will be subject of local and global evolution.

A. Language Paradigm: ECA Rules

According to [74], we propose an approach that is in general based on rules, more specifically, *reactive* *rules* according to the *Event-Condition-Action* (ECA) paradigm for the specification of reactivity. An important advantage of them is that the *content* of the communication can be separated from the *generic semantics* of the rules themselves. Cooperative and reactive behavior is then based on events (e.g., an update at a data source where possibly others depend on): If a resource detects a relevant event (either it is delivered explicitly, or it is in some way communicated or detectable on the Web), conditions are checked (either simple data conditions, or e.g. tests if the event is relevant, trustable etc.), which are queries to one or several nodes and are to be expressed in the proposed query language. Finally, an appropriate action is taken (e.g., updating own information accordingly). This action can also be formulated as a transaction whose ACID properties ensure that either all actions in a transaction are performed, or nothing of is done. The actions in course raise again events (either explicit updates, or visible as application-level events).

The focus of the development is on appropriate sublanguages for rules, events, conditions (that are in fact queries) and for the action part of the rules that continue the separation between application-specific *contents* and generic patterns (e.g. for composite events).

a) Events.: An (atomic) event is in general any detectable occurrence on the Web, i.e., (local) system events, incoming messages including queries and answers, transactional events (commit, confirmations etc), updates of data anywhere in the Web, or any occurrences somewhere in an application, that are (possibly) represented in explicit data, or signaled as the event itself. For these *atomic events*, it must also be distinguished between the event itself (carrying application-specific information), and its metadata, like the type of event (update, temporal event, receipt of message, . . .), time of occurrence, the time of detection/receipt (e.g., to refuse it, when it had been received too late), the event origin or its generator (if applicable; e.g. in terms of its URI).

Reactive rules often do not specify reactions on atomic events, but use the notion of *composite events*, e.g., "when E_1 happened and then E_2 and E_3, but not E_4 after at least 10 minutes, then do A". Complex events are usually defined in terms of *event algebras*. Thus, a declarative language for describing composite events is required, together with algorithms for handling composite events. This language should not be concerned with what the information contained in the event might be, but only with types of events. For making events themselves part of the Semantic Web, an ontology of composite events has to be defined, together with mappings from and to given event algebras and their implementations.

An important aspect is the integrability of the event meta language, the event contents languages, and the query language. It is desirable that the specification of composite events can be combined with requirements on the state of resources at given intermediate timepoints, e.g. "when at timepoint t_1, a cancellation comes in and somewhere in the past, a reservation request came in in a timepoint when all seats were booked, then, the cancellation is charged with an additional fee". In this case, the composite event handler has to state a query at the moment when a given event arrives. For being capable of describing these situations, a logic (and system) that deals with sequences of events and queries is required. Such approaches have e.g. been presented in *Transaction Logic* [29]; we will also investigate the use of Evolving Logic Programs [1] for this purpose.

So, several integrated languages have to be defined: the surrounding language for composite events, a language for atomic events and their metadata, and languages for expressing the contents of different types of events – e.g., one language based on an ontology for personalization. Note that an ontology for describing data that is relevant to personalization is needed, and a related language for events that are relevant for personalization is required.

b) Events, Knowledge, and Rules.: The view described up to this point is to result in an infrastructure for evolution and reactivity on the Web based on reaction rules that define the behavior of resources upon detection of events. These are in general composite events, based on atomic, application-level ones. Local knowledge is defined by facts, derivation rules, and reaction rules. All of this local knowledge is encoded in XML, and is updatable, in the sense that the update language to be developed must be capable of changing both facts, derivation rules and reactive rules. Here we may rely on the studies done in the context of logic programming about updating derivation and reactive rules [2].

B. Evolution and Reactivity for Personalization

Concepts for personalization and adaptivity will be implemented and supported by the above framework. "Plain" evolving and reactive applications will provide scenarios where personalization is then applied. Expressing personalization by (ECA-) rules is a usual way in today's approaches, which is then extended to a semantic level in various aspects.

For Semantic Web applications, personalization functionality is built upon an ontology-based user model. The ECA rules that implement personalized behavior use –inside the generic languages for the rules and for composite events– sublanguages that combine the user modeling ontology with the respective ontology of application-specific events.

There will prospectively be "typical" rule patterns (i.e., typical structures of composite events that include typical atomic events, and typical action patterns) for expressing personalization issues. These patterns are then "parameterized" by atomic events, special conditions, and actions to yield a certain rule which then belongs to the behavior base of an application. As stated above, this behavior base is also subject to evolution of several kinds:

- reactivity-controlled evolution, which adapts the behavior base according to events on the Web (e.g., when adapting the personal portfolio tracker when a stock to be traced is moved from MDAX to DAX),
- reactivity-controlled evolution, which adapts the behavior base according to changes in the user's profile (e.g., when he is not longer eligible for student tarifs in trains),
- users are enabled to change these rules interactively (via an appropriate graphical interface),
- intelligent evolution by reasoning about the behavior base, etc.

In the context of evolution and reactivity, personalization does not only mean personalized access to the Web –as implemented in today's portals–, but also personalized behavior that is able to raise events. A customer e.g. may have a personalized Web agent for bidding at ebay or for trading stocks. The behavior of such agents is preferably again expressed by ECA rules that can evolve in the same way as described above.

C. Knowledge Base Update and Reasoning About

As pointed out above, the dynamic nature of the desired infrastructure for describing evolution and reactivity on the Web requires not only the capability of updating facts, but also the capability of updating rules. Such updates can be handled in different ways. On the one hand, updates can be performed on an *ad hoc* basis in a static environment, exploiting and adapting methods from the area of knowledge base revision and belief change, see e.g. [48], [92]. On the other hand, and this is for a dynamic environment crucial, updates may occur event-driven. The nature and circumstances of the event which occurred may determine the way in which an entailed update has to be incorporated into the current rule and knowledge base. Here, personalization comes into play since each user might have his or her own view on how this update should materialize. This is supported by user-definable *update policies*, like event-condition-action rules, in which the general change behaviour according to the desires and preferences of the user can be described at a generic level. For instance, the user may define rules which suppress certain unwanted information, or propagate information to parts of the knowledge base which are semantically connected at the meta level.

We envisage a general formal model for expressing different such update approaches, following the method put forth in [43] for capturing different update approaches in the context of (possibly nonmonotonic) knowledge bases. More specifically, such a formal model has different components, taking care of the kind of language, the knowledge base, the change actions, an update policy, etc. under consideration, which can be instantiated in a suitable manner. The accommodation of more general evolving logic programs [1] in it remains to be explored. Moreover, such a formal model provides the basis for defining a temporal logic language for expressing different properties of the evolving knowledge base on top, based on a well-defined semantics. This logical language, in turn, can be used to specify and study general inference and reasoning tasks associated with evolving knowledge and rule bases.

V. Personalization by Reasoning about Actions

Reasoning about action and change is a kind of temporal reasoning where, instead of reasoning about *time* itself, we reason on *phenomena* that take place in time. Indeed, theories of reasoning about action and change describe a *dynamic world* changing because of the execution of actions. Properties characterizing the dynamic world are usually specified by propositions which are called *fluents*. The word *fluent* stresses the fact that the truth value of these propositions depends on time and may vary depending on the changes which occur in the world.

The problem of reasoning about the effects of actions in a dynamically changing world is considered one of the central problems in knowledge representation theory. Different approaches in the literature took different assumptions on the temporal ontology and then they developed different abstraction tools to cope with dynamic worlds. However, most of the formal theories for reasoning about action and change (*action theories*) describe dynamic worlds according to the so-called *state-action model*. In the state-action model the world is described in terms of states and *actions* that cause the transition from a state to another. Typically it is assumed that the world persists in its state unless it is modified by an action's execution that causes the transition to a new state (*persistency assumption*).

The main target of action theories is to use a logical framework to describe the effects of actions on a world where *all* changes are caused by the execution of actions. To be precise, in general, a formal theory for representing and reasoning about actions allows us to specify:

- *causal laws*, i.e. axioms that describe domain's actions in terms of their *precondition* and *effects* on the fluents;
- action sequences that are executed from the initial state;
- *observations* describing the value of fluents in the *initial state*;
- *observations* describing the value of fluents in later states, i.e after some action's execution.

The term *domain description* is used to refer to a set of propositions that express causal laws, observations of the fluents values in a state and possibly other information for formalizing a specific problem. Given a domain description, the principal reasoning tasks are *temporal projection* (or prediction), *temporal explanation* (or postdiction) and *planning*.

Intuitively, the aim of *temporal projection* is to predict an action's future effects based on even partial knowledge about the current state (reasoning from causes to effect). On the contrary, the target of *temporal explanation* is to infer something on the past states of the world by using knowledge about the current situation. The third reasoning task, planning, is aimed at finding an action sequence that, when executed starting from a given state of the world, produces a new state where certain desired properties hold.

Usually, by varying the reasoning task, a domain description may contain different elements that provide a basis for inferring the new facts. For instance, when the task is to formalize the temporal projection problem, a domain description might contain information on (a), (b) and (c), then the logical framework might provide the inference mechanisms for reconstructing information on (d). Otherwise, when the task is to deal with the planning problem, the domain description will contain the information on (a), (c), (d) and we will try to infer (b), i.e. which action sequence has to be executed on the state described in (c) for achieving a state with the properties described in (d).

An important issue in formalization is known as the *persistency problem*. It concerns the characterization of the invariants of an action, i.e. those aspects of the dynamic world that are not changed by an action. If a certain fluent f representing a fact of the world holds in a certain state and it is not involved by the next execution of an action a, then we would like to have an efficient inference mechanism to conclude that f still hold in the state resulting from a's execution.

Various approaches in the literature can be broadly classified in two categories: those choosing classical logics as the knowledge representation language [63], [76] and those addressing the problem by using non-classical logics [36], [52], [84], [90] or computational logics [11], [13], [50], [72]. Among the various logic-based approaches to reasoning about actions one of the most popular is still the situation calculus, introduced by Mc Carthy and Hayes in the sixties [76] to capture change in first order classical logic. The situation calculus represents the world and its change by a sequence of *situations*. Each situation represents a state of the world and it is obtained from a previous situation by executing an action. Later on, Kowalski and Sergot have developed a different calculus to describe change [63], called *event calculus*, in which *events* producing changes are temporally located and they initiate and terminate action effects. Like the situation calculus, the event calculus is a methodology for encoding actions in first-order predicate logic. However, it was originally developed for reasoning about events and time in a logic-programming setting.

Another approach to reasoning about actions is the one based on the use of modal logics. Modal logics adopts essentially the same ontology as the situation calculus by taking the state of the world as primary and by representing actions as state transitions. In particular, actions are represented in a very natural way by modalities whose semantics is a standard Kripke semantics given in terms of accessibility relations between worlds, while states are represented as sequences of modalities.

Both situation calculus and modal logics influenced the design of logic-based languages for agent programming. Recently the research about situation calculus gained a renewed attention thanks to the cognitive robotic project at University of Toronto, that has lead to the development of a high-level agent programming language, called GOLOG, based on a theory of actions in situation calculus [68]. On the other hand, in DyLOG [12], a modal action theory has been used as a basis for specifying and executing agent behaviour in a logic programming setting, while the language IMPACT is an example of use of deontic logic for specifying agents: the agent's behavior is specified by means of a set of rules (the agent program) which are suitable to specify, by means of deontic modalities, agent policies, that is what actions an agent is obliged to take in a given state, what actions it is permitted to take, and how it chooses which actions to perform.

Let us now show how these concepts can be useful in the Semantic Web, by describing two scenarios where personalization is required. The idea of exploiting reasoning techniques for obtaining adaptation derives from the observation that in many application domains the goal of the user and the interaction occurring with a resource play a fundamental role.

A. Reasoning about Web Services

In the first scenario that we consider, the action metaphor is used for describing (and handling) *Web services*. Generally speaking, a Web service can be seen as any device that can automatically be accessed over the Web. It may alternatively be a software system or a hardware device; a priori no distinction is made. The main difference between a Web service and other devices that are connected to a network stands in the kind of tasks that can be performed: a Web service can be automatically retrieved after a search (that can be thought of as analogous to finding Web pages by means of a search engine, given a set of keywords), it can be automatically invoked, composed with other Web services so to accomplish more complex tasks, it must be possible to monitor its execution, and so on. In order to allow the execution of these tasks, it is necessary to enrich the Web service with a machine-processable description, that contains all the necessary information, such as what the service does, which inputs it requires, which results are returned, and so forth. A lot of research is being carried on in this area and none of the problems that we have just enumerated has met its final solution yet. Nevertheless, there are some proposals, especially due to commercial coalitions, of languages that allow the description of the single services, and their interoperation. In this line, the most successful are WSDL [93] and BPEL4WS. This initiative is mainly carried on by the commercial world, with the aim of standardizing registration, look-up mechanisms and interoperability,

Among the other proposals, OWL-S [82] (formerly DAML-S [38]) is more concerned with providing greater expressiveness to service description in a way that can be *reasoned about* [34]. In particular, a service description has three conceptual levels: the *profile*, used for advertising and discovery, the *process model*, that describes how a service works, and the *grounding*, that describes how an agent can access the service. In particular, the process model describes a service as atomic, simple or composite in a way inspired by the language GOLOG and its extensions [51], [68], [77]. In this perspective, a wide variety of agent technologies based upon the *action metaphor* can be used. In fact, we can view a service as an action (atomic or complex) with preconditions and effects, that modifies the state of the world and the state of agents that work in the world. The process model can, then, be viewed as the description of such an action; therefore, it is possible to design agents, which apply techniques for reasoning about actions and change to Web service process models for producing new, composite, and customized services.

Quoting McIlraith [78]: "[...] *Our vision is that agents will exploit user's constraints and preferences to help customize user's requests for automatic Web service discovery, execution, or composition and interoperation* [...]". In different words, personalization is seen as *reasoning* about the user's constraints and preferences and about the *effects*, on the user's knowledge and on the world, of the *action* "interact with a Web service". Techniques for reasoning about actions and change are applied to produce composite and customized services.

We claim that a better personalization can be achieved by allowing agents to reason also about the *conversation protocols* followed by Web services. Conversation protocols rule the interactions of a service with its interlocutors: the protocol defines all the possible "conversations" that the service can enact. Roughly speaking, we can consider it as a procedure built upon atomic speech acts. So far, however, OWL-S does not represent in a way that can be reasoned about, the communicative behaviour of a service. Let us explain with a simple example how this would be useful: an agent, which is a user's *personal assistant*, is requested to book a ticket at a cinema where they show a certain movie; as a further constraint, the agent does not have to use the user's credit card number along the transaction. While the first is the *user's goal*, the additional request constrains the way in which the agent will *interact* with the service. In this case, in order to personalize the interaction according to the user's request, it is indeed necessary to reason about the service communications.

In [7] a Web service is supposed to follow some (possibly non-deterministic) procedure for interacting with other services/agents. The authors show that by reasoning on the (explicitly given) conversation protocols followed by Web services, it is possible to achieve a *better personalization* of the service fruition. More recently, the same authors have shown that the same kind of reasoning can be exploited for *composing* a set of Web services, which must interoperate in order to accomplish a complex task, that none of them can execute by itself alone. Consider, as an example, the organization of a journey: it is necessary to find and make work together services for finding a flight, renting a car, making a reservation at some hotel, maybe the user's personal calendar, etc. All services that have been developed independently and for simpler purposes. The problem of describing and reasoning about conversation protocols is faced in an *agent logic programming* setting, by exploiting the reasoning capabilities of agents written in the DyLOG language, introduced in [12]. In particular, integrated in the language, a communication kit [8], [83] allows reasoning about the possible interactions ruled by a protocol by answering to existential queries of the kind: is there a possible execution of the protocol, after which a set of beliefs of interest (or goal) will be true in the agent's mental state?

B. *Reasoning about Learning Resources*

The second scenario is set in an e-learning framework: a system has to manage a repository of learning resources, helping users to retrieve the documentation that they need, for acquiring some desired expertise. The goal of the system is returning a *personalized reading sequence* through a (sub)set of the available resources, that will allow the specific user to reach his/her learning goal. Notice that resources may be of different kinds, e.g. text, examples, tests, programming patterns, references to books, and so forth.

The same learning object can be used in different reading sequences, maybe aimed at different learning goals. Moreover, a sequence might contain learning objects that are physically located in different repositories.

Based on the experience gained in previous work [9], [10], an approach is to carry on the construction of reading sequences by means of techniques for reasoning about actions, like planning and temporal explanation, applying them to semantically annotated learning resources. Indeed, also in this scenario the adoption of the "action metaphor" is quite straightforward: a *learning resource* can, in fact, be considered as an *action*, with preconditions (what the student should know for understanding the knowledge contents) and effects (what the student is supposed to learn by reading the resource) on the knowledge of the reader. This choice is also supported by research in pedagogy that shows that human learning is goal-driven, and the notions of prerequisite and effect (in our case, knowledge gain) play a fundamental role. In the action-based representation of learning resources, prerequisites and effects are supposed to be expressed by means of "knowledge entities", i.e. terms from a reference ontology.

In this scenario the goal of personalization is to produce reading sequences that fit the specific user's characteristics (i.e. users with different initial knowledge will be suggested different solutions) and the user's learning goal. Notice that, differently than what happens in other approaches, adaptation occurs at the level of the *reading sequence* rather than at the level of page contents (no link hiding or semaphore annotation is supposed to be used), and it is done w.r.t. the user's *learning goal*.

Many reasoning techniques can be applied in this scenario. One way for building personalized reading sequences is to apply planning techniques; on the other hand, temporal explanation can be used to motivate the user to read documents, that apparently have no direct relation to the learning goal. Also techniques for dealing with *failure* and *replanning* are useful: failure occurs when a user is not satisfied of the proposed solution, on the whole or of part of it, and the system is asked to find alternatives. *Non-monotonic* reasoning techniques could help in this case.

In the literature, it is possible to find programming languages based on action logics (like DyLOG and GOLOG) that support some of the mentioned reasoning techniques and many others. For instance, in DyLOG it is possible to exploit a kind of planning, known as *procedural planning*, that rather than combining in all the possible ways the available actions (documents, or resources) searches for solutions in a restricted space, consisting of the set of possible executions of a given procedure. In this case the procedure describes the general schema of the solution to be found, which is kept separate from the specific resources. At planning time, depending on the initial situation and on the available resources, a solution will be built. The use of procedures as schemas allows the achievement of a form of personalization that not only depends on the user's characteristics and goal (whose description is contained in the initial state) but it also depends on preferences given by the providers of the resources. In the scenario in issue, the procedure would correspond to a *learning strategy* described by the lecturer of the course, which takes into account the experience of the teacher and his/her preferences on how the topic should be thought.

VI. XCERPT: A QUERY AND TRANSFORMATION LANGUAGE FOR WEB AND SEMANTIC WEB APPLICATIONS

Querying the Web, i.e. retrieving Web and Semantic Web data using queries expressed in a high level language, can considerably ease the realization of personalized information systems on the Web. Doing this using a query language capable of deduction can further simplify conceiving and implementing personalized information systems on the Web.

Xcerpt [24], [89], [94] is an experimental deductive query language developed at the Institute for Informatics of the University of Munich since 2001.

The goal of the Xcerpt project is to investigate ways to ease realizing Web as well as Semantic Web applications, in particular realizing personalized information systems on the Web. One might see the Semantic Web meta-data added to today's Web as semantic indexes similar to encyclopedias. A considerable advantage over conventional encyclopedias printed on paper is that the relationships expressed by Web meta-data can be followed by computers, very much like hyperlinks can be followed by programs, and be used for drawing conclusion using automated reasoning methods:

> For the Semantic Web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. [28]

A central principle of the Web query language Xcerpt presented in this section is that a common query language capable of inference to be used for querying both the conventional Web and the Semantic Web is desirable and possible. This working hypothesis is one of the salient features of Xcerpt which makes it different from all Web as well as Semantic Web query languages developed so far.

1) *Xcerpt's Principles:*

a) Referential Transparency.: Referential transparency means that, within a definition scope, all occurrences of an expression have the same value, i.e. denote the same data. Referentially transparent programs are easier to understand and therefore easier to develop, to maintain, and to optimize. Referential transparency surely is one of the essential properties a query language for the Web should satisfy.

b) Answer-Closedness.: We call "answer-closed" a query language such that replacing a subquery in a compound query by possible (not necessarily actual) answers always yields a syntactically valid query. Answer-closed query languages ensure in particular that every data item, i.e. every possible answer to some query, is a syntactically valid query. Functional programs can – but need not – be answer-closed. Answer-closedness eases the specification of queries because it keeps limited the unavoidable shift in syntax from the data sought for, i.e. the expected answer, and the query specifying these data. Xcerpt is answer-closed.

c) Answers as Arbitrary XML Data.: XML is the lingua franca of data interchange on the Web. As a consequence, answers should be expressible as every possible XML application. This includes both text without markup and freely chosen markup and structure. This requirement is obvious and widely accepted for conventional Web query languages but it is not enforced by many Semantic Web query languages.

d) Answer Ranking and Top-k Answers.: It is often desirable to rank answers according to some application-dependent criteria. It is desirable that Web and Semantic Web query languages offer (a) basic means for specifying ranking criteria and, for efficiency reasons, (b) evaluation methods computing only the top-k answers (i.e. a given number k of best-ranked answers according to a user-specified ranking criterium). Xcerpt supports the specification of orders on XML documents and the retrieval of k answers of a query, possibly sorted according to a specified order.

e) Pattern Queries.: Xcerpt uses patterns for binding variables in query expressions instead of path expressions – like e.g. the Web query languages XQuery and XSLT. Query patterns are especially well-suited for a visual language because queries have a structure very close to that of possible answers.

f) Incomplete Query Specifications.: Incomplete queries specifying only part of data to retrieve, e.g. only some of the children of an XML element (referring to the tree representation of XML data called "incompleteness in breadth") or an element at unspecified nesting depth (referring to the tree representation of XML data called "incompleteness in depth"), are important on the conventional Web because of its heterogeneity: one often knows part of the structure of the XML documents to retrieve. For similar reasons, incomplete queries are important on the Semantic Web. Xcerpt supports queries that are incomplete in breadth, in depth, with respect to the element order, and because of optional elements or attributes.

g) Incomplete Data Selections.: Because Web data are heterogeneous in their structures, one is often interested in "incomplete answers". Two kinds of incomplete answers can be considered. First, one might not be interested in some of the children of an XML (sub)document retrieved by a query. Second, one might be interested in some child elements if they are available, but would accept answers without such elements. Xcerpt's construct `except` gives rise to discard a child of an element retrieved by a query, i.e. to express queries of the first kind. Xcerpt's construct `optional` gives rise to select elements only if available, i.e. to express queries of the second kind.

h) Rule-Based, Chaining, and Recursion.: Rules are understood here as means to specify novel, maybe virtual data in terms of queries, i.e. what is called "views" in (relational) databases, regardless of whether this data is materialized or not. Views, i.e. rule-defined data are desirable for both conventional and Semantic Web applications. Xcerpt supports (unrestricted) recursion on possibly cyclic data (relying on a so-called "memorization" or "tabulation" technique).

i) Separation of Queries and Constructions.: Two standard and symmetrical approaches are widespread, as far as query and programming languages for the Web are concerned:

- queries or programs are embedded in a Web page or Web page skeleton giving the structure of answers or data returned by calls to the programs
- parts of a Web page specifying the structure of the data returned to a query or program evaluation are embedded in the queries or programs.

It is a thesis of the Xcerpt project that both approaches to queries or programs are hard to read (and, therefore, to write and to maintain). Instead of either approach, Xcerpt strictly separates queries and "constructions", i.e. expressions specifying the structure of answers. With Xcerpt, constructions are rule heads and queries are rule bodies. In order to relate a rule's construction, i.e. the rule's head, to a rule's query, i.e. the rule's body, Xcerpt uses (logic programming) variables.

j) A Query Language for both the Standard Web and the Semantic Web.: A thesis underlying the Xcerpt project is that a common query language for both conventional Web and Semantic Web applications is desirable.

k) Specific Reasoning as Theories.: Many practical applications require special forms of reasoning. For this reason, it is desirable that a query language for the (conventional and Semantic) Web can be extended with so-called "theories" implementing specific forms of reasoning.

l) Two Syntaxes: XML Syntax and Compact Human Readable Syntax.: While it is desirable that a query language for the (conventional and/or Semantic) Web has an XML syntax, because it makes it easier to exchange query programs on the Web and to manipulate them using the query language, a second, more compact syntax easier for human being to read and write is desirable.

2) Flavors of Xcerpt: Xcerpt's Core Constructs: An Xcerpt program consists of at least one goal and of some (possibly zero) rules. Goals and rules are built from data, query, and construct terms representing respectively XML documents, query, and XML documents constructed from the answers to queries.

Data, query, and construct terms represent tree-like (or graph-like) structures. In data, query, and construct terms, square brackets (i.e. [ ]) denote ordered term specification (as in standard XML), i.e. the matching subterms in the queried resource are required to be in the same order as in the query term. Curly braces (i.e. { }) denote unordered term specification (as is common in databases), i.e. the matching subterms in the queried resource may be in arbitrary order. Single (square or curly) braces (i.e. [ ] and { }) denote that a matching term must contain matching subterms for all subterms of a term and may not contain additional subterms (total term specification). Double braces (i.e. [[ ]] and {{ }}) denote that the data term may contain additional subterms as long as matching partners for all subterms of the query term are found (partial term specification).

Non-tree graph structures are expressed using references, i.e. symbolic addresses: The construct `id @ t` is a defining occurrence of the identifier `id` as reference handle of a term `t` and the construct `^id` is a referring occurrence.

a) Data terms: represent XML documents (we speak of "XML in disguise"). They are similar to ground functional programming expressions and logical atoms. Data terms may only contain single square and curly braces, but no double braces expressing partial specifications, as an XML document is complete.

The data term in Figure 1 is the shortened representation of an article in Xcerpt syntax. Note that some parts of the article use unordered term specification (e.g. the author entries) since the order is irrelevant.

b) Query terms: are partial patterns that are matched with data terms, augmented by an arbitrary number of variables for selecting data items from a data term. In addition to the constructs used in data terms, query terms have the following additional properties:

1) partial specifications omitting subterms irrelevant to the query are possible (indicated by double square brackets [[ ]] or curly braces {{ }}),
2) it is possible to specify subterms at arbitrary depth using the construct `desc`),
3) query terms may contain term variables and label variables to "select" data.

In the following examples, upper case characters are chosen for variables. The Xcerpt construct `X -> t` (read "X as `t`") associates a variable to a query term, so as to specify a restriction of its bindings. The Xcerpt construct `desc` (read "descendant") is used to specify subterms at arbitrary depth. Suppose that the articles of the proceedings of a conference are contained in a `proceedings` element. The following query term selects title and author pairs for each article:

```
proceedings {{
         article {{
             var T -> title {{ }},
             var A -> author {{ }}
         }}
      }}
```

Query terms (in general containing variables) are unified with data or construct terms (in which variables may occur) using a non-standard unification expecially conceived for Xcerpt and called simulation unification [33]. Simulation unification is based on "graph simulation", a relation similar to graph homomorphisms.

The result of unifying a query term with a data term (construct term, resp.) is a set of substitutions for the variables in the query term (in the query term and construct term, resp.), where each substitution represents an alternative solution.

c) Construct terms: serve to reassemble variables (the bindings of which are specified in query terms) so as to construct new data terms. They may only contain single brackets (i.e. [ ] or { }) and variables, but no partial specification (i.e. no double braces [[ ]] or {{ }}) or variable restrictions (i.e. `X -> t`). The rationale of this is to keep variable specifications within query terms, ensuring a strict separation of purposes between query and construct terms. The following construct term creates an Author-Title pair wrapped in a "`result`" element:

Fig. 1. Representation of an article in Xcerpt syntax

```
paper [
        front [
           title [ "Reasoning Methods for Personalization
                    on the Smantic Web " ],
           author {
              fname [ "Grigoris" ],
              surname [ "Antoniou" ],
              address { ... },
              bio [ ... ]
           },
           author {
              fname [ "Nicola" ],
              surname [ "Henze" ],
              address { ... },
              bio [ ... ]
           },
        ],
        body [
           section [
              title [ "Introduction" ], ],
              ...
        ],
        rear [
            acknowl [ ... ],
            bibliog {
                 bibitem [
                    bib [ "XQuery" ],
                    pub [ "XQuery: The XML Query Language ..." ]
                 ],
                 ...
            }
        ]
     ]
```

```
  result {
           var A, var T
      }
```

In a construct term, the Xcerpt construct `all t` serves to collect (in the construct term) all instances of `t` that can be generated by alternative substitutions for the variables in `t` (returned by the associated query terms in which they occur). Likewise, `some n t` serves to collect at most `n` instances of `t` that can be generated in the same manner. Referring to the previous query, the following construct term creates a list of publications for each author:

```
results {
          result {
              var A,
              all var T
           }
      }
```

Referring again to the previous query, the following construct term collects all titles for each author:

```
      results {
           all result { var A, all var T }
      }
```

Referring again to the previous query, the following construct term collects all titles for each author:

```
results {
    all result { all var A, var T }
}
```

d) Queries: Query terms are (atomic) *queries*. Query terms can be "and" or "or"-connected yielding (complex) *queries*. A query is always (implicitly or explicitly) associated with a resource, i.e. the program itself, an external Xcerpt program or an (XML or other) document specified by a URI (uniform resource identifier). All occurrences of a variable in a query term and in and-connected queries are always evaluated identically: this is the usual approach to variable binding in the database query language SQL and in logic programming. The query in Figure 2 selects all authors that have published an article in the proceedings of the 2003 and 2004 venues of a conference (it is assumed that the articles are contained in a `proceedings03` resp. `proceedings04` element):

e) Construct-query rules and goals.: An Xcerpt program consists of zero or more *construct-query rules*, one or more *goals* and zero or more data terms. In particular, an XML document, i.e. a data term, is an Xcerpt program. Rules and goals have the forms:

Fig. 2. An example query

```
and {
    in { resource { "file:proceedings03.xml" },
        desc author {{
            fname { var First }, surname { var Last }
        }}
    },
    in { resource { "file:proceedings04.xml" },
        desc author {{ fname { var First }, surname { var Last }
        }}
    }
}
```

```
CONSTRUCT construct term
FROM query
END

GOAL construct term
FROM query
END
```

where a `construct term` is constructed depending on the evaluation of a `query`, i.e. shared variables.

f) Further constructs.: Besides the core constructs presented above, Xcerpt has so-called “advanced constructs”. These constructs give rise to expressing (1) functions and aggregations (such as count, average, etc.), (2) that part of a query is “optional”, i.e. to be retrieved only if present in the data considered, (3) to express positions of subterms searched for, and (4) negation in queries. Xcerpt’s advanced constructs are detailed in [89].

3) Languages Related to Xcerpt: Two companion languages of Xcerpt deserve to be mentioned: visXcerpt and XChange. visXcerpt [23], [25] is a visual language based on the same principles as the textual language presented above. XChange is a reactive language based on Xcerpt for expressing updates and exchanging events on the Web [31], [32].

VII. Web Data Extraction

If, on a hand, today the Semantic Web [27] is still a vision, on the other, the *unstructured Web* already contains millions of documents which are not queryable as a database and heavily mix layout and structure. Moreover, they are not annotated at all. There is a huge gap between Web information and the qualified, structured data as usually required in corporate information systems. According to the vision of the Semantic Web, all information available on the Web will be suitably structured, annotated, and qualified in the future. However, until this goal is reached, and also, towards a faster achievement of this goal, it is absolutely necessary to (semi-)automatically extract relevant data from HTML documents and automatically translate this data into a structured format, e.g., XML. Once transformed, data can be used by applications, stored into databases or populate ontologies.

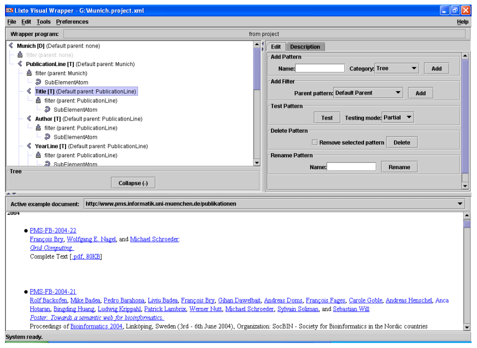
Whereas information retrieval targets to analyze and categorize documents, information extraction collects and structures entities inside of documents. For Web information extraction languages and tools for accessing, extracting, transforming, and syndicating the Data on the Web are required. The Web should be useful not merely for human consumption but additionally for machine communication. A program that automatically extracts data and transforms it into another format or markups the content with semantic information is usually referred to as *wrapper*. Wrappers bridge the gap between unstructured information on the Web and structured databases.

A number of classification taxonomies for wrapper development languages and environments have been introduced in various survey papers [47], [64], [66].

High-level languages have been developed for Web extraction. These *stand-alone wrapper programming languages* include *Florid* [75], *Jedi* [62], *Tsimmis* and *Araneus* [6]. In general, all manual wrapper generation languages are difficult to use by laypersons.

Machine learning approaches generally rely on learning from examples and counterexamples of a large number of Web pages (*Stalker* [80], Davulcu et al. [39], *Wien* [65]). The *RoadRunner* [37] approach does not need labelled examples, but derives rules from a number of given pages by distinguishing the structure and the content. It uses an interesting generation of pattern names based on offset-criteria in addition to the applied semi-structured wrapping technology. Some approaches such as [46] offer generic wrapping techniques. Such approaches have the advantage that they can wrap arbitrary Web pages never seen before, on the other hand the disadvantage that they are restricted to particular domains (such as detecting addresses).

Interactive approaches allow for semi-automatic extraction generation and offer convenient visual dialogues to generate a wrapper based on a few examples and user interaction. *Supervised interactive wrapper*



Munich [D] (Default parent: none)
filter (parent: none)
PublicationLine [T] (Default parent: Munich)
filter (parent: Munich)
SubElementAtom
Title [T] (Default parent: PublicationLine)
filter (parent: PublicationLine)
SubElementAtom
Author [T] (Default parent: PublicationLine)
filter (parent: PublicationLine)
SubElementAtom
YearLine [T] (Default parent: PublicationLine)
filter (parent: PublicationLine)
SubElementAtom

http://www.pms.informatik.uni-muenchen.de/publikationen

2004

- PMS-FB-2004-22
  François Bry, Wolfgang E. Nagel, and Michael Schroeder:
  Grid Computing.
  Complete Text [.pdf, 80KB]

- PMS-FB-2004-21
  Rolf Backofen, Mike Badea, Pedro Barahona, Liviu Badea, François Bry, Gihan Dawelbait, Andreas Doms, François Fages, Carole Goble, Andreas Henschel, Anca Hotaran, Bingding Huang, Ludwig Krippahl, Patrick Lambrix, Werner Nutt, Michael Schroeder, Sylvain Soliman, and Sebastian Will:
  Poster: Towards a semantic web for bioinformatics.
  Proceedings of Bioinformatics 2004, Linköping, Sweden (3rd - 6th June 2004), Organization: SocBIN - Society for Bioinformatics in the Nordic countries

Fig. 3. Lixto Visual Wrapper: Wrapping Publication Pages

generation tools include *W4F* [88], *XWrap* [70], *Wiccap* [71], *SGWrap* [79], and *Wargo* [85] and *DEByE* [86]. In general, many systems neglect the capabilities of Deep Web navigation such as form filling; however, in practice this is highly required, as most information is hidden somewhere in the Deep Web [26].

A. Lixto

Lixto [17] is a methodology and tool for visual and interactive wrapper generation developed at the University of Technology in Vienna. It allows wrapper designers to create so-called “XML companions” to HTML pages in a supervised way. As internal language, Lixto relies on *Elog*. Elog is a Datalog-like language especially designed for wrapper generation. Examples of programs in Elog are given in [16]. The Elog language operates on Web objects, that are HTML elements, lists of HTML elements, and strings. Elog rules can be specified fully visually without knowledge of the Elog language. Web objects can be identified based on internal, contextual, and range conditions and are extracted as so-called “pattern instances”.

In [53], [54], the expressive power of a kernel fragment of *Elog* has been studied, and it has been shown that this fragment captures monadic second order logic, hence is very expressive while at the same time easy to use due to visual specification.

Besides expressiveness of a wrapping language, robustness is one of the most important criteria. Information on frequently changing Web pages needs to be correctly discovered, even if e.g. a banner is introduced.

Visual Wrapper offers robust mechanisms of data extraction based on the two paradigms of tree and string extraction. Moreover, it is possible to navigate to further documents during the wrapping process. Predefined concepts such such as “is a weekday” and “is a city” can be used. The latter is established by connecting to an ontological database. Validation alerts can be imposed that give warnings in case user-defined criteria are no longer satisfied on a page.

Visually, the process of wrapping is comprised of two steps: First, the identification phase, where relevant fragments of Web pages are extracted (see Figure 3). Such extraction rules are semi-automatically and visually specified by a wrapper designer in an iterative approach. This step is succeeded by the structuring phase, where the extracted data is mapped to some destination format, e.g. enriching it with XML tags. With respect to populating ontologies with Web data instances, another phase is required: Each information unit needs to be put into relation with other pieces of information.

B. Visual Data Processing with Lixto

Heterogeneous environments such as integration and mediation systems require a conceptual information flow model. The usual setting for the creation of services based on Web wrappers is that information is obtained from multiple wrapped sources and has to be integrated; often source sites have to be monitored for changes, and changed information has to be automatically extracted and processed. Thus, push-based information systems architectures in which wrappers are connected to pipelines of postprocessors and integration engines which process streams of data are a natural scenario, which is supported by the Lixto

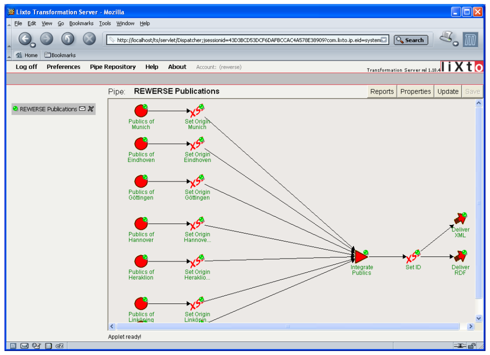


Fig. 4. Lixto Transformation Server: REWERSE Publication Data Flow

Transformation Server [21], [59]. The overall task of information processing is composed into stages that can be used as building blocks for assembling an information processing pipeline. The stages are to

- acquire the required content from the source locations; this component resembles the Lixto Visual Wrapper plus Deep Web Navigation;
- integrate and transform content from a number of input channels and tasks such as finding differences, and
- format and deliver results in various formats and channels and connectivity to other systems.

The actual data flow within the Transformation Server is realized by handing over XML documents. Each stage within the Transformation Server accepts XML documents (except for the wrapper component, which accepts HTML), performs its specific task (most components support visual generation of mappings), and produces an XML document as result. This result is put to the successor components. Boundary components have the ability to activate themselves according to a user-specified strategy and trigger the information processing on behalf of the user. From an architectural point of view, the Lixto Transformation Server may be conceived as a container-like environment of visually configured information agents. The pipe flow can model very complex unidirectional information flows (see Figure 4). Information services may be controlled and customized from outside of the server environment by various types of communication media such as Web services.

C. Web Data Extraction Application Domains

Better software connections are a key challenge to rapid progress in collaborative and e-commerce applications. Rather than waiting for suppliers to recode entire applications to Web service and Semantic Web standards, one can choose the route to better Web connectivity, using today's existing systems. Extraction technologies help to unfold the structure of the desired pieces of information from HTML documents and translate it into XML in a very cost-effective way.

With Lixto some functions that will be tangible only in the future Semantic Web are already turning into reality today. Lixto applications collect data, transform the information into a homogeneous structure and syndicate the semantically enriched data to applications or devices. Lixto's advantages in respect to other wrapper tools and screen-scrapers are its high flexibility, robustness, expressiveness, usability, and its ability to provide interfaces to various data formats and delivery channels [19], [20].

The application domains of extraction technologies are manifold. They e.g. include automatizing portal-based interactions between automotive suppliers, repackaging content for mobile devices, monitoring e.g. price and news data for business intelligence frameworks, and updating address data for CRM databases [15], [18]. Moreover, Web data harvested and syndicated by Lixto can be ideally used by personalization systems, e.g. to offer personalized views on extracted news or publications, as described in Section VIII-D.

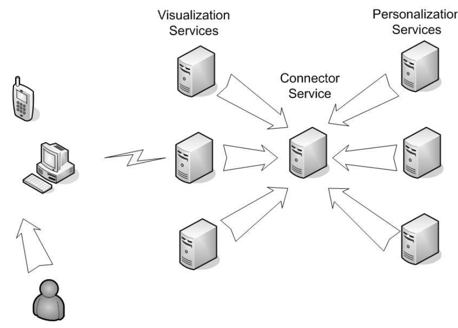


Fig. 5. Architecture of the Personal Reader framework, showing the different components of the Personal Reader: Visualization (user interface), the Personal Reader backbone (consisting of the Connector services, the Reasoning service(s)), and some data-provision services, for RDF data and for the connection with some database for storing user profile information.

VIII. Personalization Services for the Semantic Web: The Personal Reader Framework

How can we establish personalization for the Semantic Web? Personalization can provide guidance, recommendations, hints for a user browsing the Web, it makes the retrieval process of information more effective, it supports users in managing their view on information on the Web, etc. To sum it up, personalization provides an *added value* or a *service* to the end user. One approach for bringing personalization functionality to the (Semantic) Web is therefore to realize *Personalization Web services* which are offered to end user for selection according to their convenience, or to applications for retrieving and integrating additional functionality, as discussed in Section V. In this section, we describe two demonstrator applications for implementing personalization functionality in the Semantic Web, following the approach discussed in Section IV: a Personal Reader Instance for the e-Learning domain, and a Personal Publication Reader.

A. Architectural Overview of the Personal Reader Framework

The Personal Reader Framework[1] is an environment for designing, implementing and maintaining personal Web-content Readers [41], [56]. These personal Web-content Readers allow a user to browse information (the *Reader* part), and to access personal recommendations and contextual information on the currently regarded Web resource (the *Personal* part). We will briefly outline the underlying architecture of the Personal Reader framework, and discuss in more detail how personalization services for two instances of Personal Readers have been implemented.

The architecture of the Personal Reader is a rigorous approach for applying recent Semantic Web technologies. A modular framework of Web services – for constructing *the user interface*, for *mediating* between user requests and currently available personalization services, for *user modeling*, and for offering *personalization functionality* – forms the basis for the Personal Reader. The communications between all components / services is syntactically based on RDF descriptions (see Figure 5).

