Towards Agent-Oriented Knowledge Base Maintenance for Description Logic $\mathcal{ALCN}$

Stanislav Ustymenko[1] and Daniel G. Schwartz[2]

[1] Meritus University, School of Information Technology,
30 Knowledge Park Drive (Suite 301), Fredericton, New Brunswick, Canada E3C 2R2 ,
sustymenko@meritusu.ca
[2] Department of Computer Science, Florida State University,
Tallahassee, Florida, USA
schwartz@cs.fsu.edu

Abstract

Artificial agents functioning in the Semantic Web are to be capable of getting knowledge from diverse sources. This implies the capability to continuously update their knowledge bases. New stream reasoning concepts make this need even more pressing. Semantic Web ontologies are commonly represented using description logic knowledge bases. We propose an agent architecture with such features, utilizing a Dynamic Reasoning System (DRS). This explicitly portrays reasoning as a process taking place in time and allows for manipulating inconsistent knowledge bases. We sketch a procedure for user-directed ontology debugging. This same mechanism can be used for automated belief revision. We identify important research directions that may benefit from this approach.

1 Introduction

The Semantic Web (SW) [4] is a common name of a family of technologies extending the Web with rich, machine-interpretable knowledge. The SW retains the massively decentralized nature of current World Wide Web, with an unlimited number of knowledge sources identifiable by unique URIs. It supports rich metadata annotation, including expressive ontology languages. Description Logics (DLs) [2] emerged as leading formalism for knowledge representation and reasoning on the Semantic Web.

Once widely implemented, the Semantic Web will support intelligent software agents that will work with massive, decentralized ontologies, while other agents modify them in possibly inconsistent ways. Agents will need a way to absorb new knowledge in a timely fashion, all the while protecting the consistency of their knowledge bases, or, alternatively, be able to draw useful inferences from inconsistent premises.

Several approaches have been proposed to model knowledge evolution over time. One of the most well-researched formalisms is *belief revision* [9, 10], specifically the classic AGM framework [1, 9]. Substantial efforts have been extended to apply this approach to description logics [13, 14, 19, 20], and the work is ongoing. However, the belief revision framework does not explicitly address knowledge evolution in time. Also, in its original formulation, belief revision postulates are stated in terms of potentially infinite belief sets (although work has been done to address this issue). We believe that belief revision is a mature paradigm that can be valuable source for important insight. However, there is a need for formal approaches that address the practical challenges more directly.

New research direction under the tentative title "stream reasoning" [6] emerged within the Semantic Web community. It explicitly deals with reasoning over rapidly changing and time-dependent data in a way that can deliver answers to the user while they are still relevant. Stream reasoning is defined as "*the new multi-disciplinary approach which will provide the abstractions, foundations, methods, and tools required to integrate data streams and reasoning systems*" [7]. Della Valle et al. [5] write: "*Stream-reasoning research definitely needs new theoretical investigations that go beyond data-stream management systems, event-based systems, and complex event processing. Similarly, we must go beyond current existing theoretical frameworks such as belief revision and temporal logic*". Currently, there is no consensus on a logic formalism appropriate for stream reasoning. There is an obvious practical need for such formalism to be able to integrate current, description logic-based Semantic Web standards.

Dynamic Reasoning Systems (DRS) [17] provide a formal framework for modeling the reasoning process of an artificial agent that "explicitly portrays reasoning as an activity that takes place in *time*". It sidesteps the logical omniscience assumption of the classical AGM framework and has means of working with inconsistent knowledge bases by keeping track of a proposition's *derivation path*. The DRS framework has been shown to support non-monotonic reasoning in a natural way.

A DRS can be defined for any language. DLs present a challenge in that they do not have explicit derivation rules. Instead, DLs rely on *inference algorithms* to accomplish common reasoning tasks. One of the basic tasks is checking *subsumption* of concepts.

The goal of this paper is to present the DRS framework as a suitable formalism for Semantic Web reasoning. To this end, we give an instance of DRS capable of building a concept subsumption hierarchy for a well-known description logic.

