A Case Study on the Use of Semantic Web Technologies for Learner Guidance

Andrea Zielinski
Fraunhofer IOSB
Fraunhoferstr. 1
76131 Karlsruhe, DE
andrea.zielinski@iosb.fraunhofer.de

Jürgen Bock
FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, DE
bock@fzi.de

ABSTRACT

Personalized learning pathways have been advocated by didactic experts to overcome the problem of disorientation and information overload in technology enhanced learning (TEL). They are not only relevant for providing user-adaptive navigational support, but can also be used for composing learning objects into new personalized courses (sequencing and assembly). In this paper we investigate, how Semantic Web technologies can effectively support these tasks, based on a proper representation of learning objects and courses according to didactic requirements.

We claim that both eLearning tasks, adaptive navigation and course assembly, call for a representational model that can capture the syntax and semantics of learning pathways adequately. In particular: (1) a new type of navigation that takes into account ordering information and the hierarchical structure of an eLearning course complemented with adaptive constraints; (2) closely tied to it, a semantic layer to guarantee interoperability and validation of the correctness of the learning pathway descriptions.

We investigate to what extend Semantic Web Languages like RDF/S and OWL are expressive enough to handle different aspects of learning pathways. While both share a structural similarity with DAGs, only OWL ontologies - formally underpinned by description logics (DLs) - are expressive enough to validate the correctness of the data and infer semantically related learning resources on the pathway.

For tasks that are more related to the syntax of learning pathways, in particular navigation similar to a guided tour, we test the time efficiency on various synthetic OWL ontologies using the HermiT reasoner. Experimental results show that the course structure and the density of the knowledge graph impact on the performance. We claim that in a dynamically changing environment, where the computation of reachability of a vertex is computed on demand at run-time, OWL-based reasoning does not scale up well. Using a real-world case study from the eLearning domain, we compare an OWL 2 DL implementation with an equivalent graph algorithm implementation with respect to time efficiency.

Categories and Subject Descriptors

K.3 [**Computers and Education**]. I.2.4 [**Knowledge Representation Formalisms and Methods**] *Representation languages*, *Information systems education*

WWW 2015 Companion, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3473-0/15/05.
http://dx.doi.org/10.1145/2740908.2743047

Keywords

Adaptive Learning Pathways, user guidance, adaptive navigation, Semantic Web, guided tour.

1. INTRODUCTION

Adaptive navigation support in the Semantic Web is seen as a basic requirement for search engine users, because it can help in terms of efficiency [1] and user satisfaction, where studies have shown that users exploring the web tend to find more diverse and novel items [2].

In E-learning, adaptive navigation has been advocated specifically to overcome the problem of disorientation and information overload. Based on the metadata description of the learning resources, domain concepts, and a user model that reflects the user's actual learning progress, adaptation techniques to reason over Semantic Web data have been suggested (see, e.g., [3,4,15]).

These systems, however, do not support navigation based on a more elaborate sequencing model that has been suggested for the description of learning pathways according to e-Learning requirement specifications [5], based on modular composition, nested composition, optional parts, and sequencing.

First, we observe that ontology-based approaches to navigation exclusively explore binary semantic relationships between individual instances. However, more complex relations for defining sequences are not easily encoded in RDF/S or OWL, including compositionality aspects to account for the structure of a course.

Second, for courseware generation, finding reachable learning objects that lay on the learner's path relative to his/her current learning position, is an important subtask for more elaborate tasks like inferring successors/predecessors that are similar to a given reference learning path. Even though this task mainly involves structural aspects of learning pathways and can be handled efficiently in isolation, a uniform approach for both syntactic and semantic aspects is required.

The Semantic Web has promising application potentials for TEL. It is useful to combine pedagogical, course and domain knowledge, which can be highly heterogeneous, into a coherent and interoperable framework. To support adaptive navigation and courseware assembly, the following basic requirements need to be satisfied:

i. An expressive formalism that is capable of modelling personalized learning pathways

ii. An efficient algorithm to guide the learner and sequence the course material based on this formalism.