The common "understanding" of the services is realized by referring to semantics in the ontologies which provide the valid vocabulary for describing functionality, user interface components, requests, etc. In particular, we employ the following ontologies for describing our objects of discourse, following the logic-based definition of adaptive hypermedia systems [58]:

1) a domain ontology describing the application domain, and a document ontology.
2) a user model ontology (attribute–value pairs for user characteristics, preferences, information on the devices the user is using for accessing the Personal Reader, etc.);
3) an observation ontology (for describing the different kinds of user observations made during runtime);
4) and an adaptation ontology for describing the adaptation functionality which is provided by the adaptation services.

The underlying architecture of the Personal Reader Framework allows to design, implement and maintain

[1]www.personal-reader.de

Fig. 6. Determining details for the currently used learning resource

```
FORALL LO, LO_DETAIL detail_learningobject(LO, LO_DETAIL) <-
  EXISTS C, C_DETAIL(detail_concepts(C, C_DETAIL)
    AND concepts_of_LO(LO, C) AND concepts_of_LO(LO_DETAIL, C_DETAIL))
    AND learning_resource(LO_DETAIL) AND NOT unify(LO,LO_DETAIL).
```

Personal Web Content Readers. In the following, we describe two Personal Reader instances which have been recently developed: A Personal Reader for the e-Learning domain, and a Personal Publication Reader developed for the publications of the Network of Excellence REWERSE[2].

B. A Personal Reader Instance: Personal Reader for e-Learning

Let us start with a specific scenario, involving a user, Alice, interested in learning Java programming:

> Alice is currently learning about variables in Java by accessing some learning resource in an online tutorial. During her studies she realizes that she needs some clarifications on naming variables. The Personal Reader shows where detailed information on variables can be found in this online tutorial, and also points out recommended references for deeper understanding. For ensuring that Alice understands the use of variables, the Personal Reader provides several quizzes. When practicing, Alice does some of the recommended exercises. For the chosen exercises, the Personal Reader provides Alice with appropriate links to the Java API, and some already solved exercises. A further source of information are the JAVA FAQ references pointed out to Alice by the Personal Reader.

The Personal Reader for e-Learning (PR-eL) provides a learner with such a personal interface for studying learning resources: the *Personal Annotation service* recommends the learner next learning steps to take, points to examples, summary pages, more detailed information, etc., and always recommends the most appropriate of these information according to the learner's current knowledge, his/her learning style, learning goal, background, etc.

We provide some examples of personalization rules from the Personal Annotation services of the PR-eL for learning the Java programming language. This Personal Reader helps the learner to view the learning resources from the Sun Java Tutorial [35], a freely available online Tutorial on Java programming, in a context: more *details* related to the topics of the learning resource, the *general topics* the learner is currently studying, *examples*, *summaries*, *quizzes*, etc. are generated and enriched with personal recommendations according to the learner's current learning state.

[2]rewerse.net

For implementing the reasoning rules, we currently use the TRIPLE [91] query and rule language for the Semantic Web. Rules defined in TRIPLE can reason about RDF-annotated information resources (required translation tools from RDF to triple and vice versa are provided). An RDF statement (which is a triple) is written as `subject[predicate -> object]`.

RDF *models* are explicitly available in TRIPLE: Statements that are true in a specific model are written as "@model". This in particular is important for constructing the *temporal knowledge bases* as required in the Personal Reader. Connectives and quantifiers for building logical formulae from statements are allowed as usual: `AND`, `OR`, `NOT`, `FORALL`, `EXISTS`, `<-`, `->`, etc. are used.

In the following, we will describe some of the rules that are used by the Personal Reader for learning resources to determine appropriate adaptation strategies.

a) Providing a Context by Displaying Details of a Learning Resource.: Generating links to more detailed learning resources is an adaptive functionality in this example Personal Reader.

The adaptation rule takes the isA hierarchy in the domain ontology, in this case the domain ontology for Java programming, into account to determine domain concepts which are details of the current concept or concepts that the learner is studying on the learning resource. In particular, more details for the currently used learning resource are determined by `detail_learningobject(LO, LO_DETAIL)`, see Figure 6, where `LO` and `LO_Detail` are learning resources, and where `LO_DETAIL` covers more specialized learning concepts which are determined with help of the domain ontology.

N. B. the rule does neither require that `LO_DETAIL` covers all specialized learning concepts, nor that it exclusively covers specialized learning concepts. Further refinements of this adaptation rule are of course possible and should, in a future version of the Personal Reader, be available as tuning parameters under control of the learner. The rules for embedding a learning resource into more general aspects with respect to the current learning progress are similar.

b) Providing Pointers to Quizzes.: Another example, Figure 7, of an *adaptation rule* for generating embedding context is the recommendation of quiz pages. A learning resource `Q` is recommended as a quiz for a currently learned learning resource `LO` if it is a quiz (the rule for determining this is not displayed) and if it provides questions to at least some of the concepts learned on `LO`.

c) Calculating Recommendations.: Recommendations are personalized according to the current learning progress of the user, e. g. with respect to the current set of course materials. The rule in Figure 8 determines that a learning resource `LO` is `recommended` if the learner studied at least one more general learning resource (`UpperLevelLO`).

Additional rules deriving stronger recommendations (e. g., if the user has studied *all* general learning resources), less strong recommendations (e.g., if one or two of these haven't been studied so far), etc., are possible, too.

Recommendations can also be calculated with respect to the current domain ontology, Figure 9. This is necessary if a user is regarding course materials from different courses at the same time.

However, the first recommendation rule, which reasons within one course will be more accurate because it has more fine–grained information about the course and thus on the learning process of a learner taking part in this course.

d) Reasoning Rules for User Modeling.: The Personal Reader requires only view information about the user's characteristics. Thus, for our example we employed a very simple user model: This user model traces the user's path in the learning environment and registers whenever the user has visited some learning resource. This information is stored in the user's profile, which is bound to RDF as shown in Figure 10.

From this information, we derive whether a particular user learned some concept. The rule in Figure 11 derives all learned concepts.

Similarly, it can be determined whether a learning object has been learned by a user.

C. A Personal Reader Instance: The Personal Publication Reader

Again, let us consider a scenario first for describing the idea of the Personal Publication Reader:

> Bob is currently employed as a researcher in a university. Of course, he is interested in making his publications available to his colleagues, for this he publishes all his publications at his insitute's Web page. Bob is also enrolled in a research project. From time to time, he is requested to notify the project coordination office about his new publications. Furthermore, the project coordination office maintains a member page where information about the members, their involvement in the project, research experience, etc. is maintained.

Can we simplify this process? And, furthermore, can we use this information to provide new, syndicated information? From the scenario, we may conclude that most likely the partners of a research project have their own Websites where they publish their research papers. In addition, information about the role of researchers in the project like "Bob is participating mainly in working group X, and working group X is strongly cooperating with working groups Y and Z" might be available. If we succeed in making this information available to machines to reason about, we can derive new information like: "This research paper of Bob is related to working group X, other papers of working group X on the same research aspects are A, B, and C, etc."

To realize a Personal Publication Reader (PR-R), we extract the publication information from the various websites of the partners in the REWERSE project: All Web-pages containing information about publications of the REWERSE network are periodically crawled and new information is automatically detected, extracted and indexed in the repository of semantic descriptions of the REWERSE network (see Section VIII-D). This information, together with extracted information on the project REWERSE, on people involved in the project, their research interests, etc., is used to provide more information on each publication: who has authored it, which research groups are related to this kind of research, which other publications are published by the research group, which other publications of the author are available, which other publications are on the similar research, etc. (see Section VIII-E)

D. Gathering Data for Semantic Web Applications

Each institute and organization offers access to its publications on the Web. However, each presentation is usually different, some use e.g. automatic conversions of *bibtex* or other files, some are manually maintained. Such a presentation is well suited for human consumption, but hardly usable for automatic processing. Consider e.g. the scenario that we are interested in all publications of REWERSE project members in the year 2003 which contain the word "personalization" in their title or abstract. To be able to formulate such queries and to generate personalized views on heterogeneously presented publications it is necessary to first have access to the publication data in a more structured form.

Fig. 7. Adaptation rule example

```
 FORALL Q quiz(Q) <-
   Q['http://www.w3.org/1999/02/22-rdf-syntax-ns#':type ->
         'http://ltsc.ieee.org/2002/09/lom-educational#':'Quiz']
FORALL Q, C concepts_of_Quiz(Q,C) <- quiz(Q) AND concept(C)
   AND  Q['http://purl.org/dc/elements/1.1/':subject -> C].
FORALL LO, Q quiz(LO, Q) <- EXISTS C (concepts_of_LO(LO,C)
   AND concepts_of_Quiz(Q,C)).
```

Fig. 8. Recommending a resource

```
 FORALL LO1, LO2 upperlevel(LO1,LO2) <-
   LO1['http://purl.org/dc/terms#':isPartOf -> LO2].
FORALL LO, U learning_state(LO, U, recommended) <-
   EXISTS UpperLevelLO (upperlevel(LO, UpperLevelLO)
   AND p_obs(UpperLevelLO, U, Learned)).
```

Fig. 9. Recommendation with respect to the current domain ontology

```
 FORALL C, C_DETAIL detail_concepts(C, C_DETAIL) <-
   C_DETAIL['http://www.w3.org/2000/01/rdf-schema#':subClassOf -> C]
   AND concept(C) AND concept(C_DETAIL).

FORALL LO, U learning_state(LO, U, recommended) <-
  EXISTS C, C_DETAIL (concepts_of_LO(LO, C_DETAIL)
     AND detail_concepts(C, C_DETAIL) AND p_obs(C, U, Learned) ).
```

Fig. 10. Storing information in the user’s profile

```
 <rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:j.0="http://semweb.kbs.uni-hannover.de/rdf/l3s.rdf#" >
 <rdf:Description rdf:about="http://semweb.kbs.uni-hannover.de/user#john">
   <rdf:type rdf:resource="http://hoersaal..../rdf/l3s.rdf#User"/>
   <j.0:hasVisited>http://java.sun.com/.../variables.html</j.0:hasVisited>
    ...
```

Fig. 11. Rule deriving all learned concepts

```
 FORALL C, U  p_obs(C, U, Learned) <-
   EXISTS LO (concepts_of_LO(LO, C) AND
   U['http://semweb.kbs.uni-hannover.de/rdf/l3s#':hasVisited ->LO]).
```

Fig. 12. Sample RDF output entry

```
<rdf:Description rdf:about="http://www.example.org/id/16">
  <rewerse:origin>University of Heraklion</rewerse:origin>
  <rewerse:title>Describing Knowledge Representation Schemes:
   A Formal Account</rewerse:title>
  <rewerse:author>
    <rdf:Seq>
      <rdf:li rdf:resource="#Giorgos Flouris" />
      <rdf:li rdf:resource="#Dimitris Plexousakis" />
      <rdf:li rdf:resource="#Grigoris Antoniou" />
    </rdf:Seq>
  </rewerse:author>
  <rewerse:year>2003</rewerse:year>
  <rewerse:link>ftp://ftp.ics.forth.gr/tech-reports/2003/
   2003.TR320.Knowledge_Representation_Schemes.pdf</rewerse:link>
  <rewerse:abstract>The representation and manipulation of knowledge
   has been drawing a great deal of attention since the early [...]
  </rewerse:abstract>
</rdf:Description>
```

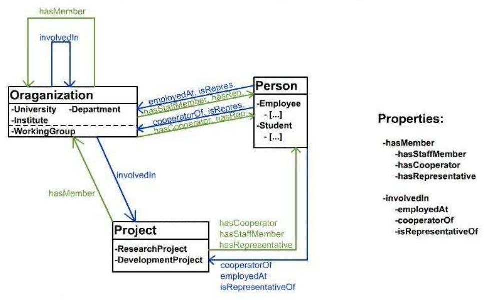


Fig. 13. Part of the Ontology on Researchers used in the Personal Publication Reader

Fig. 14. Example of a rule determining all authors of a publication

```
FORALL A, P all_authors(A, P) <-
  EXISTS X, R (
  P['http://.../rewerse#':author -> X]@'http:...#':publications
  AND X[R -> 'http://www.../author':A]@'http:...#':publications).
```

Fig. 15. Example rule determining the employer of a project member

```
FORALL A,I works_at(A, I) <-
   EXISTS A_id,X (name(A_id,A)
     AND ont:A_id[ont:involvedIn -> ont:I]@'http:...#':researcher
     AND ont:X[rdfs:subClassOf ->
               ont:Organization]@rdfschema('http:...#':researcher)
     AND ont:I[rdf:type -> ont:X]@'http:...#':researcher).
```

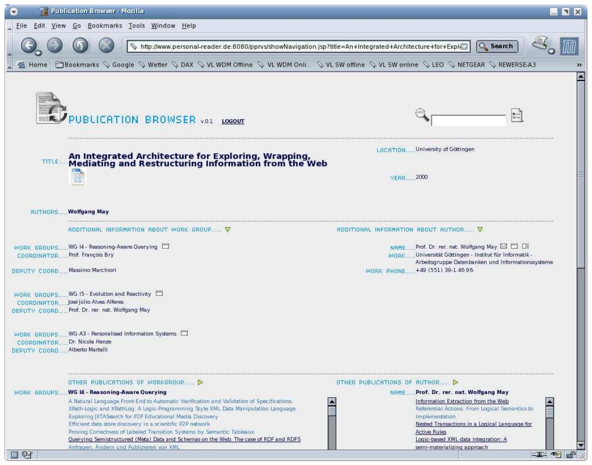
In Section VII we discussed data extraction from the Web and the Lixto methodology. Here, we apply Lixto to regularly extract publication data from all REWERSE members. As Figure 4 illustrates, the disks are Lixto wrappers that regulary (e.g. once a week) navigate to the page of each member (such as Munich, Hannover, Eindhoven) and apply a wrapper that extracts at least author names, publication titles, publication year and link to the publication (if available). Figure 3 illustrates the visual wrapper specification on the Munich page.

In the “XSL” components publication data is harmonized to fit into a common structure and an attribute “origin” is added containing the institution’s name. The triangle in Figure 4 represents a data integration unit; here data from the various institutions is put together and duplicate entries are removed. IDs are assigned to each publication in the next step. Finally, the XML data structure is mapped to a predefined RDF structure (this happens in the lower arc symbol in Figure 4) and passed on to the Personal Publication Reader as described below. A second deliverer component delivers the XML publication data additionally in RDF. One sample RDF output entry is depicted in Figure 12.

This Lixto application can be easily enhanced by connecting further Web sources. For instance, abstracts from `www.researchindex.com` can be queried for each publication lacking this information and joined to each entry, too. Moreover, using text categorization tools one can rate and classify the contents of the abstracts.

E. Content Syndication and Personalized Views

In addition to the extracted information on research papers that we obtain as described in the previous section, we collect the data about the members of the research project from the member’s corner of the REWERSE project. We have constructed an ontology for describing researchers and their envolvment in


Publication Browser - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.personal-reader.de:8080/pprvs/showNavigation.jsp?title=An+Integrated+Architecture+for+Expl
Search
Home Bookmarks Google Wetter DAX VL WDM Offline VL WDM Onli... VL SW offline VL SW online LEO NETGEAR REWERSE-A3
PUBLICATION BROWSER v.0.1 LOGOUT

TITLE...... An Integrated Architecture for Exploring, Wrapping, Mediating and Restructuring Information from the Web

LOCATION...... University of Göttingen

YEAR...... 2000

AUTHORS...... Wolfgang May

ADDITIONAL INFORMATION ABOUT WORK GROUP......

WORK GROUPS...... WG I4 - Reasoning-Aware Querying
COORDINATOR...... Prof. François Bry
DEPUTY COORD...... Massimo Marchiori

WORK GROUPS...... WG I5 - Evolution and Reactivity
COORDINATOR...... José Júlio Alves Alferes
DEPUTY COORD...... Prof. Dr. rer. nat. Wolfgang May

WORK GROUPS...... WG A3 - Personalised Information Systems
COORDINATOR...... Dr. Nicola Henze
DEPUTY COORD...... Alberto Martelli

ADDITIONAL INFORMATION ABOUT AUTHOR......

NAME...... Prof. Dr. rer. nat. Wolfgang May
WORK...... Universität Göttingen - Institut für Informatik - Arbeitsgruppe Datenbanken und Informationssysteme
WORK PHONE...... +49 (551) 39-1 46 96

OTHER PUBLICATIONS OF WORKGROUP......

WORK GROUPS...... WG I4 - Reasoning-Aware Querying
A Natural Language Front-End to Automatic Verification and Validation of Specifications.
XPath-Logic and XPathLog: A Logic-Programming Style XML Data Manipulation Language
Exploring JXTASearch for P2P Educational Media Discovery
Efficient data store discovery in a scientific P2P network
Proving Correctness of Labeled Transition Systems by Semantic Tableaux
Querying Semistructured (Meta) Data and Schemas on the Web: The case of RDF and RDFS
Anfragen, Ändern und Publizieren von XML

OTHER PUBLICATIONS OF AUTHOR......

NAME...... Prof. Dr. rer. nat. Wolfgang May
Information Extraction from the Web
Referential Actions: From Logical Semantics to Implementation
Nested Transactions in a Logical Language for Active Rules
Logic-based XML data integration: A semi-materializing approach

Fig. 16. Screenshot of the Personal Publication Reader

REWERSE. A part of this ontology can be seen in Figure 13

All the collected information is then used in a Personalization service which provides the end user with an interface for browsing publications of the REWERSE project, and having instantly access to further information on authors, the working groups of REWERSE, recommended related publications, etc.

The Personalization service of the PR-R uses, similar to the PR-eL, personalization rules for deriving new facts, and for determining recommendations for the user. As an example, the rule in Figure 14 determines all authors of a publication:

Further rules combine information on these authors from the researcher ontology with the author information. E.g. the rule in Figure 15 determines the employer of a project member, which might be a company, or a university, or, more generally, some instance of a subclass of an organization:

The screenshot in fig. 16 depicts the output of the visualization service of the PR-R.

By further exploiting the Web service architecture of the Personal Reader, it is possible to *link* to the PR other (reasoning) services, such as a personal sequencing service, implemented as a planner by exploiting the action metaphor, or making use of the non-monotonic reasoning functionality or the ECA paradigm and expressiveness for more advanced personalization functionality.

IX. Acknowledgement

This research has been carried out in connection with the Network of Excellence REWERSE[3] which strives for a (minimal) set of rule and reasoning languages for the Semantic Web.

X. Conclusions

This paper discusses recent approaches for shaping the logic layer of the Semantic Web, and for supporting approaches to personalization in the Semantic Web. We demonstrate approaches for rules and rule-languages in the logic layer of the Semantic Web. Special attention is devoted to the important aspects of evolution, updates and events, and their consequences for personalization and reasoning. Approaches to personalization via reasoning about actions is examplified for different scenarios.

Query- and transformation languages as well as Web data extraction for maintaining and constructing semantic descriptions are discussed. Finally, personalized Web systems making use of these reasoning techniques, semantic descriptions and extractions, are introduced.

[3]REWERSE - Reasoning on the Web, Network of Excellence founded in the 6th European Framework Programme, rewerse.net

References

[1] Alferes, J. J., Brogi, A., Leite, J. A., and Pereira, L. M. Evolving logic programs. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA'02)* (2002), S. Flesca, S. Greco, N. Leone, and G. Ianni, Eds., vol. 2424 of *LNCS*, Springer-Verlag, pp. 50–61.

[2] Alferes, J. J., Leite, J. A., Pereira, L. M., Przymusinska, H., and Przymusinski, T. C. Dynamic updates of non-monotonic knowledge bases. *The Journal of Logic Programming 45*, 1–3 (2000), 43–70. A shorter version appeared in "Principles of Knowledge Representation and Reasoning '98".

[3] Antoniou, G., Bikakis, A., and Wagner, G. A system for nonmonotonic rules on the web. In *Proc. of RuleML-2004* (2004), Springer LNCS.

[4] Antoniou, G., Billington, D., Governatori, G., and Maher, M. Representation Results for Defeasible Logic. *ACM Transactions on Computational Logic 2,2* (2002), 255–287.

[5] Antoniou, G., and van Harmelen, F. *A Semantic Web Primer*. MIT Press, 2004.

[6] Atzeni, P., and Mecca, G. Cut and paste. In *Proc. of PODS* (1997).

[7] Baldoni, M., Baroglio, C., Martelli, A., and Patti, V. Reasoning about interaction for personalizing web service fruition. In *Proc. of WOA 2003: Dagli oggetti agli agenti, sistemi intelligenti e computazione pervasiva* (Villasimius (CA), Italy, September 2003), G. Armano, F. De Paoli, A. Omicini, and E. Vargiu, Eds., Pitagora Editrice Bologna.

[8] Baldoni, M., Baroglio, C., Martelli, A., and Patti, V. Reasoning about self and others: communicating agents in a modal action logic. In *Proc. of ICTCS'2003* (2003), vol. 2841 of *LNCS*, Springer, pp. 228–241.

[9] Baldoni, M., Baroglio, C., and Patti, V. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review 22*, 1 (2004), 3–39.

[10] Baldoni, M., Baroglio, C., Patti, V., and Torasso, L. Reasoning about learning object metadata for adapting scorm courseware. In *AH 2004: Workshop Proceedings, Part I, EAW 2004: Engineering the Adaptive Web* (Eindhoven, Holland, August 2004), L. Aroyo and C. Tasso, Eds., CS-Report 04-18, Technische Universiteit Eindhoven, pp. 4–13.

[11] Baldoni, M., Giordano, L., Martelli, A., and Patti, V. An Abductive Proof Procedure for Reasoning about Actions in Modal Logic Programming. In *Proc. of NMELP'96* (1997), J. Dix et.al., Ed., vol. 1216 of *LNAI*, Springer-Verlag, pp. 132–150.

[12] Baldoni, M., Giordano, L., Martelli, A., and Patti, V. Programming Rational Agents in a Modal Action Logic. *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation 41*, 2-4 (2004), 207–257.

[13] Baral, C., and Son, T. C. Formalizing Sensing Actions - A transition function based approach. *Artificial Intelligence 125*, 1-2 (January 2001), 19–91.

[14] Bassiliades, N., Antoniou, G., and Vlahavas, I. A defeasible logic system for the semantic web. In *Principles and Practice of Semantic Web Reasoning* (2004), Springer LNCS 3208.

[15] Baumgartner, R., Eichholz, S., Flesca, S., Gottlob, G., and Herzog, M. Semantic Markup of News Items with Lixto. In *Annotation for the Semantic Web* (2003).

[16] Baumgartner, R., Flesca, S., and Gottlob, G. Declarative Information Extraction, Web Crawling and Recursive Wrapping with Lixto. In *Proc. of LPNMR* (2001).

[17] Baumgartner, R., Flesca, S., and Gottlob, G. Visual web information extraction with Lixto. In *Proc. of VLDB* (2001).

[18] Baumgartner, R., Flesca, S., Gottlob, G., and Herzog, M. Building dynamic information portals - a case study in the agrarian domain. In *Proc. of IS* (2002).

[19] Baumgartner, R., Gottlob, G., and Herzog, M. Lixto - Halfway to the Semantic Web. *OEGAI-Journal 1* (2003), 19–24.

[20] Baumgartner, R., Gottlob, G., Herzog, M., and Slany, W. Interactively Adding Web Service Interfaces to Existing Web Applications. In *Proc. of SAINT* (2004).

[21] Baumgartner, R., Herzog, M., and Gottlob, G. Visual programming of web data aggregation applications. In *Proc. of IIWeb-03* (2003).

[22] Beckett, D. Rdf/xml syntax specification. http://www.w3.org/TR/rdf-syntax-grammar/.

[23] Berger, S., Bry, F., and Schaffert, S. A Visual Language for Web Querying and Reasoning. In *Proceedings of Workshop on Principles and Practice of Semantic Web Reasoning, Mumbai, India (9th–13th December 2003)* (2003), vol. 2901 of *LNCS*.

[24] Berger, S., Bry, F., Schaffert, S., and Wieser, C. Xcerpt and visXcerpt: From Pattern-Based to Visual Querying of XML and Semistructured Data. In *Proceedings of 29th Intl. Conference on Very Large Data Bases, Berlin, Germany (9th–12th September 2003)* (2003).

[25] Berger, S., Bry, F., and Wieser, C. Visual Querying for the Semantic Web. In *Proceedings of 23rd International Conference on Conceptual Modeling, Shanghai, China (8th–12th November 2004)* (2004).

[26] Bergman, M. K. The deep web: Surfacing hidden value. BrightPlanet White Paper, http://www.brightplanet.com/technology/deepweb.asp.

[27] Berners-Lee, T., Hendler, J., and Lassila, O. The semantic web. *Scientific American* (May 2001).

[28] Berners-Lee, T., Hendler, J., and Lassila, O. The Semantic Web – A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American* (May 2001).

[29] Bonner, A. J., and Kifer, M. An overview of transaction logic. *Theoretical Computer Science 133* (1994).

[30] Brickley, D., and Guha, R. Rdf vocabulary description language 1.0: Rdf schema. http://www.w3.org/TR/rdf-schema/.

[31] Bry, F., Furche, T., Pătrânjan, P.-L., and Schaffert, S. Data Retrieval and Evolution on the (Semantic) Web: A Deductive Approach. In *Proceedings of Workshop on Principles and Practice of Semantic Web Reasoning, St. Malo, France (6th–10th September 2004)* (2004), REWERSE.

[32] Bry, F., and Pătrânjan, P.-L. Reactivity on the Web: Paradigms and Applications of the Language XChange. In *20th Annual ACM Symposium on Applied Computing (SAC'2005)* (2005).

[33] Bry, F., and Schaffert, S. Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In *Proceedings of International Conference on Logic Programming, Copenhagen, Denmark (29th July–1st August 2002)* (2002), vol. 2401 of *LNCS*.

[34] Bryson, J., Martin, D., McIlraith, S., and Stein, L. A. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web, 2002.

[35] Campione, M., and Walrath, K. The java tutorial, 2003. http://java.sun.com/docs/books/tutorial/.

[36] Castilho, M., Gasquet, O., and Herzig, A. Modal tableaux for reasoning about actions and plans. In *Proc. ECP'97* (1997), S. Steel, Ed., LNAI, pp. 119–130.

[37] Crescenzi, V., Mecca, G., and Merialdo, P. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases* (2001), pp. 109–118.

[38] DAML-S. http://www.daml.org/services/daml-s/0.9/. version 0.9, 2003.

[39] Davulcu, H., Yang, G., Kifer, M., and Ramakrishnan, I. Computational aspects of resilient data extraction from semistructured sources. In *Proc. of PODS* (2000).

[40] Dean, M., and Schreiber, G. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/.

[41] Dolog, P., Henze, N., Nejdl, W., and Sintek, M. The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies. In *Proccedings of the 3nd*

International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004) (Eindhoven, The Netherlands, 2004).

[42] EITER, T., FINK, M., SABBATINI, G., AND TOMPITS, H. Declarative update policies for nonmonotonic knowledge bases. In *Logics for Emerging Applications of Databases*, J. Chomicki, R. van der Meyden, and G. Saake, Eds. Springer-Verlag, 2003, ch. 3, pp. 85–129.

[43] EITER, T., FINK, M., SABBATINI, G., AND TOMPITS, H. Reasoning about Evolving Nonmonotonic Knowledge Bases. *ACM Transactions on Computational Logic* (2004). To appear.

[44] EITER, T., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. Combining answer set programming with description logics for the semantic web. In *Proceedings KR-2004* (2004), pp. 141–151. http://www.kr.tuwien.ac.at/staff/roman/semweblp/.

[45] EITER, T., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. Well-founded semantics for description logic programs in the semantic web. In *Proceedings RuleML 2004 Workshop, ISWC Conference, Hiroshima, Japan* (2004), Springer, pp. 81–97. http://www.kr.tuwien.ac.at/staff/roman/semweblp/.

[46] ETZIONI, O., CAFARELLA, M., DOWNEY, D., KOK, S., POPESCU, A., SHAKED, T., SODERLAND, S., WELD, D. S., AND YATES, A. Web-Scale Information Extraction in KnowItAll (Preliminary Results). In *Proceedings of the World Wide Web Conference 2004* (2004).

[47] FLESCA, S., MANCO, G., MASCIARI, E., RENDE, E., AND TAGARELLI, A. Web wrapper induction: a brief survey. *Journal of the ACM, 51(1)* (2004).

[48] GABBAY, D., AND PH.SMETS, Eds. *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, vol. III: Belief Change. Kluwer Academic, 1998.

[49] GELFOND, M., AND LIFSCHITZ, V. Classical negation in logic programs and disjunctive databases. In *In* New Generation Computing (1991), vol. 9, pp. 365–385.

[50] GELFOND, M., AND LIFSCHITZ, V. Representing action and change by logic programs. *Journal of Logic Programming 17* (1993), 301–321.

[51] GIACOMO, G. D., LESPÈRANCE, Y., AND LEVESQUE, H. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence 121* (2000), 109–169.

[52] GIORDANO, L., MARTELLI, A., AND SCHWIND, C. Dealing with concurrent actions in modal action logic. In *Proc. ECAI-98* (1998), pp. 537–541.

[53] GOTTLOB, G., AND KOCH, C. Monadic datalog and the expressive power of languages for Web Information Extraction. In *Proc. of PODS* (2002).

[54] GOTTLOB, G., AND KOCH, C. Monadic Datalog and the Expressive Power of Web Information Extraction Languages. *AI Communications Vol.17/2* (2004).

[55] GROSOF, B. N., HORROCKS, I., VOLZ, R., AND DECKER, S. Description logic programs: Combining logic programs with description logic. In *Twelfth International World Wide Web Conference* (Budapest, Hungary, May 2003).

[56] HENZE, N., AND HERRLICH, M. The Personal Reader: A Framework for Enabling Personalization Services on the Semantic Web. In *Proceedings of the Twelfth GI- Workshop on Adaptation and User Modeling in Interactive Systems (ABIS 04)* (Berlin, Germany, 2004).

[57] HENZE, N., AND KRIESELL, M. Personalization functionality for the semantic web: Architectural outline and first sample implementation. In *Proccedings of the 1st International Workshop on Engineering the Adaptive Web (EAW 2004), held at the Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)* (Eindhoven, The Netherlands, 2004). To appear.

[58] HENZE, N., AND NEJDL, W. A logical characterization of adaptive educational hypermedia. *New Review of Hypermedia 10*, 1 (2004).

[59] HERZOG, M., AND GOTTLOB, G. InfoPipes: A flexible framework for M-Commerce applications. In *Proc. of TES workshop at VLDB* (2001).

[60] HEYMANS, S., AND VERMEIR, D. Integrating semantic web reasoning and answer set programming. In Answer Set Programming, Advances in Theory and Implementation, Proc. 2nd Intl. ASP’03 Workshop, Messina, Italy (2003), pp. 194–208.

[61] HORROCKS, I., PATEL-SCHNEIDER, P., BOLEY, H., TABET, S., AND GROSOF, B. Swrl: A semantic web rule language combining owl and ruleml. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[62] HUCK, G., FANKHAUSER, P., ABERER, K., AND NEUHOLD, E. JEDI: Extracting and synthesizing information from the web. In *Proc. of COOPIS* (1998).

[63] KOWALSKI, R., AND SERGOT, M. A Logic-based Calculus of Events. *New Generation of Computing 4* (1986), 67–95.

[64] KUHLINS, S., AND TREDWELL, R. Toolkits for generating wrappers. In *Net.ObjectDays* (2002).

[65] KUSHMERICK, N., WELD, D., AND DOORENBOS, R. Wrapper induction for information extraction. In *Proc. of IJCAI* (1997).

[66] LAENDER, A. H., RIBEIRO-NETO, B. A., DA SILVA, A. S., AND TEIXEIRA, J. S. A brief survey of web data extraction tools. In *Sigmod Record 31/2* (2002).

[67] LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic* (2004). To appear. Available via `http://www.arxiv.org/ps/cs.AI/0211004`.

[68] LEVESQUE, H. J., REITER, R., LESPÉRANCE, Y., LIN, F., AND SCHERL, R. B. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming 31* (1997), 59–83.

[69] LEVY, A., AND ROUSSET, M.-C. Combining horn rules and description logics in carin. *Artificial Intelligence 104(1-2)* (1998), 165–209.

[70] LIU, L., PU, C., AND HAN, W. XWrap: An extensible wrapper construction system for internet information. In *Proc. of ICDE* (2000).

[71] LIU, Z., LI, F., AND NG, W. K. Wiccap Data Model: Mapping Physical Websites to Logical Views. In *Proceedings of the 21st International Conference on Conceptual Modelling (ER2002)* (Tempere, Finland, October 7-11 2002).

[72] LOBO, J., MENDEZ, G., AND TAYLOR, S. R. Adding Knowledge to the Action Description Language $\mathcal{A}$. In *Proc. of AAAI’97/IAAI’97* (Menlo Park, 1997), pp. 454–459.

[73] The mandarax project. http://www.mandarax.org.

[74] MAY, W., ALFERES, J. J., AND BRY, F. Towards generic query, update, and event languages for the semantic web. In *Principles and Practice of Semantic Web Reasoning (PPSWR)* (2004), no. 3208 in LNCS, Springer, pp. 19–33.

[75] MAY, W., HIMMERÖDER, R., LAUSEN, G., AND LUDÄSCHER, B. A unified framework for wrapping, mediating and restructuring information from the web. In *WWWCM* (1999), Sprg. LNCS 1727.

[76] MCCARTHY, J., AND HAYES, P. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence 4* (1963), 463–502.

[77] MCILRAITH, S., AND SON, T. Adapting Golog for Programming the Semantic Web. In *5th Int. Symp. on Logical Formalization of Commonsense Reasoning* (2001), pp. 195–202.

[78] MCILRAITH, S. A., SON, T. C., AND ZENF, H. Semantic Web Services. *IEEE Intelligent Systems* (March/April 2001), 46–53.

[79] MENG, X., WANG, H., LI, C., AND KOU, H. A schema-guided toolkit for generating wrappers. In *Proc. of WEBSA2003* (2003).

[80] MUSLEA, I., MINTON, S., AND KNOBLOCK, C. A hierarchical approach to wrapper induction. In *Proc. of 3rd Intern. Conf. on Autonomous Agents* (1999).

[81] NIEMELÄ, I., AND SIMONS, P. Implementation of the stable model and well-founded semantics for normal logic programs. In *In J. Dix, U. Furbach, and A. Nerode, editors*, Proc. 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-97) (1997), Springer, pp. 420–429.

[82] OWLS. http://www.daml.org/services/owl-s/. version 1.0, 2004.

[83] PATTI, V. *Programming Rational Agents: a Modal Approach in a Logic Programming Setting*. PhD thesis, Dipartimento

di Informatica, Università degli Studi di Torino, Italy, 2002. Available at `http://www.di.unito.it/˜patti/`.

[84] PRENDINGER, H., AND SCHURZ, G. Reasoning about action and change. a dynamic logic approach. *Journal of Logic, Language, and Information 5*, 2 (1996), 209–245.

[85] RAPOSO, J., PAN, A., ALVAREZ, M., HIDALGO, J., AND VINA, A. The Wargo System: Semi-Automatic Wrapper Generation in Presence of Complex Data Access Modes. In *Proceedings of DEXA 2002* (Aix-en-Provence, France, 2002).

[86] RIBEIRO-NETO, B., LAENDER, A. H. F., AND DA SILVA, A. S. Extracting semi-structured data through examples. In *Proc. of CIKM* (1999).

[87] The rule markup initiative. http://www.ruleml.org.

[88] SAHUGUET, A., AND AZAVANT, F. Building light-weight wrappers for legacy web data-sources using W4F. In *Proc. of VLDB* (1999).

[89] SCHAFFERT, S., AND BRY, F. Querying the Web Reconsidered: A Practical Introduction to Xcerpt. In *Proceedings of Extreme Markup Languages 2004, Montreal, Quebec, Canada (2nd–6th August 2004)* (2004).

[90] SCHWIND, C. B. A logic based framework for action theories. In *Language, Logic and Computation* (1997), J. Ginzburg et al., Ed., CSLI, pp. 275–291.

[91] SINTEK, M., AND DECKER, S. TRIPLE - an RDF Query, Inference, and Transformation Language. In *International Semantic Web Conference (ISWC)* (Sardinia, Italy, 2002), I. Horrocks and J. Hendler, Eds., LNCS 2342, pp. 364–378.

[92] WINSLETT, M. *Updating Logical Databases*. Cambridge University Press, 1990.

[93] WSDL. http://www.w3c.org/tr/2003/wd-wsdl12-20030303/. version 1.2, 2003.

[94] http://xcerpt.org.

Personalization for the Semantic Web*

Matteo Baldoni[1], Cristina Baroglio[1], and Nicola Henze[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino
c.so Svizzera 185, I-10149, Torino, Italy
E-mail: {baldoni,baroglio}@di.unito.it
[2] ISI - Semantic Web Group, University of Hannover,
Appelstr. 4, D-30167 Hannover, Germany
E-mail: henze@kbs.uni-hannover.de

Abstract. Searching for the meaning of the word "personalization" on a popular search engine, one finds twenty-three different answers, including *"the process of matching categorized content with different end users based on business rules ... upon page request to a Webserver"*, *"using continually adjusted user profiles to match content or services to individuals"*, and also *"real-time tailoring of displays, particularly Web pages, to a specific customer's known preferences, such as previous purchases"*. A little more generally, personalization is a process by which it is possible to give the user optimal support in accessing, retrieving, and storing information, where solutions are built so as to fit the preferences, the characteristics and the taste of the individual. This result can be achieved only by exploiting machine-interpretable semantic information, e.g. about the possible resources, about the user him/herself, about the context, about the goal of the interaction. Personalization is realized by an inferencing process applied to the semantic information, which can be carried out in many different ways depending on the specific task. The objective of this paper is to provide a coherent introduction into issues and methods for realizing personalization in the Semantic Web.

1 Introduction

Personalized information systems aim at giving the individual user optimal support in accessing, retrieving, and storing information. The individual requirements of the user are to be taken into account in such different dimensions like the current task, the goal of the user, the context in which the user is requesting the information, the previous information requests or interactions, the working process s/he is involved in, the level of expertise, the device s/he is using to display the information, the bandwidth and availability of the communication channel, the abilities (disabilities or handicaps) of the user, his/her time constraints, and many, many more. Different research disciplines have contributed to explore personalization techniques and to evaluate their usefulness within various application areas: E.g. hypertext research has studied personalization in the area of so-called adaptive hypertext systems, collaborative filtering research has investigated recommender systems, artificial intelligence techniques have been widely used to cluster Web data, usage data, and user data, reasoning and uncertainty management has been adopted to draw conclusions on appropriate system behavior, and so forth.

* This research has partially been funded by the European Commission and by the Swiss Federal Office for Education and Science within the 6th Framework Programme project REWERSE number 506779 (cf. http://rewerse.net). Matteo Baldoni and Cristina Baroglio have also been supported by MIUR Cofin 2003 "Logic-based development and verification of multi-agent systems (MASSiVE)" national project.

Many attempts have been done to apply personalization techniques to the *World Wide Web* as a natural extension of work on hypertext and hypermedia, however, the Web is an information space thought for human to human communication, while personalization requires software systems (broadly speaking "machines") to take part to the interaction and help. Such systems require *knowledge* to be expressed in a machine-interpretable format, which in the Web is not available. The development of languages for expressing information in a machine-processable form is characteristic of the *Semantic Web* initiative, as Tim Berners-Lee pointed out since 1998. Over this knowledge layer, the use of *inferencing mechanisms* is envisioned as a fundamental means for performing a content-aware navigation, producing an overall behavior that is closer to the user's intuition and desire. This is the reason why the Semantic Web is the most appropriate environment for realizing personalization. In other words, the Semantic Web is *deeply connected* to the idea of personalization in its very nature.

In the following we will see how the notion of personalization applies to the Semantic Web, overview the expectations, the mechanisms, the languages and tools, and set the state of the art. The paper is organized as follows. Section 2 introduces personalization as the key feature of the Semantic Web. Section 3 reports the state of the art in personalized Web systems, mainly based on the concept of "user model". Section 4 explains how WWW adaptive systems can take advantage of the Semantic Web. Conclusions follow.

2 Demands of personalization in the (Semantic) Web

The objective of the Semantic Web is a content-aware navigation and fruition of the resources. This means being able, by means of proper mechanisms, to identify those resources that better satisfy the requests not only on the basis of descriptive keywords but also on the basis of *knowledge*. There is, in fact, a general agreement that the use of knowledge increases the precision of the answers. Such a knowledge, as we will see, represents different things, information about the user, the user's intentions, the context. One of the key features that characterize the Semantic Web is that its answers are always personalized or adapted so to meet specific requirements. It will not be the case that the answer to a query about "book" will contain links to bookshops and links to travel agencies. This Web of knowledge is currently being built on top of the more traditional World Wide Web and requires the definition of proper languages and mechanisms. Let us now introduce a few basic concepts.

The first key concept is that of *user model*, that is a machine-interpretable representation of knowledge about the user. The user model, however, may contain different kinds of information; depending on what the user model contains, different reasoning technique might be necessary. Often the user model contains general information, e.g. age and education. In this case, in the tradition of works on personalization, the adaptation occurs at the level of information selection and, especially, presentation. Different users better understand different ways of explaining things. Choosing the best possible communication pattern is fundamental in application systems that supply a kind of information which, because of its nature, might be difficult to understand but that it is important for the user to comprehend. Think, for example, to health-care systems, where medical information is supplied to persons of different age and education. In order for this kind of task to be executed, it is necessary to enrich the data sources and the data itself with semantic information. To this aim, one of the greatest difficulties is to define adequate ontologies.

More and more frequently, however, the Semantic Web is not seen as an information provider but as a *service provider*. This is actually in the line with the latest view of the World Wide Web as a platform for sharing resources and services. We can divide services in two families: "world services" and "web services". A world service is, for instance, a shop, a museum, a restaurant, whose address, type and description is accessible over the Web. A Web service, instead, is a resource, typically a software device, that can be automatically retrieved and invoked over the Web, possibly by another service.

