Applying DLs to workflow reuse and repurposing

Antoon Goderis, Ulrike Sattler and Carole Goble
University of Manchester, UK
{goderis,sattler,carole}@cs.man.ac.uk

1 Reuse and repurposing in e-Science

Workflow techniques are an important part of *in silico* experimentation, potentially allowing a scientist to describe and enact their experimental processes in a structured, repeatable and verifiable way. The ^{my}Grid (www.mygrid.org.uk) workbench, a set of components to build workflows in bioinformatics, currently allows access to a thousand globally distributed services and a hundred workflows, some of which orchestrate up to fifty services. Figure 2 shows the example of a ^{my}Grid workflow which gathers information about genetic sequences in support of research on Williams-Beuren syndrome [10].

Much of the research geared towards the construction of on-line processes (*i.e.* workflows) is led by a vision of automatic composition of services based on extensive formalisation (see for example www.daml.org/services/owl-s/pub-archive.html). Such research can be complemented with techniques that exploit those cases where existing workflows and *fragments* of workflows can be *reused*, thereby benefitting from hard-won human experience in composing services. A *workflow fragment* is a piece of an experimental description that is a coherent sub-workflow that makes sense to a domain specialist. Each fragment forms a useful resource in its own right and is identified and annotated at publication time. We distinguish between *reuse*, where workflows and workflow fragments created by one user might be used as is, and *repurposing*, where they are used as a starting point by others. The idea of repurposing is that a user looks for workflows that are *close enough* to the user's requirements so that these workflows can be fit to a new purpose. In Figure 1, we show the lifecycle of a repurposed workflow.

1. Before embarking on a new design the scientist consults a registry of existing workflows. Search facilities based on an ontology and a database repository identify any existing workflows that are relevant to them.
2. Workflows or their fragments are potentially edited; services are parameterised or bound to end points but rarely altered. Other services, workflows or workflow fragments are sought, or new ones are created.

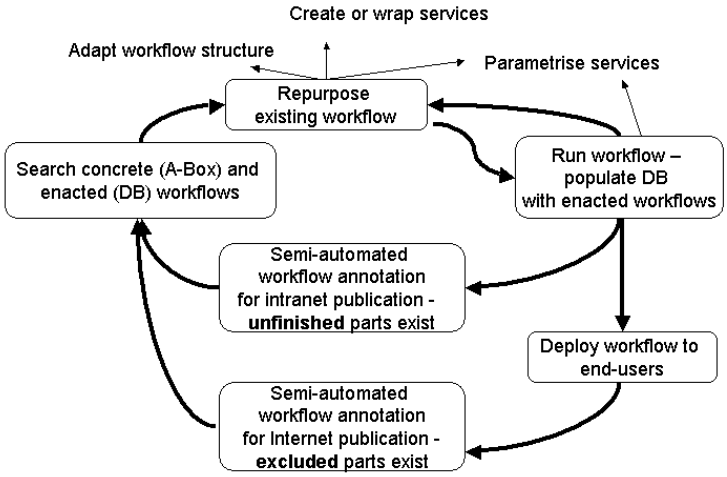

Figure 1: The role of annotation in the workflow repurposing lifecycle

3. The workflow is tested and generates results which are stored in a database.
4. It must then be a simple task to publish the workflow, annotate with a description and additional knowledge on the suitability of the original workflow for this task, so that others can benefit. Building workflows can take months, even years. Keeping others in the organisation informed of new, even incomplete, workflows and to publish early results is important.
5. We cycle through this process until the scientist is happy, and the workflow has proven its worth.
6. The user publishes the workflow to the wider Web community, for instance to back up results in a journal paper. Because workflows often contain sensitive information or refer to services that are unavailable outside an organisation, not all of the information in the workflow can be shared with the outside world. The excluded parts are left incomplete.

In this paper, we reflect on how the description logic (DL) based Web Ontology Language OWL Lite [7] could be used for searching workflow fragments. We use this expressive DL because of: (i) it being a standardised KR language; (ii) the support it offers for classifying a large collection of workflow fragments; (iii) the potential to describe and query for workflows at a level of abstraction suited for a domain scientist through query languages; (iv) the support for representing incomplete workflows. Various authors have experimented with service discovery based on DL reasoning, typically based on the OWL-S *Profile* [11] or WSMO *Capability* descriptions [9]. Unlike this DL discovery work, we envisage the discovery of workflow fragments. Fragments incorporate a simple notion of control flow, which is exploited for discovery. This is in contrast to the Profile or Capability descriptions, where control flow is not used during discovery. We