We offer a reasoning-based approach that can handle all these requirements. It builds on DL and is thus expressed in a formalism with a well-defined semantics.
We present an implementation of learning pathways in OWL 2 DL where expressive constraints on nodes and edges as well as sequential and hierarchical relationships between nodes can all be represented in a uniform and declarative way.
In a dynamically changing environment where the computation has to be carried out efficiently at run-time and cannot be pre-determined, the performance of OWL 2 DL based reasoning becomes significant. Reasoning is known to be cost expensive with respect to runtime performance due to its theoretical time complexity of up to N2EXPTIME in the worst case.

The main contributions of the paper are as follows:

(1) First: we define a representation in OWL 2 DL that supports semantically-driven navigation and courseware construction. Class expressions are set up to perform tasks such as reachability within a course unit supplemented with additional semantic filters to constrain the navigation. Due to its high expressivity, as opposed to DAGs and RDF/S, adaptivity can be achieved without duplication of nodes, keeping the size of the learning path network small.

(2) Its limitation is, of course, the scalability: the amount of data involved could be high, making the navigation costly. Therefore we conduct experiments on the runtime performance. In a quantitative evaluation, we test the scalability for the HermiT reasoner for a specific query/ontology combination, using a graph generator to build various synthetic ontologies varying over the course structure and the density of the knowledge graph.

The remainder of the paper is organized as follows. We begin with a presentation of related work on navigation in knowledge graphs (Section 2). Then, we define the basic properties and the structural descriptions of learning pathways on a formal level, using the graph metaphor. We show the differences in expressive power for various formalisms (DAG, RDF/S, OWL) based on the notions sequencing, compositionality, adaptivity, reachability, validation and inferencing on learning pathways (Section 3). Considering an eLearning scenario in Web Didactics [6], we briefly present our OWL 2 DL modelling for navigation in the space of learning objects (Section 4). The scalability is measured on a case study for a sophisticated navigation task using HermiT. This is compared with an alternative graph algorithm in Java of the same navigational sequences with respect to run time efficiency (Section 5). The paper concludes with a final discussion (Section 6).

2. RELATED WORK

Pioneering work by Bruslikovsky et al. [7] in the field of adaptive hypermedia systems offer an account to navigation in terms of a guided tour, i.e. a composite of a sequence of navigational objects for which an order is provided, allowing to visit previous, next and/or back items, including a structured access. However, while the document structure and HTML can provide such an access based on a model of hierarchy and sequence, this is not directly implemented in RDF [8].

The general need for a navigation language for the Web of Data has been recognized by various authors (see, e.g. [9]). Some expressive declarative specification languages that exploit regular expressions on RDF predicates, featuring semantic tests to orient the navigation, have been implemented. While SPARQL 1.1 already allows to define patterns to search for paths of arbitrary length via property paths [10], an advanced query optimization and runtime processing technique specifically designed for reachability queries based on an index integrated into a state-of-the art RDF processor has been proposed by Gubivech et al. [18]. They show that fast graph reachability queries on huge RDF data with 100 million facts is feasible within milliseconds.

In our TEL scenario, the choice of the more expressive Semantic Web language OWL has been motivated by the ability to represent basic knowledge about learning pathways in an intuitive and adequate way from the tutor's perspective. Therefore, representing hierarchically structured sequences and semantic constraints in OWL is required.

A number of best practice pattern have been offered for specific tasks involving sequences. Hirsh and Kudenko [12] describe a DL pattern based on suffix trees and rewriting. Drummond et al. [13] use a linked list pattern in OWL. Finally, Hayes uses n-ary relations to define sequences [14]. All of these accounts were implemented with a focus on pattern matching, and do not support inference of successors/predecessors on structured sequences.

3. Formal Description

In this section, we formally describe the notion of 'navigation along learning pathways' by means of a graph representation of the knowledge space.

We use *KO* to denote a learning object (small, self-contained, reusable unit of learning) that generally forms part of a course or Concept Container, denoted by *CC*. For simplicity, we assume that learning pathways are specified as macro-level learning pathways (*Macro LPs*) on the level of *CCs*, and micro-level learning pathways (*Micro LPs*) on the level of *KO*s.

