A Semantic Framework for Sequential Decision Making

Patrick Philipp[1(✉)], Maria Maleshkova[1], Achim Rettinger[1], and Darko Katic[2]

[1] Institute AIFB, Karlsruhe Insitute of Technology, Karlsruhe, Germany
{patrick.philipp,maria.maleshkova,achim.rettinger}@kit.edu
[2] HIS, Karlsruhe Institute of Technology, Karlsruhe, Germany
darko.katic@kit.edu

Abstract. Current developments in the medical domain, not unlike many other sectors, are marked by the growing digitalisation of data, including patient records, study results, clinical guidelines or imagery. This trend creates the opportunity for the development of innovative decision support systems to assist physicians in making a diagnosis or preparing a treatment plan. To this end, complex tasks need to be solved, requiring one or more interpretation algorithms (e.g. image processors or classifiers) to be chosen and executed based on heterogeneous data. We, therefore, propose a semantic framework for sequential decision making and develop the foundations of a Linked agent who executes interpretation algorithms available as Linked APIs [9] on a data-driven, declarative basis [10] by integrating structured knowledge formalized in RDF and OWL, and having access to meta components for optimization. We evaluate our framework based on image processing of brain images and ad-hoc selection of surgical phase recognition algorithms.

Keywords: Sequential decision making · Linked APIs · Meta learning · Planning

1 Introduction

Inspired by the medical domain, we are interested in sequential decision making under uncertainty for complex tasks of arbitrary complexity. Consider a scenario in image processing which comprises several subtasks. An image, first, has to be filtered for distorting elements and normalized in terms of color. The image, then, has to be aligned with other images in the knowledge base to ease interpretation. Depending on the request of an enduser, one eventually has to segment the image and annotate it by using machine learning approaches. We are given access to a large set of interpretation algorithms eligible for each of these and other subtasks (e.g normalization, registration or segmentation). There is no trivial way to automate this process for a variety of user requests and scenarios. With no domain expert around, we might already fail to execute an interpretation algorithm with the correct data or to decide if the resulting segmentation is good or not. To make it even more complex, we might also be faced with multiple

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 392–409, 2015.
DOI: 10.1007/978-3-319-19890-3_25

eligible interpretation algorithms for a single subtask and need to decide which one to choose.

We summarize such an environment as highly heterogeneous in terms of tasks, interpretation algorithms and data. Endusers of such complex tasks often are no domain experts or only experts in a small subset of available interpretation algorithms. In addition, with a growing number of possibilities, it becomes intractable for domain experts to manually optimize complex tasks. Enabling automatic execution of complex tasks by sequential decision making, hence, is an important subject. The longterm goal is to develop a system (or agent) which knows how to optimally choose sequences of available interpretation algorithms for a given complex task as illustrated in fig. 1.

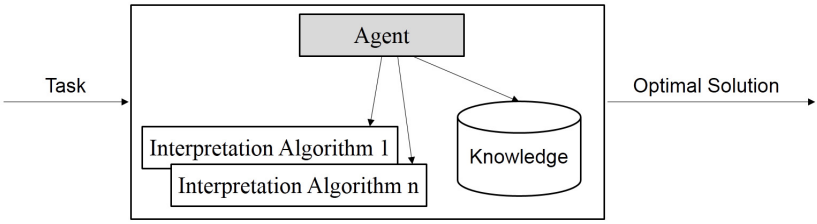

Fig. 1. An agent able to find optimal solutions for tasks

We build on prior work in the Semantic Web which centers around semantically enriched web services (so-called Linked APIs) [9], [2] and their data-driven, declarative execution (with Linked Data-Fu [10]). Equipped with these powerful technologies, we now concentrate on building a framework to enable decision making under uncertainty. Based on access to interpretation algorithms for a complex task, one can develop *meta components* and easily plug them into the current workflow. These meta *components* can comprise any strategy to choose interpretation algorithms for a task.

Although these strategies do not have to be sophisticated, we especially focus on enabling adequately complex methodologies to be flexibly and easily used. We argue that such complex approaches are necessary, if a vast amount of different kinds of information is available. Besides training samples, this might comprise manually modelled domain expert knowledge, statistical knowledge from studies or enduser feedback given in arbitrary situations. We will introduce two such *meta components* which make use of available structured knowledge to choose among interpretation algorithms.

Our contributions are threefold; we

(i) disclose different meta strategies for sequential decision making in heterogeneous domains and provide first results on their interplay with Linked Data,

(ii) enable flexible integration and testing of meta strategies within the semantic framework and
(iii) describe two meta components for (sequential-) decision making and their applications to medical scenarios.

The remainder of this paper is structured as follows. We formalize the problem in section 2 and show where our work has commonalities and differences to other approaches. The single components of our framework are, subsequently, being introduced in section 3. Section 4, then, integrates the components and shows how we solve complex tasks. In section 5, we show how our framework works in practise based on two medical scenarios and thereby dwell on a *meta learning* and an *abstract planning* component. We discuss current developments and possible improvements in section 6 and conclude the paper in section 7.

2 Problem Formulation and Related Work

We, first, formalize the essentials of the problem of solving complex tasks in heterogeneous environments and disclose our core challenges. We then summarize the work related to ours and point out our contributions.