To begin with, let us consider services of the former kind, world services. The scenarios in which these services are considered adopt user models, in which a different kind of information is considered: the location of the user, which is supposed to vary along time. Typically this information is *not* supplied by the user but it is obtained by the system in a way that is transparent to him/her. In the simplest case, the user (a tourist or a person who is abroad for work) describes in a qualitative way a service of interest, as done with the regular Web browsers. The answer, however, contains only information about world services that are located nearby. The scenario can be made more complex if one adds the time dimension. In this case the user is not necessarily interested in a service that is available now, the system is requested to store the user's desire and alert the user whenever a matching event occurs, that refers to a service that is nearby. As an example, consider a user who loves classical ballet. He is traveling, and has just arrived at Moscow. After a couple of days he receives an SMS informing him that in the weekend Romeo and Juliet is going to be held at the Boljsoi Theater and that tickets are available. Notice that besides a different kind of information contained in the user model, also the mechanism by which personalization is obtained is very different from the previous case: here the answer changes according to the *context*, in this case given by the position of the user in space and time, and the answer is not always immediately subsequent the query. As we have seen, in fact, a *triggering mechanism* is envisioned that alerts the user whenever an event that satisfies the description occurs. The word "triggering mechanism" makes one think of a sort of reactive system, nevertheless, many alternatives might be explored and, in particular, inference mechanisms. Moreover, this approach is suitable also to a very different application domain, such as *ambient intelligence*, where appliances are the world services to be handled.

Strongly related to these topics, the recent success of decentralized applications has caused a growing attention towards decentralized approaches to user modeling. In this framework, the target applications include personal guides for navigation or ambient devices, integrated Web-sites (e.g. newspapers), portals (e.g. Yahoo), e-commerce Web-sites (e.g. Amazon), or recommender sites (e.g. MovieLens). In ubiquitous environments distributed sensors follow a user's moves and, based on the tasks typically performed by him/her, on preferences induced from history and on the specific characteristics of the given context, perform adaptation steps to the ambient-dependent features of the supported functionalities.

As a last observation, when the answer is time-delayed, as described, the descriptions of the services (or more in general, of the events) of interest are sometimes considered as part of the user model. In this case the user model does not contain general information about the user but a more specific kind of information. Alternatively, this can be seen as a configuration problem: I configure a personalized assistant that will warn me when necessary. It is interesting to observe that no-one considers these as queries. An example application is a personalized agenda: the idea is to use an automatic configuration system for filling the agenda of a tourist, taking into account his/her preferences and the advertisements of cultural events in the visited area as they are published. Indeed, filling the agenda could be considered as the topmost level of a system that also retrieves services triggered by events and biased by the user's location. This kind of systems should perform also personalization w.r.t. the device by which the user interacts with the system (mobile, laptop).

Many scenarios that refer to world services could naturally be extended so as to include *Web services*. In this case, the meaning of localization should be revised, if at all applicable, while the idea of combining services, as proposed in the case of the tourist agenda, should be explored with greater attention; Web service automatic composition is, actually, quite a hot topic as research in the field proves [20, 5]. Both, Web-service-based and ubiquitous computing, applications can be considered as conglomerates of independent, autonomous services developed by independent parties. Such components are not integrated at design time, they are integrated dynamically at run-time, according to the current needs. A frequently used metaphor is a free-market of services where the user buys a complex service, that is composed dynamically on the basis of the available (smaller) services. For example, an e-learning course can be assembled dynamically by composing learning objects stored in independent repositories. The composition is performed so as to satisfy the specific characteristics of the student. For instance, a vision-impaired student will be returned audio materials.

Another, orthogonal, case is the one in which the user model contains (or is accompanied by) the description of what the user would like to *achieve*. There are situations in which this description cannot be directly related to specific resources or services, but it is possible to identify (or compose) a set of resources so as to satisfy the user's *desires*. In this case a planning process is to be enacted. Sometimes besides the planning process other reasoning techniques are envisioned in order to supply a more complete support to the user. An application domain in which the goal-driven approach seems particularly promising is, once again, *e-learning*. In this case the goal is the learning goal of the user, that is to say a high-level description of the knowledge that s/he would like to acquire, and the plan contains the learning resources that the user should use for acquiring the desired expertise. The whole interaction with the user is supposed to be carried on through a browser. It is important to remark that students are not the only kind of users of this sort of systems. Also teachers should access them but with a different aim. For instance, a teacher might look for learning resources for a new course that s/he will teach. A new notion is, then, introduced, that of *role*. Not only user models contain general or specific information about the users' interests but they also contain the role that the user plays. Depending on the role, different views might be supplied by the system (personalization at the level of presentation) and different actions might be allowed. Rather than being just one of the many features from a user model, the role could, actually, be considered as orthogonal to it (the role is independent from the specific user). Beyond e-learning, the concept of role is useful in many application domains. In health care, there are patients and there are doctors and nurses. In tourism, there are tourists and there are travel agencies.

Another basic concept is that of *domain knowledge*. For understanding the meaning of this word, let us consider the intuitive application case of e-learning. Here the system needs to be supplied with a body of knowledge that not only contains the semantic description of the single learning resources, but it also contains definitions of *more abstract* concepts, not directly related to the courses and defined on the basis of other concepts. This knowledge is used to bias the construction of solutions that make sense from a *pedagogical* point of view. The use of a knowledge of this kind might be exported also to other application domains, whenever similar reasoning techniques are adopted.

Summarizing, the goal of personalization in the Semantic Web is to make easier the access to the right resources. This task entails two orthogonal processes: *retrieval* and *presentation*. Retrieval consists in finding or constructing the right resources when they are needed, either on demand or (as by the use of automatic updates) when the information arises in the network. Once the resources have been defined they are presented in the most suitable way to the user, taking into account his/her own characteristics and preferences. To these aims it is necessary to have a model of the user, that is, a representation of those characteristics according to which personalization will occur. It is also necessary to apply inferencing techniques which, depending on the task, might range from the basic ontology reasoning mechanisms supplied by Description Logics (like subsumption and classification) to the most various reasoning techniques developed in Artificial Intelligence.

3 Personalization in the World Wide Web

Personalization in the World Wide Web can be compared to creating individual views on Web data according to the special interests, needs, requirements, goals, access-context, etc. of the current beholder. The ideas and solutions for creating these individual views are manifold and require interdisciplinary engagement: human computer interaction specialists, e.g. for creating meaningful user interfaces with good usability rankings; artificial intelligence experts, e.g. for mining Web data, or for creating dynamic and accurate models of users; and software engineers for creating generic infrastructure for maintaining personalized views on Web data, and for sufficient user interaction support.

In this article, we focus on those aspects of personalization which aim at improving the selection, access and retrieval of Web resources. The creation of appropriate user interfaces and user awareness is out of scope of this article.

Definition 1 (Personalizing the access to Web data) *Personalizing the access to Web data defines the process of supporting the individual user in finding, selecting, accessing, and retrieving Web resources (or meaningful sub-sets of this process).*

With this definition, we can more precisely say that the process of personalization is a process of *filtering the access* to Web content *according to the individual needs and requirements of each particular user*. We can distinguish two different classes of filters: those filter which have been created for a certain *hypermedia system*, and those, which have been created for a *network of Web resources*. The difference between these filters is in the way how they treat the underlying document space: if they have precise information on the structure and relations between the documents (this means the hypertext system), or whether they use dynamics and networking effects in the Web in order to provide individual views on Web data.

The first class of filters has been investigated since the beginnings of the nineties of the last century under the topic of *Adaptive Hypermedia Systems*. The second belongs to *Web Mining* techniques, both Web usage and Web content mining. The personalized systems based on Web mining are often called *recommender systems*, which are in focus of research since the mid-nineties of the last century.

In the following, we describe techniques and methods for personalization in the field of adaptive hypermedia (see Section 3.1), and Web mining (see Section 3.2). Afterwards, we will summarize approaches to user modeling.

3.1 Adaptive Hypermedia Systems

An *adaptive hypermedia system* enlarges the functionality of a hypermedia system. It *personalizes* a hypermedia systems for the individual users: Each user has her or his individual view and individual navigational possibilities for working with the hypermedia system. A general definition of hypertext / hypermedia is given in [58]:

Definition 2 (Hypertext) *A set of nodes of text which are connected by links. Each node contains some amount of information (text) and a number of links to other nodes.*

Definition 3 (Hypermedia) *Extension of hypertext which makes use of multiple forms of media, such as text, video, audio, graphics, etc.*

Discussions on the definitions of hypertext can be found for example in [24, 47]. The terms hypertext and hypermedia are often synonymous [47]. Throughout this text, we use the term *hypermedia*. For a general, functionality-oriented definition of adaptive hypermedia systems, we follow the proposal of P. Brusilovsky [17].

Definition 4 (Adaptive hypermedia system) *"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user."*

The support of adaptive methods in hypermedia systems is advantageous if there is *one* common system which serves *many* users with *different* goals, knowledge, and experience, *and* if the underlying hyperspace is relatively *large* [17]. Adaptation of hypermedia systems is also an attempt to overcome the "lost in hyperspace problem" (for a discussion, see for example [47]). The user's goals and knowledge can be used for limiting the number of available links in a hypermedia system.

Techniques in Adaptive Hypermedia. As we have explained, a hypermedia system consists of documents which are connected by links. Thus, there are generally two aspects which can be adapted to the users: the *content* and the links. Let us begin with *content level adaptation*.

By adapting the content to a user, the document is tailored to the needs of the user, for example by hiding too specialized information or by inserting some additional explanations. According to [17], we can identify the following methods for content level adaptation:

– *Additional explanations*: Only those parts of a document are displayed to a user which fit to his goals, interest, tasks, knowledge, etc.
– *Prerequisite explanations*: Here the user model checks the prerequisites necessary to understand the content of the page. If the user lacks to know some prerequisites, the corresponding information is integrated in the page.
– *Comparative explanations*: The idea of comparative explanations is to explain new topics by stressing their relations to known topics.
– *Explanation variants*: By providing different explanations for some parts of a document, those explanations can be selected which are most suited for the user. This extends the method of prerequisite explanations.
– *Sorting*: The different parts of a document are sorted according to their relevance for the user.

The following techniques are used for implementing the above stated adaptation methods [17]:

– *Conditional text*: Every kind of information about a knowledge concept is broken into text parts. For each of these text parts, the required knowledge for displaying it to the user is defined.
– *Stretch text*: Some keywords of a document can be replaced by longer descriptions if the user's actual knowledge requires that.
– *Page or page fragment variants*: Here, different variants of whole pages or parts of them are stored.
– *Frame based technique*: This technique stores page and fragment variants into concept frames. Each frame has some slots which present the page or page fragments in a special order. Certain rules decide which slot is presented to the user.

Content level adaptation requires sophisticated techniques for improved presentation. The current systems using content level adaptation do so by enriching their documents with meta information about prerequisite or required knowledge, outcome, etc. The documents or fragments contained in these systems have to be written more than once in order to obtain the different explanations.

Link Level Adaptation. By using link level adaptation, the user's possibilities to navigate through the hypermedia system are personalized. The following methods show examples for adaptive navigation support:

– *Direct guidance*: Guide the user sequentially through the hypermedia system. Two methods can be distinguished, "next best" and "page sequencing" (or "trails"). The former provides a next-button to navigate through the hypertext. The latter generates a reading sequence through the entire hypermedia or through some part of it.
– *Adaptive sorting*: Sort the links of a document due to their "relevance" to the user. The relevance of a link to the user is based on the system's assumptions about him/her. Some systems sort links depending on their similarity to the present page. Or by ordering them according to the required prerequisite knowledge. These methods are known as "similarity sorting" and "prerequisite knowledge sorting".
– *Adaptive hiding*: Limit the navigational possibilities by hiding links to irrelevant information. Hiding of links can be realized by making them unavailable or invisible.
– *Link annotation*: Annotate the links to give the user hints about the content of the pages they point to. The annotation might be text, coloring, an icon, or

dimming. The most popular method for link annotation (in the educational area) is the so called "traffic light metaphor". Here the educational state of a link is estimated by the system with respect to the user's actual knowledge state. The link pointing to the page is then annotated by a colored ball. A *red* ball in front of a link indicates that the user lacks some knowledge for understanding the pages; thus the page is not recommended for reading. A *yellow* ball indicates links to pages that are not recommended for reading; this recommendation is less strict than in case of a red ball. A *green* ball is in front of links which lead to recommended pages. *Grey* balls give the hint that the content of the corresponding page is already known to the user. Variants in the coloring exist. A mix of traffic light metaphor and adaptive hiding is also used in some systems. For an evaluation about adaptive hiding and adaptive navigation we refer to [67].
– *Map annotation*: Here, graphical overviews or maps are adapted with some of the above mentioned methods.

Techniques for link level adaptation depend on the specific system and are, for example, discussed in [17]. Here the assumptions that the system makes about the user play an important role to decide what and how to adapt. Link level adaptation restricts the number of links and thus the number of navigational possibilities. It is useful to prevent the user from "getting lost in hyperspace". As in the case of content level adaptation, a description of the content of the documents is required for implementing the adaptation tasks.

Case Study: Adaptive Educational Hypermedia Systems Adaptive educational hypermedia systems (AEHS) have been developed and tested in various disciplines and have proven their usefulness for improved and goal-oriented learning and teaching. In this section, we propose a component-based logical description of AEHS, in contrast to the functionality-oriented definition 4. This component-based definition is motivated by Reiter's theory of diagnosis [62] which settles on characterizing systems, observations, and diagnosis in first-order logic (FOL). We decompose adaptive educational hypermedia systems into basic components, according to their different roles: Each adaptive (educational) hypermedia system is obviously a hypermedia system, therefore it makes assumptions about documents and their relations in a *document space*. It uses a *user model* to model various characteristics of individual users or user groups. During runtime, it collects *observations* about the user's interactions. Based on the organization of the underlying document space, the information from the user model and from the system's observation, the *adaptive functionality* is provided.

In this section, we will give a logic-based definition for AEHS. We have chosen first order logic (FOL) as it allows us to provide an abstract, generalized formalization. The notation chosen in this paper refers to [64]. The aim of this logic-based definition is to accentuate the main characteristics and aspects of adaptive educational hypermedia.

Definition 5 (Adaptive Educational Hypermedia System (AEHS)) *An Adaptive Educational Hypermedia System (AEHS) is a Quadruple*

$$(DOCS,\ UM,\ OBS,\ AC)$$

with

DOCS: Doc*ument* **S***pace: A finite set of first order logic (FOL) sentences with constants for describing documents (and knowledge topics), and predicates for defining relations between these constants.*
UM: U*ser* **M***odel: A finite set of FOL sentences with constants for describing individual users (user groups), and user characteristics, as well as predicates and rules for expressing whether a characteristic applies to a user.*
OBS: Obs*ervations: A finite set of FOL sentences with constants for describing observations and predicates for relating users, documents/topics, and observations.*
AC: A*daptation* **C***omponent: A finite set of FOL sentences with rules for describing adaptive functionality.*

The components "document space" and "observations" describe basic data (DOCS) and run-time data (OBS). The user model and adaptation components process this data, e.g. for estimating a user's preferences (UM), or for deciding about beneficial adaptive functionalities for a user (AC). A collection of existing AEHS, described according to this logic-based formalism, is reported in [36, 35]. In these works a characterization is given of the systems belonging to the first generation of AEHS (e.g. Interbook [18]), to the second generation of adaptive educational hypermedia systems (e.g. NetCoach [71] and KBS Hyperbook [34]), as well as of a recent system, which is also an authoring framework for adaptive educational hypermedia (AHA!2.0 [15]).

To make an example, let us then describe by the above formalism an AEHS, called Simple, having the following functionality. Simple can annotate hypertext-links by using the traffic light metaphor with two colors: red for non recommended, green for recommended pages. Later, we will extend this system to demonstrate the use (and the usefulness) of a domain model in an AEHS. Simple can be modeled by a quadruple $(DOCS_s, UM_s, OBS_s, AC_s)$, whose elements are defined as follows:

– $DOCS_s$: This component is made of a set of n constants and a finite set of predicates. Each of the constants represents a document in the document space (the documents are denoted by $D_1, D_2, \ldots, D_n$). The predicates define pre-requisite conditions, i.e. they state which documents need to be studied before a document can be learned, e.g. $preq(D_i, D_j)$ for certain $D_i \neq D_j$ means that D_j is a prerequisite for D_i. N.B.: This AEHS does not employ an additional knowledge model.
– UM_s: it contains a set of m constants, one for each individual user U_1, U_2, $\ldots$, U_m.

- OBS_s: A special constant (*Visited*) is used within the special predicate *obs* to denote whether a document has been visited: $obs(D_i, U_j, Visited)$ is the observation that a document D_i has been visited by the user U_j.
- AC_s: This component contains constants and rules. One constant (*Recommended_for_reading*) is used for describing the values of the "learning_state" of the adaptive functionality, two constants (*Green_Icon* and *Red_Icon*) for representing values of the adaptive functionality. The learning state of a document is described by a set of rules of kind:

$$\begin{aligned}&\forall U_i \forall D_j (\forall D_k preq(D_j, D_k) \Longrightarrow \\ &\quad obs(D_k, U_i, Visited)) \Longrightarrow \\ &\qquad learning\_state(D_j, U_i, Recommended\_for\_reading)\end{aligned}$$

  This component contains also a set of rules for describing the adaptive link annotation with traffic lights. Such rules are of kind:

$$\begin{aligned}&\forall U_i \forall D_j learning\_state(D_j, U_i, Recommended\_for\_reading) \\ &\quad \Longrightarrow document\_annotation(D_j, U_i, Green\_icon)\end{aligned}$$

  or of kind:

$$\begin{aligned}&\forall U_i \forall D_j \neg learning\_state(D_j, U_i, Recommended\_for\_reading) \\ &\quad \Longrightarrow document\_annotation(D_j, U_i, Green\_icon)\end{aligned}$$

We can extend this simple AEHS by using a *knowledge graph* instead of a domain graph. The system, called Simple1, is able to give a more differentiated traffic light annotation to hypertext links than Simple. It is able to recommend pages (green icon), to show which links lead to documents that will become understandable (dark orange icon), which might be understandable (yellow icon), or which are not recommended yet (red icon). As in the previous case, let us represent Simple1 by a quadruple ($DOCS_{s1}$, UM_{s1}, OBS_{s1}, AC_{s1}):

- $DOCS_{s1}$: The document space contains all axioms of the document space of Simple, $DOCS_s$, but it does not contain any of the predicates. In addition, it contains a set of s constants which name the knowledge topics $T_1, T_2, \ldots, T_s$ in the knowledge space. It also contains a finite set of predicates, stating the learning dependencies between these topics: $depends(T_j, T_k)$, with $T_j \neq T_k$, means that topic T_k is required to understand T_j.
  The documents are characterized by predicate *keyword* which assigns a non-empty set of topics to each of them, so $\forall D_i \exists T_j keyword(D_i, T_j)$, but keep in mind that more than one keyword might be assigned to a same document.
- UM_{s1}: The user model is the same as in Simple, plus an additional rule which defines that a topic T_i is assumed to be learned whenever the corresponding document has been visited by the user. To this aim, Simple 1 uses the constant *Learned*.
  The rule for processing the observation that a topic has been learned by a user is as follows (*p_obs* is the abbreviation for "processing an observation"):

$$\begin{aligned}&\forall U_i \forall T_j (\exists D_k keyword(D_k, T_j) \wedge obs(D_k, U_i, Visited) \\ &\quad \Longrightarrow p\_obs(T_j, U_i, Learned)\end{aligned}$$

- OBS_{s1}: Are the same as in Simple.
- AC_{s1}: The adaptation component of Simple1 contains two further constants (w.r.t. Simple), representing new values for the learning state of a document. Such constants are: *Might_be_understandable* and *Will_become_understandable* (the meaning is intuitive). Two more constants are added for representing new values for adaptive link annotation. They are: *Orange_Icon* and *Yellow_Icon*. Such constants appear in the rules that describe the *educational state* of a document, reported hereafter.
  The first rule states that a document is recommended for learning if *all* the prerequisites to the keywords of this document have already been learnt:

$$\begin{aligned}&\forall U_i \forall D_j (\forall T_k keyword(D_j, T_k) \Longrightarrow \\ &\quad (\forall T_l depends(T_k, T_l) \Longrightarrow p\_obs(T_l, U_i, Learned) \\ &\qquad \Longrightarrow learning\_state(D_j, U_i, Recommended\_for\_reading)))\end{aligned}$$

  The second rule states that a document might be understandable if at least some of the prerequisites have already been learnt by this user:

$$\begin{aligned}&\forall U_i \forall D_j (\forall T_k keyword(D_j, T_k) \Longrightarrow \\ &\quad (\exists T_l depends(T_k, T_l) \Longrightarrow \\ &\qquad p\_obs(T_l, U_i, Learned) \\ &\qquad \wedge \neg learning\_state(D_j, U_i, Recommended\_for\_reading) \\ &\qquad\quad \Longrightarrow learning\_state(D_j, U_i, Might\_be\_understandable)))\end{aligned}$$

  The third rule entails that a document will become understandable if the user has some prerequisite knowledge for at least one of the document's keywords:

$$\begin{aligned}&\forall U_i \forall D_j (\exists T_k keyword(D_j, T_k) \Longrightarrow \\ &\quad (\exists T_l depends(T_k, T_l) \Longrightarrow \\ &\qquad p\_obs(T_l, U_i, Learned) \\ &\qquad \wedge \neg learning\_state(D_j, U_i, Might\_be\_understandable) \\ &\qquad\quad \Longrightarrow learning\_state(D_j, U_i, Will\_become\_understandable)))\end{aligned}$$

  Four rules describe the adaptive link annotation:

  1) $\forall U_i \forall D_j learning\_state(D_j, U_i, Recommended\_for\_reading)$
     $\Longrightarrow document\_annotation(D_j, U_i, Green\_Icon)$
  2) $\forall U_i \forall D_j learning\_state(D_j, U_i, Will\_become\_understandable)$
     $\Longrightarrow document\_annotation(D_j, U_i, Orange\_Icon)$
  3) $\forall U_i \forall D_j learning\_state(D_j, U_i, Might\_be\_understandable)$
     $\Longrightarrow document\_annotation(D_j, U_i, Yellow\_Icon)$
  4) $\forall U_i \forall D_j \neg learning\_state(D_j, U_i, Recommended\_for\_reading)$
     $\Longrightarrow document\_annotation(D_j, U_i, Red\_Icon)$

Discussion: Why a logical characterization of adaptive (educational) hypermedia is needed. With Brusilovsky's definition of adaptive hypermedia, we can describe the general functionality of an *adaptive* hypermedia system, and we can compare which kind of adaptive functionality is offered by such a system.

In the literature, we can find reference models for adaptive hypermedia, e.g. the AHAM Reference Model [16], or the Munich Reference Model [43]. Both the AHAM and Munich Reference models extend the Dexter Hypertext Model [31], and provide a framework for describing the different components of adaptive hypermedia systems. In both cases, the focus is posed on process modeling and on the engineering of adaptive hypermedia applications, so we can say that these models are *process-oriented*.

However, a formal description of adaptive educational hypermedia, which allows for a system-independent characterization of the adaptive functionality, is still missing. Currently, we cannot answer a request like the following: "I want to apply the adaptive functionality X in my system. Tell me what information is required with the hypermedia-documents, which interactions at runtime need to be monitored, and what kind of user model information and user modeling is required". At the moment, we can only describe the functionality with respect to a *specific* environment, which means we can describe the functionality only in terms of the system that implements it. We cannot compare different implementations nor can we benchmark adaptive systems. A benchmark of adaptive systems would require at least a comparable initial situation, observations about a user's interactions with the system during some defined interaction period, before the *result* of the system is returned, the *adaptive functionality* as well as the changes in the user model.

The logical definition of adaptive educational hypermedia given here focuses on the components of these systems, and describes which kind of processing information is needed from the underlying hypermedia system (the document space), the runtime information which is required (observations), and the user model characteristics (user model). The adaptive functionality is then described by means of these three components, or more precisely: how the information from these three components, the static data from the document space, the runtime-data from the observations, and the processing-data from the user model, is used to provide the adaptive functionality. The aim of this logical definition of adaptive educational hypermedia is to provide a *language* for describing the adaptive functionality, to allow comparison of adaptive functionality in a well-grounded way, and to enable the re-use of an adaptive functionality in different contexts and systems.

There is, actually, a need for a formalism expressing adaptive functionalities in a system-independent and re-usable manner, which allows their application in various contexts. In the educational context, a typical scenario where re-usable adaptive functionality is required would be: Imagine a learner who wants to learn a specific subject. The learner registers to some learning repository, which stores learning objects. According to his/her current learning progress, some of the learning objects which teach the subject s/he is interested in are useful, some of them require additional knowledge that the learner does not have so far (in accordance to his/her user model), and some might teach the subject only on the surface and are too easy for this learner. This kind of situation has been studied in adaptive educational hypermedia in many applications, and with successful solutions. However, these solutions are specific to certain adaptive hypermedia applications, and are hardly generalizable for re-use in different applications. Another reason why the adaptive functionality is not re-usable today is related to the so-called *open corpus problem* in adaptive (educational) hypermedia [33, 19], which states that currently, adaptive applications work on a fixed set of documents which is defined at the design time of the system, and directly influences the way adaptation is implemented, e.g. that adaptive information like "required prerequisites" is coded on this fixed set of documents.

3.2 Web Mining

In contrast to the approach in adaptive hypermedia, personalization with aid of Web mining does not work on such well-defined corpora like a hypertext system. Instead, it uses effects and dynamics in the network structure in order to detect (virtual) relations between Web resources.

The World Wide Web is seen as the *Web graph*. In this graph, Web resources are the nodes, and links between the Web resources are the edges. NB: as it is practically impossible to create a complete snapshot of the World Wide Web at a certain time point, this Web graph is not a completely known structure. On the contrary, in the case of adaptive hypermedia systems, the underlying hypermedia graph models completely the hypertext.

The approaches in Web Mining-based personalization are centered around detecting relations between Web resources. These relations can be *existing relations*, this means hyperlinks between Web resources, or *virtual relations*, this means that two or more Web resources are related to each other but are not connected via some hyperlink. These existing or virtual relations between Web resources are *mined* on basis of the Web graph. We can distinguish two main approaches for detecting the relations: *Mining based on the content* of the Web resources, or *mining based on the usage* of the Web resources. The two approaches can of course be combined.

Normally, Web Mining-based personalization has no external models like domain or expert models, as those used in adaptive hypermedia, but instead create dynamic models which grow with the number of Web resources integrated into the model.

Recommendation Techniques for Web Mining. In the following, we summarize major recommendation techniques according to Burke [21]. We can distinguish between *content-based*, *collaborative*, *demographic*, *utility-based*, and *knowledge-based* recommendations. Let U and I be respectively a set of users and a set of items, and $\mathcal{U}$ denotes an individual user. Let us outline these techniques:

– *Content-based recommendation*:

- each user is assumed to operate independently of other users;
- recommendations can exploit information derived from document contents;
- The system builds user models in the following way: initially, users apply *candidate profiles* against their own preferences. For example, a candidate user profile for the rating of today's news article is presented, the user can accept or reject the ratings for the articles. The profile is maintained by exploiting keywords and content descriptors which contribute to the rating of each article.
- The quality of the learnt knowledge is measured against the classical measures of Information Retrieval, i.e. precision and recall (see e.g. [4]).
- The typical background consists of features of items in I, the typical input to the mining process consists of the user's ratings of some items in I. A learning process is enacted that generates a classifier fitting the user's preferences, expressed by the ratings. The constructed classifier is applied to all the items in I, for finding out which might be of interest.
- limitations:
  - ∗ as in all inductive approaches, items must be machine-parsable or with assigned attributes;
  - ∗ only recommendations based on what the user has already seen before (and indicated to like) can be taken into account but negative information is as well important;
  - ∗ stability vs. plasticity of recommendations;
  - ∗ no filtering based on quality, style, or point-of-view (only based on *content*;

– *Collaborative recommendations (social information filtering)*:

- This technique is basically a process of "word-of-mouth", in fact the items are recommended to a user based upon values assigned by other people with *similar taste*. The underlying hypothesis is that people's tastes are not randomly distributed: there are general trends and patterns within the taste of a person as well as between groups of people. Also in this case a *user model* is to be built. To this aim the users are initially required to explicitly rank some sample objects.
- The input used for computing the predictions is a set of "Ratings of *similar users*", where the similarity is measured on the basis of the user profile values.
- The mining process begins with the identification of those users in U that result similar to $\sqcap$, and extrapolates the possible interests and likings of the user at issue from the ratings that similar users gave to items in I.
- Limitations:
  - ∗ a critical mass of users is required before the system can make recommendations;
  - ∗ how to get the first rating of a new object?
  - ∗ stability vs. plasticity of recommendations.

Demographic recommendation, utility-based recommendation and knowledge-based recommendation are variants which require additional data about the user beyond rating of items:

– *Demographic recommendations*
  In this case, demographic information about all the users in U is exploited: similarly to the previous case, the users that are close to $\mathcal{U}$ are identified, but in this case similarity is computed on the demographic data. Then, the ratings of these users on items in I are used to produce recommendations to the user at issue.

– *Utility-based recommendations*
  In this case the preferences of $\mathcal{U}$ are coded by a *utility function*, which is applied to all the items in I for defining recommendations.

– *Knowledge-based recommendations*
  The knowledge-based approach to recommendation works on a description of the user's needs and on a body of knowledge that describes how items can meet various needs. An inferencing process is used to match the description of the user's needs with the items that can help the user and, thus, are to be recommended.

Case Study: Web Usage Mining in an online shop. In this case study, we will see how we can improve selling strategies in an artificial online shop. Our online shop sells a small variety of products. Our goal is to find out which items are commonly purchased together in order to make for example some selected frequent-customers special bundle-offers which are likely to be in their interest.

To detect relations between data items, the concept of *association rules* can be used. Association rules aim at detecting *uncovered relations* between data items, this means relationships which are not inherent in the data like functional dependencies, and normally do not necessarily represent a sort of causality or correlation between the items. A database in which an association rule is to be found is viewed as a set of tuples: each tuple contains a set of items; the items represent the items purchased, and the tuples denote the list of items purchased together. For a definition of association rules, we follow [26]:

Definition 6 (Association Rule) *Given a set of items $I = \{I_1, I_2, \ldots, I_m\}$ and a database of transactions $D = \{t_1, t_2, \ldots, t_n\}$ where $t_i = \{I_{i1}, I_{i2}, \ldots I_{ik}\}$ and $I_{ij} \in I$, an* **association rule** *is an implication of the form $X \Longrightarrow Y$, where $X, Y \subset I$ are sets of items classed* itemsets *and $X \cap Y = \emptyset$.*

To identify the "important" association rules, the two measures *support* and *confidence* are used (see [26]):

Definition 7 (Support) *The* **support (s)** *for an association rule $X \Longrightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$.*

$$\textbf{support}(X \Longrightarrow Y) = \frac{|\{t_i \in D : X \cup Y \subset t_i\}|}{|D|}$$

Definition 8 (Confidence / Strength) *The* **confidence** *or* **strength** (α) *for an association rule $X \Longrightarrow Y$ is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X.*

$$\textbf{confidence}(X \Longrightarrow Y) = \frac{|\{t_i \in D : X \cup Y \subset t_i\}|}{|\{t_i \in D : X \subset t_i\}|}$$

The support measures how often the rule occurs in the database, while the confidence measures the strength of a rule. Typically, large confidence values and smaller support values are used, and association rules are mined which satisfy at least a *minimum support* and a *minimum confidence*.

The hard part in the association rule mining process is to detect the high-support (or *frequent*) item-sets. Computationally less costly is then the checking of the confidence. Algorithms for uncovering frequent item-sets exist in the literature [26], most prominent is the Apriori-algorithm [1], which uses the property of frequent itemsets that all subset of a frequent itemset must be frequent, too.

Example: An online Book Shop This (artificial) online book shop sells five different books: Semantic Web, Winnie the Pooh, Data Mining, Faust, and Modern Statistics.

Transaction	Items
t1	Semantic Web, Winnie the Pooh, Data Mining
t2	Semantic Web, Data Mining
t3	Semantic Web, Faust, Data Mining
t4	Modern Statistics, Semantic Web
t5	Modern Statistics, Faust

Customer X is a very good customer, and to tighten the relationship to customer X, we want to make a personal and attractive offer. We see him ordering a book on "Semantic Web". Which bundle offer might be interesting for him? Which book shall we offer to a reduced price: Winnie the Pooh, Data Mining, Faust, or Modern Statistics? We are looking for association rules which have a minimum-support of 30% and a confidence of 50%. The association rules we are interested in are thus :

Semantic Web $\Longrightarrow$ **Data Mining** support: 60%, confidence: 75 %
Semantic Web $\Longrightarrow$ **Faust** support: 20%, confidence: 25%
Semantic Web $\Longrightarrow$ **Winnie the Pooh** support: 20%, confidence: 25 %
Semantic Web $\Longrightarrow$ **Modern Statistics** support: 20%, confidence: 25 %

An often seen pattern is that the books "Semantic Web" and "Data Mining" are bought together, and the association rule "Semantic Web $\Longrightarrow$ Data Mining" satisfies the minimum support of 30%. In 60% of the cases in which customers bought the book "Semantic Web", they also bought the book "Data Mining" (confidence: 60%). Thus, we decide to offer our valuable customer the book "Data Mining" in a personal offer for an attractive price.

NB: The general "association rule problem" is to mine association rules which satisfy a given support and confidence; in the above example, we simplify the approach by asking whether a certain item is obtained in some association rule.

3.3 User Modeling

In a user model, a system's estimations about the preferences, often performed tasks, interests, and so forth of a specific end user (or group of users) are specified (in the following, we will only refer to "the user" wherever a single user a sufficient homogeneous group of users can be meant). We can distinguish between the *user profile* and the *user model*. A *User profile* provides access to certain characteristics of a user. These characteristics are modeled as attributes of the user. Thus, a user profile of user $\mathcal{U}$ gives the instantiations of attributes for $\mathcal{U}$ at a certain timepoint t. Instead, the task of the *user model* is to ascertain the values in the user profile of a user $\mathcal{U}$. Thus, the user model must provide updating and modification policies of the user profile, as well as instructions to detect and evaluate incidents which can lead to update or modification processes. Methods for drawing appropriate conclusions about the incidents must be given, as well as mechanisms for detecting discrepancies in the modeling process. Advanced user modeling approaches also provide mechanisms for dealing with uncertainty in the observations about a user, appropriate error detection mechanisms, and can prioterize the the conclusion on observed incidents.

A very simple user profile identifies all the pages that a user $\mathcal{U}$ has visited, therefore, it is a set of couples:

$$(\mathcal{P}, visited)$$

A simple user model which can create this-like user profiles contains the following rule for interpreting incidents:

"if $\mathcal{U}$ visits page $\mathcal{P}$ then insert $(\mathcal{P}$, visited) into the user profile of $\mathcal{U}$"

An extension of this simple user model is to recognize the observation that a user $\mathcal{U}$ has bookmarked some page $\mathcal{P}$ and note this in the user profile:

"if $\mathcal{U}$ bookmarks page $\mathcal{P}$ then insert $(\mathcal{P}$, important) into the user profile of $\mathcal{U}$"

We will not go into detail on user modeling in this article (for more in-depth information refer to [41]). But even from this simple user models above, we can see that interpretation about the user interactions is not at all an easy task. E.g. if we observe a user $\mathcal{U}$ bookmarking a page $\mathcal{P}$: How can we distinguish that $\mathcal{U}$ has stored this page for future reference based on the content of the page from the fact that $\mathcal{U}$ stored this page only because he liked the design of the page? Can we really be sure that bookmarking expresses favor for a page in contrast to denial? Appropriate mechanisms for dealing with uncertainty in the observations about the user, and for continuous affirmation of derived conclusions are essential for good user models (a good reference for studying numerical uncertainty management in user modeling is e.g. given in [38]).

User modeling approaches for *Adaptive Hypermedia* can take advantage of the underlying hypermedia structure or the domain models associated with the hypermedia system. Task models, expert models, or other, external models are used to model the user with respect to this external model. This approach is called *overlay modeling* [30]. As an example, for educational hypermedia systems, the learner's state of knowledge is described as a subset of the expert's knowledge of the domain, hence the term "overlay". Student's lack of knowledge is derived by comparing it to the expert's knowledge.

The critical part of overlay modeling is to find the initial knowledge estimation. The number of observations for estimating the knowledge sufficiently well must be small. In addition, a student's misconceptions of some knowledge concepts can not be modeled. A great variety of approaches for user modeling is available, see e.g. [42, 69]

User Modeling for Web Mining For Web Mining, the absence of a structured corpus of documents leads to different approaches for user modeling. An interest and/or content-profile of a user is generated (with the aid of classification or clustering techniques from machine learning) based on observations about the user's navigation behavior. A *stereotype user modeling* approach [63] classifies users into *stereotypes*: Users belonging to a certain class are assumed to have the same characteristics. When using stereotype user modeling, the following problem can occur: the stereotypes might be so specialized that they become obsolete (since they consist of at most one user), or a user cannot be classified at all.

Discussion The user modeling process is the core of each personalization process, because here the system's estimations about the user's needs are specified. If the system identifies the needs not correctly, the personalization algorithms –regardless how good they are– will fail to deliver the expected results for this erroneous modeled user.

3.4 Conclusion: Personalization in the World Wide Web

To develop systems which can filter information according to the requirements of the individual, which can learn the needs of users from observations about previous navigation and interaction behavior, and which can continuously adapt to the dynamic interests and changing requirements is still one of the challenges for building smart and successful Web applications. Although the necessity to "support the users in finding what they need at the time they want" is obvious, building and running personalized Web sites is still a cost-intensive venture which sometimes underachieves [40].

Looking at the techniques in adaptive hypermedia, we can see that re-usability of these techniques is still an unsolved problem. We require a formalism expressing adaptive functionality in a system-independent and re-usable manner, which allows us to apply this adaptive functionality in various contexts, as it has been done e.g. for the adaptive educational hypermedia systems (see Section 3.1). Another reason why adaptive functionality is not re-usable today is related to the so-called *open corpus problem* in adaptive hypermedia, which states that currently, adaptive applications work on a fixed set of documents which is defined at the design time of the system, and directly influences the way adaptation is implemented, e.g. that adaptive information like "required prerequisites" is coded on this fixed set of documents. The introduction of standards for describing such metadata is a step forwards - and is currently undertaken in the Semantic Web.

Looking at the personalization techniques based on Web mining, we can see that the filtering techniques (content-based, collaborative-based, demographic-based, utility-based, knowledge-based, or others) are limited as they require a critical mass of data before the underlying machine learning algorithms produce results of sufficient quality. Explicit, machine-readable information about single Web resources as given in the Semantic Web could be used for improving the quality of the input data for the algorithms.

4 Personalization for the Semantic Web

Functionalities for performing personalization require a machine-processable knowledge layer that is not supplied by the WWW. In the previous section we have studied techniques for developing adaptive systems in the WWW with all the difficulties and limitations brought by working at this level. Let us now see how adaptive systems can evolve benefiting of the Semantic Web. In particular, since the capability of performing some kind of inferencing is fundamental for obtaining personalization, let us see how the introduction of machine-processable semantics makes the use of a wide variety of *reasoning techniques* possible, thus widening the range of the forms that personalization can assume.

4.1 An overview

The idea of exploiting reasoning techniques for obtaining adaptation derives from the observation that in many (Semantic Web) application domains the goal of the user and the interaction occurring with the user play a fundamental role. Once the goal to be achieved is made clear, the system strives for achieving it, respecting the constraints and the needs of the user and taking into account his/her characteristics. In this context, the ability of performing a semantic-based retrieval of the necessary resources, that of combining the resources in a way that satisfies the user's goals, and, if necessary, of remotely invoking and monitoring the execution of a resource, are fundamental. All these activities can be performed by adopting automated reasoning techniques. To make an example, suppose that, for some reason, a student must learn something about the Semantic Web for a University course. Suppose that the student has access to a repository of educational resources that does not contain any material under the topic "Semantic Web". Let us suppose, however, that the repository contains a lot of information about XML-based languages, knowledge representation, ontologies, and so forth: altogether this information gives knowledge about the Semantic Web, the problem is retrieving it. A classical search engine would not be able to do it, unless the word "Semantic Web" is explicitly contained in the documents. This result can be obtained only by a system that is able to draw as an inference the fact that all these topics are elements of the Semantic Web.

In the Semantic Web every new feature or functionality is built as a new layer that stands on top of the previous ones. Tim Berners-Lee has described this process and structure as the "Semantic Web Tower". In this representation reasoning belongs to the logic and proof layers that lay on the ontology layer. This vision allows the Semantic Web to be developed incrementally.

Data on the Web is basically considered as the set of the available Web resources, each identified by a URI (uniform resource identifier). Such resources are mainly represented by plain XML (eXtensible Markup Language) descriptions. XML stands at the bottom of the tower. It allows a Web document to be written in a structured way, exploiting a user-defined vocabulary. It is perfect as a data interchange format, however, it does not properly supply any semantic information. Sometimes, when the domain is very closed and controlled, the tags can be considered as being associated with a meaning but the solution is risky and the application as such cannot be safely extended.

Semantic annotation of data is done by means of RDF (Resource Description Framework). RDF [59] is the basic Semantic Web (XML-based) language for writing simple statements about Web resources. Each statement is a binary predicate that defines a relation between two resources. These predicates correspond to logical facts. Given semantically-annotated data it is possible to perform some kinds of reasoning. In particular, some query languages have been developed that allow the automatic transformation of RDF-annotated data. Two of the main query languages that are used to transform data encoded in RDF are TRIPLE and RDQL. They are both quite simple in the inferencing that they allow.

TRIPLE [66] is a rule language for the Semantic Web which is based on Horn logic and borrows many basic features from F-Logic but is especially designed for querying and transforming RDF models. In contrast to procedural programming languages, such as C or Java, it is a declarative language which shares some similarities with SQL or Prolog. TRIPLE programs consist of facts and rules, from which it is possible to draw conclusions for answering queries. The language exploits reasoning mechanism about RDF-annotated information resources; translation tools from RDF to TRIPLE and vice versa are provided. An RDF statement, i.e. a "triple", is written as `subject[predicate -> object]`. RDF *models* are explicitly available in TRIPLE: statements that are true in a specific model are written as "@model". Connectives and quantifiers (e.g. `AND`, `OR`, `NOT`, `FORALL`, `EXISTS`) for building logical formulae from statements are allowed as usual.

RDQL [61] is a query language for RDF and is provided as part of the Jena Semantic Web Framework [39] from HP labs, which also includes: an RDF API, facilities for reading and writing RDF in RDF/XML, N3 and N-Triples, an OWL API, and in-memory and persistent storage. RDQL provides a *data-oriented* query model so that there is a more declarative approach to complement the fine-grained, procedural Jena API. It is "data-oriented" in that it only queries the information held in the models; there is no inference being done. Of course, the Jena model may be "smart" in that it provides the impression that certain triples exist by creating them on-demand. However, the RDQL system does not do anything other than take the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings.