We follow the formal characteristics of a proper learning pathway specification as defined by Janssen et al. [5] based on the notions of modular composition, nested composition, optional parts, and sequencing. We show the differences in expressive power for various formalisms such as DAGs, RDF/S and OWL.

3.1 From DAGs to RDF to OWL

DAGs

The correspondence between learning pathways and Directed Acyclic Graphs (DAGs) is straightforward: A learning path can be defined syntactically as a sequence of consecutive edges in the DAG, i.e. a directed graph with no directed cycle. The length of the path is the number of edges traversed. There are selected start and end nodes for each path. A directed graph $G = (V, E)$ is an ordered pair of disjoint sets (V, E), where $E \subseteq V \times V$. Set V is called the vertex or node set, while set E is the edge set of graph G. It is assumed that self-loops (i.e. edges of the form (u, u), for some $u \in V$) are not contained in the graph.

A generic task on DAGs is to calculate the Reachability Set of the current node v, i.e. the set of vertices for which G contains a path (of at least one edge) from v.

Deciding whether a vertex in a graph is reachable from another vertex has been studied intensively in complexity theory and is well understood. For DAGs, this can be accomplished in linear time with respect to the number of edges |E| in the graph using algorithms such as Depth-First or Breadth-First traversal. The direct computation of the transitive closure of G may take

O(|V||E|) of query time in the worst-case. Depending on the setting, the reachability of a vertex in a DAG is either computed on demand or pre-computed and stored in a separate data structure. In a dynamically changing environment with only a few queries, the first option is generally preferred. An alternative would be to compute and store the transitive closure of the graph. It answers reachability queries in constant time but needs $O(n^2)$ space to store the transitive closure of an n-vertex graph [16].

Labeled DAGs

Unlabeled DAGs can only express one single relationship between any two vertices and generally fall short to define optional parts, and thus non-deterministic choices.

An example of a labeled DAG (see Fig. 1) illustrates the difference: For each sequence, i.e. either alternative pathways of the same type or different types of pathway (e.g. chronologically forward/backward), different labels have to be set up, as indicated by the different types of arrows. The three LPs cannot be defined in a single DAG (*KO13* and *KO14* would introduce a loop).

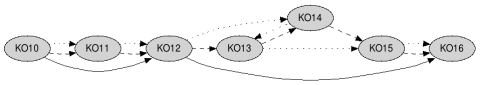

Fig. 1. Capturing different learning pathways

Furthermore, when traversing the graph from the current learner position, individual nodes might need to be semantically checked to provide user-adaptation. (Labeled) DAGs can be used not only to express syntactic but also semantic information, i.e. by means of feature structures. Typically, feature checks on atomic symbols or sets of label/value pairs are carried out to guide the navigation: Comparing the learning resources to the current learner state (e.g., *learning speed, etc.*), the best matching path can be chosen. As indicated in the graph below (see Fig. 2), feature checks are used to trigger the selection of KOs. The blue vertex represents the set of KOs that fulfil all feature constraints.

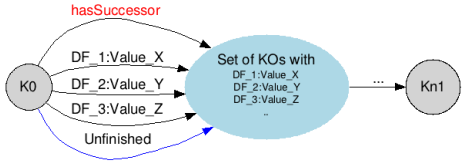

Fig. 2. Adaptive Navigation based on Feature Constraints

In practice, incorporation of feature constraints is also useful to minimize the graph, because the learning path description can be left partially unspecified. It is possible to abstract away from some dynamic features that are only known at run-time, minimizing the effort of specifying learning pathways.[1]

A major shortcoming of DAGs is, however, that they do not exhibit any hierarchical structure, as linking to subgraphs cannot be achieved. Capturing the predefined paths can only be achieved by spelling out the learning pathways at the terminal level which causes a combinatorial explosion of nodes.