2.1 Problem: Solving Complex Tasks with Access to Heterogeneous Interpretation Algorithms

Let X be the set of all tasks, Y the set of all abstract tasks and A the set of all available interpretation algorithms. Let further S be the set of *abstract states* defined by a subset of objects O, literals L and relations R. We denote, for simplicity, F_{s_k} as the set of features of a state s_k (i.e. a subset of $O \times R \times O$ and $O \times R \times L$). A grounded state $g(s_k)$ depicts an instance of s_k in nature. The set A_{s_k} defines the subset of applicable interpretation algorithms in s_k which is known to some degree. We, thus, assume that an interpretation algorithm $a_i \in A$ can be defined by a subset of features of F in a similar way as states $s_k \in S$. Knowing A_{s_k} depends on how we define features $f \in F$ for s_k and a_i. Let $T(s, a, s')$ be the transition function for some state s and interpretation algorithm a ending in s'. Our knowledge of $T(s, a, s')$, again, depends on the available features for s, a and s'. $T(g(s), a, g(s'))$ is not known and requires further knowledge to be approximated. A task $x(g(s_1), s_K)$ is a function defined on a grounded start state $g(s_1)$ and an abstract goal state s_K. Reaching an unknown grounded goal state $g(s_K)$ takes 1 to n state transitions $(g(s), a, g(s'))$. To solve $x(g(s_1), s_K)$, we need to find a sequence of interpretation algorithms a_i ending in the unknown grounded goal state $g(s_K)$ with high probability. An abstract task $y(g(s_1), s_K)$ is defined similarly but we need to find any sequence $a_1, \ldots, a_n$ to get from $g(s_1)$ to s_K. Our setting is much related to a Markov Decision Process (MDP) (S, A, T, R, γ) with R, in addition, being the reward function for state, interpretation algorithm pairs (s, a) and γ the discount factor. The latter regulates the influence of future interpretation algorithms a_i taken in

future steps s_k on the value estimations of current states and actions. Defining $R(s, a)$ for $x(g(s_1), s_K)$ is not straightforward as $g(s_K)$ is unknown. An absorbing state with $R(s_k, a_i) = 0$ can be artificially modelled to denote the goal s_K.

We define **abstract planning** as trying to solve an *abstract task* $y(g(s_1), s_K)$. Here, we ignore that multiple interpretation algorithms a_i might be available for s_k. **Meta learning** considers $|A_{g(s_k)}| > 1$ and tries to solve a subtask $x_{A_{g(s_k)}}(g(s_k), s_K))$ to find optimal the a_i (or combination of multiple a_i) for $g(s_k)$. **Planning** deals with solving $x(g(s_1), s_K)$ with known T and R, and **planning-related learning** considers T, R unknown and tries to approximate them (as, for instance, is done in model-based reinforcement learning).

We will only deal with abstract planning and meta learning in this work, as they do suffice for our practical applications (see section 5), but stress that the planning task and the planning-related learning task are highly interesting and important. Based on the problem setting, we derived the following needs for our semantic sequential decision making framework:

Need for a Controlled & Semantic Vocabulary. If we want to have tasks automatically executed based on state-goal pairs; tasks, data and algorithms need to be using a common vocabulary.

Need for Accessibility & Scalability. The pool of algorithms needs to be accessible in real-time and available for many concurrent tasks. New interpretation algorithms should be readily available to be used and evaluated.

Need for Data-driven & Declarative Execution. With a large number of available interpretation algorithms for a state s_k, it will quickly become intractable to manually define and evaluate all possible permutations. By executing interpretation algorithms when they match s_k, we gain flexibility and can delegate the optimization problem. With growing experience, one could generalize the learnt optimal decisions to similar s_k.

Need for Meta Learning Components. As the optimization problem is neither trivial nor homogeneous, it might not be solvable by a single piece of software. One rather needs several meta components which are experts for different s_k.

2.2 Existing Frameworks

There is an ongoing research interest in so-called 'workflow systems' that enable describing and executing algorithms of different kinds. The work centered around semantic workflows [3] aims to enable the automatic composition of components in large-scale distributed environments. Generic semantic descriptions support combining algorithms and enable formalizing ensembles of learners. Therefore, conditions and constraints need to be specified. The framework also automatically matches components and data sources based on user requests.

Wood et al. [12] create abstract workflows as domain models which are formalized using the Web Ontology Language (OWL) and enable dynamic instantiation of real processes. These models can then be automatically converted into more specific workflows resulting in OWL individuals. The components can be reused in another context or process, and one can share abstract representations across the Web through OWL classes.

Automatic orchestration of analytical workflows is studied in [1]. The system essentially uses a planner, a leaner and a large (structured-) knowledge base to solve complex tasks. A large amount of potential workflows are taken into account to answer a user specified query with the optimal choice. The decision process comprises complex learning and planning approaches, and entails exploring large possible feature spaces. Lastly, atomic actions are lifted with semantic annotations to better adapt to user queries.

In contrast to previous approaches, our framework benefits from a combination of RDF and OWL, minimizing the efforts required for describing the algorithms and the used data. We use a data-driven approach to execute workflows and work towards completely automatic, declarative and optimal compositions of such. Our novel contribution essentially enables to develop powerful *Linked agents* capable of solving complex tasks in heterogeneous environments. Besides, only a small fraction of the above approaches employ a structured knowledge base. We are able to store structured performance-related information and try to reuse this evidence in order to optimize results. In addition, with the *Linked agent*, we can flexibly integrate new kinds of structured knowledge as well as so-called *meta components* to optimize decision making under uncertainty.