Going back to *our example*, by using RDF we could semantically annotate the resources that give explanations about XML-based languages, ontologies, knowledge representation, etc. However, the use that we want to do of such resources requires that each of them is explicitly associated to every topic it might have correlations with. This should be done even though some of the topics are related with each other, for instance XML is related to RDF and XML is related to Semantic Web, but also RDF is related to Semantic Web, and ideally we could *exploit such relations* to infer properties of the available resources. What is still missing is the possibility of *expressing knowledge about the domain*.

RDF Schema [60] adds a new layer of functionalities by allowing the representation of *ontologies*. This is done by introducing the notion of "class" of similar resources, i.e. objects showing a set of same characteristics. Resources are then viewed as "individuals" of some class. Classes can be divided in "subclasses", the result is a *hierarchical structure*. From an extensional point of view, every instance of a class is also an instance of its super-class, as such it inherits the properties of that class. It is possible to exploit this mechanism to perform simple inferences about instances and classes w.r.t. the hierarchical structure. A more powerful ontology language is OWL [54] (Web Ontology Language). OWL, the W3C standard for ontology representation, builds on top of RDF and RDF-S and allows the representation of more complex relations, such as transitivity, symmetry, and cardinality constraints.

It is possible to reason about ontologies by means of techniques that are typical of Description Logics. Basically, such techniques are aimed at *classification*, that is, if a resource is an instance of a class, then it will also be an instance of its super-classes. Also, if a resource satisfies a set of properties that define a sufficient condition to belonging to a given class, then the resource is an instance of that class. By means of these techniques we can satisfy the goal of the user of our example: in fact, if we have an ontology in which Semantic Web has as subclasses XML-based languages, knowledge representation, and so on, and we have a set of resources that are individuals of such classes, it is possible to infer that they are also individuals of Semantic Web. The introduction of these inferencing mechanisms is a fundamental step towards personalization, although in order to have real personalization something more is to be done. Indeed, if two different users are both interested in the Semantic Web the system will return as an answer *the same set* of resources because it does not take into account any information about them.

So far, reasoning in the Semantic Web is mostly reasoning about knowledge expressed in some ontology and the ontology layer is the highest layer of the Semantic Web tower that can be considered as quite well assessed. The layers that lie on top of it, in particular the logic layer and the proof layer, are still

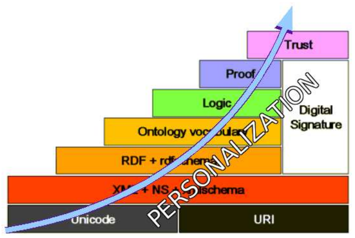


Fig. 1. The Semantic Web tower. Personalization occurs at the ontology layer but mostly at the logic and proof layers.

at a primitive level. The lesson learnt from the analysis that we have done is that for making some personalization we need to represent and reason about *knowledge* and the Semantic Web offers this possibility. Let us, then, see what kinds of knowledge are necessary for performing personalization.

4.2 Knowledge and reasoning about knowledge

A system that performs some kind of personalization needs to represent different kinds of *knowledge*: knowledge about the user, knowledge about the user's purpose (sometimes considered as included in the user's description), knowledge about the context, knowledge about the resources that can be queried, retrieved or composed, and domain knowledge that is used by the inferencing mechanism for obtaining personalization.

Knowledge about the user can roughly be viewed as partitioned in generic knowledge about the user's characteristics and preferences and in "state" knowledge. By the word "state knowledge" we hereby mean information that can change and that is relevant w.r.t. a specific application system, such as which exams have been passed in the case of e-learning.

A user's *goal* most of the times is considered as being coincident with a query but there are some distinctive features to take into account. First of all, a query presupposes an answer, and it implies a selection process, that can be performed by means of the most various techniques. The answer is supposed to be returned within a few seconds. In some applications, however, the goal corresponds to a general interest of the user. For example, the user might be a fan of a given music band and whenever the band performs in the user's town, s/he would like to be informed automatically. In this case, we can view the goals as conditions that can are embedded in rules: when some event satisfies a rule condition, the rule is triggered and, typically, the user is warned in a way that can be subject to further personalization (e.g. w.r.t. the physical device that is used –laptop, mobile, hand-held–). In this case, the answer, that depends on location and time, might be returned days or weeks after the rule has been set. Moreover, the same rule might be activated many times by many different events. A third kind of goal, that we have seen, is more abstract and not directly interpretable as a query. It is, for instance, the case of a learning goal: a learning goal is a description of the expertise that a user would like to acquire. The system uses this information to build a solution that contains many Web resources, to be used as learning materials. None of them is (possibly) directly tied with the learning goal; the goal will be reached by the user if s/he will follow the proposed reading path. In other words, the composition of resources is a *means* for reaching the goal.

In performing resource selection, also knowledge about the *context* plays a very important part. In many applications, three kinds of contextual information can be identified: location in time, location in space, and role. *Location in time and space* is used for refining resource selection, that is, only those resources that fit the context description, are shown. The context description is not necessarily expressed by the user, since it might as well be obtained in other ways. In ubiquitous and in ambient computing it could be returned by a sensor network. *Roles* are predefined views, possibly with a limitation of the actions, that the role players can execute. They are used to personalize the selection of information sources, the selection of information and, of course, presentation.

For performing semantic-based processing on the Web it is necessary that the *Web resources* are semantically annotated. This is normally done by means of ontologies. Even though semantic annotation is not so much diffused, the languages for writing such annotations are pretty well assessed. One of the major difficulties is, actually, to retrieve –if any– an ontology that is suitable to the application at hand, avoiding to write a new one unless really necessary.

The last kind of knowledge that is often necessary in personalization tasks, that we called *domain knowledge*, is aimed at giving a structure to the knowledge. Domain knowledge relates the ontological terms in a way that can be exploited by other inferencing mechanisms, and not only to perform ontological reasoning. For instance, planning is a useful reasoning technique for obtaining personalization; there are proposals in the literature that suggest to bias the search of a plan by introducing solution schemas, that correspond to abstract descriptions of solutions that "make sense". For instance, in the e-learning applications when a course is constructed out of a set of available learning materials, the course must "make sense" also from a pedagogical point of view, see [7]. One can then imagine to have a high-level description of the structure of interest, not related to specific materials, which is personalized and filled with contents on demand, in a way that fits the specific user. Moreover, in many scenarios it is useful to express some event-driven behavior (e.g. in the already mentioned touristic application domain). It is especially at this level that rules can play a fundamental role in the construction of personalization systems in the Semantic Web.

Beyond ontologies: some examples. The first scenario that we consider is set in one of the leading application areas for personalization: education. The most typical problem in this framework consists in determining an "optimal reading sequence" through a hyper-space of learning objects (a learning object is a resource with educational purposes). The word optimal does not mean that this is absolutely the best solution, it means that it specifically fits the characteristics and the needs of the given user. It is optimal for that user. So the aim is to support the user in the acquisition of some desired knowledge by identifying a reading path that best fits him/her. Considerable advancements have been yield in this field, with the development of a great number of Web-based systems, like ELM-Art [70], the KBS hyperbook system [34], TANGOW [22], WLog [6] and many others, based on different, adaptive and intelligent technologies.

Different methods have been proposed on how to determine which reading path to select or to generate in order to support in the best possible way the learner's navigation through the hyper-space. All of them require to go *one step beyond* the ontology layer. In fact, pure ontological annotation and ontological reasoning techniques (though necessary) are not sufficient to produce, in an automatic way, the desired sequencing. If in our ontology the class "Semantic Web" is divided in the classes "XML-based languages", "knowledge representation", and "ontologies" we will be able to conclude that each of the individuals that belong to the sub-classes also belong to the super-class. What we cannot do is to impose that the student will be presented resources about *all* such topics, because only the conjunction of the three will let him/her satisfy his/her learning goal. Another thing that we cannot do is to impose that a given topic is presented before another one because only in this way the student will understand them.

If, on the one hand, it is necessary to annotate the actual learning objects, with the ontological terms that represent identifiable pieces of knowledge related to the learning objects themselves, on the other, it is also necessary to structure a domain knowledge in a way that it is possible to perform the personalization task. The desire is to develop an *adaptation component*, that uses such a knowledge, together with a representation of the user's *learning goal* and of knowledge about the user, for producing sequences that fit the user's requirements and characteristics, based on the available learning objects. Such an adaptation component exploits knowledge representations that are not ontologies (though they use ontologies) and it exploits reasoning mechanisms that are not ontological reasoning mechanisms. For instance, in the application domain that has been taken into account, *goal-directed reasoning* techniques seem particularly suitable.

To this purpose, one solution is to interpret the learning resources as atomic actions. In fact, each learning resource has a set of preconditions (competences that are necessary for using it) and a set of effects (the supplied competences). Competences can be connected by causal relationships. Rational agents could use such descriptions and the user's learning goal, expressed as well in terms of competences, for performing the sequencing task. This is, for instance, the solution adopted in the WLog system [6], which exploits techniques taken from the research area of "reasoning about actions and change" (*planning*, *temporal projection*, and *temporal explanation*) for building personalized solutions.

Another example concerns *Web services*. Generally speaking, a Web service can be seen as any device that can automatically be accessed over the Web. It may alternatively be a software system or a hardware device; a priori no distinction is made. The main difference between a Web service and other devices that are connected to a network stands in the kind of tasks that can be performed: a Web service can be automatically retrieved by searching for the desired functionality (in a way that is analogous to finding Web pages by means of a search engine, given a set of keywords), it can be automatically invoked, composed with other Web services so to accomplish more complex tasks, it must be possible to monitor its execution, and so on. In order to allow the execution of these tasks, it is necessary to enrich the Web service with a machine-processable description, that contains all the necessary information, such as what the service does, which inputs it requires, which results are returned, and so forth. A lot of research is being carried on in this area and none of the problems that we have just enumerated has met its final solution yet. Nevertheless, there are some proposals, especially due to commercial coalitions, of languages that allow the description of the single services, and their interoperation. In this line, the most successful are WSDL [72] and BPEL4WS [14]. This initiative is mainly carried on by the commercial world, with the aim of standardizing registration, look-up mechanisms and interoperability.

Among the other proposals, OWL-S [55] (formerly DAML-S) is more concerned with providing greater expressiveness to service description in a way that can be *reasoned about* [20]. In particular, a service description has three conceptual levels: the *profile*, used for advertising and discovery, the *process model*, that describes how a service works, and the *grounding*, that describes how an agent can access the service. In particular, the process model describes a service as atomic, simple or composite in a way inspired by the language GOLOG and its extensions [45, 50]. In this perspective, a wide variety of agent technologies based upon the *action metaphor* can be used. In fact, we can view a service as an action (atomic or complex) with preconditions and effects, that modifies the state of the world and the state of agents that work in the world. The process model can, then, be viewed as the description of such an action; therefore, it is possible to design agents, which apply techniques for reasoning about actions and change to Web service process models for producing new, composite, and customized services.

Quoting McIlraith [51]: "[. . .] *Our vision is that agents will exploit user's constraints and preferences to help customize user's requests for automatic Web service discovery, execution, or composition and interoperation* [. . .]". In different words, personalization is seen as *reasoning* about the user's constraints and preferences and about the *effects*, on the user's knowledge and on the world, of the *action* "interact with a Web service". Techniques for reasoning about actions and change are applied to produce composite and customized services.

A better personalization can be achieved by allowing agents to reason also about the *conversation protocols* followed by Web services. Conversation protocols rule the interactions of a service with its interlocutors: the protocol defines all the possible "conversations" that the service can enact. Roughly speaking, we can consider it as a procedure built upon atomic speech acts. So far, however, no language for Web service specification, e.g. OWL-S, allows the explicit representation of the communicative behavior of Web services at an abstract level, i.e. in a way that can be reasoned about. Let us, however, explain with a simple example how this would be useful: an agent, which is a user's *personal assistant*, is requested to book a ticket at a cinema where they show a certain movie; as a further constraint, the agent does not have to use the user's credit card number along the transaction. While the first is the *user's goal*, the additional request constrains the way in which the agent will *interact* with the service. In this case, in order to personalize the interaction according to the user's request, it is indeed necessary to reason about the service communications. Another possible task of the personal assistant is the organization of a journey: it is necessary to find and make work together (*compose*) services for finding a flight, renting a car, making a reservation at some hotel, maybe the user's personal calendar, etc. All services that have been developed independently and for simpler purposes.

Personalization may involve also other kinds of reasoning, that require knowledge to be represented in other ways. Among them *defeasible reasoning*, which allows taking into account degrees of preference represented as priorities between rules (e.g. DR-DEVICE [11]), *Answer Set Programming* [27], that can deal with incomplete information and default knowledge, *reactivity to events* [48] (the so called ECA rules –event, condition, action–), that allow the propagation of knowledge updates through the Web. All these approaches and techniques conceptually lie at the logic and proof layers of the Semantic Web tower and rely on some kind of rule language.

Rule languages and rule systems are, actually, in the mainstream of research in the Semantic Web area, especially for what regards *exchange of rule sets* between applications. Works in this direction include initiatives for the definition of *rule markup languages*. The aim of introducing rules is to support in a better and wider way the interaction of systems with users as well as of systems with other systems over the Web. Rule markup languages are designed so to allow the expression of rules as modular, stand-alone units in a declarative way, and to allow the publishing and interchange of rules among different systems. Different perspectives can be considered [68]. Rules can be seen as statements that define the terms of the domain, they can be seen as formal statements, which can be directly mapped to executable statements of a software platform, and they can also be considered as statements in a specific executable language.

Two examples of rule markup languages are RuleML [52] and SWRL [37]. The former is a deductive logic language based on XML and RDF. SWRL is a more recent proposal aimed at adding to the OWL language, for defining Web ontologies, the possibility of including Horn-like clauses. The idea is to add the possibility of making deductive inferences that cannot be accomplished by the ontology reasoning techniques. For instance, a consequence of this kind: if X has a brother Y and X has a son Z, then Y is an uncle of Z.

The most important aspect of the standards is its *adoption*, which implies a diffusion of the inference engines that implement them. The hope is that in the near future browsers will support RuleML engines, SWRL engines, and so forth, enabling the use of knowledge over the Web, in the same easy way in which they currently support languages like Java and JavaScript. On the other hand, besides the standards, the way is open for building, on top of the ontology layer, languages that support *heterogeneous* reasoning mechanisms, that fit the requirements of specific personalization problems. This is the reading key of the following section, where a case study is presented together with reasoning techniques for tackling the personalization task. Further examples of personalization problems, reasoning techniques, and prototype systems can be found in [2].

4.3 Case study: personalization in an e-learning scenario

Let us focus on e-learning and see how reasoning can help personalization in this context. We will begin with the annotation of the learning resources, then, we will introduce some reasoning techniques, all of which exploit a new level of knowledge thus allowing a better personalization.

A learning object can profitably be used if the learner has a given set of prerequisite competences; by using it, the learner will acquire a new set of competences. Therefore, a learning object can be interpreted as an action: in fact, an action can be executed given that a set of conditions holds, and by executing it, a set of conditions will become true. So, the idea is to introduce at the level of the learning objects, some annotation that describes both their *pre-requisites* and their *effects*. Figure 2 shows an example of how this could be done. To make the example realistic, the annotation respects the standard for learning object metadata LOM. LOM allows the annotation of the learning objects by means of an ontology of interest (see for instance [56]), by using the attribute *classification*. A LOM classification consists of a set of ontology elements (*taxons*), with an associated role (the *purpose*). The taxons in the example are taken from the DAML version of the ACM computer classification system ontology [53]. The reference to the ontology is contained in the *source* element. Since the XML-based representation is quite long, for the sake of brevity only two taxons have been reported: the first (relational database) is necessary in order to understand the contents of the learning object, while the other (scientific databases) is a competence that is supplied by the learning object.

The proposed annotation expresses a set of *learning dependencies* between *ontological terms*. Such dependencies can be expressed in a declarative formalism, and can be used by a reasoning system. So, given a set of learning objects each annotated in this way, it is possible to use the standard planners, developed by the Artificial Intelligence community (for instance, the well-known Graphplan [13]), for building the reading sequences. Graphplan is a general-purpose planner that works in STRIPS-like domains; as all planners, the task that it executes is to build a sequence of atomic actions, that allows the transition from an initial state to a state of interest, or goal state. The algorithm is based on ideas used in graph algorithms: it builds a structure called *planning graph*, whose main

```
<lom xmlns="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
  <general>
    <title>
      <langstring>module A</langstring>
    </title>
  </general>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Prerequisite</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml
        </langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">relational database</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
  ...
  <classification>
    <purpose>
      ...
      <value><langstring>Educational Objective</langstring></value>
    </purpose>
    <taxonpath>
      <source>
        <langstring>http://daml.umbc.edu/ontologies/classification.daml
        </langstring>
      </source>
      <taxon>
        <entry>
          <langstring xml:lang="en">scientific databases</langstring>
        </entry>
      </taxon>
    </taxonpath>
  </classification>
</lom>
```

Fig. 2. Excerpt from the annotation for the learning object 'module A': "relational database" is an example of prerequisite while "scientific databases" is an example of educational objective.

property is that the information that is useful for constraining the plan search is quickly propagated through the graph as it is built.

General-purpose planners search a sequence of interest in the whole space of possible solutions and allow the construction of learning objects on the basis of any learning goal. This is not always adequate in an educational application framework, where the set of learning goals of interest is fairly limited and the experience of the teachers in structuring the courses and the learning materials is important. For instance, a teacher due to his/her own experience may believe that topic A is to be presented before topic B, although no learning dependence emerges from the descriptions of A and B. This kind of constraint cannot be exploited by a general-purpose planner, being related to the teaching strategy adopted by the teacher.

On the other hand, it is not reasonable to express schemas of this kind in terms of specific learning objects. The ideal solution is to express the aforementioned schemas as *learning strategies*, i.e. a rule (or a set of rules) that specifies the overall structure of the learning object, expressed only in terms of *competences*. The construction of a learning object can, then, be obtained by refining a learning strategy according to specific requirements and, in particular, by choosing those components that best fit the user.

Reasoning about actions. Reasoning about actions and change is a kind of temporal reasoning where, instead of reasoning about *time* itself, one reasons on *phenomena* that take place in time. Indeed, theories of reasoning about actions and change describe a *dynamic world* changing because of the execution of actions. Properties characterizing the dynamic world are usually specified by propositions which are called *fluents*. The word *fluent* stresses the fact that the truth value of these propositions depends on time and may vary depending on the changes which occur in the world.

The problem of reasoning about the effects of actions in a dynamically changing world is considered one of the central problems in knowledge representation theory. Different approaches in the literature took different assumptions on the temporal ontology and then they developed different abstraction tools to cope with dynamic worlds. However, most of the formal theories for reasoning about action and change (*action theories*) describe dynamic worlds according to the so-called *state-action model*. In the state-action model the world is described in terms of states and *actions* that cause the transition from a state to another. Typically it is assumed that the world persists in its state unless it is modified by an action's execution that causes the transition to a new state (*persistency assumption*).

The main target of action theories is to use a logical framework to describe the effects of actions on a world where *all* changes are caused by the execution of actions. To be precise, in general, a formal theory for representing and reasoning about actions allows us to specify:

1. *causal laws*, i.e. axioms that describe domain's actions in terms of their *precondition* and *effects* on the fluents;

2. action sequences that are executed from the initial state;
3. *observations* describing the value of fluents in the *initial state*;
4. *observations* describing the value of fluents in later states, i.e after some action's execution.

The term *domain description* is used to refer to a set of propositions that express causal laws, observations of the fluents values in a state and possibly other information for formalizing a specific problem. Given a domain description, the principal reasoning tasks are *temporal projection* (or prediction), *temporal explanation* (or postdiction) and *planning*.

Intuitively, the aim of *temporal projection* is to predict an action's future effects based on even partial knowledge about the current state (reasoning from causes to effect). On the contrary, the target of *temporal explanation* is to infer something on the past states of the world by using knowledge about the current situation. The third reasoning task, planning, is aimed at finding an action sequence that, when executed starting from a given state of the world, produces a new state where certain desired properties hold.

Usually, by varying the reasoning task, a domain description may contain different elements that provide a basis for inferring the new facts. For instance, when the task is to formalize the temporal projection problem, a domain description might contain information on (1), (2) and (3), then the logical framework might provide the inference mechanisms for reconstructing information on (4). Otherwise, when the task is to deal with the planning problem, the domain description will contain the information on (1), (3), (4) and we will try to infer (2), i.e. which action sequence has to be executed on the state described in (3) for achieving a state with the properties described in (4).

An important issue in formalization is known as the *persistency problem*. It concerns the characterization of the invariants of an action, i.e. those aspects of the dynamic world that are not changed by an action. If a certain fluent f representing a fact of the world holds in a certain state and it is not involved by the next execution of an action a, then we would like to have an efficient inference mechanism to conclude that f still hold in the state resulting from a's execution.

Various approaches in the literature can be broadly classified in two categories: those choosing classical logics as the knowledge representation language [49, 44] and those addressing the problem by using non-classical logics [57, 23, 65, 29] or computational logics [28, 10, 46, 8]. Among the various logic-based approaches to reasoning about actions one of the most popular is still the situation calculus, introduced by Mc Carthy and Hayes in the sixties [49] to capture change in first order classical logic. The situation calculus represents the world and its change by a sequence of *situations*. Each situation represents a state of the world and it is obtained from a previous situation by executing an action. Later on, Kowalski and Sergot have developed a different calculus to describe change [44], called *event calculus*, in which *events* producing changes are temporally located and they initiate and terminate action effects. Like the situation calculus, the event calculus is a methodology for encoding actions in first-order predicate logic. However, it was originally developed for reasoning about events and time in a logic-programming setting.

Another approach to reasoning about actions is the one based on the use of modal logics. Modal logics adopts essentially the same ontology as the situation calculus by taking the state of the world as primary and by representing actions as state transitions. In particular, actions are represented in a very natural way by modalities whose semantics is a standard Kripke semantics given in terms of accessibility relations between worlds, while states are represented as sequences of modalities.

Both situation calculus and modal logics influenced the design of logic-based languages for agent programming. Recently the research about situation calculus gained a renewed attention thanks to the cognitive robotic project at University of Toronto, that has lead to the development of a high-level agent programming language, called GOLOG, based on a theory of actions in situation calculus [45]. On the other hand, in DyLOG [9], a modal action theory has been used as a basis for specifying and executing agent behavior in a logic programming setting, while the language IMPACT is an example of use of deontic logic for specifying agents: the agent's behavior is specified by means of a set of rules (the agent program) which are suitable to specify, by means of deontic modalities, agent policies, that is which actions an agent is obliged to take in a given state, which actions it is permitted to take, and how it chooses which actions to perform.

Introducing learning strategies. Let us now show how the schemas of solution, or *learning strategies*, can be represented by means of rules. In particular, we will use the notation of the language DyLOG.

Learning strategies, as well as learning objects, should be defined on the basis of an ontology of interest. One common need is to express *conjunctions* or *sequences* of learning objects. So for instance, one can say that in his/her view, it is possible to acquire knowledge about *database management* only by getting knowledge about *all* of of a given set of topics, and, among these, *relational databases* must be known before *distributed databases* are introduced.

An example that is particularly meaningful is preparing the material for a basic computer science course: the course may have different contents depending on the kind of student to whom it will be offered (e.g. a Biology student, rather than a Communication Sciences student, rather than a Computer Science student). Hereafter, we consider the case of Biology students and propose a *DyLOG* procedure, named '*strategy('informatics_-for_biologists')*'. This procedure expresses, at an abstract level, a learning strategy for guiding a biology student in a learning path, which includes the basic concepts about how a computer works, together with a specific competence about databases. Notice that no reference to specific learning objects is done.

$$\begin{aligned}&strategy('informatics\_for\_biologists')\ \mathbf{is}\\ &\quad achieve\_goal(has\_competence('computer\ system\ organization')) \wedge\\ &\quad achieve\_goal(has\_competence('operating\ systems')) \wedge\\ &\quad achieve\_goal(has\_competence('database\ management')).\end{aligned}$$

$$\begin{array}{l} \quad \ldots \\ achieve\_goal(has\_competence('database\ management'))\ \textbf{is} \\ \quad achieve\_goal(has\_competence('relational\ databases')) \wedge \\ \quad achieve\_goal(has\_competence('query\ languages')) \wedge \\ \quad achieve\_goal(has\_competence('distributed\ databases')) \wedge \\ \quad achieve\_goal(has\_competence('scientific\ databases')). \end{array}$$

strategy is defined as a procedure clause, that expresses the view of the strategy creator on what it means to acquire competence about *computer system organization*, *operating systems*, and *database management*.

Suppose that *module A* is the name of a learning object. Interpreting it as an action, it will have preconditions and effects expressed as in Figure 2. We could represent *module A* and its learning dependencies in DyLOG in the following way:

$$\begin{array}{l} access(learningObject('module\ A'))\ \textbf{possible if} \\ \quad has\_competence('distributed\ database') \wedge \\ \quad has\_competence('relational\ database'). \\ access(learningObject('module\ A'))\ \textbf{causes} \\ \quad has\_competence('scientific\ databases'). \end{array}$$

Having a learning strategy and a set of annotated learning objects, it is possible to apply *procedural planning* (supplied by the language) for assembling a reading path that is a sequence of learning resources that are annotated as required by the strategy. Opposite to general-purpose planners, procedural planning searches for a solution in the set of the possible executions of a learning strategy. Notice that, since the strategy is based on competences, rather than on specific resources, the system might need to select between different courses, annotated with the same desired competence, which could equally be selected in building the actual learning path. This choice can be done based on external information, such as a user model, or it may be derive from a further interaction with the user. Decoupling the strategies from the learning objects results in a greater flexibility of the overall system, and simplifies the reuse of the learning objects. As well as learning objects, also learning strategies could be made public and shared across different systems.

Other approaches to rule-based personalization in an e-learning scenario The above example is just one possible way in which personalization can be realized in the Semantic Web in a practical context. Remaining in the e-learning application domain, many other forms of personalization can be thought of, which require other approaches to rule representation and reasoning. Hereafter, we report another example that is taken from a real system. The personalization rules that we will see realize some of the adaptation methods of adaptive educational hypermedia systems (see Section 3.1). The application scenario is a *Personal Reader*[3] [32, 12] for learning resources. This Personal Reader helps the learner to view the learning resources in a *context*: In this context, more *details* related to the topics of the learning resource, the *general topics* the learner is currently studying, *examples*, *summaries*, *quizzes*, etc. are generated and enriched with personal recommendations according to the learner's current learning state [32, 25]. Let us introduce and comment some of the rules that are used by the Personal Reader for learning resources to determine appropriate adaptation strategies. These personalization rules have been realized using TRIPLE.

[3] http://www.personal-reader.de

Generating links to more detailed learning resources is an adaptive functionality in this example Personal Reader. The adaptation rule takes the *isA* hierarchy in the domain ontology, in this case the domain ontology for Java programming, into account to determine domain concepts which are details of the current concept or concepts that the learner is studying on the learning resource. In particular, more details for the currently used learning resource is determined by `detail_learningobject(LO, LO_DETAIL)` where `LO` and `LO_Detail` are learning resources, and where `LO_DETAIL` covers more specialized learning concepts which are determined with help of the domain ontology.

```
FORALL LO, LO_DETAIL detail_learningobject(LO, LO_DETAIL) <-
   EXISTS C, C_DETAIL(detail_concepts(C, C_DETAIL)
     AND concepts_of_LO(LO, C) AND concepts_of_LO(LO_DETAIL, C_DETAIL))
     AND learning_resource(LO_DETAIL) AND NOT unify(LO,LO_DETAIL).
```

Observe that the rule does neither require that `LO_DETAIL` covers all specialized learning concepts, nor that it exclusively covers specialized learning concepts. Further refinements of this adaptation rule are of course possible. The rules for embedding a learning resource into more general aspects with respect to the current learning progress are similar.

Another example of a *personalization rule* for generating embedding context is the recommendation of quiz pages. A learning resource `Q` is recommended as a quiz for a currently learned learning resource `LO` if it is a quiz (the rule for determining this is not displayed) and if it provides questions to at least some of the concepts learned on `LO`.

```
FORALL Q quiz(Q) <-
   Q['http://www.w3.org/1999/02/22-rdf-syntax-ns#':type ->
   'http://ltsc.ieee.org/2002/09/lom-educational#':'Quiz']

FORALL Q, C concepts_of_Quiz(Q,C) <-
   quiz(Q) AND concept(C) AND
   Q['http://purl.org/dc/elements/1.1/':subject -> C].

FORALL LO, Q quiz(LO, Q) <-
     EXISTS C (concepts_of_LO(LO,C) AND concepts_of_Quiz(Q,C)).
```

Recommendations are personalized according to the current learning progress of the user, e. g. with respect to the current set of course materials. The following rule determines that a learning resource `LO` is `recommended` if the learner studied at least one more general learning resource (`UpperLevelLO`):

```
FORALL LO1, LO2 upperlevel(LO1,LO2) <-
   LO1['http://purl.org/dc/terms#':isPartOf -> LO2].

FORALL LO, U learning_state(LO, U, recommended) <-
   EXISTS UpperLevelLO (upperlevel(LO, UpperLevelLO) AND
   p_obs(UpperLevelLO, U, Learned) ).
```

Additional rules deriving stronger recommendations (e. g., if the user has studied *all* general learning resources), less strong recommendations (e.g., if one or two of these haven't been studied so far), etc., are possible, too. Recommendations can also be calculated with respect to the current domain ontology. This is necessary if a user is regarding course materials from different courses at the same time.

```
FORALL C, C_DETAIL detail_concepts(C, C_DETAIL) <-
   C_DETAIL['http://www.w3.org/2000/01/rdf-schema#':subClassOf -> C]
   AND concept(C) AND concept(C_DETAIL).

FORALL LO, U learning_state(LO, U, recommended) <-
   EXISTS C, C_DETAIL (concepts_of_LO(LO, C_DETAIL)
   AND detail_concepts(C, C_DETAIL) AND p_obs(C, U, Learned) ).
```

However, the first recommendation rule, which reasons within one course will be more accurate because it has more fine–grained information about the course and therefore on the learning process of a learner taking part in this course. Thus, a strategy is to prioritize those adaptation rule which take most observations and data into account, and, if these rules cannot provide results, apply less strong rules. This can be realized by defeasible rules [3]: Priorities are used to resolve conflicts, e.g. by giving external priority relations (N.B.: these external priority relations must be acyclic). For example: Rule `r1` determines that the learning state of a learning object is recommended for a particular user if the user has learnt at least one of the general, introductory learning objects in the course, while `r2` says that a learning object is not recommended if the learner has not learnt at least one of the more general concepts. In the following code, `r1 > r2` defines a degree of preference: only when the first rule cannot be applied, the system tries to apply the second.

```
r1: EXISTS UpperLevelLO (upperlevel(LO, UpperLevelLO) AND
      p_obs(UpperLevelLO, U, Learned))
    => learning_state(LO, U, recommended)

r2: FORALL C, C_DETAIL (concepts_of_LO(LO, C_DETAIL)
      AND detail_concepts(C, C_DETAIL) AND NOT p_obs(C, U, Learned)
    => NOT learning_state(LO, U, recommended)

and r1 > r2.
```

5 Conclusions

Personalization, which has become one of the major endeavors of research over the Web, has been studied since the mid 90's in fields like Adaptive Hypermedia and Web Mining. In Adaptive Hypermedia each user has a personalized view of the hypermedia system as well as individual navigation alternatives. Personalization is carried out either selecting the proper level of contents, that the user can read, or by modifying the set of links to other documents (for instance by hiding certain connections). Web Mining, on the other hand, is mostly concerned with the identification of relations between Web resources which are not directly connected through links. These new relations can be induced on the basis of resource contents or on the basis of regularities in the behavior of a set of independent users. All these approaches have been applied to the WWW, allowing the realization of adaptive systems even in absence of a universally agreed semantics and of standard languages and tools for representing and dealing with semantics. This heterogeneity entails some limitations. In fact, any technique used to deliberate whether a certain resource or link is to be shown to the user requires a lot of information, about the user, about the reasons for which the user should access that resource, and so on. Actually, most of the early personalization systems either managed "closed-world" resources, as it was the case of many systems for e-learning that handled given repositories of learning materials as well as of e-commerce tools, or they were based on user models refined during the direct interaction with the user.

The birth of the Semantic Web brought along standard models, languages, and tools for representing and dealing with machine-interpretable semantic descriptions of Web resources, giving a strong new impulse to research on personalization. Just as the current Web is inherently heterogeneous in data formats and data semantics, the Semantic Web will be heterogeneous in its reasoning forms and the same will hold for personalization systems developed in the Semantic Web. In this lecture we have analyzed some possible applications of techniques for reasoning about actions and change and of techniques for reasoning about preferences, the so called defeasible logic, but, indeed, the availability of a variety of reasoning techniques, all fully integrated with the Web, opens the way to the design and the development of forms of interaction and of personalization that were unimaginable still a short time ago. To this aim it is necessary to integrate results from many areas, such as Multi-Agent Systems, Security, Trust, Ubiquitous Computing, Ambient Intelligence, Human-Computer Interaction and, of course, Automated Reasoning.

This paper is just an introduction to personalization over the Semantic Web, that presents issues, approaches, and techniques incrementally. We have started from the World Wide Web and, then, moved to more abstract levels step by step towards semantics and reasoning, a pattern that follows the classical view of the Semantic Web as a tower of subsequent layers. More than being exhaustive w.r.t all the different techniques and methods that have been proposed in the literature, we have tried to give a complete overview, that includes historical roots, motivations, interconnections, questions, and examples. In our opinion, personalization plays a fundamental role in the Semantic Web, because what is the Semantic Web but a knowledge-aware Web, able to give each user the answers that s/he expects? Research in this field is at the beginning.

Acknowledgements

The authors are indebted with all the researchers who took part to the stimulating discussions during the meetings of REWERSE and in particular of working group A3 in Munich and Hannover. Special thanks to Viviana Patti and Laura Torasso, who actively contribute to the project.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*, 1994.
2. G. Antoniou, M. Baldoni, C. Baroglio, R. Baungartner, F. Bry, T. Eiter, N. Henze, M. Herzog, W. May, V. Patti, S. Schaffert, R. Schidlauer, and H. Tompits. Reasoning methods for personalization on the semantic web. *Annals of Mathematics, Computing & Teleinformatics (AMCT)*, 2(1):1–24, 2004.
3. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. MIT Press, 2004.
4. R. Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.
5. M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about interaction protocols for web service composition. In M. Bravetti and G. Zavattaro, editors, *Proc. of 1st Int. Workshop on Web Services and Formal Methods, WS-FM 2004*, volume 105 of *Electronic Notes in Theoretical Computer Science*, pages 21–36. Elsevier Science Direct, 2004.
6. M. Baldoni, C. Baroglio, and V. Patti. Web-based adaptive tutoring: an approach based on logic agents and reasoning about actions. *Artificial Intelligence Review*, 22(1), September 2004.
7. M. Baldoni, C. Baroglio, V. Patti, and L. Torasso. Reasoning about learning object metadata for adapting scorm courseware. In L. Aroyo and C. Tasso, editors, *Proc. of Int. Workshop on Engineering the Adaptive Web, EAW'04: Methods and Technologies for personalization and Adaptation in the Semantic Web*, pages 4–13, Eindhoven, The Netherlands, August 2004.
8. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. An Abductive Proof Procedure for Reasoning about Actions in Modal Logic Programming. In J. Dix et al., editor, *Proc. of NMELP'96*, volume 1216 of *LNAI*, pages 132–150. Springer-Verlag, 1997.
9. M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Programming Rational Agents in a Modal Action Logic. *Annals of Mathematics and Artificial Intelligence, Special issue on Logic-Based Agent Implementation*, 41(2-4):207–257, 2004.
10. C. Baral and T. C. Son. Formalizing Sensing Actions - A transition function based approach. *Artificial Intelligence*, 125(1-2):19–91, January 2001.
11. N. Bassiliades, G. Antoniou, and I. Vlahavas. A defeasible logic system for the semantic web. In *In Proc. of Principles and Practice of Semantic Web Reasoning (PPSWR04)*, volume 3208 of *LNCS*. Springer, 2004.
12. Robert Baumgartner, Nicola Henze, and Marcus Herzog. The personal publication reader: Illustrating web data extraction, personalization and reasoning for the semantic web. In *Proceedings of 2nd European Semantic Web Conference*, Heraklion, Greece, May 2005.
13. A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.
14. BPEL4WS. http://www-106.ibm.com/developerworks/library/ws-bpel. 2003.
15. P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! version 2.0: More adaptation flexibility for authors. In *Proceedings of the AACE ELearn'2002 conference*, October 2002.
16. P. De Bra, G.J. Houben, and H. Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In *ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, 1999.
17. P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
18. P. Brusilovsky, J. Eklund, and E. Schwarz. Web-based Educations for All: A Tool for Development Adaptive Courseware. In *Proceedings of the Sevenths International World Wide Web Conference, WWW'98*, 1998.
19. Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11:87–110, 2001.
20. 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. In J. Liu N. Zhong and Y. Yao, editors, *Web Intelligence*. Springer-Verlag, Berlin, 2002. Agent-Based Composite Services in DAML-S: The Behavior-Oriented Design of an Intelligent Semantic Web.
21. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.
22. R.M. Carro, E. Pulido, and P. Rodruez. Dynamic generation of adaptive internet-based courses. *Journal of Network and Computer Applications*, 22:249–257, 1999.
23. M. Castilho, O. Gasquet, and A. Herzig. Modal tableaux for reasoning about actions and plans. In S. Steel, editor, *Proc. ECP'97*, LNAI, pages 119–130, 1997.
24. P. de Bra. Hypermedia structures and systems: Online Course at Eindhoven University of Technology, 1997. http://wwwis.win.tue.nl/2L690/.
25. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. The Personal Reader: Personalizing and Enriching Learning Resources using Semantic Web Technologies. In *Proc. of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2004)*, Eindhoven, The Netherlands, 2004.
26. M. H. Dunham, editor. *Data Mining*. Prentice Hall, 2003.
27. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9, 1991.
28. M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–321, 1993.
29. L. Giordano, A. Martelli, and C. Schwind. Dealing with concurrent actions in modal action logic. In *Proc. ECAI-98*, pages 537–541, 1998.
30. I.P. Goldstein. The genetic graph: A represenation for the evolution of procedural knowledge. In D. Sleeman and J.S.Brown, editors, *Intelligent Tutoring Systems*. Academic Press, 1982.
31. F. Halasz and M. Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

32. N. Henze and M. Kriesell. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *Proccedings of the 1st International Workshop on Engineering the Adaptive Web (EAW 2004), co-located with AH 2004*, Eindhoven, The Netherlands, 2004.
33. N. Henze and W. Nejdl. Extendible adaptive hypermedia courseware: Integrating different courses and web material. In *Proccedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000)*, Trento, Italy, 2000.
34. N. Henze and W. Nejdl. Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12, 2001.
35. N. Henze and W. Nejdl. Logically characterizing adaptive educational hypermedia systems. Technical report, University of Hannover, April 2003. http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2003/TechReportHenzeNejdl.pdf.
36. N. Henze and W. Nejdl. A logical characterization of adaptive educational hypermedia. *New Review of Hypermedia*, 10(1), 2004.
37. I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, and B. Grosof. SWRL: a semantic web rule language combining OWL and RuleML, 2004. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521.
38. A. Jameson. Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User Adapted Interaction*, 5(3/4):193–251, 1996.
39. Jena - A Semantic Web Framework for Java, 2004. http://jena.sourceforge.net/.
40. Jupiter Research Report, October 14th, 2003. http://www.jupitermedia.com/corporate/releases/03.10.14-newjupresearch.html.
41. A. Kobsa. User modeling: Recent work, prospects and hazards. In M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, editors, *Adaptive User Interfaces: Principles and Practice*. Elvesier, 1993.
42. A. Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11:49–63, 2001.
43. N. Koch. *Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process.* PhD thesis, Ludwig-Maximilians-Universitt Mnchen, 2001.
44. R. Kowalski and M. Sergot. A Logic-based Calculus of Events. *New Generation of Computing*, 4:67–95, 1986.
45. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997.
46. J. Lobo, G. Mendez, and S. R. Taylor. Adding Knowledge to the Action Description Language $\mathcal{A}$. In *Proc. of AAAI'97/IAAI'97*, pages 454–459, Menlo Park, 1997.
47. D. Lowe and W. Hall. *Hypermedia and the Web.* J. Wiley and Sons, 1999.
48. W. May, J.J. Alferes, and F. Bry. Towards generic query, update, and event languages for the semantic web. In *in Proc. of Principles and Practice of Semantic Web Reasoning (PPSWR04)*, volume 3208 of *LNCS*. Springer, 2004.
49. J. McCarthy and P. Hayes. Some, Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1963.
50. S. McIlraith and T. Son. Adapting Golog for Programming the Semantic Web. In *5th Int. Symp. on Logical Formalization of Commonsense Reasoning*, pages 195–202, 2001.
51. S. A. McIlraith, T. C. Son, and H. Zenf. Semantic Web Services. *IEEE Intelligent Systems*, pages 46–53, March/April 2001.
52. Rule ML. http://www.ruleml.org.
53. Association of Computing Machinery. The ACM computer classification system, 2003. http://www.acm.org/class/1998/.
54. OWL, Web Ontology Language, W3C Recommendation, February 2004. http://www.w3.org/TR/owl-ref/.
55. OWL-S: Web Ontology Language for Services, W3C Submission, November 2004. http://www.org/Submission/2004/07/.
56. W. Nejdl P. Dolog, R. Gavriloaie and J. Brase. Integrating adaptive hypermedia techniques and open rdf-based environments. In *Proc. of The 12th Int. World Wide Web Conference*, Budapest, Hungary, 2003.
57. H. Prendinger and G. Schurz. Reasoning about action and change. a dynamic logic approach. *Journal of Logic, Language, and Information*, 5(2):209–245, 1996.
58. R. Rada. *Interactive Media.* Springer, 1995.
59. RDF. http://www.w3c.org/tr/1999/rec-rdf-syntax-19990222/. 1999.
60. RDFS. http://www.w3c.org/tr/rdf-schema/. 2004.
61. RDQL - query language for RDF, Jena, 2005. `http://jena.sourceforge.net/RDQL/`.
62. R. Reiter. A theory of diagnosis from first principles. *Artifical Intelligence*, 32, 1987.
63. E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1978.
64. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.
65. C. B. Schwind. A logic based framework for action theories. In J. Ginzburg et al., editor, *Language, Logic and Computation*, pages 275–291. CSLI, 1997.
66. M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC)*, pages 364–378, Sardinia, Italy, 2002. LNCS 2342.
67. M. Specht. Empirical evaluation of adaptive annotation in hypermedia. In *ED-Media and ED-Telekom*, Freiburg, Germany, 1998.
68. G. Wagner. Ruleml, swrl and rewerse: Towards a general web rule language framework. *SIG SEMIS Semantic Web and Information Systems*, 2004. http://www.sigsemis.org/articles/copy_of_index_html.
69. Geoffrey I. Webb, Michael J. Pazzani, and Daniel Billsus. Machine learning for user modeling. *User Modeling and User-Adapted Interaction*, 11:19–29, 2001.
70. G. Weber and P. Brusilovsky. ELM-ART: An Adaptive Versatile System for Web-based Instruction. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12, 2001.
71. G. Weber, H.C. Kuhl, and S. Weibelzahl. Developing adaptive internet based courses with the authoring system NetCoach. In *Proc. of the Third Workshop on Adaptive Hypermedia, AH2001*, 2001.
72. WSDL. http://www.w3c.org/tr/2003/wd-wsdl12-20030303/. version 1.2, 2003.