RDF

The Resource Description Framework (RDF) is a metadata model introduced by the W3C for representing information on resources in the Semantic Web. It represents data as sets of triples that can also be conceived of as a directed labeled graph. Since links are semantically labelled, they can also be used to control the navigation. While the RDF model is basically a data model reflecting the underlying graph structure, an extra semantic layer can be built above RDF using RDF Schema, i.e. a primitive ontology language which offers a restricted set of modeling primitives (e.g. class, subclass, property, etc.). Since learning resources are denoted by Internationalized Resource Identifiers (IRIs), and thus can be unambiguously identified in different contexts, i.e., within different domains, etc., they enable interoperability in the Semantic Web.

Property Graphs

While the RDF data model intrinsically only supports binary relations, at first glance, it seems not expressive enough to accommodate for the hierarchical and modular structure of learning pathways. A formalism is needed that can directly assert a property to a specific edge to represent the fact that the *hasSuccessor* relation between two *KOs* is valid only in a certain course unit (*CC*). In property graphs, which can be represented in RDF by the use of quads (e.g. an extension of a triple, supported by the W3C RDF1.1 Recommendation [17]), key/values can be associated with both vertices and edges. Figure 3 gives an example of a property graph that allows for local restrictions on the edges, i.e. the whole triple.

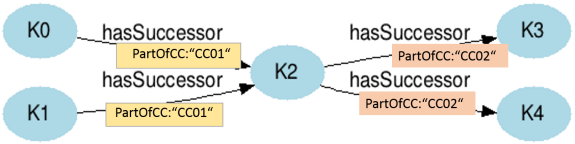

Fig. 3. Property Graph

SPARQL 1.1, primarily designed for pattern (subgraph) matching, also supports path traversal in a property graph and is capable of matching paths of arbitrary length [10].

OWL 2 DL

The expressive power of learning pathways represented in RDF can be further augmented with OWL semantics. OWL 2 DL ontologies are grounded in Description Logics, a model-theoretic semantics. Restrictions on property values and cardinalities can be used by the reasoner to infer new knowledge, such as equality of individuals and class membership. With OWL axioms, the relationships between *KOs*, *CCs* can be formalized (e.g. disjointness) and it can be stated that, e.g., every *hasSuccessor* relation is connected by *KOs*, so that the consistency of the data (e.g. existing start and end nodes, no cycles, no *KOs* appear more than once on the path) can be checked automatically. Furthermore, the network of learning resources can be further enriched, using e.g. *owl:sameAs*. Learning pathways can be inferred that are similar to a reference sequence, i.e. all explicitly given relationships in the metadata description of sharable open education resources on the Semantic Web could also be exploited.

[1] In principle, different LPs can already be pre-computed to accommodate for the learners' static features (e.g., language, age, etc.), thereby multiplying the number of learning pathways. Generally, there is a tradeoff between storage (pre-computing personalized pathways) and processing time: adding feature constraints will keep the size of the network small but impact on the performance at run-time.

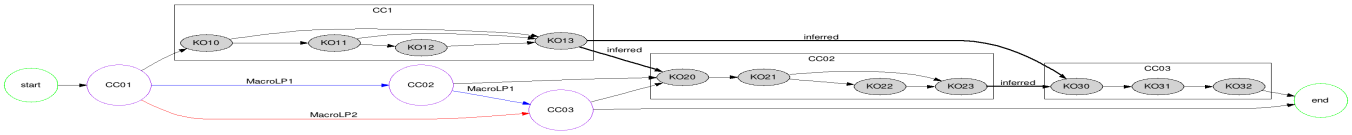

Figure 1 Fig. 4. Macro- and Micro-level Learning Pathways

Figure 4 gives an illustration of the type of learning path network that we seek to model in our approach. It supports the description of learning pathways specified as fully connected sequences (from a start to an end node) on two hierarchical levels (i.e., CC and KO), namely *Macro*- and *Micro* learning pathways, used to guide the learner towards his/her learning goal.

Our OWL 2 DL Approach

We offer a new modeling approach for navigational sequences which extends previous work in the context of the Pedagogical Ontology (PO)[20]. In contrast to related work, learning pathways are pre-defined by experts as didactically-meaningful pathways, featuring different learning and teaching processes. These might intrinsically entail (binary) constraints related to the pre-requisite relationship among concepts described in the learning objects.