We believe it to be an important foundation for research on belief dynamics for Semantic Web agents. Section 2 of this paper contains a brief formal introduction to Description Logics and the necessary definitions. Section 3 discusses Dynamic Reasoning Systems. Section 4 describes a DRS and agent reasoning process for deriving explicit subsumption hierarchies from description logic $\mathcal{ALCN}$ terminology. Short abstract of this work appears in [20]. Finally, in Section 5, we draw some conclusions and discuss directions for future research.

2 Description Logics

Languages for any description logic contain *concept names, role names,* and *individual names*. Below, we will use uppercase A and B for concept names, uppercase letters R, P for role names, and lowercase x, y, z for individual names.

DL languages combine role and concept names into *concept definitions*. Concepts of a description logic $\mathcal{AL}$ [16] are defined as follows:

$C, D \rightarrow$	A	|	(atomic concept)
	$\top$	|	(universal concept)
	$\bot$	|	(bottom concept)
	$\neg A$	|	(atomic negation)
	$C \cap D$	|	(intersection)
	$\forall R.C$	|	(value restriction)
	$\exists R.\top$		(limited existential quantification)

More expressive DLs extend $\mathcal{AL}$ by the following constructs:

Indication	Syntax	Name
$\mathcal{U}$	$C \cup D$	union
$\mathcal{E}$	$\exists R.C$	full existential quantification
$\mathcal{N}$	$\leqslant n R, \geqslant n$	number restriction
$\mathcal{C}$	$\neg C$	full negation

The commonly used DL $\mathcal{ALCN}$ extends $\mathcal{AL}$ with full negation and number restriction. In the following sections, we will restrict ourselves to $\mathcal{ALCN}$.

An *Interpretation* of a DL is a structure $I = (\Delta^I, .^I)$, where Δ^I is a nonempty set called *domain* and $.^I$ is an *interpretation function* that maps concept names to subsets of a domain, role names to subsets of $\Delta^I \times \Delta^I$, and individual names to elements of Δ^I. The function $.^I$ extends to arbitrary concept definition in a rather intuitive way (for details, see [2], chapter 2). A concept is *unsatisfiable* if for any interpretation I, $C^I = \emptyset$.

Description Logic *knowledge bases* consist of two components: a TBox, a set of statements about concepts, and an ABox, a set of assertions about individuals. In general, a TBox T contains *general concept inclusion axioms* of the form $C \subseteq D$ (inclusion axiom). The pair of axioms $C \subseteq D, D \subseteq C$ is abbreviated $C \equiv D$ (equality axiom). An interpretation I satisfies an axiom $C \subseteq D$ if $C^I \subseteq D^I$. Interpretation I satisfies a TBox T if it satisfies every axiom in T.

A *definition* is an equality axiom with an atomic concept on the left hand side. A TBox is a *terminology* if it consists of definitions and no concept name is defined more than once. A concept name is a *defined name* if it appears on the left hand side of the axiom and a *base name* if it doesn't. A definition is in the *extended form* if only base concept names appear on the right hand side. A terminology is *definitorial* if every definition has exactly one extended form (not counting equivalent syntactic variants). In further discussion, we assume that our TBoxes are definitorial terminologies. Under this condition, we can assume, wlog, that definitions contain no cycles.

An ABox contains assertions regarding individual names. These include *concept assertions* $C(a)$ and *role assertions* $R(a, b)$. An interpretation I satisfies (or is a model of) $C(a)$ if $a^I \in C^I$ and it satisfies $R(a, b)$. if $(a^I, b^I) \in R^I$. Finally, I satisfies an assertion α (or an ABox A) with respect of a TBox T if it is a model of both an assertion (or an ABox) and the TBox.

An ontology of concepts can be expressed using a DL. The term *ontology* is often applied either to a TBox or to a full DL knowledge base. We will occasionally use *ontology* in the former sense.