Artificial Intelligence Review **22:** 3–39, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Web-Based Adaptive Tutoring: An Approach Based on Logic Agents and Reasoning about Actions

MATTEO BALDONI, CRISTINA BAROGLIO and VIVIANA PATTI
Dipartimento di Informatica, Università degli Studi di Torino, C.so Svizzera, 185, I-10149 Torino, Italy (E-mails: {baldoni; baroglio; patti}@di.unito.it)

Abstract. In this paper we describe an approach to the construction of adaptive tutoring systems, based on techniques from the research area of Reasoning about Actions and Change. This approach leads to the implementation of a prototype system, having a multi-agent architecture, whose kernel is a set of rational agents, programmed in the logic programming language DyLOG. In the prototype that we implemented the reasoning capabilities of the agents are exploited both to dynamically build study plans and to verify the correctness of user-given study plans with respect to the competence that the user wants to acquire.

Keywords: adaptive systems, curriculum sequencing, curricula validation, logic programming, multiagent systems, reasoning about actions, web-based tutoring

1. Introduction

This work investigates the use of an agent logic language, based on reasoning about action effects, for performing adaptive tutoring tasks in the context of a Web-based educational application. Adaptation in Web-based educational systems is a hot research topic attracting greater and greater attention (Brusilovsky, 2001), especially after the spreading of web technologies. Many research teams have implemented Web-based courseware and other educational applications based on different, adaptive and intelligent technologies, with the common goal of using knowledge about the domain, about the student and about the teaching strategies in order to support flexible, personalized learning and tutoring (Weber and Brusilovsky 2001; Henze and Nejdl 2001; Carro et al. 1999; Macías and Castells 2001).

The problem that we faced is to provide personalized support to users (students, in our case) in the definition process of *study plans*, a study plan being a sequence of courses that the student should attend. The idea that we explored is to base adaptation on the *reasoning capabilities* of a *rational agent*, built by means of a logic language. In particular, we focused on the possible uses of three different reasoning techniques, namely *planning*, *temporal projection*, and *temporal explanation*, which have been developed for allowing software agents to build action plans and to verify whether some

properties of interest hold after the application of given sequences of actions. In both cases actions are – usually – not executed in the real world but their execution is simulated "in the mind" of the system, which has to foresee their effects (or their enabling causes) in order to build solutions. In our application framework, a group of agents, called *reasoners*, works on a dynamic domain description, where the basic actions that can be executed are of the kind "attend course X" and where also complex professional expertise can be described. Effects and conditions of actions (courses) are essentially given in terms of a set of abstract *competences*, which are connected by causal relationships. The set of all the possible competences and of their relations defines an *ontology*. This multi-level description of the domain bears along many advantages. On the one hand, the high modularity that this approach to knowledge description manifests allows course descriptions as well as expertise descriptions to be added, deleted or modified, without affecting the system behavior. On the other hand, working at the level of competences is close to human intuition and enables the application of both goal-directed reasoning processes and explanation mechanisms.

The reasoning process that supports the *definition* of a *study plan*, aimed at reaching a certain learning goal, either computes over the effects of attending courses (given in terms of competence acquisition, credit gaining, and the like) or over those conditions that make the attendance of a course reasonable from the educator point of view. The logic approach also enables the *validation* of student-given study plans with respect to some learning goal of interest to the student himself. Basically, the reasons for which a study plan may be wrong are two: either the course sequence is not correct or by attending that plan the student will not acquire the desired competence. In both cases our reasoning mechanism allows the system to detect the weak points in the plan proposed by the student and to return a precious feedback.

Summarizing, the adaptive services currently supplied by our rational agents are: study plan construction, student-given study plan validation, and, in case of validation failure, explanation of the reasons for which a student-given plan is not correct. Notice that our tutoring system is not required to monitor the students progress, its only target being study plan definition. In this perspective, our task resembles *curriculum sequencing* problems, where an "optimal reading sequence" through a hyper-space of information sources is to be found. We can consider in fact courses (or, at least, course descriptions) as information sources. The main difference with respect to many applicative domains, where curriculum sequencing is used, is that in our framework the whole path is to be constructed before the student begins to attend the courses. Therefore, we cannot apply methods that construct the solution one step at a time, always returning the *next best step*. We actually propose a *multi-step methodology*, that builds *complete*, personalized study plans (see the conclusions for a deeper comparison with other curriculum sequencing techniques).

A key feature that allows an agent of the kind described above to *adapt* to each single user is its capability of tackling mental attitudes, such as *beliefs* and *intentions*. In fact, the agent can adopt a student's *learning goal* and find a way for achieving it, which fits that specific student's interest and takes into account the student's current knowledge. As we will see, the identified solution can, then, be further adapted by interacting with the user during the solution presentation phase. Intention, as well as belief and action, has intensively been studied in *logics* and in *logic programming* (Scherl and Levesque 1993; Gelfond and Lifschitz 1993). We used an agent logic programming language, called DyLOG (Baldoni et al. 2001b; Patti 2002), that is based on a *modal formal theory of actions* and allows reasoning about action effects in a dynamically changing environment. The language will be described in the following sections, while its setting in the logic programming literature is given in the conclusions.

Users interact with the system that we implemented, Wlog,[1] by means of a web browser. All the communication with the system is performed by means of a set of dynamically generated web pages, in a process that can be considered a very simple form of *conversation*. In this perspective, the use of mental attitudes could recall some works on cooperative dialogue systems in the field of Natural Language, and in particular (Bretier and Sadek 1997); however, there are some differences. Bretier and Sadek proposed a logic of rational interaction for implementing the dialogue management components of a spoken dialogue system. This work is based, like the DyLOG language, on dynamic logic; nevertheless, while they exploit the capability of reasoning on actions and intentions in order to produce proper *dialogue* acts, we use them to produce *solutions* to the users' problems.

The article is organized as follows. Section 2 describes the application framework and informally introduces some key elements of our work: the notion of course, the notion of competence, the tasks that we have tackled. Section 3 introduces the DyLOG language and explains how both the domain knowledge and knowledge about the student are represented; moreover, this section explains how courses can be interpreted as actions and the forms of interaction that the language supports. Section 4 explains our interpretation of adaptive tutoring services as "reasoning about actions" tasks. Three kinds of reasoning processes are introduced: temporal projection, temporal explanation, and planning. We will see how rational agents, that can perform these kinds of reasoning, can accomplish the tutoring tasks that are described in Section 2. In Section 5, we show how DyLOG can be used not only for

representing the domain knowledge but also for programming the reasoners that perform the various tasks. Finally, Section 6 describes the implemented system; conclusions follow.

2. The Virtual Tutor Domain

In Italian universities students must list the courses that they want to attend in their years as undergraduate students. Every year they have the possibility to change this list (called *study plan*) according to their recent experiences and personal taste. Study plans should be compiled according to given guidelines and their consistency is verified by university professors.

Both the definition and the validation of study plans are time-consuming, difficult tasks. For instance, students tend to be attracted by courses whose names recall graphics, multimedia, or the web, disregarding those that supply the necessary theoretical backgrounds. One of the main reasons of this behavior is that when asked to build a study plan, students tend to be attracted by those courses that are described by keywords, which can intuitively be associated to the professional expertise they are interested in (the student's *learning goal*). On the other hand, also study plan verification is a difficult and time-consuming task for professors, which requires knowledge about each student situation (which courses have been attended and passed, if and what the student studied abroad, and so forth) as well as knowledge about each course (prerequisites, topics that are taught) and the general guidelines that constrain the possible alternatives. In order to resolve the possible inconsistencies of a student-defined study plan, it is often necessary for students to meet a tutor and discuss with him/her both their intentions and some of their specific choices.

For all these reasons it would be very useful to have a *software assistant* that helps both students and professors in all of the different phases of plan construction and validation. Taking a look at the literature, study plan construction can be interpreted as a special case of curriculum, or page, sequencing. *Curriculum sequencing* is a well-known technique in the field of Adaptive Educational Hypermedia (AEH) (Stern and Woolf 1998; Brusilovsky 2000; Weber and Brusilovsky 2001; Henze and Nejdl 2001; Baldoni et al. 2002); it is commonly used in this field to personalize, in an intelligent way, the navigation of a student in a hyperspace of information sources. Although the granularity and the type of information source depend on the specific application domain and can widely vary (an information source may be a definition, a page, a web site), they are usually considered as being *atomic* and thus they are handled as single unstructured objects. In the special case of our application domain, study plan construction can be described

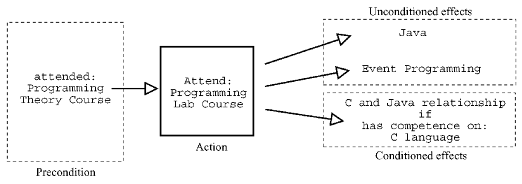

attended: Programming Theory Course
Precondition
Attend: Programming Lab Course
Action
Unconditioned effects
Java
Event Programming
C and Java relationship if has competence on: C language
Conditioned effects

Figure 1. The action of attending the Programming Lab course.

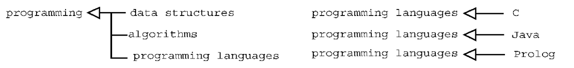


Figure 2. As an example, this is a little excerpt from our competence-based knowledge model.

as a curriculum sequencing task where the atomic units of information that compose the hyperspace are *course descriptions*.

Our approach to study plan construction is based on the observation that it is quite natural to represent each course as an *action*: the action of attending the course. For instance, Figure 1 describes a Programming Lab course as the action of "attending a Programming Lab course", which can be executed if an action "attend a Programming Theory course" has already been executed. The execution of the action "attend a Programming Lab course" has as an effect the acquisition of knowledge about Java and Event programming. Part of an action effects can be subject to more specific conditions; in the example a student can understand comparisons with C programming if he already knows the C language; observe, however, that it is not necessary to know C for attending the course. As underlined by the example, in this perspective a course can be fully described by two sets of conditions: the conditions that are to be satisfied before the course can be attended and the conditions that ideally become true by attending it (i.e., the knowledge that the student is supposed to acquire by attending the course). Besides references to explicit courses, all preconditions and outcomes are given in terms of knowledge elements, that in our approach are described separately: we call them *competences*. Generally, competences are not atomic and can be seen as composed of smaller pieces of competence, related to one another. The set of all the pieces of competence and the set of their relations define an ontology, which is a *knowledge model* of the learning domain. The ontology also defines the vocabulary of the terms used to write the course descriptions which belong to our hyperspace of information sources; last but not least, competences are also used to describe the students' learning goals.

Knowledge models can be defined in different ways by describing the relationships among the various elements; our ontology is built upon a set of causal relations. Figure 2 shows an example: the two-level tree on the left represents the fact that having competence about "data structures", "algorithms", and "programming languages" causes to have competence about "programming". The rules on the right say that competence about "programming languages" can be achieved by (alternatively) acquiring competence either about the "C", the "Java", or the "Prolog" programming language.

In this framework, study plan construction can be interpreted as a *reasoning about actions* task: the task of finding a sequence of courses that, once attended, will allow a student to achieve his learning goal. Interpreting courses as actions, described in terms of a common vocabulary, has another advantage: it enables the application of other reasoning mechanisms that are very helpful in the construction of an AEH system. For example, it is possible to validate the correctness of a student-given plan in an automatic way or, as we will explain in the following sections, to build systems that to some extent "discuss" with a student explaining what is wrong in a submitted study plan.

3. Representing the Domain in DyLOG

In our work, we have developed a multi-agent system, Wlog, whose architecture will be described in Section 6, that performs the tasks described in the previous section. The system's kernel is a set of rational agents, called *reasoners*, that have been implemented in the agent programming language DyLOG. In the current and in the following section we will respectively explain how it is possible to represent both the domain knowledge and the agents' behaviour by using the DyLOG language and how it is possible to interpret adaptive tutoring services as reasoning about actions tasks.

The reader who would like to learn more about the DyLOG language will find a thorough description of it in (Baldoni et al. 2001b; Patti 2002).

3.1. *The* DyLOG *agent programming language*

DyLOG is a language for programming agents based on a logical theory for reasoning about action and change in a *modal logic programming* setting. Agents are entities that read, understand, modify, or more generally interact with their environment by performing actions. Therefore, an agent's behavior can be fully described in a non-deterministic way by giving the *set of actions* that it can perform.

Actions have preconditions to their application, which may be conditions either about the external world or about the agent internal state, and they produce expected effects. This situation recalls what we said about courses: a course has preconditions and it has effects on the knowledge of the student. An action effect may also cause further effects, which do not directly derive from the execution of that action but from a set of causal rules which depend on the domain and that are triggered automatically when some conditions become true. For instance, let us think to a robot that plugs in an iron: an indirect effect of this action is that the iron becomes hot. The robot did not heat the iron but it caused its increase of temperature by plugging it in.

3.2. *States: Representing the knowledge of a student*

We have mentioned that actions can be applied only if their preconditions are true. Once applied, actions produce changes either to the world or to the agent's knowledge. We can, then, think to the whole reasoning process as a sequence of *transitions* between *states*. Our states represent the *knowledge* that the agent has about the world and not the state of the world itself (which we do not model). A state can also be seen as the result of an action sequence, applied to some initial state. Technically speaking, a *state* consists of a set of *fluents*, i.e., properties whose truth value may change over the time. In general we cannot assume that the value of each fluent in a state is known to an agent, so we want to have both the possibility of representing that some fluents are unknown and the capability of reasoning about the execution of actions on incomplete states. To explicitly represent the unknown value of some fluents, in (Baldoni et al. 2001b) we introduced an epistemic level in our representation language. In particular, we introduced an epistemic operator $\mathcal{B}$, to represent the beliefs an agent has about the world: $\mathcal{B}f$ means that the fluent f is known to be true, $\mathcal{B}\neg f$ means that the fluent f is known to be false. A fluent f is undefined when both $\neg\mathcal{B}f$ and $\neg\mathcal{B}\neg f$ hold at the same time ($\neg\mathcal{B}f \wedge \neg\mathcal{B}\neg f$). For expressing that a fluent f is undefined, we write $u(f)$. Thus each fluent can have one of the three values: *true*, *false* or *unknown*.

Nevertheless, in our *implementation* of DyLOG (and, for the sake of simplicity, also in the following description) we do not explicitly use the epistemic operator $\mathcal{B}$: if a fluent f (or its negation $\neg f$) is present in a state, it is intended to be believed, unknown otherwise. This choice is due to the fact that operator $\mathcal{B}$ is indeed very useful when an agent induces some information (fluent values) about the world and, thus, it cannot be certain about it. Uncertainty is not present in the case that we currently tackle, although the possibility of dealing with uncertainty will allow us, in future work, to enrich our agents with the capability of carrying on complex dialogues with the user, in the line of the works of (Bretier and Sadek 1997).

In our application, a state contains fluents that capture those pieces of information about a student's education, that a rational agent believes to be true at a certain time. In particular, the fluents that we use represent:

1. the set of already attended *courses*: for each attended course and for each competence, that is a direct effect of attending that course, there is a fluent **knows(course, competence)** that records the way in which the competence was acquired. Observe that if a course supplies many competences the state will contain as many fluents *knows*(*course*, *competence*). We call the competences that are direct effects of actions *basic* competences;
2. the *competences* that the student has acquired: the state also contains a fluent **has_competence(competence)** for each competence that the student has, independently from the way it has been acquired;
3. the *learning goal* of the student: in this case we use the fluent **requested(goal)**. Notice that we do not strictly partition the set of all the competences in a set of derived competences that is disjunct from a set of the basic competences (related to courses); actually, a same competence may be acquired in different ways: it could either be obtained by learning the smaller pieces of competence it is made of or it could be obtained by attending a single course, if any is available, aimed at teaching that topic. This characteristic is particularly important in the perspective of building an open system, in which courses can be added or removed along time.

As an example, a state may contain the following fluents:

(a) *knows*(*'programming lab', 'java'*).
(b) *knows*(*'programming lab', 'event programming'*).
(c) *has_competence*(*'programming languages'*).
(d) *requested*(*has_competence*(*curriculum*(*'web application'*))).

Fluents (a) and (b) mean that a student has passed the programming course lab exam, thus acquiring the basic competences *java* and *event programming*. The student also acquired the derived competence *programming languages*, see fluent (c). Eventually, fluent (d) states that the student's goal is to acquire the competences supplied by the "web applications" curriculum.

3.3. *Course as actions*

In DyLOG, *primitive actions* are the basic building blocks for defining an agent's behavior. From a modal logic point of view, each primitive action a is represented by a modality $[a]$ (box a). The meaning of the formula $[a]\alpha$ is that α holds after any execution of action a. We can also write: $\langle a\rangle\alpha$ (possible a), whose meaning is that there is a possible execution of action a after which

α holds. The special modality $\Box$ (box) is used to denote those formulas that hold in all states, i.e., after any action sequence.

The direct and indirect effects of primitive actions on states are described by *simple action laws*, which consist of *action laws*, *precondition laws*, and *causal laws*. Intuitively, an action can be executed in a state *s* if the preconditions to the action hold in *s*; the execution of the action modifies the state according to the action and causal laws. We also assume that the value of a fluent *persists* from one state to the next one, if the action does not cause it to change. In the following we will define the simple action laws giving also the syntax that we use in the language for defining them and, last but not least, their representation in the modal logic framework.

1. *Action laws* define *direct* effects of primitive actions on a fluent and can also be used for representing action conditional effects. In the language they have the form:

$$a \textbf{ causes } F \textbf{ if } Fs \qquad (1)$$

where *a* is a primitive action name, *F* is a fluent, and *Fs* is a fluent conjunction, meaning that action *a* has effect *F*, when executed in a state where the fluent precondition *Fs* holds. The modal logic representation for this rule is $\Box(Fs \rightarrow [a]F)$.
2. *Precondition laws* allow action *preconditions*, i.e., those conditions which make an action executable in a state, to be specified. In DyLOG they are written as:

$$a \textbf{ possible if } Fs \qquad (2)$$

meaning that when the fluent conjunction *Fs* holds in a state, the execution of action *a* is possible in that state. The modal logic representation of precondition laws is $\Box(Fs \rightarrow \langle a\rangle true)$.
3. *Causal laws* are used to express *causal dependencies* among fluents and, then, to describe *indirect* effects of primitive actions. In the language they are written as:

$$F \textbf{ if } Fs \qquad (3)$$

meaning that the fluent *F* holds if the fluent conjunction *Fs* holds too. The modal logic representation of such rules is $\Box(Fs \rightarrow F)$.

In the tutoring system that we have implemented, each course is interpreted as the *action* of attending that course, therefore it is represented as a set of simple action laws. As an example, consider Figure 3, in which the representation in DyLOG of the “Programming Lab” course described in

(a) *attend*(*course*(‘*programming lab*’)) **possible if**
    knows(‘*programming theory*’,_).
(b) *attend*(*course*(‘*programming lab*’)) **causes**
    knows(‘*programming lab*’,‘*java*’).
(c) *attend*(*course*(‘*programming lab*’)) **causes**
    knows(‘*programming lab*’,‘*event programming*’).
(d) *attend*(*course*(‘*programming lab*’)) **causes**
    knows(‘*programming lab*’,‘*C and java relationship*’) **if**
    has_competence(‘*C language*’).
(e) *attend*(*course*(‘*programming lab*’)) **causes** *credit*(*B*1) **if**
    get_credits(‘*programming lab*’,*C*) ∧
    credit(*B*) ∧ (*B*1 *is B* + *C*).
(f) *has_competence*(‘*programming*’) **if**
    has_competence(‘*data structures*’) ∧
    has_competence(‘*algorithms*’) ∧
    has_competence(‘*programming languages*’).
(g) *has_competence*(‘*programming languages*’) **if**
    has_competence(‘*java language*’).
(h) *has_competence*(*Competence*) **if**
    knows(_,*Competence*).

Figure 3. Precondition, action, and causal laws.

Figure 1 is shown. Rule (a) states that the action *attend(course(‘programming lab’))* can be executed if the action of attending the “programming theory” course has already been executed. Action laws (b)–(c) describe the unconditional effects of the action execution: adding the “programming lab” course causes to have competence about *Java* and *event programming*. Action law (d) describes the conditional effect of the action at issue. Finally, action law (e) updates other fluents (credit) that control the length of the desired study plan. Rules (f) and (g) describe the *indirect effects* of having a competence; they are used for inferring the higher-level competences of a student based on his known competences. Finally, since the agent reasons on the student’s competences independently from how they were obtained, rule (h) states that if a competence is a direct effect of attending a course (the underscore in *knows*(_,*Competence*) means that we do not care which course was actually attended) it will be a student’s competence (*has_competence*(*Competence*)).

3.4. *Curriculum schemas as procedures*

So far, we have seen the basic building blocks of the DyLOG language. However, in order to be able to represent behaviour strategies we need some compound syntax element. This is given by procedure clauses, which allow us to define complex actions. More precisely, in our language *complex actions* are defined as procedure clauses on the basis of primitive actions, sensing actions, *test* actions[2] and other complex actions. A *complex action* is a collection of *procedure clauses* of the form:

$$p_0 \textbf{ is } p_1, \ldots, p_n (n \geq 0) \tag{4}$$

where p_0 is the name of the procedure and p_i, $i = 1, \ldots, n$, is either a primitive action, a sensing action, a test action, or a procedure name (i.e., a procedure call). Procedures can be recursive and are executed in a goal directed way, similarly to standard logic programs, and their definitions can be nondeterministic as well as in Prolog.

From a theoretical point of view, procedure clauses have to be regarded as axiom schemas of the logic. More precisely, each procedure clause p_0 **is** $p_1, \ldots, p_n$, can be regarded as the axiom schema:

$$\langle p_1\rangle\langle p_2\rangle \ldots \langle p_n\rangle\varphi \supset \langle p_0\rangle\varphi.$$

Its meaning is that if in a state there is a possible execution of p_1, followed by a possible execution of p_2, and so on up to p_n, then in that state there is a possible execution of p_0.

In the curriculum sequencing application, procedures schematize the way for acquiring *professional expertise*. For instance, in Figure 4 we report a little part of the procedures that describe how to acquire a *web applications* expertise. In particular, we have expanded only part of the procedures for acquiring competence about programming. This example will be further discussed in the following sections.

3.5. *Interaction as sensing and suggesting actions*

In the previous sections we have explained that in our approach agents keep a mental state of the situation that they are currently tackling, which is modified by action execution. Generally speaking, however, the agent's knowledge may be incomplete so, in order to understand which actions can be applied, it is sometimes necessary to acquire new information from the outer world. To this aim, we have studied and integrated in the formal account of the language some informative actions, whose outcome is not under the agent's control but depends on the environment; such actions are called *sensing actions*. The difference with respect to the other kinds of actions, that we have seen so far, is that they allow an agent to acquire knowledge about the value of a fluent f rather than to change it.

(a) *achieve_goal*(*has_competence*(*curriculum*('*web applications*'))) **is**
 achieve_goal(*has_competence*('*first year competences*')) ∧
 achieve_goal(*has_competence*('*database*')) ∧
 achieve_goal(*has_competence*('*ai*')) ∧
 achieve_goal(*knows*('*distributed systems*'_)) ∧
 achieve_goal(*has_competence*('*web technology*')).
(b) *achieve_goal*(*has_competence*('*first year competences*')) **is**
 . . .
 achieve_goal(*has_competence*('*programming*')) ∧
 . . .
(c) *achieve_goal*(*has_competence*('*programming*')) **is**
 achieve_goal(*has_competence*('*data structures*')) ∧
 achieve_goal(*has_competence*('*algorithms*')) ∧
 achieve_goal(*has_competence*('*programming languages*')).
(d) *achieve_goal*(*has_competence*('*programming languages*')) **is**
 achieve_goal(*has_competence*('*c*')).
(e) *achieve_goal*(*has_competence*('*programming languages*')) **is**
 achieve_goal(*has_competence*('*java*')).
(f) *achieve_goal*(*has_competence*('*programming languages*')) **is**
 achieve_goal(*has_competence*('*prolog*')).

Figure 4. Procedure clauses.

(a) *achieve_goal*(*knows*(*Course,Competence*)) **is**
 Course ≠ *generic* ∧
 attend(*Course*).
(b) *achieve_goal*(*knows*(*generic,Competence*)) **is**
 ?*u*(*knows*(*generic,Competence*)) ∧
 offer_course_on(*Competence*) ∧
 ?*course_on*(*Competence,Course*) ∧
 attend(*Course*).
(c) *offer_course_on*(_) **possible if** *true*.
(d) *offer_course_on*(*Keyword*) **suggests** *course_on*(*Keyword,*_).

Figure 5. Suggesting actions.

In DyLOG direct effects of *sensing actions* are represented by using *knowledge laws* that have form:

$$s \textbf{ senses } f \tag{5}$$

meaning that action s causes to know whether f holds.

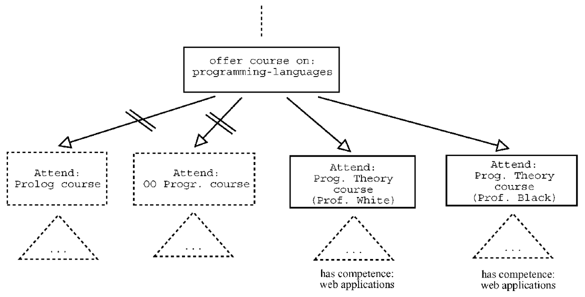

offer course on: programming-languages
Attend: Prolog course
Attend: OO Progr. course
Attend: Prog. Theory course (Prof. White)
Attend: Prog. Theory course (Prof. Black)
has competence: web applications
has competence: web applications

Figure 6. An example of selection of courses to offer for constructing a web applications curriculum.

In the current application, sensing actions are used also for allowing the agent to *interact with the user*. In the simplest case, the user is explicitly requested to enter the truth value of a fluent. This kind of interaction, however, is not sufficient, because rather than asking for a fluent's truth value, it is often more useful to offer a set of alternatives to the user, among which he will make a choice. To this aim, we have defined a special subset of sensing actions, called *suggesting actions*. For representing the effects of such actions we use the notation:

$$s \textbf{ suggests } f \tag{6}$$

meaning that action s suggests a (restricted) set of values for fluent f and causes to know the value of f.

Formally, the difference w.r.t. standard sensing actions is that while those consider as alternative values for a given fluent its whole domain, suggesting actions offer only a subset of it. The agent has an active role in selecting the possible values among which the user chooses: only those values that lead to fulfill the goal will be selected. Such values are identified by means of a reasoning process. After reasoning about a given problem, the agent finds a set of alternative items that are *equivalent* w.r.t. the task of achieving a given goal. Since using one or the other is equivalent, the reasoner may decide to leave the choice up to the user.

For example, let us consider again our application. We have seen that procedures are used to schematize the way to achieve a certain professional expertise in terms of simpler competences to acquire. We have also introduced basic competences, saying that they are all those competences that are supplied by single courses; more than one course could supply the same competence but the system does not necessarily present all of the alternatives to the student. Figure 6 shows an example where the agent is helping a student to build a bioinformatics study plan: at a certain point of the plan construction, the agent finds four alternative courses that give competence about "programming languages", however, only two subtrees allow the student to get competence about imperative languages, necessary for a bioinformatics curriculum (i.e., the actual learning goal). The other branches are cut during the reasoning phase and the corresponding courses are not offered to the student. In different words, only those alternatives that open paths that will lead to the fulfillment of the user's learning goal will be selected. Afterwards, the choice can be left to the user: in fact, whatever the choice, the goal will be reached.

Let us now consider our agent implementation. Professional expertise definitions that, as we will see, are used for accomplishing the task of building a study plan, are expressed by means of the procedure *achieve_goal*, as reported in Figure 4. The *achieve_goal* definition encompasses also two cases, that are described in Figure 5:

rule (a) – some specific course is requested: *knows*(*Course,Competence*)
rule (b) – some competence is requested: *knows*(*generic,Competence*).[3]

Rule (a) states that if a specific course is requested, this is to be added to the study plan (action *attend*(*Course*)). Rule (b), instead, formalizes the case in which we are not interested in a specific course, therefore, the agent searches for all of the possible alternatives and suggests them to the user, waiting for his choice. In the agent implementation the *suggesting action* aimed at offering a set of alternative courses is *offer_course_on*, see rules (c) and (d). In particular, the possible alternatives are extracted from the domain knowledge by *course_on* in rule (d).

4. Tutoring Adaptive Services as Reasoning about Action Tasks

In Section 3 a representation of the virtual tutor domain in terms of a DyLOG domain description has been introduced. On this basis, we can interpret the adaptive services described in Section 2 – building and validating personalized curricula – as "reasoning about actions" tasks.

In general, given a domain described in a logic action framework, the main kinds of reasoning tasks that can be performed are *temporal projection* (or prediction), *temporal explanation* (or postdiction) and *planning* (Sandewall, 1994). Intuitively, temporal projection is a method for reasoning from causes to effects; its aim is to predict the effects of actions, that have not been executed yet, based on (even partial) knowledge about the current state. On the contrary, by performing temporal explanation an agent considers some facts as effects of *already executed* actions and reasons about the possible causes that produced them. In different words, the aim of temporal explanation is to infer knowledge about the past states of the world, starting from some knowledge about the current state. The third reasoning task, *planning*, is probably the best known of the three; it is aimed at finding an action sequence that, when executed starting from a given state of the world, produces a new state where certain desired properties hold. All such reasoning tasks are supported by DyLOG.

In the following, we show how we interpreted the problem of personalized curricula construction as a procedural planning problem. Then, we describe how we interpreted the problem of validating a user-given plan as a temporal projection problem and how to exploit a simple temporal explanation mechanism for helping the student to understand the reasons of validation failure.

4.1. *Building personalized curricula by procedural planning*

Generally speaking, the planning problem amounts to determine, given an initial state *s*, if there is a sequence of actions that, when executed in *s*, leads to a goal state, in which a desired condition *Fs* holds.

In the DyLOG framework we consider a specific instance of the planning problem, in which we wonder if there is a possible execution of a *procedure* *p*, leading to a state in which some condition *Fs* holds, *Fs* being a set of fluents. In the modal language, this problem is expressed by the query $\langle p \rangle Fs$, which is to be read: "given a procedure *p*, is there a terminating execution of *p* (i.e., a finite sequence of primitive actions), leading from an initial state, that corresponds to the current situation, to a state in which *Fs* holds?". In DyLOG we refer to this query with the English-like notation:

$$Fs \textbf{ after } p \tag{7}$$

Intuitively, the *terminating executions* of *p* that lead to the goal state are *plan*s to bring about *Fs*. Indeed, the procedure *constrains* the space in which the plan is sought for; in the literature, this reformulation is known as *procedural planning*. In the curriculum sequencing application, procedures schematize how to acquire *professional expertise* (see the previous section), whereas *Fs* expresses the set of competences that a student would like to acquire. The student, who asked for support in the construction of a study plan, may have already acquired some of the necessary competences before the study plan construction; in that case, the fluents that express his current expertise will be part of the initial state.

The execution of the above query returns as a side-effect an *execution trace* of *p*, i.e., one of the terminating sequences $a_1, \ldots, a_n$ of primitive actions leading to the final state. Such a trace can either be a *linear* or, when the procedure contains sensing or suggesting actions, a *conditional* plan. In fact, if some of the p_i's include sensing or suggesting actions, the obtained execution trace contains also the foreseen communication acts with the user. Due to the fact that their outcomes are unknown at planning time, all the possible alternatives are to be taken into account; therefore, we will obtain a conditional plan, whose branches correspond each to one of the alternative values. It is important to remark that only those values that lead to success will be taken into account and will produce a branch in the conditional plan.

In Figure 4, a set of procedures describing how to acquire the body of competence for a *web applications* curriculum is shown. Let us suppose that our student asked to be supported in the design of a study plan for becoming an expert of *web applications* with the further requirement of acquiring competence about *graph theory*. Furthermore, suppose the student also added some constraints on the length of the plan: it should not exceed 132 credits. This request can be expressed by the query:

$$\{has\_competence(\text{`}graph\ theory\text{'}),\ credit(C),\ (C \leq 132)\}$$
$$\textbf{after}\ achieve\_goal(has\_competence(\text{`}web\ applications\text{'}))$$

where the set of conditions {*has_competence*('*graph theory*'), *credit*(*C*), ($C \leq 132$)} is the set *Fs*, which the student wishes to hold after the execution of the procedure *achieve_goal*(*has_competence*('*web applications*')). If such an execution trace exists, it will correspond to a personalized study plan because, besides achieving the main learning goal *has_competence*('*web applications*'), it will also fulfill the additional requirements contained in *Fs* by supplying the competence *graph theory* and not exceeding 132 credits. Let us suppose that our student is not a beginner because he already attended the *first year* university studies and also some courses that supply the *database* competence. The system will suggest a course sequence for achieving only the missing competences, which (with reference to Figure 4) are *artificial intelligence*, *distributed systems* and *web technology*. Figure 7 sketches the conditional plan that results from planning in the described situation. The plan specifies a set of courses that the student is recommended to attend. The branching points of the conditional plan are to be interpreted as questions that will be posed to the student at execution time; by answering all the questions the student will select one of the alternative study plans. Queries are inserted in the conditional plan during its construction. In fact, when the planning process finds that different courses supply a required competence, it introduces an interactive action for offering all of the alternatives to the student.

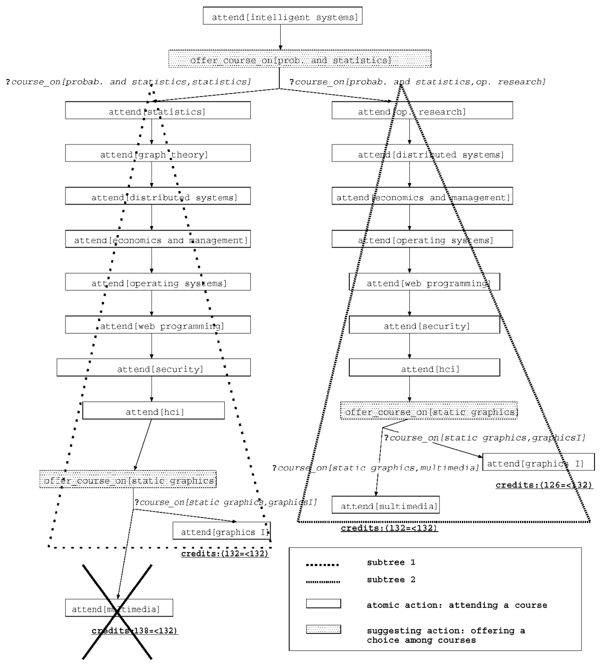

attend[intelligent systems]
offer_course_on[prob. and statistics]
?course_on[probab. and statistics,statistics]
?course_on[probab. and statistics,op. research]
attend[statistics]
attend[graph theory]
attend[distributed systems]
attend[economics and management]
attend[operating systems]
attend[web programming]
attend[security]
attend[hci]
offer_course_on[static graphics]
?course_on[static graphics,graphicsI]
attend[graphics I]
credits:(132=<132)
attend[multimedia]
credits:(138=<132)
attend[op. research]
attend[distributed systems]
attend[economics and management]
attend[operating systems]
attend[web programming]
attend[security]
attend[hci]
offer_course_on[static graphics]
?course_on[static graphics,graphicsI]
?course_on[static graphics,multimedia]
attend[graphics I]
credits:(126=<132)
attend[multimedia]
credits:(132=<132)
subtree 1
subtree 2
atomic action: attending a course
suggesting action: offering a choice among courses

Figure 7. An example of conditional plan for a personalized curriculum in "web applications", that does not exceed the requested 132 credits and guarantees to acquire competence on graph theory.

Further suggestions are committed to the user's choice. For instance, in the plan above after recommending the student to attend the "intelligent systems" course, the system offers the student the choice between two different courses – i.e., "statistics" and "operations research" – which supply competence on *probability and statistics*. Further recommendations depend on the student's decision. For instance, when the preferred option is "operations research" (right subtree, Figure 7), the "graph theory" course is not added to the study plan, because the "operations research" course already provides the compet-

ence that this course supplies. At the end of that path another branching point corresponds to a query where the student must choose between two courses ("multimedia" and "graphics I") that supply competence on *static graphics*. Notice that such a choice is not given in case the student previously selected the "statistics" course. This is because of the user's constraint on credits. In fact, at the bottom of the left subtree of Figure 7 only one of the courses allows the system to build a study plan that does not exceed 132 credits. The "multimedia" course is too long (the study plan would be 6 credits longer than requested), therefore, it is not taken into account.

Finally, let us stress that all of the branches of the conditional plan lead to a state where the condition to have competence on *graph theory* is satisfied. Intuitively, it means that, no matter the alternatives preferred by the user during the actual interaction, it is guaranteed that the resulting study plan leads to the fulfillment of the learning goal. The addition of further requirements about the competence to achieve affects the generated conditional plan, causing the cut of some branches. If, for instance, our student asks also for competence about *audio*, since such competence is provided only by the "multimedia" course, the system would insert in the conditional plan only those paths that contain the "multimedia" course; with reference to Figure 7, those paths that contain the "graphic I" course would be cut.

4.2. *Validation of curricula by temporal projection*

Another reasoning process that is extremely interesting in the context of personalized course sequencing is the *validation of a student-given study plan*. In this case, after defining the course sequence according to his personal taste and interests, the student asks the system if it satisfies the learning dependencies of the domain, allowing to achieve some desired learning goal (some specific expertise).

This task can easily be interpreted as a *temporal projection* problem. In spoken words, the temporal projection problem is defined as: "given an initial state s and an action sequence $a_1, \ldots, a_n$, does the condition Fs hold after the execution of the action sequence?". Differently than in planning, in this case the action sequence is given. In the logical framework that we defined, we formalize the temporal projection task by means of the query:

$$\langle a_1 \rangle \ldots \langle a_n \rangle F_s$$

where each a_i is a primitive action and Fs is a conjunction of fluents. Notice that, since the primitive actions defined in our domain descriptions are *deterministic* w.r.t. the epistemic state, the equivalence $\langle a \rangle Fs \equiv [a]Fs \wedge \langle a \rangle \top$ holds for all the actions a that are defined in the domain description. Therefore, the success of the existential query $\langle a_1 \rangle \ldots \langle a_n \rangle F_s$ entails the success of the universal query $[a_1] \ldots [a_n] Fs$.

In DyLOG we represent the query that formalizes the temporal projection task by the English-like notation:

$$Fs \textbf{ after } a_1, \ldots, a_n$$

In the curriculum sequencing context, the sequence of actions is a sequence of courses $c_1, \ldots, c_n$ to attend while the final condition Fs is a set of desired competences. The temporal projection task can then be read as "Given the initial student background, can the student acquire the set of competences Fs by attending the course sequence $c_1, \ldots, c_n$?". In the implementation this query becomes:

$$Fs \textbf{ after } attend(course(c_1)); \ldots ; attend(course(c_n)) \tag{8}$$

where Fs is the student's learning goal.

As a first, simple example, let us consider a student, who asks his tutor if, by attending the course sequence "programming theory" followed by "programming lab" – in the given order –, he will acquire competence about *event programming* (which is the student's learning goal). Let us also suppose that the student has no previously acquired competence about programming or other related topics. The corresponding DyLOG query will be:

has_competence('*event programming*') **after**
    attend(*course*('*programming theory*'));
    attend(*course*('*programming lab*'))

Since all the learning dependencies are respected and attending the "programming lab" course causes the acquisition of competence about *event programming* (see Figure 3), the validation of this query will succeed.

Indeed in our framework *validation may fail* for *two reasons*: either the preconditions to one of the actions do not hold in the state in which the action is executed (sequencing problem) or the learning goal is not achieved. For instance, the query:

has_competence('*event programming*') **after**
    attend(*course*('*programming lab*'));
    attend(*course*('*programming theory*'))

would fail because the "programming lab" course cannot be attended if the student does not have competence about *programming theory*, which is the effect of attending the "programming theory" course. On the other hand, the query:

has_competence('*multimedia*') **after**
    attend(*course*('*programming theory*'));
    attend(*course*('*programming lab*'))

would fail because none of the two courses in the sequence allows the student to acquire competence in *multimedia*, although the action sequencing is correct.

4.3. *Explanation of validation failure by temporal explanation*

As we have seen above, the validation of a user-built plan may fail for different reasons: the plan may be wrong because it does not allow to reach the final desired competence (the action sequence is correct but it does not lead to the learning goal) or because the sequencing does not respect some of the learning dependencies (the student does not have the necessary competence for attending the next course in the plan). In either case, it is extremely important to return a feedback to the user about the reasons of validation failure, in order to support plan correction.[4]

Course sequencing is, actually, quite a special application domain. Its peculiarity is that competences can only be added. Intuitively, no new course will ever erase from the students' memory the concepts acquired in previous courses. More formally, the domain is *monotonic*. This consideration is very helpful in the definition of a failure explanation mechanism; the one that we propose is based on the notion of state *completion* and exploits a mechanism known as *temporal explanation*. The information that we compute to return as a feedback is the set of competences that the student should already have in order for the plan to be valid. So, for instance, if a student adds to his study plan the course "programming lab" but, standing to the information that the system has, the student does not have notions about "programming theory", the system will tell the student that he can attend his plan only if he already has notions about programming theory, otherwise he will not be able to understand the contents of the "programming lab" course.