Our OWL 2 DL modelling (see also [19]) and reasoner supports

- retrieving direct and indirect successors and predecessors within a Concept Container w.r.t. to a certain learner state, including switching to the next level at the end or beginning of a Concept Container
- incorporation of semantic adaptivity constraints on successor/predecessor nodes
- inferring pathways based on semantic attributes, so called Knowledge Type or Media Type Pathways, for automatic courseware generation

In our approach, learning pathways are represented as ontology classes, i.e. generic ontological entities, which are to be extended in the ontology framework when defining specific pathways.

This modeling approach has been chosen instead of representing learning pathways as object properties connecting any two *KOs* (*CCs*, resp.), which are in a direct successor relationship w.r.t. a particular learning pathway to increase flexibility in terms of allowing tutors to individually specify learning pathways without having to alter the (generic) Pedagogical Ontology or modifying the ontology vocabulary (schema) by creating new object properties. Since OWL does not support the representation of n-ary relations, it is not possible to specify generic learning pathway object properties that can be reused for different pathways. Moreover, it is not possible to attach any additional context to a specific pathway relation between any *KOs* (*CCs*, resp.), e.g., to specify that a particular *Micro*-level learning pathway relation between two *KOs* only holds in a particular *CC*. As a consequence, every learning pathway would require its own object property for denoting successor relations. Formally, to connect two KOs via a *Micro* learning pathway, or two CCs via a *Macro* learning pathway, using a connector individual is required. We also offer a modelling of Knowledge Type and Media Type Pathways. Such pathways provide an ordering w.r.t. the Knowledge Type (example, assessment test, etc.) or Media Type (e.g., audio, video, text) associated with the resources. The actual pathways on the knowledge type level are specified in the same way as the *Micro*- and *Macro-level* pathways, i.e. Knowledge Types KT_i and KT_j are connected via a learning pathway with a connector individual. Provided, the Knowledge Type is specified for each *KO* via the object property *hasKnowledgeType* the predecessor and successor relations on the *KO* level can be inferred via a property chain axiom. Property chain rules are also adopted to refer to the transitivity of the *hasSuccessor* relation.

Altogether, four learning pathway related sets are defined by OWL class expressions based on the successor/predecessor relations and current/previous KOs.

4. A case study

For a quantitative evaluation of our sequential navigation scenario in OWL 2 DL, we conduct various tests using generated input ontologies, which simulate an instantiation of the Pedagogical Ontology. We want to access the scalability of a state of the art OWL 2 DL reasoner by generating large A-Boxes systematically, addressing increasing complexity with respect to various characteristics that might involve the reasoning performance. While the T-Box is fixed, the A-Box is populated with varying numbers of individuals and properties over them.

We conducted 5 test series, where in each series all ontology characteristics are kept constant with only one characteristic changing. Each test run was executed two times subsequently in order to observe caching effects. All tests were executed on an Apple Macbook Pro running on iOSX 10.10, with a 2.3 GHz Intel i7 Quadcore processor and 16 GB RAM.

Ontology generation.

The ontology generator creates a given number of *CCs* connected randomly via a given number of *Macro* LPs. Moreover, it generates a given number of Knowledge Objects per Concept Container connected randomly via a given number of *Micro* LPs. Both *Macro*- and *Micro-LPs* can contain branches, which is determined by a given branching factor.

Query execution.

In the experiments, runtime was measured for the complete processing of the generated ontology by the reasoner, which involves several invocations for the different instance retrieval tasks to be carried out (direct successors of current KO, all successors of current KO, all predecessors of current KO, direct successors of previous KO). The processing by the QueryBuilder also involves instance retrieval for the various criteria correlated to user-adaptive features. However, the focus in these experiments was on measuring runtime performance for the pathway-related queries, neglecting the cost for intensive feature checking.

Test series.

Experiment 1 – Increasing number of KOs

This test demonstrates how runtime performance correlates with the number of Knowledge Objects in the input ontology. The test ontologies contain 2 Concept Containers, 1 *Micro*-level LP, and 1 *Macro*-level LP. The test series is executed 2 times (4 times in

total for measuring cache influence), using a branching factor of 1 (i.e. no branching) and 10, respectively.