3 Dynamic Reasoning Systems

The classical (propositional) notion of belief set [e.g., 9] models it as an (often infinite) set of formulas of the underlying logical language. In our view, a belief set should be finite and should represent the agent's knowledge and beliefs at a given point in time. Moreover, each formula in such a belief set should contain information indicating how if was obtained and whether it has been used in subsequent deductions, thereby enabling both backtracking and forward chaining through reasoning paths for so-called "reason maintenance".

To this end, in [17] there was defined the notion of a dynamic reasoning system (DRS), which explicitly portrays reasoning as an activity that takes place in time. This is

obtained from the conventional notion of formal logical system by lending special semantic status to the concept of a derivation path (i.e., a proof). Introduction of new knowledge or beliefs into the path occurs in two ways: either new propositions are added in the form of axioms, or some propositions are derived from earlier ones by means of an inference rule. In either case, the action is regarded as occurring in a discrete time step, and the new proposition is labeled with a time stamp (an integer) indicating the step at which this occurred. Moreover, for propositions entered into the path as a result of rule applications, the label additionally contains a record of which inference rule was used and which propositions were employed as premises.

At any given time, the contents of the path is regarded is being the sum total of the agent's knowledge and beliefs as of that time. Thus we here take this path as being the agent's belief set as of that time.

This is to be contrasted with other systems of belief revision, which assume that the agent additionally knows all the logical consequences of the basic belief set. Such systems are said to exhibit "logical omniscience." For an in-depth analysis of this issue, together with a manner of addressing it, see the paper by Fagin, Halpern, Moses, and Vardi [8].

For complete details of the notion of a DRS, please see [S97]. A brief outline is as follows. A labeled formula is defined as a pair (P, λ) where $P \in L$, where L is a logical language, and the *label* λ is an ordered 4-tuple *(index, from, to, status)*, where:

1. *index* is a non-negative integer, the *index*, representing the formulas position in the belief set.
2. *from* is a *from list*, containing information about how the formula came to be entered into the belief set. Either it was received from an outside source (obtained from some other agent or through interaction with its environment), in which case the *from* list contains the token *rec*, or it was derived from some formulas occurring earlier in the belief set, in which case the from list contains the name of the derivation rule and the indexes of the formulas used as premises in the derivation. The types of formulas that can be received are understood to include both axioms of the propositional calculus and statements about the agents environment (sometimes distinguished as "logical" and "nonlogical" axioms).
3. *to* is a *to list*, containing the indexes of all formulas in the belief set for which the given formula served as a premise in the indexed formula's derivations.
4. *status* is a *status indicator*, taking values on or off, indicating whether the belief represented by the formula is currently held, i.e., whether the formula may or may not be used in any future derivations. Whenever a formula is initially entered into the belief set, its status is on.

For a given agent, let us denote the agent's belief set at time step i by Λ_i. Let $\Lambda_0 = \emptyset$. Thus the agent initially has no knowledge or beliefs. Then, given Λ_i, for $i \geq 0$, Λ_{i+1} can be obtained in any of the following ways:

1. A new formula is received from an outside source,
2. A formula is derived from some formulas in Λ_i by means of an inference rule,
3. A formula in Λ_i has its status changed from *on* to *off*.

Changing a formula's status from on to off occurs during a reason maintenance process that is invoked whenever an insatisfiability, i.e., a definition of the form $A \equiv \perp$ is entered into the agent's belief set. The objective of reason maintenance is to remove this insatisfiability.

This has two phases. First one starts back tracking from the insatisfiability, using the from lists in the formula labels, looking for the "culprit" formulas that occurred earlier and which led to the inconsistency. A decision then must be made to turn the status of at least one of these formulas to "off". Then one forward chains from this formula, using the to lists, to find all formulas whose derivations stemmed from the culprit formula, and likewise turns their status to "off". This will include the inconsistent formula that triggered the reason maintenance process.

Which culprit formula to deactivate can be determined by the various culprits' degrees of belief, to wit, remove the one that is least believed. In case the culprits are all believed equally, one can be chosen at random. Alternatively, an agent can remove the culprit formula that is the least important according to some reasonable criteria. One such criteria is a cumulative belief level of formulas derived from the culprit. This criteria provides a finite version of the AGM epistemic entrenchment relation.