More formally, a task of temporal explanation amounts to reconstruct, starting from some given observations, what has happened (more generally, what should have happened) in order for those observations to be true. For dealing with temporal explanation, we adopt an *abductive approach* by determining the assumptions on the initial state that are needed for explaining observations on later states. In the case of courses and study plans, such assumptions will be a set of competences: all those competences that the student did not declare to have and that are not supplied by the courses in the sequence up to a state in which they result to be necessary. The intuitive idea is that a study plan is always applicable, given that the student has as background knowledge the missing competences.

While the reasoning mechanisms of planning and temporal projection, used in Sections 4.2 and 4.1, are based on the proof procedure described in (Baldoni et al. 2001b), *temporal explanation* is based on the work contained in (Baldoni et al. 1997). In that work an abductive proof procedure is defined in terms of an auxiliary nondeterministic procedure *support*, which carries out the computation for the monotonic part of the action language. Given a domain description Π and a query of form:

$$Fs \ \mathbf{after} \ attend(course(c_1)); \ldots ; attend(course(c_n))$$

let us pose, for the sake of simplicity, $Q = \langle attend(course(c_1))\rangle \ldots \langle attend(course(c_n))\rangle Fs$, then, the procedure $support(Q,\Pi)$, described hereafter, returns an abductive support for the query Q in Π, which is a set Δ of abducibles that, when added to the domain description, allows us to derive our query:

$$\Pi \cup \Delta \vdash_{vs} [add(course(c_1))] \ldots [add(course(c_n))]Fs$$

Briefly, abducibles are atomic propositions of the form $\mathbf{M}[a_1]\ldots[a_n]F$ (F being a fluent), where $\mathbf{M}$ is not to be regarded as a modality: this notation has been adopted in analogy to default logic and $\mathbf{M}[a_1]\ldots[a_n]F$ means that F is consistent after the execution of the sequence of actions $a_1, \ldots, a_n$.

All the details concerning the implementation of the monotonic part of the language are hidden in the definition of the procedure *support*, and they can be ignored by the abductive procedure. The abductive procedure is defined in the style of Eshghi and Kowalski's abductive procedure for logic programs with negation as failure (Eshghi and Kowalski 1989), and it is similar to the procedures proposed in (Toni and Kakas 1995) to compute the acceptability semantics. Here we report only the support procedure, that was modified in the following way, while the abductive procedure remained unchanged (see (Baldoni et al. 1997, Section 4):

1. $a_1, \ldots, a_m \vdash_{vs} \top$ with $\emptyset$;
2. $a_1, \ldots, a_m \vdash_{vs} F$ with Δ if
   a) $a_1, \ldots, a_{m-1} \vdash_{vs} Fs'$ with Δ, where $m > 0$ and $\Box(Fs' \supset [a_m]F) \in \Pi$; or
   b) $a_1, \ldots, a_{m-1} \vdash_{vs} F$ with Δ_1 and $\Delta = \Delta_1 \cup \{\mathbf{M}[a_1, \ldots, a_m]F\}$; or
   c) $m = 0$ and $\Delta = \emptyset$ if $F \in S_0$, $\Delta = \{\mathbf{M}F\}$ otherwise;
3. $a_1, \ldots, a_m \vdash_{vs} Fs_1 \wedge Fs_2$ with $\Delta_1 \cup \Delta_2$ if $a_1, \ldots, a_m \vdash_{vs} Fs_1$ with Δ_1 and $a_1, \ldots, a_m \vdash_{vs} Fs_2$ with Δ_2;
4. $a_1, \ldots, a_m \vdash_{vs} \mathcal{M}\ell$ with Δ if $a_1, \ldots, a_m \vdash_{vs} \mathcal{B}\ell$ with Δ;
5. $a_1, \ldots, a_m \vdash_{vs} [a'_1, a'_2; \ldots ; a'_n]Fs$ with $\Delta_1 \cup \Delta_2$ if $a_1, \ldots, a_m \vdash_{vs} Fs'$ with Δ_1 and $a_1, \ldots, a_m, a \vdash_{vs} [a'_2; \ldots ; a'_n]Fs$ with Δ_2.

(a) *attend*(*course*(‘*database*’)) **possible if**
has_competence(‘*matrices*’) $\wedge$
has_competence(‘*dynamic structures*’).

(b) *attend*(*course*(‘*database*’)) **causes**
knows(‘*database*’,‘*db*’).

(c) *attend*(*course*(‘*database*’)) **causes** *credit*(*B*1) **if**
get_credits(‘*database*’,*C*) $\wedge$ *credit*(*B*) $\wedge$ (*B*1 *is B* + *C*).

Figure 8. Action and precondition laws for the course *database*.

To prove a fluent F, we can either select a clause in the domain description Π, rule 2(a), or add a new assumption to the assumption set Δ, rule 2(b) and 2(c). A query $\langle a_1\rangle \ldots \langle a_m\rangle Fs$ can be derived from a domain description Π with assumptions Δ if, using the rules above, we can derive $\varepsilon \vdash_{vs} [a_1] \ldots [a_m]Fs$.

As an example, let us consider again a student who would like to acquire competence about *event programming*, this time by attending the courses “database” and “programming lab” in the given order. Let us suppose that, standing to the information that the system has, the student has no notion about *programming theory*, which is a prerequisite for attending the *programming lab* course (Figure 3, clause (a)), nor about *dynamic structures* and *matrices*, that is necessary in order to attend the “database” course (Figure 8). The request of the student is represented by the following query:

(q_1) *has_competence*(‘*event programming*’) **after**
attend(*course*(‘*database*’));
attend(*course*(‘*programming lab*’))

Given a domain description Π, that includes both the simple action laws for “programming lab” and the simple action laws for the course “database”, the proof procedure returns a support Δ for the query (q_1), that contains the following assumptions on the initial state: $\mathbf{M}\mathcal{B}$ *has_competence* (‘*matrices*’), $\mathbf{M}\mathcal{B}$ *has_competence* (‘*dynamic structures*’) and $\mathbf{M}\mathcal{B}$ *knows* (‘*programming theory*’, ‘*object languages*’). Intuitively, Δ is the tutor feedback about the proposed plan, which can be read as: “The plan that you proposed would allow you to learn *event programming* if you already had notions about programming theory, matrices and dynamic structures”.

This failure explanation mechanism is quite simple and we would like to remark that it works due to the monotonicity of the domain that we are tackling, where new courses are not supposed to erase from the students' memory the concepts acquired in previous courses.

5. The Virtual Tutor as a Logic Agent

In our work we used the DyLOG language not only for representing the knowledge domain (as described in details in the previous sections) but also for programming the agents that use such a knowledge, i.e., for implementing the reasoners. In the specific application that we are presenting, at a very high-level reasoners have the following behavior:

1. acquire a problem definition from the user;
2. acquire the initial situation;
3. solve the problem and present a solution to the user;
4. further adapt the solution by interacting with the user.

Sometimes the solution can be achieved with no further interaction; more generally, however, the agent will ask the user for further information or for choosing among equivalent (w.r.t. the goal) alternatives, when it is the case. The whole communication between the two actors (the reasoner and the user) takes place by means of web pages that are constructed on the fly for presenting (requesting) information to (from) the user. Of course, although at an abstract level the different kinds of problem are fixed ("help me to build a study plan" or "validate this study plan"), there may be a wide variety of specific interests and interactions depending on the user and on his/her goals and situation. For each triple $\langle$*user,goal,situation*$\rangle$, a specific interaction will occur and, therefore, an ad hoc web page sequence will be generated.

Our reasoners are executed by the DyLOG interpreter, which is a straightforward implementation of the language proof procedure (Baldoni et al. 2001b). Every *primitive action* has some *code* associated to it, that is to be performed when the action is executed (the association is done by means of the keyword **performs**); such a code actually produces the effects of the action in the world. For instance, when the reasoner must show some information to the user, it executes a *showpage* action, which has associated some code for asking another agent, the actual execution device (see Section 6), to show an appropriate web page to the user. As a consequence, when the interpreter *executes* an action it must commit to it and it is not allowed to backtrack by retracting the effects of the executed action.[5]

However, reasoners perform rational tasks by *reasoning about* actions effects. In Section 4 we have seen that all the different kinds of reasoning exploit a query of the form *Fs* **after** *p*; the language interpreter provides a few meta-predicates for reasoning about actions in order to answer to this kind of query. More specifically, the meta-predicate:

$$plan(Fs\ \textbf{after}\ p,\ as)$$

extracts a primitive action sequence *as* that, given a specific initial state, is a possible execution of procedure *p* that leads to a state in which *Fs* holds. Procedure *plan* works by executing *p* in the same way as the language interpreter with a main difference: primitive actions are executed "in the mind of the reasoner", without any effect on the external environment and, as a consequence, they are backtrackable. The meta-predicate *plan* is used both to perform *study plan construction* (see Section 4.1) and for *validation* (see Section 4.2). The explanation of validation failure, instead, is accomplished by means of the meta-predicate *explain*(*Fs* **after** *as,d*), that collects in *d* all the fluents that should be true in the initial state in order for *Fs* to hold after the execution of the sequence *as* of primitive actions.

5.1. *Implementing the virtual tutor in* dyLOG

The *behaviour* of a reasoner is described by a collection of procedures. In the case of *study plan construction*, see Section 4.1, the top level procedure, called *advice*, extracts a plan that will be executed. In the following, a question mark in front of a fluent means that the value of that fluent is to be checked. So, for instance, ?*requested*(*Curriculum*) will check which professional expertise the student declared to be interested in. In this case the fluent has a predefined finite domain.

(R1) $advice(Plan)$ **is**
$$\begin{array}{l} ask\_user\_preferences \wedge ?requested(Curriculum) \wedge \\ plan(credits(C) \wedge max\_credits(B) \wedge (C \leq B)\ \textbf{after} \\ \quad achieve\_goal(has\_competence(Curriculum), Plan) \wedge Plan. \end{array}$$

Intuitively, the reasoner asks the student what kind of final expertise he wants to achieve and his background knowledge (e.g., if he already attended some of the possible courses). Afterwards, it adopts the user's goals and builds a *conditional plan* for reaching them, predicting also the future interactions with the user. That is, if it finds different courses that supply a same competence, whose prerequisites are satisfied, it plans to ask the user to make a choice. *plan* is the *meta-predicate* that actually builds the plan, in this case by extracting those executions of the procedure *achieve_goal* that satisfy the user's goals as well as the further conditions that are possibly specified (e.g., that the number of credits gained by following the study plan is not bigger than a predefined maximum).[6]

Eventually the conditional plan that is returned by the reasoning process is executed. This means that the code that is associated to every primitive or suggesting action, that is part of the returned plan, is executed, possibly modifying the environment. In our application a plan can only consist of two different kinds of actions: the primitive action *attend* and the suggesting action *offer_course_on* (see Section 3.5). Rules (R2) and (R3) contain the code associated to such actions:

(R2) *attend*(*Course*) **performs** (
  showCourse(*Course*)).

(R3) *offer_course_on*(*Keyword*) **performs** (
  build_question(*Keyword_Question*) $\wedge$
  ask_choice(*Question,Choice*)).

showCourse is a Prolog predicate that performs a FIPA-like communication with the executor (see next section) for commanding the visualization of a web page containing all the information about the course *Course*. The predicate *build_question* composes a question that suggests a set of alternative courses to the user asking for his preference; the question is stored into variable *Question*. Last but not least, *ask_choice* takes care of asking the composed question to the user, and then waits for the answer (which is stored in the variable *Choice*).

Initially the agent does not have explicit goals, because no interaction with the student has been performed. The student's inputs are obtained after the first interaction phase, carried on by the procedure *ask_user_preferences*:

(R4) *ask_user_preferences* **is**
  verify_student_competence $\wedge$
  offer_curriculum_type.

verify_student_competence is an action that allows the system to acquire knowledge about the current student's situation: mainly, which courses have been attended and successfully passed. *offer_curriculum_type*, instead, is used to acquire knowledge about the professional expertise the student would like to achieve. In Section 3.5, we have seen that in DyLOG information is acquired by means of special actions, called *sensing actions*. Differently than "normal" actions, they increase (or revise) the knowledge of the agent but they do not change its environment; indeed, *offer_curriculum_type* is an example of sensing action:

(R5) *offer_curriculum_type* **possible if** *true*.
(R6) *offer_curriculum_type* **senses** *requested*(*Curriculum*).

It is defined by means of both a precondition law that states that this action can always be executed (R5) and a sensing action law (R6), which states that, after the execution of *offer_curriculum_type*, the value of the fluent *requested*(*Curriculum*) – used in (R1) – will be known. Here the *goal adoption* occurs: the goal of the user becomes the goal of the reasoner.

In the case of *study plan validation*, see Section 4.2, the top level procedure is *check_study_plan*. This procedure – see (R7) –, after executing *verify_student_competence* that we have already explained, first interacts with the student so to get the study plan that he built (*ask_curriculum*(*Plan*)) and then asks him to input the competences he is interested in (*ask_desired_competence*(*Competence*)). Afterwards, it executes *Check*(*Plan, Competence*), which performs the actual validation.

(R7) *check_study_plan* **is**
  verify_student_competence $\wedge$
  ask_curriculum(*Plan*) $\wedge$
  ask_desired_competence(*Competence*)$\wedge$
  check(*Plan,Competence*).

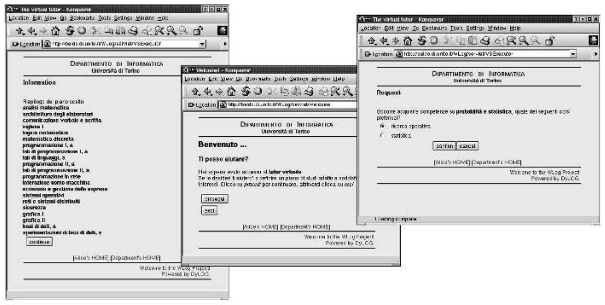
(R8) *check*(*Plan,Competence*) **is**
  plan(*Competence* **after** *Plan*,_) $\wedge$
  showpage("Your plan is OK").

(R9) *check*(*Plan,Competence*) **is**
  showpage("Your plan is not OK") $\wedge$
  explain(*Competence* **after** *Plan*,*Delta*) $\wedge$
  showpage("Explanation:",*Delta*).

(R8) uses again the meta-predicate *plan*, which executes the query *Competence* **after** *Plan*; however, in this case, we are only interested in checking if the sequence of actions contained in *Plan* allows to achieve the requested competences: for this reason we discard the meta-predicate return value (second argument). The agent will return to the user an appropriate feedback by using the primitive action *showpage*, according to the result of the validation procedure. If the plans turns out to be wrong, rule (R9) executes the metapredicate *explain*, which, according to the approach described in Section 4.3, collects in variable *Delta* a list of competences that the student should already have in order to acquire the target *Competences* by following *Plan*.

6. The Multiagent System

Wlog, the prototype system that we developed, has the *multi-agent architecture* that is sketched in Figure 10. Agent technology allows complex systems to be easily assembled by means of the creation of distributed artifacts, that can accomplish their tasks through cooperation and interaction. Systems of this kind have the advantage of being modular and, therefore, flexible and


The virtual tutor - Konqueror

Dipartimento di Informatica
Università di Torino

Information

Riepilogo del piano scelto:
analisi matematica
architettura degli elaboratori
comunicazione verbale e scritta
inglese I
logica matematica
matematica discreta
programmazione I, a
lab di programmazione I, a
lab di linguaggi, a
programmazione II, a
lab di programmazione II, a
programmazione in rete
interazione uomo macchina
economia e gestione delle imprese
sistemi operativi
reti e sistemi distribuiti
sicurezza
grafica I
grafica II
basi di dati, a
sperimentazioni di basi di dati, a

continue

[Alice's HOME] [Department's HOME]

Welcome to the WLog Project
Powered by DyLOG.

Welcome! - Konqueror

Dipartimento di Informatica
Università di Torino

Benvenuto ...

Ti posso aiutare?

Hai appena avuto accesso al tutor virtuale.
Se lo desideri ti aiuterò a definire un piano di studi adatto a soddisf
interessi. Clicca su *prosegui* per continuare, altrimenti clicca su *esci*

prosegui
esci

[Alice's HOME] [Department's HOME]

Welcome to the WLog Project
Powered by DyLOG.

The virtual tutor - Konqueror

Dipartimento di Informatica
Università di Torino

Request

Occorre acquisire competenze su **probabilità e statistica**, quale dei seguenti corsi preferisci?
(●) ricerca operativa
( ) statistica

confirm cancel

[Alice's HOME] [Department's HOME]

Welcome to the WLog Project
Powered by DyLOG.

Figure 9. Interacting with Wlog.

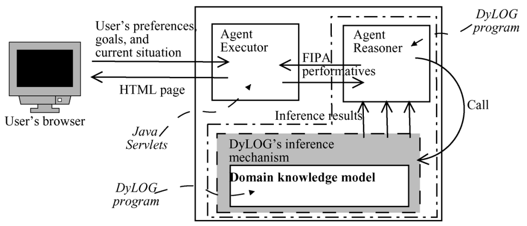


Figure 10. A sketch of Wlog architecture.

scalable. So, on one hand, each module can be developed by exploiting the best, specific technology for the service that it supplies, on the other, new components can be added for supporting either new functions or a wider number of users.

Wlog consists mainly of two kinds of agents: *reasoners* and *executors*. Reasoners are written in DyLOG, whereas executors are Java servlets embedded in a Tomcat web server. Executors are the interface between the rational agents and the users; they mainly produce HTML pages, driven by the directives sent by reasoners, and they forward the collected data to the reasoners themselves. Reasoners collect inputs from the users (preferences, goals, information about the current educational situation) and invoke the inference mechanism of the DyLOG language (see Section 4) on the domain knowledge model in order to accomplish one of the possible adaptive services, i.e., building a study plan or validating a student-given study plan.

As we have seen, also the domain knowledge model is defined in the DyLOG language. We would like to remark that while the use of DyLOG for representing the knowledge model and performing inferences is fundamental, the agent implementation described in Section 5 is written in DyLOG for convenience and it could, actually, be written in other programming languages, such as Java, the important thing being that the implementations call the DyLOG meta-predicates *plan* and *explain*, which perform the actual reasoning process.

The communication among the agents has the form of a FIPA-like message exchange in a distributed system (FIPA 1997). Each agent is identified by its location, which can be obtained by other agents from a facilitator, and has a private mailbox where it receives messages from other agents.

6.1. *Interaction between a tutor, an executor, and a user*

A user accesses the system by means of a normal web browser (Figure 9); from this moment until the end of the interaction, the user will be served by an *executor*. First the executor looks for a free reasoner by consulting the facilitator; since at the moment reasoners are not differentiated and can all perform all the different kinds of reasoning, that we have described in the previous sections, if any is available the interaction will begin.

Supposing that the previous step was successful, the user selects the service he is interested in and starts his interaction with the system. The next step will be the declaration of the user's goal, e.g., "I want to become an expert of web applications". The user's goal is adopted by the reasoner, that will start a conversation aimed at collecting information about the user's current situation. For instance, in the case of study plan construction the user will be asked about successfully passed exams. In the case of study plan validation, instead, the system will ask the study plan to validate. Of course, if our reasoning system were integrated in a wider system that is, for instance, connected with the secretariat databases, part of the information would be available without asking and the resulting interaction with the user would be simpler, although the kernel of the system would not change. At this point it is extremely interesting to understand how the interaction between the reasoner and the executor is carried on.

Figure 11, reports a finite state automaton, that represents the *interaction protocol* between the members of each couple ⟨reasoner, executor⟩. States are numbered and arcs are labelled with the speech acts that cause the various transitions. Different shading on states are used for specifying which agent will continue the conversation (white for the executor, gray for the reasoner). States with double border are terminating states.

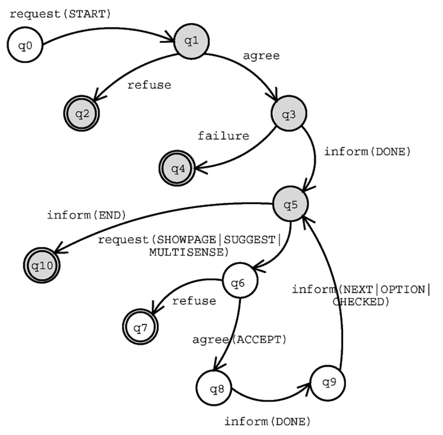


Figure 11. Communication protocol between an executor and a reasoner.

The part of graph that contains the states q_1 through q_4 encodes the connection of an executor with a reasoner (initialization phase). The part of graph consisting of the $q_5 - q_{10}$ states rules the actual *action-execution* cycle, i.e., the execution of primitive actions, commanded by the reasoner and performed by the executor. As we have seen in the previous Sections, in the application hereby described only a little number of primitive actions are defined: those necessary for sensing the inputs from the user plus *attend*(*Course*) and *offer_course_on*(*Keyword*). The former causes the description of a course to be displayed to the user, the latter causes an interaction in which, first, some alternatives are shown and, then, the user's choice is expected. Whenever a primitive action is executed an appropriate web page is to be produced and sent to the user's browser. The action-execution cycle takes care of this phase. The reasoner sends the executor the request of showing an HTML page by means of the *request* FIPA speech act from q_5 to q_6, completed with values that specify what to show, suggest or sense. The executor composes a proper HTML page and sends it to the user's browser; sometimes the page will contain a form to be filled. In either cases, when the user finishes to consult/fill the page/form, he asks the system to continue by clicking a button. The executor, then, informs the reasoner that the page has been consulted and, if necessary, also transmits to the reasoner the user's

data (*inform*() speech act from q_9 to q_5). Afterwards, it waits for the next command.

Both agents perform various controls on the messages that they receive, for guaranteeing the integrity of the interaction. For instance, if an executor receives a command from a reasoner, which is not serving its user, it will refuse to execute it. The same would happen if it were asked to execute an action that is not allowed in the current state. So if it has sent to the user's browser a form and it has not received any information in return, it will refuse to send to the browser any other page.

7. Conclusion and Related Work

In this article, we have presented an approach to adaptive tutoring, based on the use of a logic programming language that supports reasoning about actions and change. In our approach a group of agents, called reasoners, works on a real-world domain description, given in terms of a set of competences, which are connected by causal relationships. The set of all the possible competences and of their relations defines an ontology. This multi-level description of the domain bears along many advantages. The most straightforward is the simplicity of use of the system: on the one hand, no initialization phase is required (differently than in other, e.g., statistical, approaches); on the other hand, we can add, delete, modify course descriptions as well as expertise descriptions without affecting the system behavior because of the high modularity that this approach to knowledge description manifests. Last but not least, working at the level of competences is close to human intuition and enables both goal-directed reasoning processes and explanation mechanisms.

The logic approach also enables the validation of student-given study plans with respect to some learning goal of interest to the student himself. Basically the reasons for which a study plan may be wrong are two: either the sequentialization of courses is not correct or by attending that plan the student will not acquire the desired competence. In both cases we can detect all the weak points in the plan and return a precious feedback to the user. An interesting extension would be to automatically build what, according to the terminology proposed in (Baral et al. 2000), is known as a *repair plan*: an automatic correction of the wrong proposal. At the moment, however, we do not enact repair-planning policies. In fact, although at a first glance, it could seem that repairing a study plan means to complete it by adding some missing courses, the problem is actually not trivial. For instance, what to do if the patched plan violates some constraint (e.g., it is too long)? Should the system eliminate courses that the student chose but that are not really necessary for acquiring the declared learning goal? What about adaptation in this case? We believe that repairing requires a close interaction between the system and the user, whose dynamics are yet to be investigated.

In our implementation, both the study plan construction and the study plan validation tasks are performed on-line. In the case of planning we could actually have followed an alternative approach: to build off-line the most general conditional plan for each professional expertise and to limit the on-line phase to a tree pruning, according to the inputs given by the user. However, this solution would not be efficient in the case in which the user asks to build a plan for achieving a generic set of competences (rather than a professional expertise out of the set offered by the system) nor in the case of plan validation. In fact, discovering whether a sequence of actions is an instance of a schema by matching the schema tree has a higher computational complexity than verifying its correctness by applying *temporal projection*, which is linear in the number of the elements in the sequence.

The approach that we proposed can generally be adopted for building recommendation systems. For instance, besides the application that we presented in this article, we used procedural planning also for building a prototype system that helps users to assemble personal computers according to their needs (Baldoni et al. 2001a). However, it is possible to widen the set of possible applications moving to the design of virtual supervisors. Presently we are, actually, working at a new application in which a student learns how to use an application software in a learning by doing framework. A virtual tutor silently monitors the user by verifying if and how he/she reaches a learning goal proposed by the system. One of the problems to solve in this context is to ignore useless actions, that the user performs either because he/she has little acquaintance with the software or because he/she is actually exploring menus and commands. Systems for helping the users to familiarize with softwares are already being developed, the problem is that usually they are simulators, whose design and implementation are very expensive. By reasoning on the user's actions, we think that the production of such systems would be simplified.

7.1. *Other approaches to curriculum sequencing*

In the Information Technology society, the field of adaptive hypermedia applied to educational issues is attracting greater and greater attention (Brusilovsky 2001). In the last few years considerable advancements have been yield in the area, with the development of a great number of systems, like ELMArt (Weber and Brusilovsky 2001), the KBS hyperbook system (Henze and Nejdl 2001), TANGOW (Carro et al. 1999), the authoring tool for course designing ATLAS (Macías and Castells 2001) and many others. Among the

technologies used in Web-based education for supporting student adaptation and guidance, *curriculum sequencing* is one of the most popular. Different methods have been proposed on how to determine which reading (or study) path to select or to generate in order to support in an optimal way the learner navigation through the hyperspace of knowledge items, see e.g. (Brusilovsky 2000; Weber and Brusilovsky 2001; Stern and Woolf 1998; Henze and Nejdl 2001).

In the last sections we described the usefulness of three techniques for reasoning about actions, based on logic, in a curriculum sequencing applicative framework; in the following we compare our application with some other Adaptive Educational Hypermedia systems which also implement curriculum sequencing even if in slightly different application frameworks. Our analysis will not be exhaustive – we have focused on a set of representative systems – and it is inspired by the concept-driven comparison framework defined in (Baldoni et al. 2002).

Let us start with the *KBS Hyperbook System* (Henze and Nejdl 1999, 2000), an AEH system which personalizes the access to information according to the particular needs of a student. KBS implements various adaptational functionalities, among which the generation and proposal of reading sequences through a hyperspace of information sources about Java programming. As in our case, in the KBS Hyperbook framework knowledge and actual information units are kept separate. The learning dependencies, used by the adaptation component of the system for the sequencing task, are expressed at the knowledge level. They are stored in a so called *knowledge model*, which contains the knowledge prerequisites required for understanding some concept, as well as the resulting knowledge. Curriculum sequencing allows the KBS system to compile a multi-step sequence of pages for helping a user to reach a certain learning goal. Such a sequence is compiled by following a stochastic approach that performs a depth first traversal of the knowledge model.

Our work also focuses on dependencies among knowledge elements (competences), even though, when necessary, also dependencies among the actual information items (the courses) can be expressed. As a difference, while sequencing in Henze and Nejdl (1999) is based upon a Bayesian approach, producing a partial order of knowledge elements, we adopted a symbolic approach based on a *modal logic theory of action*. In KBS dependencies between knowledge elements have the form $K1 < K2$, expressing the fact that $K1$ should be learned before $K2$. Therefore, the inferencing mechanism that enables the system to understand the dependencies between sets of knowledge elements is the transitive closure of the "$<$" relation. In our case, since information items are represented as "attend course" actions that require or produce competences (our knowledge elements), the dependencies between information items and knowledge elements emerge by logical reasoning about "attend course" actions, using all the information modelled in the action theory. Indeed learning dependencies are inferred by logical derivation not only from the knowledge elements, which are in the precondition and effects of the courses, but also from the hierarchical structure among knowledge elements, encoded by causal laws, and from the specification of schematic professional expertises expressed by procedures.

One characteristic of our approach, the decoupling of knowledge from the set of the courses that are available at a specific time, makes it suitable to extensions to a more open framework and, in particular, to the development of open systems, where different sources of information are integrated. An example application could be supporting those students that apply to the Erasmus (or other) interchange program in choosing a set of courses to attend abroad. In this context an advantage of our approach is that, due to the fact that by means of DyLOG procedures we can express different composition strategies, we could specify different teaching policies (organizations of information presentation).

In the ELM-Art system (Weber and Brusilovsky 2001) curriculum sequencing is used for compiling a sequence of hypermedia documents that a student will follow for reaching a certain learning goal. As a difference with respect to KBS, in ELM-Art there is no distinction between knowledge elements and information items, thus the learning dependencies used by the adaptation component are coded at the level of the information units. Based on this model of dependencies, the system does not produce a multi-step reading sequence but suggests to the student the *next best page* to visit, which is calculated based on the reading path actually followed by the student and on the page prerequisites. A similar approach is taken in ACE (Specht and Oppermann 1998), a WWW-based tutoring framework that influenced the development of the recent WIND project (Specht et al. 2002). In ACE the domain model is built on a conceptual network of learning units: it describes a set of learning units and their interrelations and dependencies, without drawing a distinction among knowledge elements and information units. Besides prerequisite relationships among units, that specify a partial order of units in the learning space, the model can contain also a default curriculum sequence. The combination of these elements is used for adapting the student learning sequence step by step, according to the student's current knowledge. In particular ACE computes the *next best unit* to work on, depending on the probabilistic overlay model of a learner's knowledge and the prerequisites of the possible next units.

MetaLinks (Murray 2002), an authoring tool for adaptive hyperbooks, implements a functionality, "the narrative flow", that allows it to suggest step by step a reading path in a hyperspace of documents. Instead of coding learning dependencies in the usual way, i.e., by associating preconditions and outcomes to information units, MetaLinks represents *decompositional dependencies* by structuring the documents hierarchically in a way that parents are introductions or summaries of their children, while children detail the matter introduced by the parents. Based on this structure, the next page to visit is suggested by adopting a breadth first visit strategy, that exploits the concept of sibling, allowing a horizontal reading of the hierarchy: the next page to visit must be at the same level in the hierarchy of the current one, which intuitively means that they contain information at the same level of granularity.

7.2. DyLOG *in the context of the literature about agent programming languages*

The language that we used for programming our reasoners and for implementing the adaptive intelligent services provided by our tutoring system is DyLOG, a logic language developed in (Baldoni et al. 2001b; Patti 2002) with the aim of modelling and programming agents with reasoning capabilities. Formalizing rational agents by means of logic languages is one of the main topics of interest in the AI community (Levesque et al. 1997; Hindriks et al. 2001; Herzig and Longin 2002) and recent years have witnessed a growing interest in non-classical logics, such as modal and non-monotonic logics, because of their capability of representing and reasoning about structured and dynamic knowledge. Nonetheless, there is a gap between the expressive power of agent logical models and the practical implementation of agent systems, mainly due to the computational difficulties to verify that properties granted by the models hold also in the implemented systems. Indeed, the leading idea in developing DyLOG was to integrate the expressive capabilities of modal logic and non-monotonic reasoning techniques, within the logic programming framework, in order to define a language which can be used both for specifying and for programming agents, bridging the gap mentioned above.

DyLOG is based on a *modal action theory* that has been developed in (Baldoni et al. 1997, 2001b; Giordano et al. 2000; Patti 2002). The logical framework allows us to deal with complex actions as well as with sensing actions, and to address the most classical reasoning about actions tasks, such as planning, temporal projection and postdiction. In general the framework allows the user to specify the behavior of an intelligent (goal directed or reactive) agent, that chooses a course of actions conditioned on its beliefs on the environment and that can use sensors and communication for acquiring or updating its knowledge on the real world. The reasoning capabilities supported by the language were essential in the implementation of the adaptive intelligent services provided by our virtual tutor. Moreover, there was a major advantage in using DyLOG in the current work, rather than other languages, developed in the literature for reasoning about dynamic domains and for agent programming, such as GOLOG (Levesque et al. 1997): DyLOG has a sound proof procedure, which practically allows reasoners to perform the planning task in presence of *sensing*. The consequence, in our application framework, is that we can treat the problem of *interactively* generating adapted study plans as a *conditional plan* extraction problem.

The adoption of modal logic in order to tackle reasoning about actions and change, is common to many proposals, such as (De Giacomo and Lenzerini 1995; Prendinger and Schurz 1996; Castilho et al. 1997), and it is motivated by the fact that modal logic allows a very natural representation of actions as *state transitions*. Since the mental attitudes used for describing agents are usually represented as modalities, our modal action theory is also well suited to incorporate such attitudes. The formalization of complex actions draws considerably from dynamic logic, and it refers to a Prolog-like paradigm: complex actions are defined through (possibly recursive) definitions, given by means of Prolog-like clauses. The nondeterministic choice among actions is allowed by defining sets of alternative clauses.

Acknowledgements

We would like to thank prof. Alberto Martelli, Alessandro Chiarotto, Andrea Molia and Laura Torasso for their precious support.

Notes

[1] Technical information about the Wlog system can be found at the URL: `http://www.di.unito.it/~alice`.

[2] Test actions are needed for testing if some fluent holds in the current state and for expressing conditional complex actions. They are written as "?*Fs*", where *Fs* is a fluent conjunction.

[3] The atom *generic* is used to express that we do not care about which course supplies a given competence.

[4] Currently we do not tackle the problem of building repair plans, aimed at fixing a student-given, wrong study plan.

[5] Thus procedures are deterministic or at most they can implement some "don't care" determinism.

[6] Note that the above formulation of the behaviour of the agent, bears many similarities with agent programming languages based on the BDI paradigm such as dMARS (d'Inverno et al. 1997). As in dMARS, plans are triggered by goals and are expressed as sequences of primitive actions, tests or goals.

References

Baldoni, M., Baroglio, C., Chiarotto, A. & Patti, V. (2001a). Programming Goal-driven Web Sites using an Agent Logic Language. In Ramakrishnan, I. V. (ed.) *Proc. of the Third International Symposium on Practical Aspects of Declarative Languages*, Vol. 1990 of *LNCS*. Las Vegas, Nevada, USA, 60–75. Springer.

Baldoni, M., Baroglio, C., Henze, N. & Patti, V. (2002). Setting up a Framework for Comparing Adaptive Educational Hypermedia: First Steps and Application on Curriculum Sequencing. In *Proc. of ABIS-Workshop 2002: Personalization for the Mobile World, Workshop on Adaptivity and User Modeling in Interative Software Systems*. Hannover, Germany, 43–50.

Baldoni, M., Giordano, L., Martelli, A. & Patti, V. (1997). An Abductive Proof Procedure for Reasoning about Actions in Modal Logic Programming. In Dix J. et al. (eds.) *Proc. of NMELP'96*, Vol. 1216 of *LNAI*, 132–150. Springer-Verlag.

Baldoni, M., Giordano, L., Martelli, A. & Patti, V. (2001b). 'Reasoning about Complex Actions with Incomplete Knowledge: A Modal Approach. In Restivo, A., Ronchi Della Rocca, S. & Roversi, L. (eds.) *Proc. of Theoretical Computer Science, 7th Italian Conference, ICTCS'2001*, Vol. 2202 of *Lecture Notes in Computer Science*, 405–425. Springer.

Baral, C., McIlraith, S. A. and Son, T. C. (2000). Formulating Diagnostic Problem Solving Using an Action Language with Narratives and Sensing. In *Principles of Knowledge Representation and Reasoning, KR 2000*, 311–322.

Bretier, P. & Sadek, D. (1997). A Rational Agent as the Kernel of a Cooperative Spoken Dialogue System: Implementing a Logical Theory of Interaction. In Müller, J., Wooldridge, M. & Jennings, N. (eds.) *Intelligent Agents III, Proc. of ECAI-96 Workshop on Agent Theories, Architectures, and Languages (ATAL-96)*, Vol. 1193 of *LNAI*. Springer-Verlag.

Brusilovsky, P. (2000). Course Sequencing for Static Courses? Applying ITS Techniques in Large-ScaleWeb-Based Education. In *Proceedings of the fifth International Conference on Intelligent Tutoring Systems ITS 2000*. Montreal, Canada.

Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction* **11**: 87–110.

Carro, R., Pulido, E. & Rodriguez, P. (1999). Dynamic Generation of Adaptive Internet-Based Courses. *Journal of Network and Computer Applications* **22**: 249–257.

Castilho, M., Gasquet, O. & Herzig, A. (1997). Modal Tableaux for Reasoning about Actions and Plans. In Steel, S. (ed.) *Proc. ECP'97*, 119–130.

De Giacomo, G. & Lenzerini, M. (1995). PDL-based Framework for Reasoning about Actions. In *Proc. of AI∗IA '95*, Vol. 992 of *LNAI*, 103–114.

d'Inverno, M., Kinny, D., Luck, M. & Wooldridge, M. (1997). A Formal Specification of dMARS. In *Proc. of ATAL'97*, Vol. 1365 of *LNAI*, 155–176.

Eshghi, K. & Kowalski, R. (1989). Abduction Compared with Negation by Failure. In *Proc. 6th ICLP'89*. Lisbon, 234–254.

FIPA (1997). FIPA 97, Specification Part 2: Agent Communication Language. Technical report, Foundation for Intelligent Physical Agents.

Gelfond, M. & Lifschitz, V. (1993). Representing Action and Change by Logic Programs. *Journal of Logic Programming* **17**: 301–321.

Giordano, L., Martelli, A. & Schwind, C. (2000). Ramification and Causality in a Modal Action Logic. *Journal of Logic and Computation* **10**(5): 625–662.

Henze, N. & Nejdl, W. (1999). Bayesian Modeling for Adaptive Hypermedia Systems. In *Proc. of ABIS99, 7. GI-Workshop Adaptivität und Benutzermodellierung in Interaktiven Softwaresystemen*. Magdeburg.

Henze, N. & Nejdl, W. (2000). Extendible Adaptive Hypermedia Courseware: Integrating Different Courses and Web Material. In Brusilovsky, P., Stock, O. & Strapparava, C. (eds.) *Adaptive Hypermedia and Adaptive Web-Based Systems, International Conference, AH 2000*, 109–120.

Henze, N. & Nejdl, W. (2001). Adaptation in Open Corpus Hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems* **12**: 325–350.

Herzig, A. & Longin, D. (2002). Sensing and Revision in a Modal Logic of Belief and Action. In van Harmelen, F. (ed.) *Proc. of 15th European Conference on Artificial Intelligence, ECAI 2002*. Lyon, France, 307–311. IOS Press.

Hindriks, K. V., de Boer, F., van der Hoek, W. & Meyer, J. (2001). Agent Programming with Declarative Goals. In Castelfranchi, C. & Lespérance, Y. (eds.) *Intelligent Agents VII. Agent Theories, Architectures and Languages*, Vol. 1986 of *LNAI*, 228–243. Springer-Verlag.

Levesque, H. J., Reiter, R., Lespérance, Y. Lin, F. & Scherl, R. B. (1997). GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming* **31**: 59–83.

Macías, J. A. & Castells, P. (2001). Interactive Design of Adaptive Courses. In Ortega, M. & Bravo, J. (eds.) *Computer and Education – Towards an Interconnected Society*, 235–242. Kluwer Academic Publishers.

Murray, T. (2002). MetaLinks: Authoring and Affordances for Conceptual and Narrative Flow in Adaptive Hyperbooks. *International Journal of Artificial Intelligence in Education*, to appear.

Patti, V. (2002). Programming Rational Agents: a Modal Approach in a Logic Programming Setting. Ph.D. thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy. Available at http://www.di.unito.it/∼patti/.

Prendinger, H. & Schurz, G. (1996). Reasoning about Action and Change. A Dynamic Logic Approach. *Journal of Logic, Language, and Information* **5**(2): 209–245.

Sandewall, E. (1994). *Features and Fluents. The Representation of Knowledge about Dynamical Systems*, Vol. I. Oxford University Press.

Scherl, R. & Levesque, H. J. (1993). The Frame Problem and Knowledge-producing Actions. In *Proc. of the AAAI-93*. Washington, DC, 689–695.

Specht, M., Kravcik, M., Klemke, R., Pesin, L. & Hüttenhain, R. (2002). Personalized eLearning and eCoaching in WINDS. In *Proc. of Workshop on Integrating Technical and Training Documentation, ITS 2002*. San Sebastian, Spain.

Specht, M. & Oppermann, R. (1998). ACE – Adaptive Courseware Environment. *The New Review of Hypermedia and Multimedia* **4**: 141–162.

Stern, M. & Woolf, B. (1998). Curriculum Sequencing in a Web-Based Tutor. In *Proc. Of Intelligent Tutoring Systems 1998*, Vol. 1452 of *LNCS*. Springer.

Toni, F. & Kakas, A. (1995). Computing the Acceptability Semantics. *LNAI* **928**: 401–415.

Weber, G. & Brusilovsky, P. (2001). ELM-ART: An Adaptive Versatile System for Webbased Instruction. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems* **12**(4): 351–384.

Artificial immune system approach to adaptation

Sławomir Wierzchoń, Warsaw

1 Introduction

Immune algorithms, IAs for short, are representatives of still growing family of biologically inspired algorithms, like genetic algorithms, ant algorithms, particle swarm algorithms, etc. – consult [7] for extensive review of such algorithms and their applications. Shortly speaking, IAs are inspired by works on theoretical immunology and some mechanisms, described in Section 1.1, used by the natural immune system to cope with external and internal invaders. IAs are adaptive algorithms in which learning takes place by evolutionary mechanisms similar to biological evolution. Their adaptability relies on continuous generation of novel elements to handle varying set of situations and on deletion of inefficient elements. Hence, IAs can be viewed as an instance of "generate-and-test" algorithm or, as proposed in [10], as an instance of an algorithm that adapt by innovation, i.e. by constant generation of "something genuinely new". Since the aim of natural immune system is production of antibodies (or immunoglobulins) that are able to neutralize external intruders called antigens, we call the problem to be solved as antigen, and the aim of an IA is production of antibody being a solution to this problem.