3 Components for Learning Optimal Web Service Pipelines

Our system infrastructure comprises four core component types:

1. Linked Interpretation Algorithms
2. A Structured Knowledge Base
3. Linked Meta Components
4. A Data-Driven Execution Engine (the Linked Agent)

Fig. 2 illustrates the framework components. We will now explain their respective functionality and put things together in section 4.

3.1 Structured Knowledge Base

A structured knowledge base stores both metadata and data, and provides a common and controlled vocabulary. Our framework uses the Resource Description framework (RDF) and the Web Ontology Language (OWL) for annotating raw data and modelling ontologies. As the Linked Data principles suggest, persistent URIs to resources have to be available and provide sufficient information

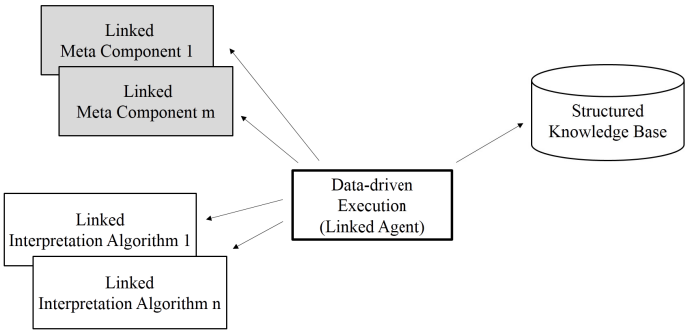

Fig. 2. Schematic Overview of the Framework

for lookups. Appropriate concepts for interpretation algorithms and data were modelled to enable the integration of new components. Fig. 5 depicts the components of the structured knowledge base used in our medical scenarios.

3.2 Linked Interpretation Algorithms

We deploy interpretation algorithms as web services to make them easily accessible in our infrastructure. We follow the idea of Linked Data web services (i.e. Linked APIs [9]) and applied them to medical interpretation algorithms in [2]. A *Linked interpretation algorithm* (an interpretation algorithm lifted to a Linked API) provides a standardized description of its functionality by reusing elements of the structured knowledge base. The description also defines how to communicate with the Linked interpretation algorithm and how to execute its methods. A minimal set of information of the description is summarized in table 1.

Table 1. Minimal description for Linked interpretation algorithms

Non-functional requirements	**Functional requirements**
Domain Experts	*Inputs*
Service Endpoint	*Preconditions*
Example request & response	*Outputs*
Algorithm class	*Postconditions*

An intuitive example of an arbitrary image processor is given in fig. 3. An image, defined in a data type ontology, is part of a pre- and postcondition of a Linked interpretation algorithm. Pre- and postconditions define strict rules about the states before- and after executing the Linked interpretation algorithm. The degree of detail of both pre- and postcondition is strongly dependent on the wrapping process of the respective interpretation algorithm. If semantics and interpretation algorithm are strongly intertwined, fine-grained semantics with rich information are available.

We use **kb** and **kbont** namespaces to describe instances and ontologies available in the knowledge base. The **msm** namespace corresponds to the minimal service model [5] which advocates and enables lightweight semantics for web services.

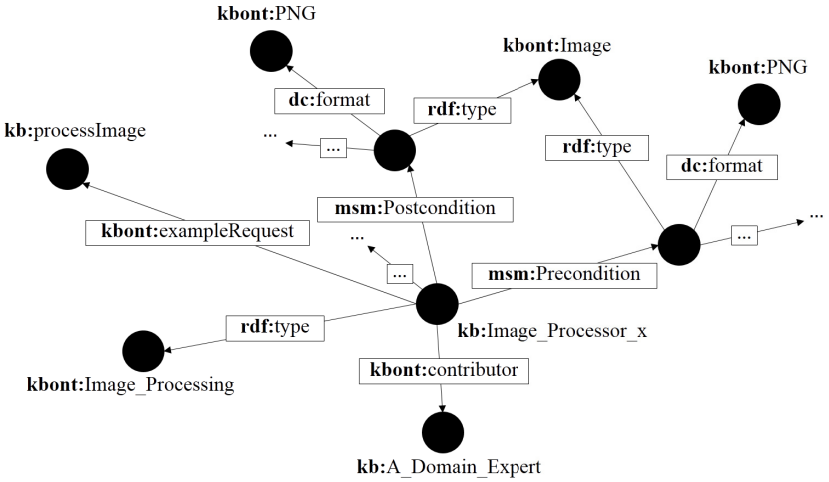

Fig. 3. An exemplary Linked image processor (namespaces omitted)

3.3 Linked Meta Components

There might be numerous approaches to choose, on a meta level, among Linked interpretation algorithms given $g(s_k)$. Such strategies generate policies which can be naive, sophisticated, biased on subjective criteria or otherwise. We enable flexible using, testing and exchanging of potentially powerful meta approaches in terms of so-called *Linked meta components*. Linked meta components are essentially Linked APIs and implement any decision making strategy of arbitrary complexity. To have access to all available interpretation algorithms, we assume a structured knowledge base linking to them.

A Linked meta component specifies the amount of information it needs by its precondition and is only called *iff* the agent can provide for all information. Besides a list of all available interpretation algorithms, a learner might, for instance, want to access a performance table which stores a history of validated results. We will discuss two cases of meta components, namely *abstract planning* and *meta learning*, in section 4 and show their practical application in section 5.