This model of agent-oriented reasoning reflects that view that, at any given time, the agent's beliefs may harbor an inconsistency, but the agent does not become aware of this unless an inconsistent formula is explicitly entered into its belief set.

This, in our opinion, is a realistic model of natural human reasoning. Humans can comfortably maintain inconsistent beliefs for long periods of time without ever realizing this.

But once they become consciously aware of a contradiction, they typically rethink their position and modify their beliefs so that the contradiction is removed.

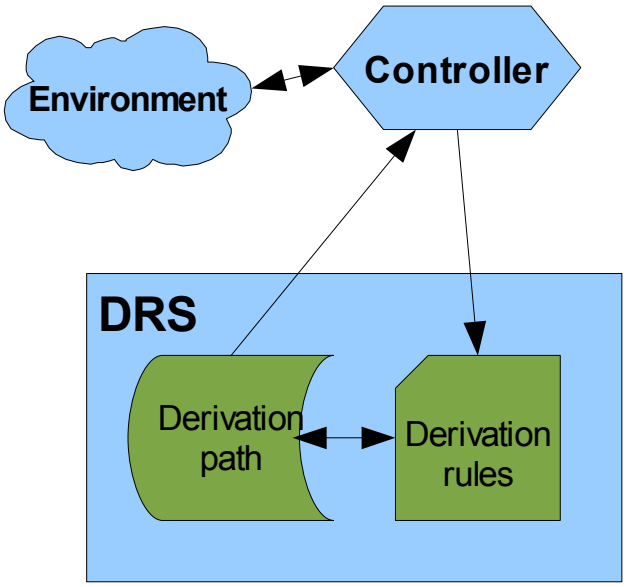

Fig.1 Reasoning agent employing a Dynamic Reasoning System

The reasoning agent (Fig. 1) uses a Dynamic Reasoning System to reach conclusions that help advance the agent's goals. A *controller* directs DRS behavior to steer it to such conclusions. The controller performs the following actions:

1. Receive information from the outside environment. The information can come from a human user, other agents, or be harvested by an agent through sensors. The latter can get information from any external data source.
2. Enter information, as a "nonlogical" axioms expressed in language *L*, into the DRS's inference path.
3. Apply an inference rule.
4. Act to remove insatisfiability, by invoking belief revision procedure described above.

The agent performs these actions in the order dictated by the agent's and environment's current state, presumably in a manner that would advance its goals. In the following, we are constructing an agent that would accept an ontology in the form of TBox definitions and construct a subsumption hierarchy of concept names implicit in this ontology.

4 Dynamic Reasoning for DL $\mathcal{ALCN}$

A Dynamic Reasoning System is a model for knowledge base and reasoning process for artificial agent that assists a user. We describe an agent that extracts ontological knowledge from the Web and uses it to support a user's browsing and querying activities. To this end, an agent maintains two information stores:

1. Local copy of the ontology, expressed as an $\mathcal{ALCN}$ TBox. This ontology consists of $\mathcal{ALCN}$ definitions that occur in the derivation path.
2. A subsumption tree of concept names.

The latter can be used to support both browsing and user querying on both a TBox and an ABox. The user has a preference for satisfiable ontologies, so the agent has to detect and remove unsatisfiable concepts. Thus, our DRS needs to support 2 types of DL reasoning:

1. Check if a defined concept A is satisfiable
2. Deduce atomic subsumption, that is, a statement of the form $A \subseteq B$,

where *A*, *B* are concept names.