This new paradigm offers some exciting possibilities of designing flexible algorithms that: (i) adapt to ***new*** situations as well as (ii) solve problems that are ***similar*** to already solved problems. Particularly, the mechanism of so-called primary immune response (described in Section 1.1) allows solving new problems, i.e. the system produces antibodies (i.e. solution) that can bind to a new pathogen (i.e. problem to be solved). On the other hand, secondary immune response, search for antibodies that can bind successfully pathogens structurally similar to already recognized pathogens. One of conceptual tools explaining how the population of antibodies is controlled is the theory of idiotypic networks proposed by N.K. Jerne in 1974. According to this theory, interactions among antibodies of different type, as well as among antibodies and pathogens result in emergent properties like learning and memory, self-tolerance, and size and diversity of immune repertoire. Broadly speaking, the evolution of the immune network is governed by the set of differential equations of general form:

$$\begin{array}{c}\textit{Rate of}\\ \textit{population}\\ \textit{variation}\end{array} = \begin{array}{c}\textit{Network}\\ \textit{stimulation}\end{array} - \begin{array}{c}\textit{Network}\\ \textit{suppresion}\end{array} + \begin{array}{c}\textit{Influx}\\ \textit{of new}\\ \textit{clones}\end{array} - \begin{array}{c}\textit{Death of}\\ \textit{unstimulated}\\ \textit{clones}\end{array} \qquad (*)$$

Section 1.2 offers detailed description of such a model originally proposed in the paper [11]. Although its authors recognized very soon that "the kinetic equations used in our original paper were highly idealized" ([9], p. 172) this model still inspires many works in the field of Artificial immune systems (AISs for brevity). It would be interesting to review alternative models used in theoretical immunology with the hope, that these models will improve behavior of currently designed AISs. Such a review can be found in [16], [20], [23].

1.1 Basic notions from immunology

The main actors of the adaptive immune system are so-called lymphocytes, i.e. white blood cells. Briefly, we distinguish lymphocytes of type B (or B-cells) and lymphocytes of type T (or T cells).

Each B-cell admits about 10^5 receptors located on its surface and called antibodies (or immunoglobulin). These antibodies are soluble proteins which have high affinity towards antigens. The key portion of antigen that is recognized by the antibody is called "epitope"; it can be viewed as the identifier of that antigen. Similarly, the "paratope" is a specific region of antibody that attach to the epitope. The antigenic determinant of an antibody (i.e. its epitope) is referred to as "idiotope". The set of idiotopes that characterizes an antibody is called its "idotype".

Real paratope and epitope are 3D molecules. If they are complementary with respect to their geometric or physico-chemical characteristics, we say that the paratope recognizes just introduced epitope; alternatively, we say that the paratope has high affinity with the epitope. To study analytically the interactions among paratopes and epitopes, Perelson and Oster assumed in [19] that the structural characteristics of these molecules can be adequately specified by a finite list of *d* parameters. The set of all possible lists form so-called "shape-space"[1], which typically is assumed to be a *d*-dimensional vector space, usually *d*-dimensional Euclidean or Hamming space. This way the affinity between paratope and epitope can be specified as a function of distance between two *d*-dimensional vectors representing these objects (i.e. points in referential shape-space).

Suppose a large number of copies of a specific, and never seen, antigen are introduced into an organism. Once a sufficient number of paratopes binds the epitopes describing just introduced antigen, so-called primary immune response occurs. It relies upon clonal expansion and somatic hypermutation. The former term means rapid replication of those B-cells which have a high affinity to the antigen. To "tune" molecular shapes of the paratopes characterizing produced clones to the shapes of invading epitopes, each clone is subjected very intensive mutation what leads to variation of immune response. Mutated clones with highest affinity to the antigen are subjected further expansion and cloning, while mutants with lowest affinity are removed from the organism. The process is continued until the concentration of epitopes decreases below sufficiently low threshold. This is the core mechanism of so called "clonal selection theory".

It should be noted that during primary immune response the interaction with T-cells is crucial to the dynamics of the system. These lymphocytes control the activity of B-cells and they may have excitatory or inhibitory role. A reader interested in details on how B- and T-cells cooperate is referred to e.g. [15].

A crucial effect of all these interactions is that the response to a new antigen has a bell-shaped form. There exists a minimum concentration (θ_1) of epitopes that will elicit the immune response while for very high concentration of epitopes (exceeding the second threshold $\theta_2 >> \theta_1$) the response decreases. In other words, in the immune system we observe low- and high-dose tolerance i.e. lack of any reaction against antigens. Only medium dose of the antigen causes immune response manifested with rapid production of antibodies. In theoretical immunology the response function $f(h_i)$, representing the concentration of antibodies of *i*-th type, is modeled by the equation (see e.g. [20])

$$f(h_i) = \frac{h_i}{(\theta_1 + h_i)} \cdot \frac{\theta_2}{(\theta_2 + h_i)} \qquad (1)$$

where h_i stands for the "field" representing the strength of influence of all epitopes present in the system on a given antibody. Usually, if m_{ij} stands for the affinity between *i*-th paratope and

[1] It has been observed in [2] that the idea of shape space may be misleading in theoretical studies and may produce artifacts which do not reflect any underlying biological reality what means that it should be used with caution.

j-th epitope, x_j denotes concentration of *j*-th epitope, and N is the number of different types of epitopes then the field h_i is computed according to the equation

$$h_i = \sum_{j=1,\ldots,N} m_{ij} \cdot x_j \tag{2}$$

Equation (2) says that *i*-th antibody can be stimulated by all the epitopes present in the organism, no matter they come from antigens or other antibodies constituting given immune system. This is because a new antibody, say Ab_1, generated e.g. through somatic mutation is a new protein for the organism, and its intensive reproduction during clonal expansion causes new immune response resulting in production of antibody of other type, say Ab_2. In summary, the production of antibody Ab_i stimulates production of other types of antibodies[2] and these subsequent generations of proteins form a kind of network called by Jerne "idiotypic network" (consult [17], or [11] for details). Its characteristic feature is that it can be maintained even in the absence of antigens inducing immune response. This is due to symmetric interactions between antibodies: if Ab_i stimulates Ab_{i+1} then Ab_{i+1} stimulates production of Ab_i. Since antibody Ab_1 was induced by an antigen Ag, one of its descendants[3], Ab_i, must have epitope structurally similar to the epitope characterizing the Ag. It is obvious that during absence of the antigen Ag the antibody Ab_i will maintain production of other antibodies belonging to the chain $Ab_1 \rightarrow \ldots \rightarrow Ab_i \ldots$ resembling auto-catalytic loop (consult [11] or [9]). Now, if the antigen Ag enters the organism next time, its shape is **"**remembered**"** by such a loop and effective antibodies are produced almost immediately. This phenomenon is referred to as "immune memory" and fast production of effective antibodies is termed "secondary immune response". The set of antibodies with epitopes structurally similar to the epitopes characterizing the already introduced antigen is said to be "internal image" of the antigen. The ability of producing internal image of any antibody seems to be very promising for creation of pattern recognition systems.

The notions of pattern recognition and immunological memory in the theory of clonal selection and idiotypic network theory are quite different. As stated by Lord in [18], "*In the clonal selection theory, recognition is the amplified response of a few specific cells and elimination of an antigenic stimulus through numeric superiority*; *memory is the persistence of a population of specific cells. In idiotypic networks, recognition is the systemic disruption of a dynamic equilibrium and the creation of a new equilibrium with different population and lineages*; *memory is the persistence of a network of anti-idiotypic reactions around a population of specific cells*."

1.2 Mathematical model of the immune network

Many different mathematical approaches have been developed to reproduce and analyze the main immune functions. Particularly, Jerne's hypothesis of idiotypic networks inspired a series of models describing the interactions between B-cell clones. The goal of mathematical modeling in theoretical immunology it to "*deduce macroscopic properties of the system from the properties and interactions among the elementary components*" [20]. Broadly speaking we distinguish between continuous and discrete models. Ordinary differential equations are typical for the first group of models; they often resemble ecological models. These models can be labeled as "B-models" if no distinction between free and cell-bound antibodies is made, and "AB-models" if both forms of antibodies are described. Surprisingly, both B- and AB-models lead to similar conclusions as regards the fixed point properties (stable fixed points for dynamics are necessary for achieving tolerance in a model), [2]. It is important, since immune memory can be potentially modeled by a fixed point of the network, [9], and the fixed points of

[2] This idea was confirmed experimentally. Namely, it was observed that e.g. in the case of *polio* virus infection, Ab_2 has the internal image of *polio* antigen. It means that Ab_2 is induced by the paratope of Ab_1 rather than by its idiotype. See: Fons, U., et al., "From Jenner to Jerne: towards idiotypic vaccines". *Immunol. Rev.* **90**:93-113, 1986

[3] and called anti-idiotypic antibody

the differential models corresponds to the immune memory. Cellular automata, on the other hand, are commonly used in the second group of models. An advantage of cellular automata over differential equations model is its numerical stability. Further the dynamics of cellular automata models can be easily tuned so as to mimic the behavior of the real system, and it does not rely on global information (consult [18] for a deeper discussion). A reader interested in detailed description of these models is referred to [20], [12], [13] and [8].

Below we briefly describe the "*bit-string model*" proposed in [11] and commonly used by the AIS community. This model takes into account only interactions among paratopes and epitopes of antibodies and antigens represented by binary strings ignoring interactions among B-cells and other agents of the immune system (e.g. T-cells, macrophages, etc). Interestingly, this model can be used to study both clonal selection and idiotypic networks theory. The affinity m_{ij} between *i*-th epitope and *j*-th paratope is computed here in a way reflecting partial matching between the two molecules. The dynamics of the system consisting of N antibody types with concentrations $\{x_1, \ldots, x_N\}$ and M antigens with concentrations $\{y_1, \ldots, y_M\}$ is described by the following system of ordinary differential equations:

$$\frac{dx_i(t)}{dt} = k_1 \cdot \left[\sum_{j=1}^{N} m_{ji} x_i(t) x_j(t) - k_2 \sum_{j=1}^{N} m_{ij} x_i(t) x_j(t) + \sum_{j=1}^{M} m_{ji} x_i(t) y_j(t) \right] - k_3 x_i(t), \quad i = 1, \ldots, N \tag{3a}$$

The first term in equation (3a) represents the stimulation of *i*-th paratope by the epitope of an antibody of type *j*, the second term represents the suppression of antibody of type *i* when its epitope is recognized by the paratope of type *j*, third term represents the stimulation of *i*-th antibody by the antigens, and the last term models the tendency of cells to die. The parameter k_1 is a rate constant that depends on the number of collisions per unit time and the rate of antibody production stimulated by a collision. Constant k_2 represents a possible inequality between stimulation and suppression and constant k_3 represents the rate of natural death. The process of elimination of antigens from the system describes the set of ordinary differential equations proposed in [10]:

$$\frac{dy_j(t)}{dt} = -c \sum_{i=1}^{N} m_{ij} x_i(t) y_j(t), \; j = 1, \ldots, M \tag{3b}$$

where c is an arbitrary constant. The model was used by Bagley *et al*. [1] who studied another important concept of theoretical immunology – plasticity in an immune network, i.e. the process of removing and recruiting certain types of antibodies from/into the network. This process enables the immune system to decide which idiotypic determinants should be included/removed in/from the network without referring to an explicit fitness function. Consequently, the network is flexible and is able to modify its structure. Soon, it became obvious, [9], that the model is too simple to describe emergence of a self-asserted structure which would be able to sustain immune functions. Thus in [1] the authors proposed another model similar to the B-model while in [9] they introduced a counterpart of the "AB-model".

2 Immune-based recommenders

This section briefly describes theoretical foundations of an immune-based recommender as well as its implementation.

2.1 Theoretical foundations

Immune-based recommender system was proposed by Cayzer and Aickelin in [4], [5] and [6]. Their aim was to apply idiotypic effects to mimic "intelligent" behavior of immune agents and to tune it to the problem of "intelligent" preference matching and recommendation. They used publicly accessible software SWAMI, [14]. Its central part acts according to the pseudocode

```
1. select a set T of test users randomly from the database
2. for each test user t
   2a. hide a vote of the user t from predictor
   2b. from remaining votes of user t create a new training
       user t'
   2c. select neighborhood of k reviewers based on t'
   2d. use neighborhood to predict vote
   2e. compare this with actual vote and collect statistics
3. Return final statistics
```

Pseudocode 1. Main part of SWAMI prediction

Steps (2c) and (2d) are implementation-dependent (i.e. they depend on the similarity/correlation measure used to compare different users).

The SWAMI package uses EchMovie[4] database in which 2,811,983 votes taken from 72,916 users on 1,628 films are recorded. The algorithm acts according to a "standard" recipe: it uses information from a neighborhood to make useful predictions and recommendations.

Each user u is encoded as the n-tuple $u = \{\{id_1, score_1\}, \{id_n, score_n\}\}$ where *id* stands for the unique identifier of the movie being rated and $score \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ stands for the evaluation of that movie by the user. The most popular measure of similarity between two users u and v is the Pearson measure

$$r(u,v) = \frac{\sum_{i=1}^{n}(u_i - <u>)(v_i - <v>)}{\sqrt{\sum_{i=1}^{n}(u_i - <u>)^2 \sum_{i=1}^{n}(v_i - <v>)^2}} \quad (4)$$

where n is the number of overlapping votes (i.e. movies evaluated by both u and v), u_i is the vote of user u for i-th movie, and $<u>$ is the average vote of user u over *all* movies seen by him. Then the predicted vote of the active user u for j-th movie, u_j, is a weighted sum of the votes of the other users

$$u_j = <u> + \kappa \cdot \sum_{v \in N(u)} w(u,v)(v_j - <v>) \quad (5)$$

where $N(u)$ stands for the neighborhood of u, and κ is a normalizing constant such that the absolute values of the weights w sum to unity. In general $w(u, v)$ is any measure reflecting distance, correlation, or similarity between users u and v.

Cazyer and Aickelin proposed to use as $w(u, v)$ the next measure

$$w(u, v) = r(u, v) \cdot x_v \quad (6)$$

where $r(u, v)$ is Pearson correlation score and x_v is the concentration of the antibody corresponding to the user v. This concentration is computed from simplified equation (3a) which now takes the form

[4] available from http://www.research.compaq.com/SRC/eachmovie.

$$\frac{dx_v(t)}{dt} = k_1 \cdot |r(v,u)| - \frac{k_2}{card(Ab)} \sum_{a \in Ab} |r(v,a)| \cdot x_a \cdot x_v - k_3 x_v(t), \quad v \in Ab \quad (7)$$

Here it is assumed that: (a) antigen is the user for whom we make prediction, (b) the set *Ab* of antibodies form other test users, (c) the affinity measure m_{uv} in equation (3a) is defined as the absolute value of Pearson score $r(u, v)$. Particularly, from (a) it follows that we have only one antigen in the system and its concentration is fixed.

In the Pearson predictor, neighborhood selection is based on choosing the k users with best absolute correlation scores (k is a predefined neighborhood size). A disadvantage of such an approach is that not every potential neighbor rated predicted movie. A set of differential equation (7) allows to choose neighbors in a more elaborated way. The whole procedure can be described by the next pseudocode, [4]:

```
1. Initialize AIS
2. Encode user for whom to make prediction as antigen, $u$
3. while ((AIS not stabilized) & (reviewers available))
   3a. Add next user as an antibody, $v$
   3b. Calculate matching score $r(u,v)$
   3c. Calculate matching scores between $v$ and other antibodies
   3d. while ((AIS at full size) & (AIS not stable)
       3d$\alpha$.   iterate AIS
```

Pseudocode 2. Immune-based creation of neighborhood for a given antigen

The AIS, described by the set of differential equations (7), is considered as stable if after ten iterations its size does not change. Hence, stabilization means that "good" candidate neighbors have been identified; typically "poor" neighbors are washed out from the system after a few iterations. As stated in [4], "*we require a set of antibodies that are a close match but which at the same time distinct from each other for successful recommendation. This is where we propose to harness the idiotypic effects of binding antibodies to similar antibodies to encourage diversity*". Particularly, the pool contains antibodies (users) that are both positively and negatively correlated (in the sense of Pearson r coefficient) with a given antigen; this increases the diversity of neighboring antibodies.

This description suggests further modification for the AIS. First of all we should experiment with different definitions of the affinity measure m_{uv} (e.g. cosine measure, etc.). Introducing non-symmetric affinity measure we can also test both suppressive and stimulatory effects of other antibodies. These last effects are omitted in the equation (7). We can also experiment with other types of equations used in theoretical immunology.

2.2 Prototypical re-implementation

To get an idea on how the approach works, a re-implementation (in Java environment) of Cayzer and Aickelin approach was performed. Like in original approach the system collaborates with SWAMI package.

All the algorithms used during prediction must implement interface **Predictor** and – obviously – they must define body of the implemented methods. Classes of all the algorithms are placed in the package `edu.berkeley.swami.predict`.

Interface *Predictor:*
This interface is an element of the *SWAMI* package; it is a bridge between testing module and concrete predictor. This interface contains two two basic methods:

- *train*() – a method called during loading test data.
- *predictRating*() – basic method used during predictor. It returns prediction for a given movie.

There are four classes implementing ***Predictor*** interface. The first two classes, **PredictByUserAverage** and **PredictByMovieAverage** return values which can be used as reference values allowing to assess the quality of a given predictor. The former class returns average value of all scores of an active user, while the later returns average evaluation of a given movie by all the users. Below we briefly describe two remaining classes

Class *SimplePearsonPredictor*: This class implements simple predictor in which correlation between users is defined in terms of Pearson coefficient. It basic methods are:
- *computePearson*() – computes Pearson score according to the equation (4)
- *findNeigbors*() – finds *k* neighbors with highest absolute value of Pearson score.
- *didReviewerRateMovie*() – checks if the reviewer has evaluated a given movie.
- *addToNeigborsOrDont*() – decides if a given user shoul be included to the neighborhood.

Class *AisPredictor*: This class implements immune-based predictor. Its most important methods are:
- *addIntialNeighbors*() – initializes a group of best neighbors of a given user (antibody).
- *addOrDontToNeigborhood*() – decides if a given user should be added to the neighbors.
- *constructNeighborhood*() – produces the list of neighbors.
- *constructReviewer*() – creates instance (object) of **Reviewer** class by using an identifier taken from the test set.
- *startAisSystem*() – runs immune-based prediction algorithm which is placed in the class **AisEngine**.

Immune-based recommender implements two interfaces: **MatchingFunction** and **AisConstans**. The main class of this recommender is **AisEngine**. It is responsible for the interactions between antibodies and antigen.

Interface *AisConstans*: It contains only variables (corresponding to the parameters of the system):
- *DEFAULT_OVERLAP_PENALTY* – threshold representing minimal value of overlaping scores nedded to compute Pearson coefficient.
- *DEFAULT_KEEP_IN_MEMORY* – determines if the test set shoul be kept in memory or shoul be read from file in each prediction.
- *STIMULATION_RATE* – stimulation rate (k_1 coefficient).
- *SUPPRESSION_RATE* – suppression rate (k_2 coefficient).
- *DEATH_RATE* – death rate (k_3 coefficient).
- *MAX_CONCENTRATION* – maximal concentration of an antibody (= 100).
- *MIN_CONCENTRATION* – minimal concentration of an antibody (= 0).
- *INITIAL_CONCENTRATION* – initial concentration of an antibody (= 10).
- *LOW_CONCENTRATION* – if the concentration of an antibody is below this threshold value, the antibody is deleted from the system.
- *USE_CONCENTRATION* – informs if the concentration value is needed by predictor.
- *USE_ABSOLUTE_VALUE* – informs if the absolute value of a similarity measure is needed by predictor.
- *NUMBER_OF_ANTIBODIES* – maximal number of antibodies in the system.
- *AIS_STABILIZATION_VALUE* – number of iterations after which stabilization is tested.
- *USE_CONCENTRATION_IN_PREDICTION* – parametr informuje nas o tym czy używamy koncentracji danego przeciwciała podczas predykcji.

- *TOP_N_RECOMMENDATION_VALUE* – number of best top recomendations returned by the recommender.
- *REVIEWERS_SIZE* – total size of training and test data.

Class *AisEngine*: This is engine for all the algorithms which use artificial idiotypic network. This class implements equation (7). This class is used by the classes **AisPredictor**, **AisPredictorTest** (the immune-based recommender) and **AisPredictorFrame** (a version of the recommender with graphical interface). Its basic methods are:
- *addAntigen*() – adds antigen to the system.
- *addAntibody*() – adds antibody to the system.
- *getAntigen*() – gets antigen with a given identifier.
- *getAntibody*() – gets antibody with a given identifier.
- *removeAntigen*() – deletes antigen with a given identifier.
- *removeAntibody*() – deletes antigen with a given identifier.
- *isAisAtFullSize*() – checks if the size of antibodies included to the system equals the parameter NUMBER_OF_ANTIBODIES.
- *isAisStable*() – checks if the system is stable, i.e. if during AIS_STABILIZATION-_VALUE its size has not changed.
- *tryToRemoveLowAntibodies*() – removes antibody if its concentration is below LOW_CONCENTRATION threshold.
- *initialise*() – initializes AIS: assigns initial concentration to all antibodies and antigen.
- *reset*() – resets AIS.
- *clearAntigens*() – removes all antigens from the AIS.
- *clearAntibodies*() – removes all antibodies from the AIS.
- *isMaximumConcentration*() – checks if there is antibody with maximal concentration defined by MAX_CONCENTRATION parameter.
- *addAntigenMatches*() – computes similarity measures between a given antigen and all antibodies.
- *addAntibodyMatches*() – computes similarity measures among all the antibodies.
- *iterate*() – solves equation (7) and controls all the interactions between antigen and antibodies and among the anibodies.

Class *AisPredictorTest*: It is responsible for making prediction. Its basic methods are:
- *constructReviewerHideVote*() – creates instance of the class ***Reviewer*** and hides a given vote of this user.
- *predictionMAE*() – computes mean absolute error of prediction according to the equation MAE = $\sum$|actual_score – predicted_score|/*n*, where *n* is the number of predictions
- *meanAccuracyOfRecomendations*() – computes Kendall's Tau (τ) statistics (consult eq. (6) in [4]).
- *constructOverlappedFilmList*() – creates list of movies evaluated both by the active user and by his neighbors.
- *disJointDataSets*() – divides dataset into test set and training set.
- *getMoviesFromReviewer*() – reads all movies evaluated by a given user.
- *computeRecall*() – computes *recall* value of the recommendation.
- *computePrecision*() – computes *precision* value of the recommendation.

The interface **MatchingFunction** allows to use any similarity measure (Pearson measure and Cosine measure in current implementation).

Our preliminary experiments – reported in Annex c) – show that the immune approach is comparable with other "classical" approaches and results produced by the system are even slightly better. Our further effort will be focused on experimenting with other than Pearson match–measures (for example, cosine measure behaves definitely poorly in comparison with Pearson correlation) and on modification of the dynamics of the equation (*).

3. Plastic clustering

The idea of plastic clustering, proposed in [21] and [22] is closely related to the already presented application of the idiotypic theory. Plastic clustering was invented to create keyword map as well as document clusters. The algorithm is as follows:

```
1.  Extract keywords (nouns) from a document set. (In all studies only keywords contained in
    more than two documents were extracted)
2.  Construct keyword network by connecting the extracted keyword $k_i$ to other keywords $k_j$,
    or to other documents $d_j$:
    (a) Connection between $k_i$ and $k_j$ ($D_{ij}$ stands for the number of documents containing both
        the keywords):
        Strong connection (SC): $D_{ij} \geq T_k$
        Weak connection (WC): $2 < D_{ij} < T_k$
    (b) Connection between $k_i$ and $d_j$ ($TF_{ij}$ stands for the term frequency of $k_i$ in $d_j$):
        Strong connection (SC): $TF_{ij} \geq T_d$
        Weak connection (WC): $0 < TF_{ij} < T_d$
3.  Calculate keywords' activation values on the constructed network based on the immune
    network model
4.  Extract the keywords with highest concentration and treat them as landmarks.
5.  Generate document clusters according to the landmarks.
```

Pseudocode 3. Plastic clustering

Here the keywords are treated as antibodies and documents as antigens. As the immune network model, so-called B-model has been adopted:

$$\frac{dx_i}{dt} = s + x_i \cdot (f(h_i^b) - k_b) \tag{8a}$$

$$h_i^b = \sum_j J_{ij}^b x_j + \sum_j J_{ij}^g y_j \tag{8b}$$

$$\frac{dy_i}{dt} = (r - k_g h_i^g) \cdot x_i \tag{8c}$$

$$h_i^g = \sum_j J_{ij}^g x_j \tag{8d}$$

$$f(h) = \frac{h}{(\theta_1 + h)} \cdot \frac{\theta_2}{(\theta_2 + h)} \cdot p \tag{8e}$$

Here x_i and y_i are the concentration values of antibody and antigen, respectively. The s is a source term modeling a constant cell flux from the bone marrow and r is a reproduction rate of the antigen; k_b and k_g are the decay terms of the antibody and antigen, respectively. The J^b_{ij} and $J^g_{ij} \in \{0, \text{WC}, \text{SC}\}$ indicate the strength of the connectivity between the antibodies i and j, and that between antibody i and antigen j, respectively. Typical values of these parameters are given in the table below:

parameter	value	parameter	value
s	10	$x_i(0)$	10
r	0.01	$y_i(0)$	10^5
k_g	10^{-4}	T_k	3
k_b	0.4	T_d	3
θ_1	10^3	SC	1
θ_2	10^6	WC	10^{-3}
		p	1

References

[1] Bagley, R.J., Farmer, J.D., Kauffman, S.A., Packard, N.H., Perelson, A.S., Stadnyk, I.M. Modeling adaptive biological systems. *BioSystems* **23**: 113-138, 1989.

[2] Bonabeau, E. A simple model for the statistics of events in idiotypic networks. *BioSystems*, **39**: 25-34, 1996

[3] Breese, J.S., Heckerman, D., Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. *Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann 1998, pp. 43-52*

[4] Cayzer, S. Aickelin U. A recommender system based on the immune system. HP Laboratories, Bristol, HPL-2002-1

[5] Cayzer, S. Aickelin U. A recommender system based on the immune network. Proc. CEC2002, Honolulu, USA, pp. 807-813

[6] Cayzer, S. Aickelin U. On the effects of idiotypic interactions for recommendation communities in artificial immune systems. Proc. of the 1st Internat. Conference on Artificial Immune Systems, ICARIS'2002, University of Kent at Canterbury, 2002, pp. 154-160

[7] Corne, D., Dorigo, M., Glover, F. (eds.) *New Ideas in Optimization*, McGraw-Hill 1999

[8] De Castro, L.N, Timmis, J. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London, Berlin, Heidelberg 2002

[9] Farmer, J.D. A Rosetta Stone for connectionism. *Physica D*, **42**: 153-187, 1990

[10] Farmer, J.D., Kauffman, S.A., Packard, N.H., Perelson, A.S. Adaptive dynamic networks as models of the immune system and autocatalytic set. *Ann. of the N.Y. Acad. of Sci.*, **504**:118-131, 1987

[11] Farmer, J.D., Packard, N.H., Perelson, A.S. The immune system, adaptation, and machine learning. *Physica D*, **22**:187-204, 1986

[12] Faro, J., Velasco, S. Studies on a recent class of network models of the immune system. *J. theor. Biol.*, **164**: 271-290, 1993

[13] Faro, J., Carneiro, J. Velasco, S. Further studies on the problem of immune network modelling. *J. theor. Biol.*, **184**: 405-421, 1997

[14] Fisher, D. et al. SWAMI: a framework for collaborative filtering algorithm development and evaluation (available from http://guir.cs.berkeley.edu/projects/swami/).

[15] Hofmeyr, S.A. Introduction to the immune system. In: L.A. Segel, I. Cohen (eds.) *Design Principles for the Immune System and Other Distributed Autonomous Systems*, Santa Fe Institute Studies in the Sciences of Complexity. New York: Oxford University Press 2001

[16] Itaya, S., Uezu, T. Analysis of an immune network dynamical system model with a small number of degrees of freedom. *Progress in Theoretical Physics*, **104**: 903-924, 2000

[17] Jerne N.J. Idiotypic networks and other preconceived ideas. *Immunol. Rev.* **79**: 5-25,1984

[18] Lord, C.C. An emergent model of immune cognition. MSc Thesis. Information Networking Institute, Cornegie Mellon University, Pittsburgh, PA, 2003 (available from http://www.andrew.cmu.edu/user/clord/Portfolio/ Thesis.pdf)

[19] Perelson, A.S., Oster, G.F. The shape space model. *J. theor. Biol.*, **81**: 645-670, 1979

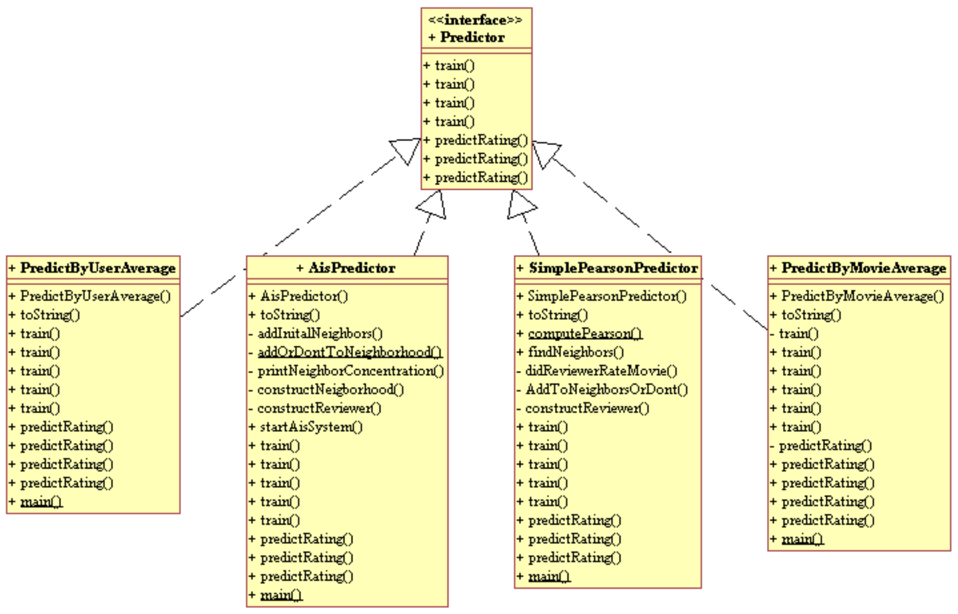
[20] Perelson, A., Weisbuch, G. Immunology for physicists. *Reviews of Modern Physics*, **69**: 1219-1265, 1977

[21] Takama, Y. and Hirota, K. Application of immune network model to keyword set extraction with variety. 6th *Int'l Conf. on Soft Computing* (*IIZUKA2000*), pp. 825-830

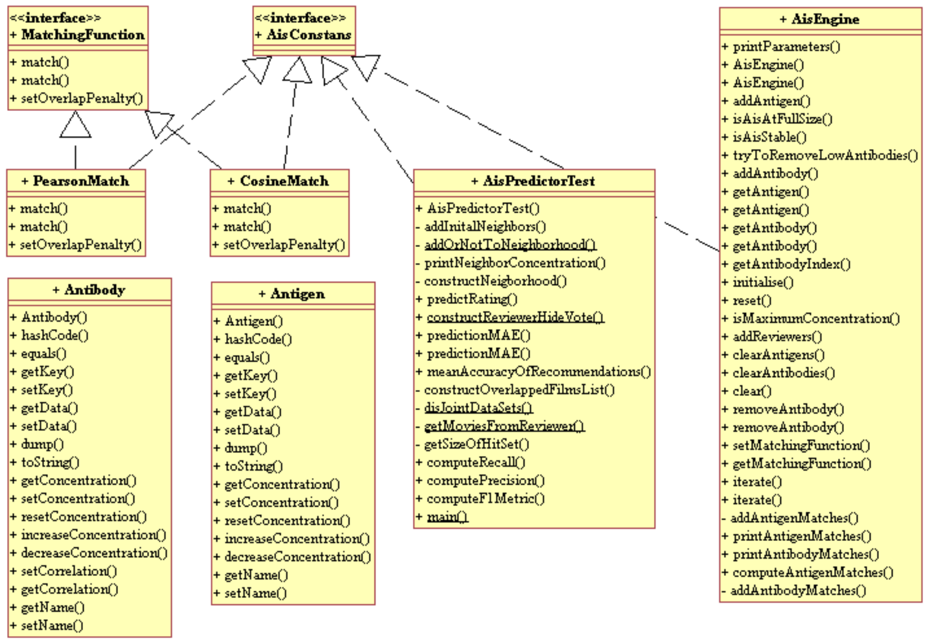
[22] Takama, Y. and Hirota, K. Web information visualization method employing immune network model for finding topic stream from document-set sequence. *J. of New Generation Computing*, **21**: 49-59, 2003

[23] Wierzchoń, S.T. Idiotypic networks as the metaphor for immune algorithms (submitted)

Annex a) Class diagram of the algorithms tested in SWAMI package



Annex b) Class diagram of the immune recommender.



Annex c) Exemplary results for SWAMI package (40% of input data are used)

			Metrics					
Type of tested algorithm		**Number of votes**	**MAE**	**Variation of MAE**	**Weighted Avg.**	**Probability P**	**Average time (in seconds)**	**Averaged error**
Standard prediction algorithm	**By User Average**	5	1.29	0.95	2.61	0.47	0.000	-0.28
		20	1.18	0.69	2.08	0.48	0.000	0.14
		40	0.94	0.74	1.63	0.40	0.000	-0.45
		A	0.99	0.51	1.48	0.41	0.000	-0.12
	By Movie Average	5	1.16	0.66	1.90	0.69	0.414	-0.12
		20	0.94	0.62	1.37	0.60	0.197	0.05
		40	0.94	0.36	1.19	0.68	0.113	-0.59
		A	0.98	0.69	1.20	0.60	0.092	-0.17
Advanced prediction algorithm	**Pearson correlation**	5	1.15	0.71	2.07	0.61	0.049	0.20
		20	0.95	0.50	1.47	0.73	0.077	0.60
		40	0.78	0.36	1.06	0.72	0.099	0.17
		A	0.90	0.47	1.20	0.62	0.091	0.33
	Clustered Pearson algorithm	5	1.16	0.68	2.12	0.63	0.006	0.15
		20	0.98	0.59	1.62	0.72	0.006	0.52
		40	0.76	0.45	1.13	0.64	0.007	0.08
		A	0.92	0.46	1.23	0.59	0.007	0.24
	Immune algorithm	5	1.16	0.78	2.22	0.63	14.122	0.13
		20	1.02	0.52	1.66	0.72	5.861	0.49
		40	0.73	0.39	1.09	0.72	3.648	0.09
		A	0.92	0.49	1.28	0.61	3.497	0.22
Ideal prediction algorithm			0	0	0	1	0	0

A – all votes of a user (except the vote for which prediction is performed).
P – probability of correct binary (“good/bad”) prediction of the user vo

The Personal Publication Reader

Fabian Abel[1], Robert Baumgartner[2,3], Adrian Brooks[3], Christian Enzi[2], Georg Gottlob[2,3], Nicola Henze[1], Marcus Herzog[2,3], Matthias Kriesell[4], Wolfgang Nejdl[1], and Kai Tomaschewski[1]

[1] Research Center L3S & Information Systems Institute, University of Hannover, {abel,henze,nejdl,tomaschewski}@kbs.uni-hannover.de
[2] DBAI, Institute of Information Systems, Vienna University of Technology {baumgart,enzi,gottlob,herzog}@dbai.tuwien.ac.at
[3] Lixto Software GmbH, Donau-City-Strasse 1/Gate 1, 1220 Vienna, Austria {baumgartner,brooks,gottlob,herzog}@lixto.com
[4] Inst. f. Math. (A), University of Hannover kriesell@math.uni-hannover.de

Abstract. This application demonstrates how to provide personalized, syndicated views on distributed web data using Semantic Web technologies. The application comprises four steps: The **information gathering step**, in which information from distributed, heterogenous sources is extracted and enriched with machine-readable semantics, the **operation step** for timely and up-to-date extractions, the **reasoning step** in which rules reason about the created semantic descriptions and additional knowledge-bases like ontologies and user profile information, and the **user interface creation step** in which the RDF-descriptions resulting from the reasoning step are interpreted and translated into an appropriate, personalized user interface. We have developed this application for solving the following real-world problem: We provide personalized, syndicated views on the publications of a large European research project with more than twenty geographically distributed partners and embed this information with contextual information on the project, its working groups, information about the authors, related publications, etc.

keywords: web data extraction, web data syndication, personalized views.

Introduction

In today's information society, the World Wide Web plays a prominent role for disseminating and retrieving information: lots of useful information can be found in the web, from train departure tables to consultation hours, from scientific data to online auctions, and so on. While this information is already available for consumption by human users, we lack applications that can collect, evaluate, combine, and re-evaluate this information. Currently, users retrieve online content in separate steps, one step for each information request, and evaluate the information chunks afterwards according to their needs: e.g. the user compares the train arrival time with the starting time of the meeting he is requested to participate in, etc. Another common scenario for researchers is that a user reads some scientific publication, gets curious about the authors, other work of the authors, on related work targeting on similar research questions, etc. Linking these information chunks together is a task that can currently not be performed by machines. In our application, we show how to solve this information integration problem for the latter mentioned "researcher scenario". We show, how to

1. extract information from distributed and inhomogeneous sites, and create semantic descriptions of the extracted information chunks,
2. maintain the web data extraction to ensure up-to-date information and semantic descriptions,
3. reason about the created semantic descriptions and additional, ontological knowledge, and
4. create syndicated, personalized views on web information.

The Personal Publication Reader (PPR) extends the idea of Semantic Portals like e.g. SEAL [4] or others with the capability of extracting and syndicating web data from various, distributed sites or portals which do not belong to the ownership of the application itself.

1 Extraction & Annotation with Semantic Descriptions

In our application, the web pages from which we extract the information are maintained by partners of the research project REWERSE, thus the sources of the information are distributed and belong to different owners which provide their information in various ways and formats (HTML, Java-script, PHP-generated pages, etc.). Moreover, in each list, authors, titles and other entities are potentially characterized in a different way, and different order criteria are enforced (e.g. by year or by name). Such a web presentation is well suited for human consumption, but hardly usable for automatic processing. Nevertheless, the web is the most valuable information resource in this scenario. In order to access and understand these heterogeneous information sources one has to apply web extraction techniques. The idea of our application is to "wrap" these heterogeneous sources into a formal representation based on Semantic Web standards. In this way, each institution can still maintain their own publication list and at the same way we can offer an integrated and personalized view on this data by regularly extracting web data from all member sites.

This application is open in the sense that it can be extended in an easy way, i.e. by connecting additional web sources. For instance, abstracts from `www.researchindex.com` can be queried for each publication lacking this information and joined to each entry. Moreover, using text categorization tools one can rate and classify the contents of the abstracts. Another possibility is to extract organization and person data from the institution's web pages to inform the ontology to which class in the taxonomy an author belongs (such as full professor). Web extraction and annotation in the PPR is performed by the

Lixto Suite. Web data extraction is a hot topic in both the academic and commercial domain – for an extensive overview of methods and tools refer to [3]. First, with the *Lixto Visual Wrapper* [1] for each type of web site a so-called wrapper is created; the application designer visually and semi-automatically defines the characteristics of publication elements on particular web sites based on characteristics of the particular HTML presentation and some possible domain knowledge. After a wrapper has been generated it can be applied to a given web site (e.g. publications of University of Munich) to generate an "XML companion" that contains the relevant information stored in XML using (in this application context meaningful) XML tags.

2 Extraction Maintenance

In the next step, in the *Lixto Transformation Server* application designer visually composes the information flow from web sources to an RDF presentation that is handed over to the PPR once a week. Then the application designer defines a schedule how often which web source is queried and how often the information flow is executed. Additionally, deep web navigation macros possibly containing logins, cookies and web forms as well as iteration over forms are created. As a next step in the data flow, the data is harmonized to fit into a common structure, and e.g. an attribute "origin" is added containing the institution's name, and author names are harmonized by being mapped to a list of names known by the system. Finally, the XML data structure is mapped to a pre-defined RDF schema structure. Once the wrappers are in place, the complete application runs without further human interference, and takes care of publication updates. In case future extractions fail the application designers will receive a notification.

3 Reasoning for Syndicated & Personalized Views on Distributed Web Data

In addition to the extracted dynamic information, we maintain data about the members of the research project from the member's corner of the REWERSE project web site. We have constructed an ontology for describing researchers and their involvement in scientific projects like REWERSE, which extends the known Semantic Web Research Community Ontology (`http://ontobroker.semanticweb.org/ontos/swrc.html`) with some project-specific aspects.

Personalization rules reason about all this dynamic and static data in order to create syndicated and personalized views. As an example, the following rule (using the TRIPLE[5] syntax) determines all authors of a publication:

```
FORALL A, P authors(A, P) <- P[dc:creator -> A]@'http:..':publications.
```

In this rule, `@'http:..':publications` is the name of the model which contains the RDF-descriptions of the extracted publication informations. Further rules combine information on these authors from the researcher ontology with the author information. E.g. the following rule determines the employer of a project member, which might be a company, or a university, or, in general, some instance of a subclass of an organization (see line three below: here, we query for some subclass (direct or inferred) of the class "Organization"):

```
FORALL A,I works_at(A, I) <- EXISTS A_id,X (name(A_id,A)
  AND ont:A_id[ont:involvedIn -> ont:I]@'http:...#':researcher
  AND ont:X[rdfs:subClassOf -> ont:Organization]@rdfschema('..':researcher)
  AND ont:I[rdf:type -> ont:X]@'http:...#':researcher).
```

Disambiguation of results – here especially resource identification problems caused by varying author names – is achieved by an additional name identification step. For a user with specific interests, for example "interest in personalized information systems", information on respective research groups in the project, on persons working in this field, on their publications, etc., is syndicated.

4 User Interface Provision

We run the PPR within our Personal Reader framework for designing, implementing and maintaining personal Web Content Readers [2]. These personal Web Content Readers allow a user to browse information (the *Reader* part), and to access personal recommendations and contextual information on the currently regarded web resource (the *Personal* part). For the PPR, we instantiated a personalization Web service in our Personal Reader framework which holds the above mentioned rules. An appropriate visualization Web service for displaying the results of the reasoning step (which are provided as RDF documents and refer to an ontology of personalization functionality) has been implemented.

Availability of the Personal Publication Reader

The concept of the Personal Publication Reader and its functionality are summarized in a video, and so are the web data extraction and maintenance tasks. All demonstration videos and access to the application itself are available via `http://www.personal-reader.de/semwebchallenge/sw-challenge.html`.