3.4 Data-Driven Execution (Linked Agent)

We integrate the prior components by using Linked Data-Fu [10]. The Linked Data-Fu rule-based execution engine describes and implements a formalism to

virtually integrate data from different sources in real-time and have Linked APIs executed based on rules. In our framework, Linked Data-Fu searches eligible Linked meta components and Linked interpretation algorithms for (newly arrived) annotated data, and uses the structured knowledge base to execute them. Each Linked interpretation algorithm is represented as single rule which we automatically generate based on its description. Fig. 4 summarizes this process. When a state $g(s_k)$ fulfils the preconditions of a Linked interpretation algorithm, a HTTP POST request with grounded preconditions is issued to its service URI.

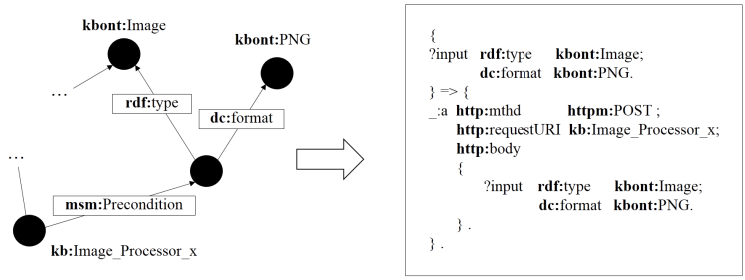

Fig. 4. A Linked Data-Fu rule for the Linked image converter

The automatic matching between Linked interpretation algorithms and structured data is highly advantageous. If interpretation algorithms have to be trained by samples, the agent can directly feed all annotated training data to the Linked interpretation algorithm. This is generally possible by merely defining rules for Linked Data-Fu. Even more important, with growing number of diverse Linked interpretation algorithms, we can automatically solve new complex tasks without additional manual effort.

We will refer to the Linked Data-Fu engine instantiated in our framework as *Linked agent*. The title is justified when combining the engine with appropriate Linked meta components, Linked interpretation algorithms and a structured knowledge base.

4 Solving Complex Tasks with Meta Learning and Abstract Planning

We will now explain the required interplay between the four components to enable sequential decision making for complex (abstract-) tasks $y(g(s_1), s_K)$ and $x(g(s_1), s_K)$. We, first, address the Linked abstract planning case where an abstract task $y(g(s_1), s_K)$ has to be solved by finding appropriate Linked interpretation algorithms a_i . Here, we do not try to distinguish between high- and low quality results (as this involves checking the instances), and trust the descriptions of Linked interpretation algorithms. Each Linked interpretation algorithm applicable in $g(s_k)$ *and* needed for reaching s_K will be chosen. The second case deals with Linked meta learning and builds on Linked interpretation algorithms

selected by Linked abstract planning. An 'optimal' Linked interpretation algorithm is returned for $g(s_k)$ if $|A_{g(s_k)}| > 1$.

In fig. 5, we give a generic overview of interactions between the Linked agent and Linked meta components, Linked interpretation algorithms and the structured knowledge base. The Linked agent first queries the knowledge base to get all available Linked interpretation algorithms and then calls a Linked abstract planner. Based on the resulting set of candidate Linked interpretation algorithms for reaching goal s_K, it executes a Linked meta learner capable of dealing with the Linked interpretation algorithms. Finally, all 'optimal' Linked interpretation algorithms are being executed.

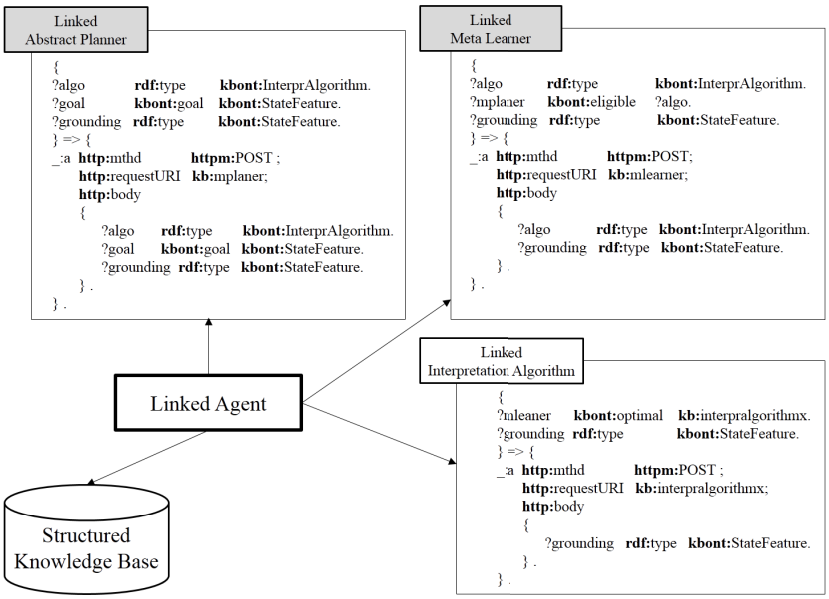

Fig. 5. Interactions within the framework to integrate Linked meta components

4.1 Linked Abstract Planning

Depending on a new task $y(g(s_1), s_K)$, the Linked agent evaluates the grounded state $g(s_k)$ in terms of rule checking. The Linked agent, therefore, keeps a set of automatically generated rules for each Linked interpretation algorithm. We cannot assume, however, that $g(s_k)$ only triggers interpretation algorithms which help reaching the goal s_K, as there might be a large amount of Linked interpretation algorithms. We, thus, need a Linked abstract planner to only return candidate Linked interpretation algorithms for reaching s_K.