To construct the DRS, we first note that if *A* and *B* are defined by axioms $A \equiv C, B \equiv D$, where C, D are concept definition, then $A \subseteq B \; iff \; C \subseteq D.$ Second, note that $C \subseteq D iff \; C \cup \neg D$ is unsatisfiable.
So both our reasoning tasks would require checking satisfiability of concepts. We are using a generic tableau-based satisfiability algorithm [2, 3].
Now we can build our dynamic reasoning system. First, We define the language, L. The symbols of L are the same as the symbols of logic $\mathcal{ALCN}$. We use *A, B* for concept names occurring in the incoming statements and *A′, B′* for the names introduced by the agents. The formulas of *L* are the following:

1. Equivalence statements of the form $A \equiv C$, where A is a concept name and C is concept definition. Without loss of generality, we assume all concept definitions are in negation normal form, i.e. negation only occurs in front of concept names.
2. Atomic subsumption statements of the form $A \subseteq B$, where A, B are concept names. These represent arcs of the subsumption tree the agent is building.
3. TBox assertions $C(a), R(a, b)$, where C is a concept, R is a role, and a,b are individual constants
4. Explicit inequality assertions $x \neq y$, where x, y are individual names.

Then we define inference rules. Implicitly, every rule that modifies a concept definition also puts the result into negation normal form. The inference rules will be:

1. *Substitution:* from $A \equiv C$ and $B \equiv D$ infer $A \equiv E$, where E is C with all occurrences of B replaced by D . For this treatment we assume that our TBox does not contain cycles in definitions. By repeatedly applying this rule, we obtain an *extension* of definition for A that only contain *ground* concept names on the right side.
2. *Subsumption test introduction:* from $A \equiv C, B \equiv D$ infer $A' \equiv C \cap \neg D$, where

A' is a previously-unused agent-generated concept name.

3. From $A \equiv C$, $B \equiv D$ and $A' \equiv \bot$, provided that name A' was introduced using rule 2 with $A \equiv C$, $B \equiv D$ as premises, derive $A \subseteq B$.

The following rules 4-10 are added to enable tableau-based consistency checks. These are derived from the transformation rules listed in [2], p. 81. Individual names $x, y, z, \ldots$ are unique names generated by the agent. All TBox statements are derived from the same ABox statement (that is undergoing satisfiability check) $A \equiv C_0$:

4. From $A \equiv C_0$, infer $C_o(x_0)$, if no ABox statements were inferred from $A \equiv C_0$.
5. From $A \equiv C_0$ and $(C_1 \cap C_2)(x)$, infer $C_1(x)$ and $C_2(x)$, if any one of them is not yet inferred.
6. From $A \equiv C_0$ and $(C_1 \cup C_2)(x)$, infer $C_1(x)$ or $C_2(x)$, if neither of them is inferred yet.
7. From $A \equiv C_0$ and $(\exists R.C)(x)$, infer $C(y)$ and $R(x, y)$, where y is a new generated name, if no z exists such that $C(z)$ and $R(x, z)$ are already derived.
8. From $A \equiv C_0$, $(\forall R.C)(x)$ and $R(x, y)$ infer $C(y)$, if not already derived.
9. From $A \equiv C_0$ and $(\geqslant n\, R)(x)$, infer $R(x, y_1), \ldots, R(x, y_n)$ and ($y_i \neq y_j$, and $R(x, y)$, unless $R(x, z_1), \ldots, R(x, z_n)$ are already inferred.
10. From $A \equiv C_0$ and $(\leqslant n\, R)(x)$, if $R(x, y_1), \ldots, R(x, y_{n+1})$ are in the derivation path and $y_i \neq y_j$ is not in the path for some $i \neq j$: replace all occurrences of y_i with y_j.

The following rules 11-13 detect inconsistency in TBoxes built using rules 4-10. As above, TBox statements are derived from $A \equiv \bot$:

11. From $A \equiv C_0$ and $\bot(x)$, derive $A \equiv \bot$, where x is any individual name.
12. From $A \equiv C_0$, $A_1(x)$ and $\neg A_1(x)$, derive $A \equiv \bot$, where x is any individual name and A_1 is any concept name.
13. From $A \equiv C_0$, $(\leqslant n\, R)(x)$, set $\{R(x, y_i) \vee 1 \leqslant i \leqslant n+1\}$ and set $\{y_i \neq y_j \vee 1 \leqslant i \leqslant j \leqslant n+1\}$ derive $A \equiv \bot$, where $x, y_{1,} \ldots, y_n + 1$ are individual names, R is a role name and $n > 0$.