As Figure 2 shows, runtime increases with the total number of *KOs*.

Moreover it can be observed that a branching factor of 10 roughly doubles the runtime compared to no branching. Caching seems to have a significant impact on the runtime performance. Regarding the cache influence, we could observe an improvement of factor 3.11 on average with no branching, and factor 2.72 with heavy branching, where caching has more effect on smaller ontologies than on larger ones. The experiment shows that with no branches in the learning pathways, a total number of 400 *KOs* can be processed in about 5 seconds, where with a branching factor 10, the same number of *KOs* take about 11 seconds of execution time (with warm caches).

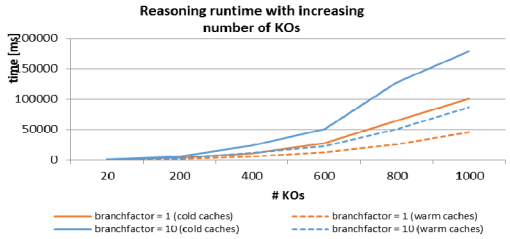

Figure 2. Increasing number of KOs. 2 CCs, 1 MLP, 1mLP

Experiment 2 – Increasing number of Concept Containers

This test demonstrates how runtime performance correlates with the number of Concept Containers in the input ontology. The test ontologies contain a constant number of Knowledge Objects (2000) which are equally distributed on the Concept Containers. The experiment was conducted with 1 *Micro*-level LP, 1 *Macro*-level LP, and no branching.

Figure 3 shows that runtime performance improves with an increasing number of *CCs*. This effect, however is most likely correlated with the decreasing number of *KOs* per *CC*. Again, it can be observed that caching has a significant impact on the performance. In this experiment, we could observe that runtime is on average about 15 times faster when caching effects can be exploited. The caching effect becomes more drastic with a larger number of *CCs*. The total number of 2000 *KOs* could be processed in about 2.5 seconds when they are distributed on 100 *CCs* (with warm caches).

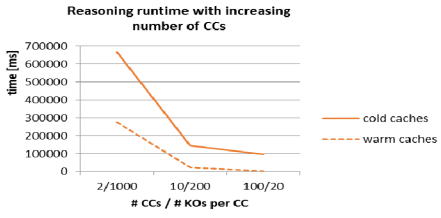

Figure 3. Increasing number of CCs, constant but equally distributed number of KOs, 1MLP, 1mLP

Experiment 3 – Increasing branching factor

This test demonstrates how runtime performance correlates with the branching of learning pathways. The test ontologies represent a constant setting of 10 *CCs*, each containing 50 *KOs*, i.e. 500 *KOs* in total. Moreover, a single *Micro*-level and a single *Macro*-level Learning Pathway are modelled. As can be observed from Figure 4, increased branching in learning pathways has negligible effects on the runtime performance when the branching factor is below 10. Again, caching has a positive effect on the runtime performance with an improvement of factor 4.68 on average.

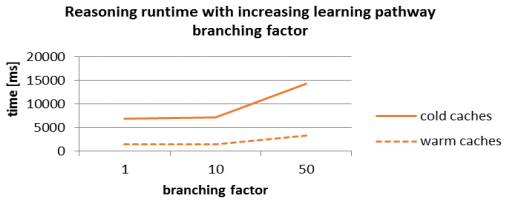

Figure 4. Increasing branching factor, 500 KOs distributed on 10 CCs, 1 mLP, 1MLP

Experiment 4 – Increasing number of Micro-LPs

This test demonstrates how runtime performance correlates with the number of Micro-level LPs. The test ontologies contain a constant number of 500 Knowledge Objects distributed on 10 Concept Containers. The experiment was conducted with 1 *Macro*-level LP and a branching factor of 2.