A Linked abstract planner has to first query the structured knowledge base for all available Linked interpretation algorithms. It, then, decides which Linked

interpretation algorithms are applicable to reach s_K based on an arbitrary mechanism and finally outputs a subset of A. Fig. 5 illustrates the agent rule to call the Linked abstract planner depending on its precondition. We describe one implementation of a Linked abstract planner in section 5.2.

4.2 Linked Meta Learning

In the meta learning setting, we want to find an optimal Linked interpretation algorithm a_i solving a subtask $x_{A_{g(s_k)}}(g(s_k), s_K))$. We assume to know candidates $A_{g(s_k)}$ of Linked interpretation algorithms to reach s_K (e.g. due to a Linked abstract planner), but are still faced with uncertainty about their performances given $g(s_k)$. Solving the learning setting can be approached with simple heuristics, but might require complex machine learning approaches to give good estimates. With our framework, we enable using any meta learning approach by wrapping it as Linked meta learner.

We reuse the Linked abstract planner and integrate a Linked meta learning component by creating a rule for the Linked agent. Fig. 5 describes the generic rule and section 5.3 introduces a Linked meta learner for one of our medical scenarios. The Linked meta learner outputs one Linked interpretation algorithm a_i for $A_{g(s_k)} > 1$ if the preconditions of Linked meta learner and a_i match. Otherwise, no decision is made on A_{s_k} and all candidates will be executed by the Linked agent. Note that multiple outputs of Linked interpretation algorithm could be combined (e.g. by a weighted majority voting) if a Linked meta learner was used after all eligible Linked interpretation algorithms have been executed, slightly changing the interactions found in fig. 5. The linked meta learner would, additionally, expect results of the Linked interpretation algorithms to compute their weights.

5 Medical Scenarios with Meta Components and Evaluations

We now introduce two scenarios set in the medical domain and illustrate possible Linked meta components. The first scenario deals with image processing. We developed a Linked abstract planner to derive eligible Linked interpretation algorithms for the so-called brain tumour progression mapping (TPM). In the second scenario, we optimized the choice among two Linked interpretation algorithms for phase recognition in minimal invasive surgeries with a Linked meta learner. Both scenarios use a common instantiation of our framework. We will start by explaining the shared components and subsequently focus on the individual scenarios.

5.1 A Semantic Framework for Medical Sequential Decision Making

The medical framework with its interpretation algorithms and data is being developed within the Cognition-Guided Surgery project [1]. Every interpretation

[1] http://www.cognitionguidedsurgery.de/

algorithm considered in the scenarios was wrapped as Linked interpretation algorithm. We modelled the descriptions with domain experts and developers of the interpretation algorithms, and integrated them in a central instance of a Semantic MediaWiki (SMW). We use an instance of XNAT [2] to store patient-relevant data and provide a RDF wrapper which lifts XNAT with semantic concepts. The knowledge base can be considered as union between the SMW and its links to other resources, such as XNAT.

Linked interpretation algorithms can automatically be executed with the Linked agent. We implemented a conversion mechanism from Linked interpretation algorithm descriptions to Linked agent rules (see fig. 4), and, thus, reduced the manual work for integrating new Linked interpretation algorithms. The Linked agent crawls the hierarchy imposed by XNAT according to simple rules and executes every Linked interpretation algorithm per patient if it is eligible. While this only covers offline scenarios, we can easily extend the setting to the online case. The complete framework with two Linked meta components is illustrated in fig. 6.

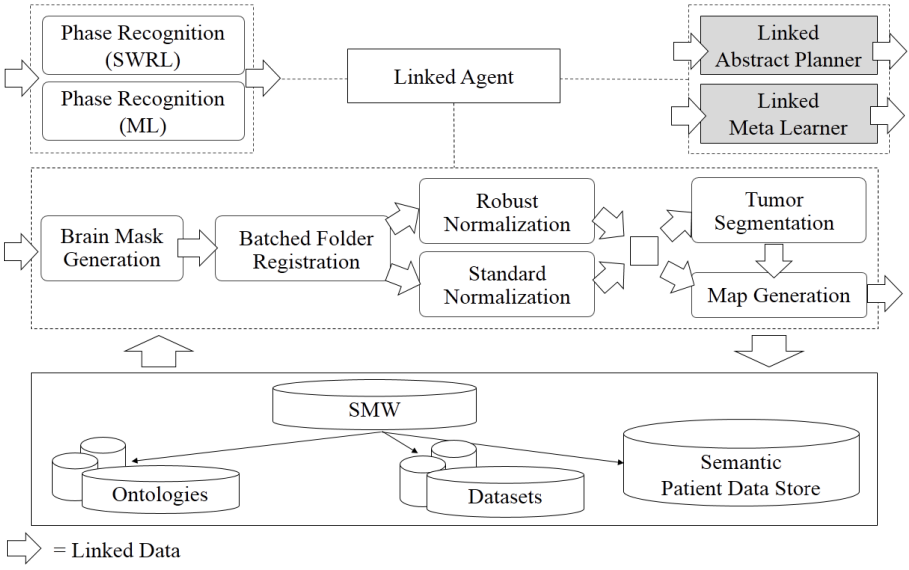

Fig. 6. The Semantic Framework for Medical Sequential Decision Making (extended based on [7])

5.2 Tumour Progression Mapping in the Semantic Framework

Tumour Progression Mapping (TPM) is an approach to visualize brain tumours in their progression over time. One, thereby, focusses on supporting radiologists

[2] http://www.xnat.org/

in their daily work. Radiologists, otherwise, would have to assess the irregular growth of brain tumours based on raw headscans which causes a lot of extra effort. When generating a TPM, different types of images are used and produced, and adequate interpretation algorithms need to be executed in correct order and with correct subsets of images.