Finally, rule 14 derives a subsumption axiom, using reduction to unsatisfiability:

14. From $A \equiv C$, $B \equiv D$, $A_1 \equiv C \cup \neg D$ and $A_1 \equiv \bot$, derive $A \subseteq B$

A Dynamic Reasoning System based on language *L* and rules 1-14 is capable of supporting an agent that builds an explicit subsumption hierarchy. We will now describe a controller that can achieve this goal.

An agent starts with an empty derivation path and empty subsumption hierarchy. It will receive TBox definitions from the user. To start the hierarchy, before receiving the first axiom, the controller will enter a root concept, $R \equiv \top$, as a first formula in the derivation path and R as a root node in the hierarchy.

Upon entering a new axiom of the form $A \equiv C$, the controller will perform the following actions:

1. Derive an expanded definition of A by repeatedly employing Rule 1 until the right side of the resulting definition contains no defined concept names.
2. Test satisfiability of A using Rules 4-13. If it is unsatisfiable, flag it for a belief revision procedure
3. Expand all (extended) definitions that depend on using Rule 1. Test the affected concepts' satisfiability, flagging for a belief revision process if unsatisfiable. Update the hierarchy of concepts affected by this step, testing subsumption by using Rules 2-14.
4. Place A into its appropriate place in the subsumption hierarchy, using Rules 2-14 to test subsumption with definitions of concept names already there.

To test satisfiability by employing Rules 3-13, an agent follows a tableau-based algorithm. Details of the appropriate algorithm, with discussion of termination and complexity, can be found in [2].

Rules 6 and 10 are *non-deterministic:* for a given ABox, they can be applied in finitely many different ways, leading to finitely many ABox'es. The concept is satisfiable if at least one such ABox is consistent. Each ABox is a branch in the satisfiablilty algorithm. The controller may handle branches by setting the belief status of statements on inactive branches to *off.* In practice, it may be useful to remove such statements from the path to save space.

We did not specify the details of modifying subsumption hierarchy on steps 3 and 4. In principle, the controller may simply search the existing hierarchy starting at the root, testing the concept in question's subsumption with each node. This is a natural and decidable procedure that will result in the correct hierarchy. Studying the complexity of such an

algorithm and exploring possible optimizations is a task left for future research.

In case an unsatisfiable concept is detected, an agent will generate and display to the user the list of definitions that lead to it. The user will have a choice to delete and modify one of them. Methods for assisting the user or for achieving this task without user interaction can be developed, based on research in ontology debugging and belief revision for description logics [12, 13, 14, 21]. Developing such algorithms is another task left for future research.

5 Conclusions and Further Research

In the present work, we argued for the suitability of Dynamic Reasoning Systems [17, 18] as formalism for agent reasoning on the Semantic Web. To this end, we presented a limited but realistic example of a DRS for performing a common reasoning task on a Description Logic ontology. We sketched a procedure for user-directed ontology debugging. This same mechanism can be used for automated belief revision.

Research in reasoning dynamics for the Semantic Web is a major part of the overall Semantic Web effort. The problem has been approached from belief revision [14], ontology debugging [12], and now stream reasoning [6] perspectives. We believe the present approach has the potential to contribute to all these efforts.

There are several directions for future research. First, the agent presented needs to be described in greater detail. Procedures need to be fleshed out, and potential performance problems need to be identified and addressed. Complexity issues need to be discussed. There is also a possibility to use data stored in the derivation path to speed up new reasoning. For example, incremental algorithms can be designed to utilize and extend existing derivation paths when a concept gets updated through incorporating new definitions.