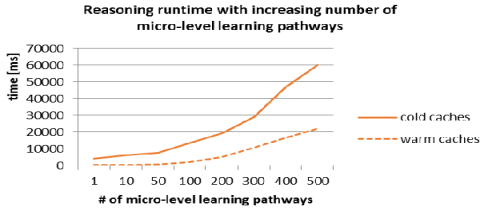

Figure 5. Increasing number of mLPs, 500 KOs distributed on 10 CCs, 1MLP, branching factor of 2

Figure 5 demonstrates that an increasing number of *Micro*-level LPs has an impact on the runtime performance. Caching has a positive effect on the runtime performance and improves it by factor 8.44 (on average), with decreasing influence with increasing number of *Micro* LPs.

Experiment 5 – Increasing number of Macro-LPs

This test demonstrates how runtime performance correlates with the number of *Macro* LPs. The test ontologies contain a constant number of 500 Knowledge Objects distributed on 50 Concept Containers. The experiment was conducted with 1 *Micro* LP and a branching factor of 2.

As Figure 6 illustrates, an increasing number of *Macro* LPs has a negative influence on the runtime performance. This experiment shows a drastic effect of caching, in particular with a larger number of *Macro* LPs. The improvement factor of warm caches is 11.9 on average, slightly decreasing with increasing number of MLPs. In absolute numbers, runtime performance remains below 1.5 seconds for up to 200 *Macro* LPs with warm caches.

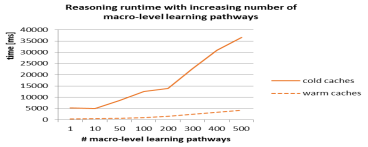

Figure 6. Increasing number of MLPs, 500 KOs distributed on 50CCs, 1 mLP, branching factor of 2

Reachability Query using a Graph Algorithm

We compare the performance for the OWL/HermiT implementation and a graph algorithm in Java[2]. Random DAGs were generated, varying on the size of the vertices and the branching factor (defined as BF= |E | / |V|). Experiments were repeated with different seeds. For a real curriculum of 125 hours from the domain of Philosophy of Didactics which has been modeled in this framework and consists of 103 *CCs*, 1,133 *KOs*, 4 *Macro*-level LPs, 2 *Micro*-level LPs, the response time with OWL/Hermit was approx. 2.5 sec with caching, as opposed to 32 msec with the Java-based graph algorithm which clearly demonstrates the efficiency of the graph-analytic algorithm in reachability queries.

Yet, the query processing cost has been measured, neglecting the cost of feature checking. Since graph algorithms have not been optimized for datatype queries, they are generally inferior in this respect. While numerous studies on the performance of OWL2 DL reasoners for datatype queries exist (see, e.g. [21]), we only considered the reachability task in this section.

5. Conclusion

We have presented a novel approach for modelling complex navigational sequences in OWL 2 DL for user guidance. The approach is applicable in the Semantic Web context, where distributed resources often are already annotated according to metadata standards and various open-source domain and user ontologies exist. Using the reasoning framework, they can be sequenced in a meaningful and user-adaptive way.

In terms of expressiveness, we show that OWL 2 DL is rich enough for interoperability, linking learning resources, as well as validating learning pathways. In particular, abstract learning paths based on Knowledge and Media Types can be defined.

We conducted studies on the efficiency of graph traversal, in particular reachability queries in OWL 2 DL. Within a dynamic setting like the Semantic Web, where storing all successors/predecessors for all possible learner states is not feasible and feature checks have to be carried out to guide the user in an adaptive way, runtime becomes a limiting factor.

While the focus of our modelling approach to learning pathways was mainly on expressivity so far, future work is directed to improve its scalability. Therefore, we seek to explore property graphs for the syntactic subtask of navigation. Also, for the semantic feature queries, we plan to experiment with approximate reasoning approaches and less expensive OWL profiles. Optimizing query answering for Description Logics is currently still very much a research issue.

Acknowledgments: The research leading to these results has received funding from the EC's 7th Framework Programme (FP7/2007-2013) under grant agreement N 318496.

References

[1] Kaplan, C., Fenwick, J., & Chen, J. (1993). Adaptive Hypertext Navigation based on User Goals and Context. *User Modeling and User-Adapted Interaction*, 3(3), 193-220.