The TPM generation process is illustrated in the framework overview (fig. 5). The images are stored in our knowledge base and converted into a common format. A mask for the brain region is created by the next interpretation algorithm, ensuring that subsequent tasks are not influenced by bones or other structures. A registration algorithm, then, spatially registers all brain images of a patient. The following normalization task adapts the intensities of MRI scans and generates similar values for similar tissue types. If additional annotations for a patient are available, the normalization becomes more robust by making use of a different normalization interpretation algorithm. The TPM can now be created by invoking the appropriate interpretation algorithm. An optional additional interpretation algorithm can automatically segment tumours and integrate the results into the map.

We studied how to wrap interpretation algorithms used in the TPM setting in [2] and initially applied Linked Data-Fu in [8], [7]. We now integrate these ideas into the semantic framework and introduce a Linked meta planner.

A Linked Abstract Planner for TPM. We developed Linked interpretation algorithms for every step in the TPM generation process. Listing 1.1 contains the preconditions of the brain stripping algorithm ('Brain Mask Generation') with headscan and initialization images as inputs.

```
?inputImage      rdf:type       kbont:Headscan;
                 dc:format      "image/nrrd".

?brainImage      rdf:type       kbont:BrainAtlasImage;
                 dc:format      "image/mha".

?brainAtlasMask  rdf:type       kbont:BrainAtlasMask;
                 dc:format      "image/mha".
```

Listing 1.1. Preconditions of the Linked brain mask generation algorithm

We created a finite MDP by using the pre- and postconditions of Linked interpretation algorithms to define abstract states s_k with local scopes, i.e. we only consider preconditions of s_k. The transition probabilities T are defined in equation 1 and make up an $S \times (A + 1) \times S$ matrix by adding a dummy interpretation algorithm pointing to the goal state, when the latter was reached. The reward function R is a $S \times (A + 1)$ matrix (see equation 2). By using any strategy to solve the MDP (e.g value iteration), we find eligible Linked

interpretation algorithms to solve the task.

$$T(s,a,s') = \begin{cases} t_{sas'} = \frac{1}{|A_{s_k}|} & \exists(s,a,s') \text{ with respect to } F_s \text{ and } F_{s'} \\ t_{sas'} = 0 & \text{otherwise} \end{cases} \quad (1)$$

$$R(s,a) = \begin{cases} r_{sa} = 1 & \text{if } a \text{ equals dummy algorithm and } s \text{ equals goal} \\ r_{sa} = 0 & \text{otherwise} \end{cases} \quad (2)$$

The resulting Linked abstract planner takes as input Linked interpretation algorithms, patient information of type **kbont:**ImageFeature (i.e. $g(s_k)$), and goal state **kbont:**TumorProgressionMapping (i.e. s_K), and returns a set of eligible Linked interpretation algorithms. The abstract Linked Data-Fu rule for the Linked abstract planner is depicted in listing 1.2. The Preconditions for the brain mask generation algorithm (see listing 1.1) could, then, replace the abstract image features.

```
{
?algo          rdf:type         kbont:InterprAlgorithm.
?goal          kbont:goal       kbont:TumorProgressionMapping.
?grounding     rdf:type         kbont:ImageFeature.
} => {
_:a   http:mthd          httpm:POST ;
      http:requestURI    kb:mplanermdp;
      http:body
    {
        ?algo          rdf:type         kbont:InterprAlgorithm.
        ?goal          kbont:goal       kbont:TumorProgressionMapping.
        ?grounding     rdf:type         kbont:ImageFeature.
    } .
} .
```

Listing 1.2. Linked Data-Fu rule for executing the Linked abstract planner

Evaluation. A part of the evaluation of the Linked TPM scenario was conducted in [7] and [2]. The TPM generation process was shown to work based on the descriptions of the single Linked image processing algorithms and an initial implementation of Linked Data-Fu without a Linked meta planner. We showed that no substantial overhead is produced while executing the interpretation algorithms on the web and that the correct pipeline is built automatically.

Our Linked abstract planner, now, creates a finite MDP and automatically constructs T and R based on the available Linked interpretation algorithms A, the goal s_K and the current grounding $g(s_k)$. Consider a grounding $g(s_k)$ for the brain stripping algorithm (listing 1.1) and the goal **kbont:**TumorProgressionMapping with all paths to the TPM being possible. Besides the 6 Linked interpretation algorithms involved in the TPM process, the algorithm pool consists of 2 Linked phase recognizers of the subsequent

scenario. We use a discount factor of 0.9, perform value iteration and derive $V =< 0.32805.0.3645, 0.405.0.405, 0.45, 0.45, 1.00, 0, 0 >$ after 6 iterations. States with values greater than zero depict preconditions of Linked interpretation algorithms which have to be executed to reach the goal (except for the absorbing goal state s_K).

5.3 Surgical Phase Recognition in the Semantic Framework

Surgical phase recognition is one step towards reducing the information overload for surgeons during surgery. Depending on the current phase, one could display an adequate subset of information, which benefits the surgeons in his or her decision making. To recognize the phase, one might leverage a variety of sensor outputs. In this scenario, only activity triples consisting of the currently used instrument, the performed action and the corresponding anatomical structure are used to determine the current phase (e.g. <Scalpel, cut, Gallbladder >).