The agent can also be extended to support more varied reasoning. It can be modified to accept more kinds of input, including, e. g. , general inclusion axioms and TBox assertions. A facility to deal with user queries on an ABox needs to be added. The agent can be used as a model to build DRSs capable of dealing with Semantic Web standards and more realistic scenarios (reasoning in the presence of loops and redefinitions of concepts). On the other hand, less expressive DLs can be investigated, in hope that they may guarantee moderate computational complexity.

Finally, the DRS formalism can be used to investigate belief revision techniques. Variants on the AGM framework's rationality postulates can be constructed for a finite DRS case, both in general and specifically for description logics. Feasible algorithms adhering to these principles need to be constructed. Finally, these postulates and algorithms can be applied to interesting practical cases, such as reasoning with multi sourced information that takes into account different degrees of the agent's belief and trust between agents.

References

1. Alchourrŏn, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions, *Journal of Symbolic Logic*, **50**, 2 (1985)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider. P. (eds.): The Description Logic Handbook, Cambridge University Press (2003)
3. Baader, F., Sattler, S.: Expressive number restrictions in Description Logics. J. of Logic and Computation, 9(3):319–350 (1999)
4. Bernes-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
5. Della Valle, E., Ceri, S., van Hamelern, F., Fensel, D.: It's a streaming world! Reasoning upon rapidly changing information. Intelligent Systems vol.24 no.6, pp. 83-89 (2009a)
6. Della Valle, E., Ceri, S., Fensel, D, van Hamelern, F., Studer, R.: (Eds.): Proc. 1st International Workshop on Stream Reasoning , CEUR vol. 466 (2009)
7. Della Valle, E., Ceri, S., Braga, D., Celino, I., Frensel, D., van Harmelen, F., . Unel, G., Research chapters in stream reasoning, In: Proc. 1st International Workshop on Stream Reasoning , CEUR vol. 466 (2009)
8. Fagin, R., Halpern, J., Moses, Y., Vardi, M. Reasoning about knowledge, MIT Press, Cambridge, MA, 1995
9. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States, MIT Press/Bradford Books, Cambridge, MA (1988).
10. Gärdenfors, P., ed.: Belief Revision, Cambridge University Press, NewYork (1992)
11. Hamilton, A.G., Logic for Mathematicians, Revised Edition , Cambridge University Press (1988).
12. Parsia, B., Sirin, E., Kalyanpur, A.:Debugging OWL ontologies, in Proc. 14th International World Wide Web Conference (WWW'05), Chiba, Japan, May 10-14, 2005. ACM Press 2005
13. Ribeiro, M. M., Wassermann, R.: First Steps Towards Revising Ontologies, In: Proc. 2nd Workshop on Ontologies and their Applications, CEUR Workshop Proc. Vol. 166 (2006)
14. Ribeiro, M. M., Wassermann, R.: Base Revision in Description Logics - Preliminary Results, In: International Workshop on Ontology Dynamics (IWOD) (2007)
15. Smolka, G: A feature logic with subsorts. Technical Report 33, IWBS, IBM Deutschland, P.O. Box 80 08 80 D-7000 Stuttgart 80, Germany (1988)
16. Schmidt-Schauß, M., Smolka. G.: Attributive concept descriptions with complements. Artificial Intelligence, 48(1):1–26, (1991)
17. Schwartz, D. G.: Dynamic reasoning with qualified syllogisms, Artificial Intelligence, 93(1-2) (1997) 103-167.
18. Schwartz, D.G.: Formal Specifications for a Document Management Assistant, In: Proc. International Conference on Systems, Computing Sciences and Software Engineering (2009)
19. Qi, G., Liu, W., Bell, D. A.: Knowledge Base Revision in Description Logics, In Proc. European Conference on Logics in Artificial Intelligence (Jelia'06), Springer Verlag (2006)
20. Ustymenko, S., Schwartz, D.G., Dynamic Reasoning for Description Logic Terminologies. Canadian Conference on AI 2010: 340-343
21. Zhuang, Z. Q., and Pagnucco, M.: *B*elief Contraction in the Description Logic EL, In: 22nd Int. Workshop on Description Logics (DL 2009), CEUR Workshop Proc. Vol. 477 (2009)