References

1. R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *Proc. of VLDB*, 2001.
2. N. Henze and M. Kriesell. Personalization Functionality for the Semantic Web: Architectural Outline and First Sample Implementation. In *1st Int. Workshop on Engineering the Adaptive Web (EAW 2004)*, Eindhoven, The Netherlands, 2004.
3. S. Kuhlins and R. Tredwell. Toolkits for generating wrappers. In *Net.ObjectDays*, 2002.
4. A. Maedche, S. Staab, N. Stojanovice, and R.Studer. Semantic portal - the seal approach. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web*, pages 317–359. MIT-Press, 2003.
5. M. Sintek and S. Decker. TRIPLE - an RDF Query, Inference, and Transformation Language. In *International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

Reasoning and Ontologies for Personalized E-Learning in the Semantic Web

Nicola Henze[1], Peter Dolog[2], and Wolfgang Nejdl[1,2]

[1]ISI- Knowledge-Based Systems,

University of Hannover, Appelstr. 4, D-30167 Hannover, Germany

{henze,nejdl}@kbs.uni-hannover.de

[2] L3S Research Center,

University of Hannover, Expo Plaza 1, D-30539 Hannover, Germany

{dolog, nejdl}@learninglab.de

Abstract:

The challenge of the semantic web is the provision of distributed information with well defined meaning, understandable for different parties. Particularly, applications should be able to provide individually optimized access to information by taking the individual needs and requirements of the users into account. In this paper we propose a framework for personalized e-Learning in the semantic web and show how the semantic web resource description formats can be utilized for automatic generation of hypertext structures from distributed metadata. Ontologies and metadata for three types of resources (domain, user, and observation) are investigated. We investigate a logic-based approach to educational hypermedia using TRIPLE, a rule and query language for the semantic web.

keywords:

Educational hypermedia, Semantic web, Ontologies, Adaptive hypermedia, Reasoning on the semantic web.

Introduction

The vision of the semantic web is to enable machines to interpret and process information in the world wide web in order to better support humans in carrying out their various tasks with the web. Several technologies have been developed for shaping, constructing and developing the semantic web. Many of the so far developed semantic web technologies provide us with tools for describing and annotating resources on the web in standardized ways, e.g. with the Resource Description Framework (RDF [RDF, 2002]) and its binding to XML (eXtensible Markup Language [XML, 2003]). In this paper we will show how semantic web technologies and in particular ontologies can be used for building adaptive educational hypermedia systems. Adaptive educational hypermedia systems are able to adapt various visible aspects of the hypermedia systems to the individual requirements of the learners and are very promising tools in the area of e-Learning: Especially in the area of e-Learning it is important to take the different needs of learners into account in order to propose learning goals, learning paths, help students in orienting in the e-Learning systems and support them during their learning progress.

We propose a framework for such adaptive or personalized educational hypermedia systems for the semantic web. The aim of this approach is to facilitate the development of an adaptive web as envisioned e.g. in [Brusilovsky and Maybury, 2002]. In particular, we show how rules can be enabled to reason over distributed information resources in order to dynamically derive hypertext relations. On the web, information can be found in various resources (e.g. documents), in annotation of these resources (like RDF-annotations on the documents themselves), in metadata files (like RDF descriptions), or in ontologies. Based on these sources of information we can think of functionality allowing us to derive new relations between information.

Imagine the following situation: You are currently writing e-Learning materials for higher education. Especially in e-Learning, it is important to overcome the *one-size-fits-all* approach and provide learners with individual learning experiences. Learners have different requirements (like their individual learning style, their actual progress in the learning process, their individual background knowledge, but also more technical requirements like the device they are currently using for accessing the E-Learning materials, etc.). The e-Learning system you would like to use should provide such a personalized delivery of e-Learning materials. How can you describe instructional material in a way allowing for personalized e-Learning?

In our solution for personalized e-Learning systems we envision personal learning services capable of interpreting metadata-annotated learning resource, *understanding* their annotations with respect to standard ontologies for learning materials like e.g. LOM [LOM, 2002] or IMS [IMS, 2002]), and also with respect to specific domain ontologies which describe the particular subject being taught. To enable personalized delivery of the learning resources, ontologies for describing the learner and observations about the learner's interactions with the e-Learning system are required to characterize and model a learners current profile.

Each personal learning service possess reasoning rules for some specific adaptation purposes. These rules query for resources and metadata, and reason over distributed data and metadata descriptions. A major step for reasoning after having queried user profile, domain ontology and learning objects is to construct a temporally valid task knowledge base as a base for applying the adaptation rules. The concluded results of these personal learning services are described using the presentation format of the open hypermedia standard.

The paper is structured as follows: In the following section, we will compare our approach with related work. Section 3 describes the representation of resources with semantic web technologies, and shows our use of a domain, user, and observation ontologies. Section 4 discusses our approach to generate hypertext structures / associations, and an example set of rules for dynamically generating personalized associations between information. A comparison of our approach to related work and a conclusion end the paper.

Related Work

To describe and implement personalized e-Learning in the semantic web, there are at least three related research areas which contribute: *open hypermedia*, *adaptive hypermedia,*, and *reasoning for the semantic web*. Open hypermedia is an approach to relationship management and information organization for hypertext-like structure servers. Key features are the separation of relationships and content, the integration of third party applications, and advanced hypermedia data models allowing, e.g., the modeling of complex relationships . In open hypermedia, data models like FOHM (Fundamental Open Hypertext Model) [Millard et al., 2000] and models for describing link exchange formats like OHIF (Open Hypermedia

Interchange format) [Gronbaek et al., 2000] have been developed. The use of ontologies for open hypermedia has e.g. been discussed in [Kampa et al., 2001]. Here, an ontology is employed that clarifies the relations of resources. On base of this ontology, inference rules can derive new hypertext relations. In [Weal et al., 2001] the open hypermedia structures are used as an interface to ontology browsing. The links at the user interface are transformed to queries over ontology. Thus links serves as contexts for particular user.

The question whether conceptual open hypermedia is the semantic web has been discussed in [Bechhofer et al., 2001]. In [Carr et al., 2001], a *metadata space* is introduced, where the openness of systems and their use of metadata is compared. On the *metadata dimension* (x-axis), the units are the use of *keywords, thesauri, ontologies*, and *description logic*. The y-axis describes the *openness dimension* of systems starts from *CD ROM / file system, Internet, Web*, and ends with *Open* systems. Our approach can be seen as employing reasoning capabilities for Web-resources, or, concrete, to be on the crossings of description logic in the metadata dimension and Web in the openness dimension.

Adaptive hypermedia has been studied normally in closed worlds, i.e. the underlying document space / the hypermedia system has been known to the authors of the adaptive hypermedia system at design time of the system. As a consequence, changes to this document space can hardly be considered: A change to the document space normally requires the reorganization of the document space (or at least some of the documents in the document space). To open up this setting for dynamic document or information spaces, approaches for so called *open corpus adaptive hypermedia systems* have been discussed [Brusilovsky, 2001,Henze and Nejdl, 2001]. Our approach to bring adaptive hypermedia techniques to the web therefore contribute to the open corpus problem in AH. The relation of adaptive hypermedia and open hypermedia has for example been discussed in [Bailey et al., 2002].

In our approach, we use several ontologies for describing the features of *domains*, *users*, and *observations*. Compared to the components of adaptive hypermedia systems [Henze and Nejdl, 2003], an ontology for adaptive functionality is missing. However, such an ontology can be derived using the "updated taxonomy of adaptive hypermedia technologies" in [Brusilovsky, 2001]. Reasoning over these distributed ontologies is enabled by the RDF-querying and transformation language TRIPLE. Related approaches in the area of querying languages for the semantic web can be found, e.g., in [Bry and Schaffert, 2002]. Here, a rule-based querying and transformation language for XML is proposed. A discussion of the interoperability between Logic programs and ontologies (coded in OWL or DAML+OIL) can be found in [Grosof et al., 2003].

Reasoning in open worlds like the semantic web is not fully explored yet, sharing and reusing of resources with high quality is still an open problem. In this paper, we discussed first ideas on the application of rules and rule-based querying and transformation language for the domains of open hypermedia and adaptive hypermedia.

Representation of Resources

Semantic web technologies like the Resource Description Format (RDF) [Lassila and Swick, 2002] or RDF schema (RDFS) [RDF, 2002] provide us with interesting possibilities. RDF schemas serve to define vocabularies for metadata records in an RDF file. RDF schemas can be used to describe resources, e.g. the RDF bindings of Learning Object Metadata (LOM) [Nilsson, 2001] can be used for these purposes, or RDF bindings of Dublin Core [Dublin Core, 2004]. There is no restriction on the use of different schemas together in one RDF file or RDF model. The schema identification comes with attributes being used from that schema so backward dereferencing is again easily possible.

For example the RDF model of a lecture can use an attribute `subject` from Dublin Core Standard together with `isPartOf` from dublin core metadata terms, etc. Part of an RDF-description for a course on Java programming can be seen in the following example. We have annotated the online version of the Sun Java tutorial [Campione and Walrath, 2000], which is a freely available online tutorial on Java programming.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<rdf:RDF xml:lang="en"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms="http://purl.org/dc/terms#">

<rdf:Description
rdf:about="http://java.sun.com/docs/books/tutorial/index.html">
   <rdf:type rdf:resource="http://ltsc.ieee.org/2002/09/lom-
educational#lecture"/>
   <dc:title>The Java Tutorial (SUN)</dc:title>
   <dc:description>A practical guide for programmers with hundreds of
      complete, working examples and dozens of trails - groups of lessons
      on a particular subject.
   </dc:description>
...
</rdf:Description>

<rdf:Description rdf:about="Object-Oriented_Programming_Concepts">
   <dc:title>Object-Oriented Programming Concepts</dc:title>
   <dcterms:isPartOf
rdf:resource="http://java.sun.com/docs/books/tutorial/index.html"/>
   <dcterms:hasPart>
      <rdf:Seq>
         <rdf:li rdf:resource="#What_Is_an_Object"/>
         <rdf:li rdf:resource="#What_Is_a_Message" />
         <rdf:li rdf:resource="#What_Is_a_Class"/>
         <rdf:li rdf:resource="#What_Is_Inheritance"/>
         <rdf:li rdf:resource="#What_Is_an_Interface"/>
         <rdf:li
rdf:resource="#How_Do_These_Concepts_Translate_into_Code"/>
         <rdf:li rdf:resource="#Questions_and_Exercises_Object-
Oriented_Concepts"/>
     </rdf:Seq>
   </dcterms:hasPart>
</rdf:Description>

....

<rdf:Description rdf:about="What_Is_an_Object">
   <dc:title>What Is an Object?</dc:title>
   <dc:description>An object is a software bundle of related variables
       and methods. Software objects are often used to model real-world
       objects you find in everyday life. </dc:description>
   <dc:language rdf:resource=
            "http://www.kbs.uni-hannover.de/~henze/lang.rdf#en"/>
   <dc:subject rdf:resource=
            "http://www.kbs.uni-hannover.de/~henze/java.rdf#OO_Objects"/>
   <dcterms:isPartOf rdf:resource="#Object-Oriented_Programming_Concepts"/>
</rdf:Description>

...

</rdf:RDF>
```

While RDF schema provides a simple ontology language, more powerful ontology languages which reside on top of RDF and RDF schema are available, too. For example, ontology languages like DAML+OIL [DAML+OIL, 2001] (the joint initiative of DAML (Darpa Agent Markup Language) and OIL (Ontology Inference Layer)) provide ontology layers on top of RDF / XML. Recently, OWL [OWL, 2003] (Web Ontology Language) has been developed, further enriching RDF.

An open question is how we can combine reasoning mechanisms on these (distributed) metadata and data resources, in order to generate hypertext presentations, link structures, etc., to bring the interoperability ideas from OHS to the WWW. This section will first describe semantic web tools that we employ in our approach, and then describe some structures for metadata components which allow us to generate link structures according to user features.

Bringing together Resources and Reasoning

On top of the RDF and ontology-layer, we find the layer of logic in the semantic web tower, or, more recently, the layers of rules and logic framework [Berners-Lee, 2002]. In our approach, the communication between reasoning rules and the open information environment will take place by exchanging RDF annotations: the rules reason over distributed RDF-annotations, results will be given back as RDF-files, too.

A rule language especially designed for querying and transforming RDF models is TRIPLE [Sintek and Decker, 2002]. Rules defined in TRIPLE can reason about RDF-annotated information resources (required translation tools from RDF to triple and vice versa are provided).

TRIPLE supports *namespaces* by declaring them in clause-like constructs of the form *namespaceabbrev := namespace*, resources can use these namespaces abbreviations.

```
sun_java := "http://java.sun.com/docs/books/tutorial".
```

Statements are similar to F-Logic object syntax: An RDF statement (which is a triple) is written as `subject[predicate` $\rightarrow$ `object]`. Several statements with the same subject can be abbreviated in the following way:

```
sun_java:'index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
```

RDF *models* are explicitly available in TRIPLE: Statements that are true in a specific model are written as "@model", e.g.

```
doc:OO_Class[rdf:type->doc:Concept]@results:simple.
```

Connectives and quantifiers for building logical formulae from statements are allowed as usual, i.e. $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$, etc. For TRIPLE programs in plain ASCII syntax, the symbols `AND`, `OR`, `NOT`, `FORALL`, `EXISTS`, `<-`, `->`, etc. are used. All variables must be introduced via quantifiers, therefore marking them is not necessary.

Domain Ontologies

First of all we need to determine a domain ontologies. Domain ontologies comprise usually classes (classifies objects from a domain) and relationships between them. One possible domain in hypermedia application can be a domain of documents and concepts described in an application domain.

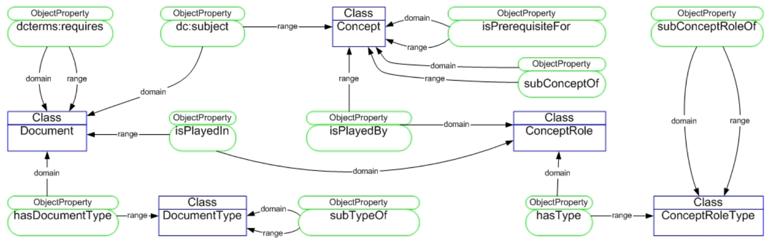


Figure 1: Ontology of documents

A simple ontology for documents and their relationships to other components is depicted in fig. 1. The class `Document` is used to annotate a resource which is a document. Documents describe some concepts. We use class `Concept` to annotate concepts. Concepts and documents are related through `dc:subject` property. Documents can be ordered by `dcterms:requires` relationship. Concepts and documents have a certain role in their collaboration in certain document. We represent these facts by instances of `DocumentRole` class and its two properties: `isPlayedIn` and `isPlayedBy`. Concepts, document roles and concept roles can form hierarchies. We define `subRoleOf`, `subConceptRoleOf`, and `subConceptOf` properties for these purposes. Concepts play a certain role in a document. We recognize `Introduction` and `FullDescription` concept roles.

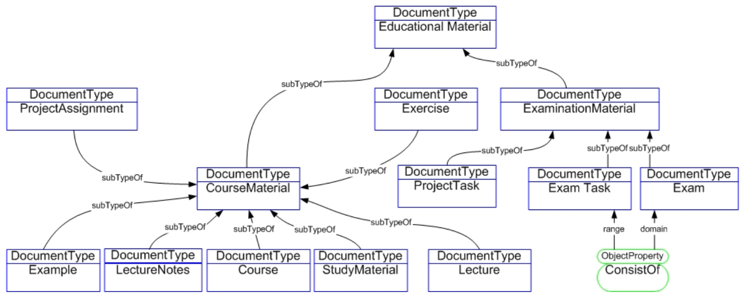


Figure 2: Ontology for documents types

Document can have a type. Figure 2 depicts the ontology with several document types for educational domain. The most general document type is `Educational Material`. `Educational Material` has two subtypes: `Course Material` and `Examination Material`. `Examination Material` can be further specialized to `Project Task`, `Exam Task`, and `Exam`. The `Exam` can consist of the `Exam Task`-s.

`Course Material` can be further specialized into `Lecture`, `Example`, `LectureNote`, `Course`, `Exercise`, and `Project Assignment`.

The document roles represent intended usage of the document in general. When a document is authored it is already known whether it will be a Lecture, Example and so on and it hardly fits to another role. Besides document roles, we recognize document types as well. Document types represent different context of a document. It means that we can differentiate at least between examination and study material. These are represented as separate document types `StudyMaterial` and `ExaminationMaterial`.

Figure 3 depicts `Programming_Strategies` concept with its subconcepts: `Object_Oriented`, `Imperative`, `Logical`, and `Functional`. `OO_Class`, `OO_Method`, `OO_Object`, `OO_Inheritance`, and `OO_Interface` are depicted as subconcepts of `Object_Oriented`.

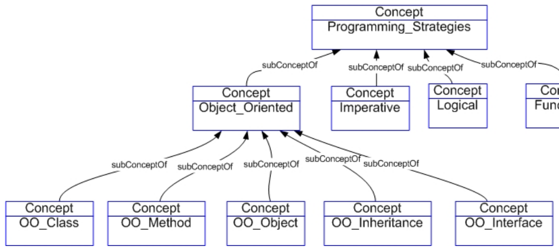


Figure 3: Concept ontology for Java e-lecture

Above described ontologies are used then in annotations of concrete documents/resources. An example of such resource can be a page describing sun_java:'java/concepts/class.html'. Following example shows how such a page can be annotated based on ontologies.

```
sun_java:'java/concepts/class.html'[
rdf:type->doc:Document;
dc:subject->doc:OO_Class].

doc:OO_Class[
rdf:type->doc:Concept;
doc:isPrerequisiteFor->doc:OO_Inheritance;
doc:subConceptOf->doc:Classes_and_objects].

doc:ClassesIntroduction[
rdf:type->doc:ConceptRole;
doc:isPlayedBy->doc:OO_Class;
doc:isPlayedIn->sun_java:'java/concepts/class.html';
doc:hasType->doc:Introduction].

doc:Introduction[
rdf:Type->doc:ConceptRoleType;
doc:subConceptRoleOf->doc:Cover].
```

The page is a document (RDF type `Document`). It describes information about classes. Thus it is annotated with `OO_Class` concept covered in the page. The `OO_Class` concept is annotated with type `Concept` and is subconcept of the `Classes_and_objects` concept. The `OO_Class` concept is prerequisite for the `OO_Inheritance`. A page can have prerequisites. Then the `dcterms:requires` property can be used in the annotation.

The `OO_Class` concept plays a role of introduction in the `sun_java:'java/concepts/class.html'` document. This is annotated by `ClassesIntroduction` resource, which is of type `ConceptRole`. The reference to `OO_Class` concept and the document where it plays the introduction role is annotated by using properties isPlayedBy and isPlayedIn respectively. The role has type `Introduction`. The `Introduction` is of type `ConceptRoleType` and is subtype of `Cover` concept role type.

Users

Data about a user serves for deriving contextual structures. It is used to determine how to adapt the presentation of hypertext structures. Here we define an ontology for a user profile based on IEEE Personal and Private Information (PAPI) [IEEE, 2000]. PAPI distinguishes *personal*, *relations*, *security*, *preference*, *performance*, and *portfolio* information. The *personal* category contains information about names, contacts and addresses of a user. *Relations* category serves as a category for specifying relationships between users (e.g. classmate, teacherIs, teacherOf, instructorIs, instructorOf, belongsTo, belongsWith). *Security* aims to provide slots for credentials and access rights. *Preference* indicates the types of devices and objects, which the user is able to recognize. *Performance* is for storing information about measured performance of a user through learning material (i.e. what does a user know). *Portfolio* is for accessing previous experience of a user. Each category can be extended. For more discussion on learner modeling standards see for example [Dolog and Nejdl, 2003].

Figure 4 depicts an example of an ontology for a learner profile. The ontology is based on *performance* category of PAPI. We are storing sentences about a learner which has a Performance. The Performance is based on learning experience (learningExperienceIdentifier), which is taken from particular document. The experience implies a Concept learned from the experience, which is maintained by learningCompetency property. The Performance is certified by a Certificate, which is issued by a certain Institution. The Performance has a certain PerformanceValue, which is in this context defined as a float number and restricted to interval from 0 to 1.

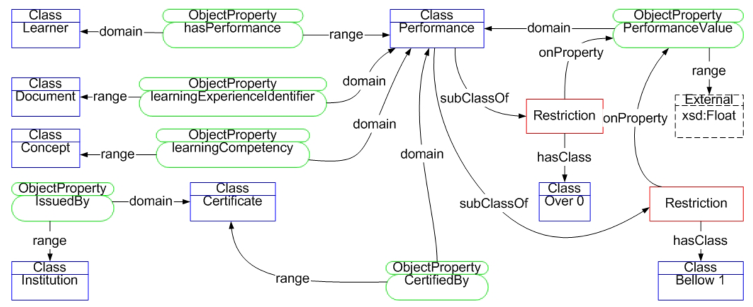


Figure 4: Ontology for learner performance

Another possibility to restrict the PerformanceValue is to define it with a range of LevelOf Knowledge. Then the instances of the class can be taken as measures of the learner performance.

The example of simple learner profile can look as follows.

```
user:user2[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user2P].

user:user2P[
  rdf:type->learner:Performance;
  learner:learningExperienceIdentifier-
>sun_java:'java/concepts/object.html';
  learner:learningCompetency->doc:OO_Object;
  learner:CertifiedBy->KBScerturi:C1X5TZ3;
  learner:PerformanceValue->0.9
].

KBScerturi:C1X5TZ3[
rdf:type->learner:Certificate;
learner:IssuedBy->KBSuri:KBS
].

KBSuri:KBS[
rdf:type->learner:Institution
].
```

The learner user2 has the performance (user2P) record. The performance contains a learning experience about the KBS Java objects resource. The concept covered in the resource is stored in the performance as well. Then a certificate about the performance with performance value and institution who issued the certificate is recorded into the learner performance as well.

Observations

During runtime, users interact with a hypertext system. The user's interactions can be used to draw conclusions about possible user interests, about user's goal, user's task, user's knowledge, etc. These concluded user features can, as described in the previous section, be used for providing personalized views on hypertexts. An ontology of observations should therefor provide a structure of information about possible user observations, and - if applicable - their relations and/or dependencies.

A simple ontology for observations is depicted in fig. 5. The ontology allow us to instantiate facts that a Learner has interacted with (hasInteraction property) with a particular Document (isAbout property) via an interaction of a specific type (InteractionType). The interaction has taken place in a time interval between beginTime and endTime, and has certain level (Level) associated, the ObservationLevel. Several events (see next section) can contribute to an interaction. Example of InteractionTypes are of kind access, bookmark, annotate, examples for ObservationLevels are that a user has visited a page, has worked on a project, has solved some exercise, etc.

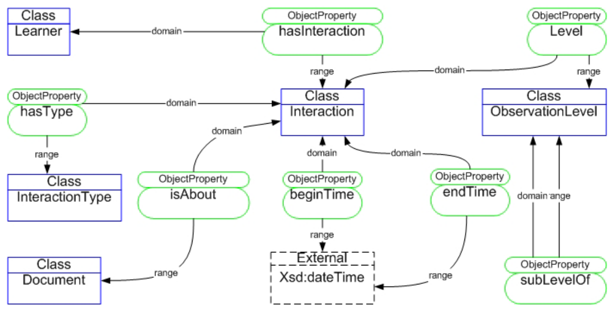


Figure 5: Ontology for observations

Generating Hypertext Structures

Hypertext structures as described in several works on open hypermedia (see e.g [Millard et al., 2000]) can be generated from metadata reported in the previous section. We do not store the hypertext structures on servers as first class entities but we allow to generate such structures on the fly. In order to generate such hypertext structures we need an ontology for structures. Then transformation rules can be used to generate instances of that structure.

Presentation Ontology

A presentation ontology is used for describing structure relevant for visualization. Such an ontology adapted from FOHM [Millard et al., 2000] is depicted in fig. 6.

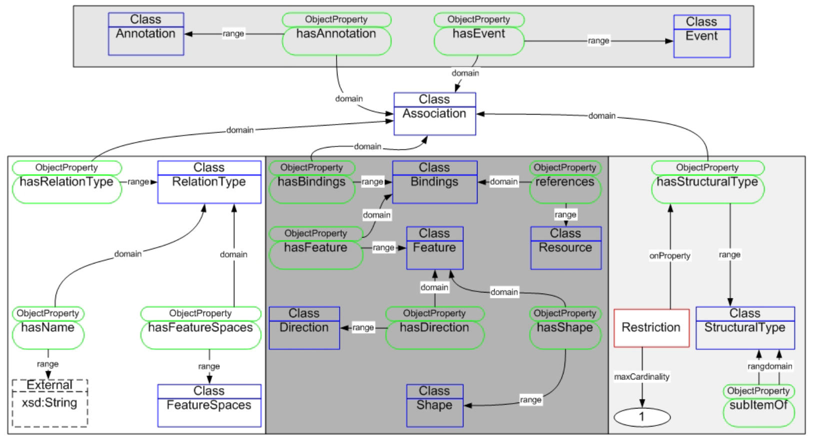


Figure 6: A part of presentation ontology

The main element of the ontology is the `Association`. Like in [Millard et al., 2000], the `Association` is built from three components: `Bindings`, `RelationType`, and `StructuralType` (in FOHM they refer to it as Cartesian product of bindings, relation type and structural type). These three components (classes) are related to association through `hasBindings`, `hasRelationType`, and `hasStructuralType` properties.

`Bindings` references a particular `Resource` on the web (document, another association, etc.), and `Feature`-s. A `Feature` can be a `Direction`, `Shape`, etc. Entries for `Direction` are depicted in figure 7b, entries for `Shape` are depicted in the figure 7c.

The `RelationType` has a `Name` which is a string. The `RelationType` also points to the `FeatureSpaces`. Entries for the `FeatureSpaces` are depicted in figure 7a. A `StructuralType` is one of stack, link, bag, or sequence of resources.

In addition, `Association` can have associated events (e.g. click events for processing user interactions) through `hasEvent` property, and an annotation (e.g. green/red/yellow icon from traffic light metaphor technique from adaptive hypermedia) through `hasAnnotation` property.

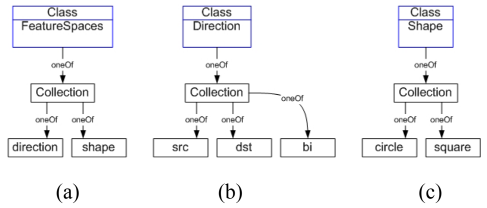


Figure 7: Members of Collection of: (a) Feature Spaces, (b) Direction, (c) Shape.

The `hasEvent` property defines an event which is provided within the document (to be able to get appropriate observation). Whenever the event is generated observation reasoning rules assigned to this type of event are triggered. The `represents` property references a resource, which is stored in observations about learner, after an event is generated as well.

FOHM introduces *context* and *behavior* objects. Filtering and contextual restrictions maintained by the *context* objects in FOHM is substituted by more richer reasoning language and rules in our approach. On the other hand, interactions and observations together with events substitute the notion of *behavior* objects.

Reasoning Rules

In this chapter we show how rules are employed to reason over distributed information sources (ontologies, user profile information, resource descriptions). The communication between reasoning rules and the open information environment will take place by exchanging RDF annotations [RDF, 2002]. Rules are encoded in the TRIPLE rule language (see section 3.1). For further examples on adaptation rules we refer the reader to [Dolog et al., 2003].

In the following, we provide a set of rules that can be used to construct an *example*-relation between resources. Assume a user `U` is visiting some page `D`. An example, illustrating the content of this page, can be found by comparing the concepts explained on the current page with the concepts shown on an example page. Several grades of how good an example is can be derived.

The easiest way for deriving an example-relation to a page `D` is by ensuring that each concept on `D` is covered by the example `E`:

```
FORALL D, E example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C2 (D[dc:subject->C2] -> E[dc:subject->C2]).
```

The second line in the rule above ensures that `D` is `StudyMaterial` and `E` is an `Example` (according to the ontology of documents "docs"). The third rule is verifying that `D` really is about some measurable concept - thus there exists a metadata annotation like `dc:subject`.

The fourth line then really expresses what our rule should check: Whether each concept on `D` will be explained in the example `E`.

Another possibility is to provide relations to examples that cover exactly the same concepts as a page `D`:

```
FORALL D, E exact_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C1 (D[dc:subject->C1] -> E[dc:subject->C1]) AND
    FORALL C2 (E[dc:subject->C2] -> D[dc:subject->C2]).
```

The second and third line in this rule are the same as in the previous rule. The fourth and fifth line ensure that each concept on `D` is covered on `E` and vice versa.

If we want to show examples which might illustrate only some aspects of a page `D`, we can derive relations to *weaker* examples by

```
FORALL D, E weaker_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C (D[dc:subject->C] AND E[dc:subject->C]).
```

which is be valid whenever at least on concept explained on `D` is part of the example `E`.

From the area of adaptive hypermedia, several methods and techniques have been provided to adapt the navigation and / or the content of a hyperspace to the needs, preferences, goals, etc. of each individual user. In [Henze and Nejdl, 2003] we have provided a logical characterization of adaptive educational hypermedia based on First Order Logic (FOL). There, an adaptive educational hypermedia system is described in FOL as a quadruple consisting of a *document space* - a hypermedia system which document nodes and their relations, a *user model* for modeling and inferencing on various individual characteristics of a user, an *observation component* which is responsible for monitoring a user's interaction with the system, and an *adaptation component* which consists of rules which describe adaptive functionality. A way to implement open adaptive hypermedia system is shown in [Dolog et al., 2003]. In this paper, we will use adaptive hypermedia to provide personalized associations. We can think of a personalized *pedagogical* recommendation of examples: The best example is an example that shows the new things to learn in context of already known / learned concepts: This would embed the concepts to learn in the previous learning experience of a user. The rule for derive this *best_example* is as follows:

```
FORALL D, E, U best_example(D,E,U) <-
    studyMaterial(D) AND example(E) AND user(U) AND example(D,E) AND
    FORALL C ( (E[dc:subject->C] AND NOT D[dc:subject->C]) ->
           p_obs(C, U, Learned) ).
```

The rule for determining whether a user has learned some concept `C (p_obs(C, U, Learned)` is derived by checking the characteristics of the user profile. A concept is assumed to be learned if we find a `Performance` of this user via the user profile, which is related to the concept in question.

```
FORALL C, U  p_obs(C, U, Learned) <- user(U) AND concept(C) AND
    EXISTS P (U[learner:hasPerformance->P] AND user_performance(P) AND
    P[learner:learningCompetency->C]).
```

The results of these rules (on the RDF-annotated and to triple translated resources provided in the Appendix) is e.g. that a page on "objects in Java (object.html)" can be related to pages which show "concepts of object orientation in Java (practical.html)" or "objects and methods in Java (objects_methods.html)". These relations are derived by using the general "example"-rule:

```
D = sun_java:'java/concepts/object.html', E =
sun_java:'java/concepts/practical.html'
D = sun_java:'java/concepts/object.html', E =
kbs_java:'java_script/examples/objects_methods.html'
```

The "exact_example-rule" from above derives for this data set that only the "overview on object-orientation in Java (OO_overview.html)" has an exact matching example.

```
D = kbs_java:'java_script/concepts/OO_overview.html',
E = sun_java:'java/concepts/practical.html'
```

The "weaker_example-rule" suggest the same example page (practical.html) which exactly fits to the document OO_overview.html also to pages about only some aspects like "methods in Java (message.html).

```
D = sun_java:'java/concepts/message.html',
E = sun_java:'java/concepts/practical.html'
```

The "best_example" for a user who is currently visiting a page on "methods in Java (message.html)" and who has already knowledge about "objects in java" is an example illustrating these two concepts (object_methods.html). In the data set provided in the appendix, user2 is currently in this position.

```
D = sun_java:'java/concepts/message.html',
E = kbs_java:'java_script/examples/objects_methods.html',
U = user:user2
```

Further rules for generating personalized hypertext associations can be used by more extensive use of facts from domain, user, and observation ontology. E.g. the mentioned `subConceptOf` relationship in the concept-ontology of the java application domain can be for example utilized to recommend either more general documents introducing a concept of programming strategies in general, or to recommend more specific documents (resources) about object oriented programming strategy based on requirements, level of knowledge, or interest of a user.

Sequencing relationship is another relationship which can be used to recommend documents. A document (resource) which describes a concept (the concept appears in `dc:subject` slot in metadata about the document) from the beginning of the sequence will be recommended sooner than a document which describes a concept from the end of such a sequence.

A dependency relationship referring to whether a concept depends on another concept can be used as well. It can be used to recommend documents which describe dependent concepts together with a document describing a concept which was recommended by another rule.

Conclusion and Further Work

In this paper, we have proposed an approach for dynamically generating personalized hypertext relations powered by reasoning mechanisms over distributed RDF annotations. We have shown an example set of reasoning rules that decide for personalized relations to example pages given some page. Several ontologies have been used which correspond to the components of an adaptive hypermedia system: a domain ontology (describing the document space, the relations of documents, and concepts covered in the domain of this document space), a user ontology (describing learner characteristics), and an observation ontology (modeling different possible interactions of a user with the hypertext). For generating hypertext structures, a presentation ontology has been introduced. We have been developing a demonstrator system showing the realization of the formalizm we presented in this paper. This demonstrator, the *Personal Reader* [Dolog et al., 2004a], generates a personalized conceptual context of learning resources. This context is generated by using adapation rules like those presented in this paper, and integrates this technology with a personalized search facility [Dolog et al., 2004b].

In further work, we plan to extend our demonstrator, and to investigate how to employ further ontologies like an ontology for educational models. This will enable us to add additional rules to enhance adaptive functionality based on the facts modeled in the knowledge-base by utilizing additional relationships.

Bibliography

[Bailey et al., 2002] Bailey, C., Hall, W., Millard, D., and Weal, M. (2002). Towards open adaptive hypermedia. In *Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, Malaga, Spain.

[Bechhofer et al., 2001] Bechhofer, S., Carr, L., Goble, C., and Hall, W. (2001). Conceptual open hypermedia = the semantic web? In *Second International Workshop on the Semantic Web*, Hong Kong, China.

[Berners-Lee, 2002] Berners-Lee, T. (2002). The semantic web - mit/lcs seminar. http://www.w3c.org/2002/Talks/09-lcs-sweb-tbl/.

[Brusilovsky, 2001] Brusilovsky, P. (2001). Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87-100.

[Brusilovsky and Maybury, 2002] rusilovsky, P. and Maybury, M. (2002). *The Adaptive Web*. Communications of the ACM.

[Bry and Schaffert, 2002] Bry, F. and Schaffert, S. (2002). A gentle introduction into xcerpt, a rule-based query and transformation language for xml. In *International Workshop on Rule Markup Languages for Buisiness Rules on the Semantic Web*, Sardinia, Italy.

[Campione and Walrath, 2000] Campione, M. and Walrath, K. (2000). *The Java(TM) Tutorial: A Short Course on the Basics (3rd Edition)*. Addison-Wesley.

[Carr et al., 2001] Carr, L., Bechhofer, S., Goble, C., and Hall, W. (2001). Conceptual linking: Ontology-based open hypermedia. In *Proceedings of the Tenth International World Wide Web Conference*, Hongkong.

[DAML+OIL, 2001] DAML+OIL (2001). DAML+OIL. http://www.daml.org/2001/03/daml+oil-index.html.

[Dolog et al., 2003] Dolog, P., Henze, N., Nejdl, W., and Sintek, M. (2003). Towards an adaptive semantic web. In *Principles and Practive of Semantic Web Reasoning (PPSWR'03)*, Mumbay, India.

[Dolog et al., 2004a] Dolog, P., Henze, N., Nejdl, W., and Sintek, M. (2004a). The personal reader: Personalizing and enriching learning resources using semantic web technologies. Technical report, Univ. of Hannover. submitted for publication.

[Dolog et al., 2004b] Dolog, P., Henze, N., Nejdl, W., and Sintek, M. (2004b). Personalization in distributed e-learning environments. In *International World Wide Web Conference*, New York, USA.

[Dolog and Nejdl, 2003] Dolog, P. and Nejdl, W. (2003). Challenges and benefits of the semantic web for user modelling. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, Budapest, Hungary.

[Dublin Core, 2004] Dublin Core (2004). Dublin Core. http://dublincore.org/.

[Gronbaek et al., 2000] Gronbaek, K., Sloth, L., and Bouvin, N. O. (2000). Open hypermedia as user controlled meta data for the web. In *Ninth International World Wide Web Conference*, pages 554-566, Amsterdam, The Netherlands.

[Grosof et al., 2003] Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: Combining logic programs with description logic. In *Twelth International World Wide Web Conference*, Budapest, Hungary.

[Henze and Nejdl, 2001] Henze, N. and Nejdl, W. (2001). Adaptation in open corpus hypermedia. *IJAIED Special Issue on Adaptive and Intelligent Web-Based Systems*, 12.

[Henze and Nejdl, 2003] Henze, N. and Nejdl, W. (2003). Logically characterizing adaptive educational hypermedia systems. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, Budapest, Hungary.

[IEEE, 2000] IEEE (2000). IEEE P1484.2/D7, 2000-11-28. draft standard for learning technology. public and private information (papi) for learners (papi learner). Available at: http://ltsc.ieee.org/wg2/. Accessed on October 25, 2002.

[IMS, 2002] IMS (2002). IMS: Standard for Learning Objects. http://www.imsglobal.org/.

[Kampa et al., 2001] Kampa, S., Miles-Board, T., Carr, L., and Hall, W. (2001). Linking with meaning: Ontological hypertext for scholars. Technical report, University of Southampton. citeseer.nj.nec.com/kampa01linking.html.

[Lassila and Swick, 2002] Lassila, O. and Swick, R. (2002). W3c resource description framework (rdf) model and syntax specification. Available at: http://www.w3.org/TR/REC-rdfsyntax/. Accessed on October 25, 2002.

[LOM, 2002] LOM (2002). LOM: Draft Standard for Learning Object Metadata. http://ltsc.ieee.org/wg12/index.html.

[Millard et al., 2000] Millard, D. E., Moreau, L., Davis, H. C., and Reich, S. (2000). FOHM: a fundamental open hypertext model for investigating interoperability between hypertext domains. In *11th ACM Conference on Hypertext and Hypermedia*, pages 93-102, San Antonio, Texas, USA.

[Nilsson, 2001] Nilsson, M. (2001). Ims metadata rdf binding guide. http://kmr.nada.kth.se/el/ims/metadata.html.

[OWL, 2003] OWL (2003). OWL. http://www.w3.org/2001/sw/WebOnt/.

[RDF, 2002] RDF (2002). Resource Description Framework (RDF) Schema Specification 1.0. http://www.w3.org/TR/rdf-schema.

[Sintek and Decker, 2002] Sintek, M. and Decker, S. (2002). Triple - an rdf query, inference, and transformation language. In Horrocks, I. and Hendler, J., editors, *International Semantic Web Conference (ISWC)*, pages 364-378, Sardinia, Italy. LNCS 2342.

[Weal et al., 2001] Weal, M. J., Hughes, G. V., Millard, D. E., and Moreau, L. (2001). Open hypermedia as a navigational interface to ontological information spaces. In *Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 227-236. ACM Press.

[XML, 2003] XML (2003). XML: extensible Markup Language. http://www.w3.org/XML/.

Appendix: Set of Rules for Deriving Relations between Information Pages and Examples

```
daml  := "http://www.daml.org/.../daml+oil#".
rdf   := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
doc   := "http://www.example.org/doc#".
results := "http://www.results.org/results#".
sun_java := "http://java.sun.com/docs/books/tutorial/".
kbs_java := "http://www.kbs.uni-hannover.de/".
java := "http://www.kbs.uni-hannover.de/~henze/java.rdf#".

@results:data{
sun_java:'index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/concepts/index.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial].
sun_java:'java/concepts/object.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Object'].
sun_java:'java/concepts/message.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Method'].
sun_java:'java/concepts/class.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Class'].
sun_java:'java/concepts/inheritance.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Inheritance'].
sun_java:'java/concepts/interface.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Interface'].
sun_java:'java/concepts/practical.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:Example;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method';
  dc:subject->java:'OO_Class';
  dc:subject->java:'OO_Inheritance';
  dc:subject->java:'OO_Interface'].

kbs_java:'java_script/examples/objects_methods.html'[rdf:type-
>doc:Document;
  doc:hasDocumentType->doc:Example;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method'].
kbs_java:'java_script/concepts/OO_overview.html'[rdf:type->doc:Document;
  doc:hasDocumentType->doc:StudyMaterial;
  dc:subject->java:'OO_Object';
  dc:subject->java:'OO_Method';
  dc:subject->java:'OO_Class';
  dc:subject->java:'OO_Inheritance';
  dc:subject->java:'OO_Interface'].

java:'OO_Object'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Method'].

java:'OO_Method'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Class'].

java:'OO_Class'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Inheritance'].

java:'OO_Inheritance'[rdf:type->doc:Concept;
  doc:isPrerequisiteFor->java:'OO_Interface'].

user:user1[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user1P].
user:user1P[
  rdf:type->learner:Performance].

user:user2[
  rdf:type -> learner:Learner;
  learner:hasPerformance -> user:user2P].

user:user2P[
  rdf:type->learner:Performance;
  learner:learningCompetency -> java:'OO_Object'].
}

@results:simple{

  FORALL O,P,V O[P->V] <-
    O[P->V]@results:data.

  FORALL D document(D) <- D[rdf:type->doc:Document].
  FORALL C concept(C) <- C[rdf:type->doc:Concept].
  FORALL U user(U) <- U[rdf:type->learner:Learner].
  FORALL P user_performance(P) <- P[rdf:type->learner:Performance].
  FORALL E example(E) <- document(E) AND
          E[doc:hasDocumentType->doc:Example].
  FORALL E studyMaterial(E) <- document(E) AND
          E[doc:hasDocumentType->doc:StudyMaterial].


  FORALL C, U  p_obs(C, U, Learned) <- user(U) AND concept(C) AND
    EXISTS P (U[learner:hasPerformance->P] AND user_performance(P) AND
    P[learner:learningCompetency->C]).

  FORALL D, E example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C2 (D[dc:subject->C2] -> E[dc:subject->C2]).

  FORALL D, E exact_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C1 (D[dc:subject->C1]) AND
    FORALL C1 (D[dc:subject->C1] -> E[dc:subject->C1]) AND
    FORALL C2 (E[dc:subject->C2] -> D[dc:subject->C2]).

  FORALL D, E weaker_example(D,E) <-
    studyMaterial(D) AND example(E) AND
    EXISTS C (D[dc:subject->C] AND E[dc:subject->C]).

  FORALL D, E, U best_example(D,E,U) <-
    studyMaterial(D) AND example(E) AND user(U) AND example(D,E) AND
    FORALL C ( (E[dc:subject->C] AND NOT D[dc:subject->C]) ->
           p_obs(C, U, Learned) ).

}

/* Several Views */
FORALL D, E <- example(D, E)@results:simple.
FORALL D, E <- exact_example(D, E)@results:simple.
FORALL D, E <- weaker_example(D, E)@results:simple.
FORALL D, E, U <- best_example(D, E, U)@results:simple.
```