The interpretation algorithms we considered for our learning scenario consisted of a rule-based interpretation algorithm using the Semantic Web Rule Language (SWRL) introduced in [4] and a machine learning (ML)-based phase recognition algorithm which takes in training samples (i.e. annotated surgeries). Both algorithms have varying degrees of performance and make mistakes in their predictions, as shown in table 2. If one could learn in which situations the respective interpretation algorithms excel, it would be highly beneficial. Our first approach to empirically learn the optimal phase recognition algorithm was mentioned in [7] and is now explained in terms of a generalizable Linked meta learner.

Table 2. Performance evaluation of phase recognition algorithms in 5 different surgeries

Algorithm	Surgery 1	Surgery 2	Surgery 3	Surgery 4	Surgery 5
ML-based	0, 9062	0, 6635	0, 9032	0, 4484	0, 6383
SWRL	0, 9315	0, 7753	0, 89	0, 8137	0, 7241

A Linked Meta Learner for Surgical Phase Recognition. We developed Linked interpretation algorithms for both phase recognition algorithms, and defined their inputs and outputs in terms of semantic pre- and postconditions [7]. See listing 1.3 for the preconditions of the Linked ML-based phase recognition algorithm. The postcondition simply states that the result has to be of type **kbont:**Phase which ensures, by inference, that only modelled phases can occur. The resulting Linked interpretation algorithms need to be initialized with a laparoscopic ontology with concepts for the surgical setting. The ML-based phase recognizer, in addition, has to be trained with samples.

```
?trainingSample   rdf:type          kbont:Surgery.

?ontology         rdf:type          kbont:Ontology.

?event            rdf:type          kbont:SurgicalEvent;
                  kbont:instrument  ?instrument;
                  kbont:action      ?action;
                  kbont:structure   ?structure.

?instrument       rdf:type          kbont:Instrument.
?action           rdf:type          kbont:InstrumentalProperty.
?structure        rdf:type          kbont:TreatedStructure.
```

Listing 1.3. Preconditions of the Linked ML-based algorithm

We developed a Linked meta learner for the setting of two competing Linked phase recognition algorithms. As it is quite specific and works only for Linked phase recognition algorithms, we define a less general description of the Linked meta learner. Listing 1.4 depicts the rule for executing the Linked meta learner. It assumes available candidate Linked interpretation algorithms to recognize surgical phases and outputs the highest weighted option. Please note that we can elegantly define the generality of the Linked meta learner based on the concept types we use. If it was able to optimally choose among two or more image processors as well, we could easily express that in the pre- and postconditions.

```
{
?algo         rdf:type         kbont:PhaseRecognitionAlgorithm.
?mplaner      kbont:eligible   ?algo.
?grounding    rdf:type         kbont:StateFeature.
} => {
_:a   http:mthd        httpm:POST;
      http:requestURI  kb:mlearnerheuristic;
      http:body
    {
        ?algo        rdf:type      kbont:PhaseRecognitionAlgorithm.
        ?grounding   rdf:type      kbont:StateFeature
    } .
} .
```

Listing 1.4. Linked Data-Fu rule for executing the Linked meta learner

The Linked meta learning component assesses the performance of a given Linked phase recognizer based on training samples close to the current state $g(s_k)$. It trains the ML-based phase recognizer on $n-1$ samples and predicts on the remaining surgery. The Linked meta learner repeats the process n times and derives the probability for a candidate with respect to $g(s_k)$ based on its performance on similar samples. The heuristic is summarized in algorithm 1.

Algorithm 1. Meta Learning Heuristic given Linked interpretation algorithms L, number of neighbours to consider k, state s_k, cut t

```
N ← nearestNeighbours(g(s_k), k)
T ← set of training samples cut into t subsets T_i
for all T_i do
    for all l ∈ L do
        train(l, T \ T_i) //if possible
        updatePerformanceTable(l, T_i)
for all l ∈ L do
    w_l ← estimatePerformance(l, N)
w_l* ← arg max_{l∈L} w_l
return l
```

Evaluation. We determined the nearest neighbours based on the similarity between the current activity triple and the ones in the training set. The total success rate of the Linked meta learning component reached a better success rate than the best phase recognizer or was at least able to compete [7]. The results are summarized in table 3. In general, meta approaches learning a probability distribution over such algorithms often provide stabler results in the longterm, but often fail to choose the optimal algorithm for every single $g(s_k)$ in hindsight.

Table 3. Performance evaluation of the Linked meta learner [7]

Algorithm	Surgery 1	Surgery 2	Surgery 3	Surgery 4	Surgery 5
Linked meta learner	0,9332	0,7786	0,9180	0,7782	0,7238

6 Discussion

Our framework builds on the use RDF and OWL to describe the functionality of meta components, interpretation algorithms and data, and their conditions for execution. We use the pre- and postconditions of a Linked interpretation algorithm to decide if it is generally eligible for a (grounded-) state, and employ Linked meta components to find and choose among candidates for a specific goal. One could relax the impact of the pre- and postconditions and shift the decision to Linked meta components. This is useful for generalization or for dealing with situations where few semantics are available. In addition, solving a task is dependent on the available pool of Linked interpretation algorithms. If the goal cannot be reached, the Linked abstract planner does not return any Linked interpretation algorithm and if no competing candidate is available, the Linked meta learner only returns the single option. Besides, we do not limit our framework to the medical domain but want to stress the added value in heterogeneous use cases.