[2] White, R. W., & Huang, J. (2010). Assessing the Scenic Route: Measuring the Value of Search Trails in Web Logs. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (pp. 587-594). ACM.

[3] Dolog, P., Henze, N., Nejdl, W., & Sintek, M. (2003). Towards the Adaptive Semantic Web. In *Principles and Practice of Semantic Web Reasoning* (pp. 51-68). Springer.

[4] Holohan, E., Melia, M., McMullen, D., Pahl, C. (2005) Adaptive E-Learning Content Generation based on Semantic Web Technology.

[5] Janssen, J., Berlanga, A., Vogten, H., & Koper, R. (2008). Towards a Learning Path Specification. *Int. Journal of Continuing Engineering Education and Life Long Learning*, 18(1), 77-97.

[6] Meder, N. 2006. Web-Didaktik: Eine neue Didaktik webbasierten, vernetzten Lernens, *Bertelsmann SE & Co. KGaA*, Bielefeld, Germany

[7] Brusilovsky, P. (1998). Methods and Techniques of Adaptive Hypermedia. In *Adaptive Hypertext and Hypermedia* (pp. 1-43). Springer Netherlands.

[8] Rutledge, L., Van Ossenbruggen, J., & Hardman, L. (2005). Making RDF Presentable: Integrated Global and Local Semantic Web Browsing. *Proc. of the 14th international conference on World Wide Web* (pp. 199-206). ACM.

[9] Fionda, V., Gutierrez, C., & Pirró, G. (2012). Semantic Navigation on the Web of Data: Specification of Routes, Web Fragments and Actions. In *Proc. of the 21st international conference on World Wide Web* (pp. 281-290). ACM.

[10] Harris, S. G.: Sparql 1.1 Query Language. http://www.w3.org/TR/sparql11-query/#propertypaths (2010).

[11] Fan, W., Li, J., Ma, S., Tang, N., & Wu, Y. (2011). Adding regular expressions to graph reachability and pattern queries. In *Data Engineering (ICDE)*, 2011.

[12] Hirsh, H., & Kudenko, D. (1997, July). Representing Sequences in Description Logics. In *AAAI/IAAI* (pp. 384-389).

[13] Drummond, N., Rector, A. L., Stevens, R., Moulton, G., Horridge, M., Wang, H., & Seidenberg, J. (2006). Putting OWL in Order: Patterns for Sequences in OWL. *OWLED*. 2006.

[14] Hayes, P., Welty, C. "Defining N-ary Relations on the Semantic Web". W3C Working Group Note, 12 April 2006.

[15] Conlan, O., Hockemeyer, C., Wade, V., & Albert, D. (2002). Metadata Driven Approaches to Facilitate Adaptivity in Personalized eLearning Systems. *The Journal of Information and Systems in Education*, 1, 38-44.

[16] Jin, R., Xiang, Y., Ruan, N., & Wang, H. (2008). Efficiently answering reachability queries on very large directed graphs. In *Proc. of the 2008 ACM SIGMOD international conference on Management of data* (pp. 595-608). ACM.

[17] Harris, S. G.: RDF 1.1 ON RDF Datasets. http://www.w3.org/TR/rdf11-datasets/

[18] Gubichev, A., Bedathur, S.J., Seufert, S.: Sparqling Kleene - Fast Property Paths in RDF-3X. *First International Workshop on Graph Data Management Experiences and Systems*. ACM, 2013.

[19] Zielinski, A., Bock, J., Kohen-Vacs, D. R., Heberle, F., Henning, P. A.: Enhanced e-Learning Experience by Pushing the Limits of Semantic Web Technologies. *Proc. of the 3rd Int. Workshop on Ordering and Reasoning.* OrdRing@ISWC, 2014.

[20] Swertz, C., Bock, J. et al., S. 2013. A Pedagogical Ontology as a Playground in Adaptive Elearning Environments. *Proc. of the Informatik 2013 conference*

[21] Horrocks, I., Motik, B., & Wang, Z. The HermiT OWL Reasoner. In *ORE*. 2012.

[2] Graph Algorithm available at http://algs4.cs.princeton.edu/code/