In case of abstract planning and MDPs, we only leveraged semantics to a small degree in terms of pre- and postcondition matchings. We want to investigate the potential advantages of richer classes such as relational MDPs [6]. In addition, linked meta components can make large use of an arbitrary amount of features besides their preconditions. Linked interpretation algorithm descriptions, although potentially modelled by domain experts, do not necessarily capture all relevant dependencies. Hence, learning the optimal feature subset of F_{s_k} to better estimate transition probabilities $T(s, a, s')$ or enriching F_{s_k} with more features seem interesting extensions to our framework. We also want to develop new Linked meta components for the pure planning and planning-related learning task as defined in section 2.

Since we enable to use multiple Linked meta components at once, one could have them compete as well. This is what Vilalta & Drissi [11] depict as curse of infinite bias. We want the system to be self-adaptive and improve with experience, which it already does to some extent by automatically considering training samples or further data sources. However, each of these Linked meta components has some kind of bias in terms of their methodology used. It is very interesting to have meta components compete with each other but the question how to deal with bias is important on its own.

7 Conclusion

We introduced our work on a semantic framework for sequential decision making in a heterogeneous environment. We reused established techniques of the Semantic Web to develop a data-driven, declarative framework for Linked interpretation algorithms and extended it with means to solve complex tasks. We, therefore, defined the problem of complex task solving in our setting and distinguished between (abstract-) planning- and (meta-) learning scenarios, with initial observations on the interplay with Linked Data (contribution (i)). By now, abstract planning and meta learning can be realized with appropriate Linked meta components, which can be naturally integrated with the Linked agent (contribution (ii)). We described two exemplary medical use cases which did benefit from our framework. The image processing and sensor interpretation algorithms were wrapped as Linked APIs and executed by the Linked agent. The two meta learning components for the use cases realized and optimized the pipeline construction for solving the complex tasks (contribution (iii)). We, finally, discussed current shortcomings of our framework, potential improvements we are investigating and the long-term goal of a self-adaptive framework.

Acknowledgments. This work was carried out with the support of the German Research Foundation (DFG) within projects I01, A01, R01, I04, SFB/TRR 125 "Cognition-Guided Surgery". We especially thank Stefanie Speidel, Christian Weber, Michael Götz, Anna-Laura Wekerle, Miriam Klauß, Hannes Kenngott and Beat Müller-Stich for support with the medical use cases.

References

1. Beygelzimer, A., Riabov, A., Sow, D., Turaga, D.S., Udrea, O.: Big data exploration via automated orchestration of analytic workflows. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), pp. 153–158. USENIX, San Jose (2013)
2. Gemmeke, P., Maleshkova, M., Philipp, P., Götz, M., Weber, C., Kämpgen, B., Zelzer, S., Maier-Hein, K., Rettinger, A.: Using linked data and web apis for automating the pre-processing of medical images. COLD (ISWC) (2014)
3. Gil, Y., Gonzalez-Calero, P.A., Kim, J., Moody, J., Ratnakar, V.: A semantic framework for automatic generation of computational workflows using distributed data and component catalogues. Journal of Experimental & Theoretical Artificial Intelligence **23**(4), 389–467 (2011)
4. Katić, D., Wekerle, A.-L., Gärtner, F., Kenngott, H., Müller-Stich, B.P., Dillmann, R., Speidel, S.: Knowledge-driven formalization of laparoscopic surgeries for rule-based intraoperative context-aware assistance. In: Stoyanov, D., Collins, D.L., Sakuma, I., Abolmaesumi, P., Jannin, P. (eds.) IPCAI 2014. LNCS, vol. 8498, pp. 158–167. Springer, Heidelberg (2014)
5. Kopecky, J., Gomadam, K., Vitvar, T.: hrests: An html microformat for describing restful web services. In: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT 2008, vol. 1, pp. 619–625. IEEE (2008)
6. van Otterlo, M.: The logic of adaptive behavior: knowledge representation and algorithms for adaptive sequential decision making under uncertainty in first-order and relational domains, vol. 192. Ios Press (2009)
7. Philipp, P., Katic, D., Maleshkova, M., Rettinger, A., Speidel, S., Wekerle, A.L., Kämpgen, B., Kenngott, H., Studer, R., Dillmann, R., Müller, B.: Towards cognitive pipelines of medical assistance algorithms. In: Proc. Computer Assisted Radiology and Surgery (CARS) (2015)
8. Philipp, P., Maleshkova, M., Götz, M., Weber, C., Kämpgen, B., Zelzer, S., Maier-Hein, K., Rettinger, A.: Automatisierte verarbeitung von bildverarbeitungsalgorithmen mit semantischen technologien. In: Bildverarbeitung fÃijr die Medizin (BVM), pp. 263–268 (2015)
9. Speiser, S., Harth, A.: Integrating linked data and services with linked data services. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 170–184. Springer, Heidelberg (2011)
10. Stadtmüller, S., Speiser, S., Harth, A., Studer, R.: Data-fu: A language and an interpreter for interaction with read/write linked data. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 1225–1236 (2013)
11. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review **18**(2), 77–95 (2002)
12. Wood, I., Vandervalk, B., McCarthy, L., Wilkinson, M.D.: OWL-DL domain-models as abstract workflows. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012, Part II. LNCS, vol. 7610, pp. 56–66. Springer, Heidelberg (2012)