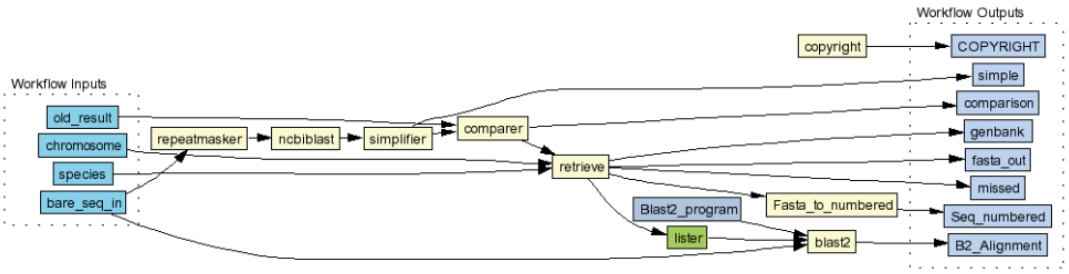

Figure 2: A ^{my}Grid workflow to annotate genetic sequence

want to capture a layer of abstraction suitable to a scientist to retrieve workflows, thereby *explicitly ignoring* complex control flow which may be present in the workflow specification.

We start by extending an existing T-Box service ontology [12] and then use this representation to answer a set of queries over a type of workflow fragments common in bioinformatics. We conclude by discussing the need for handling more complicated fragments and a hybrid approach to repurposing.

2 Example queries

From observing bioinformaticians and other scientists build workflows, we have collected the following set of practical queries. We shall refer to these queries throughout the text.

- Q1 Given a data point, service, fragment or workflow, where has this item been used before?
- Q2 Show the common data, services, service graphs or data graphs between two fragments or workflows.
- Q3 Given a set of data points, services, or fragments, have these been connected up in an existing base of workflows? If not, what are the closest available alternatives for doing so? How do these alternatives rank?
- Q4 As more and more workflows become available, fragments are reused and repurposed in a variety of workflows. How can one systematically keep track of these interrelationships?
- Q5 Which workflows are work in progress?
- Q6 Show the differences between two workflow versions.
- Q7 Show the evolution of a workflow over time.

3 Representing workflows / workflow fragments

Consider the Williams-Beuren syndrome gene annotation pipeline in Figure 2. Similar to most workflows we find in bioinformatics, this workflow is a pipeline that fans out: one starts out with a limited number of inputs, and ends up with

many more outputs. We want to represent such a tree-like workflow in a DL. We assume that the T-Box collects generic descriptions of workflows which, given their generality, hardly change, while the A-Box provides a place for scientists and workflow developers to add new workflows.

T-Box Services are the basic building blocks of a workflow. We adopt the definition of a service used in the ^{my}Grid ontology [12]. This ontology, originally in DAML+OIL and now in OWL Lite, contains 550 concepts and 69 roles and describes bioinformatics service classes. A `myGridService` service class (shown in what follows as `Service`) `usesOrProduces` one or more `BioDomainConcept`. Optionally, it includes a `performsTask` role relationship, as well as other, bioinformatics specific, roles (not shown).

$$\begin{aligned} \texttt{hasInput} &\dot{\sqsubseteq} \texttt{usesOrProduces} \\ \texttt{hasOutput} &\dot{\sqsubseteq} \texttt{usesOrProduces} \\ \texttt{Service} &\doteq \texttt{BioProcess} \sqcap \exists \texttt{usesOrProduces}.\texttt{BioDomainConcept} \end{aligned}$$

We define workflows as entities that contain at least one service, by means of the transitive *has part* role `hp`.

$$\texttt{WF} \doteq \texttt{Process} \sqcap \exists \texttt{hp}.\texttt{Service}$$

Adding an ordering is made possible through the *has direct successor* role `hds`.

The following is a partial T-Box description for the sequence analyzer `SeqAna` fragment combining 3 services of Figure 2.

$$\begin{aligned} \texttt{SyntaxTranslator} &\dot{\sqsubseteq} \texttt{Mediator} & \qquad \texttt{Mediator} &\dot{\sqsubseteq} \texttt{Service} \\ \texttt{RepeatMasker} &\dot{\sqsubseteq} \texttt{RemRedDNA} & \qquad \texttt{RemRedDNA} &\dot{\sqsubseteq} \texttt{Service} \\ \texttt{BLAST} &\dot{\sqsubseteq} \texttt{SeqSimSearch} & \qquad \texttt{SeqSimSearch} &\dot{\sqsubseteq} \texttt{Service} \\ \texttt{BLASTn} &\dot{\sqsubseteq} \texttt{BLAST} \end{aligned}$$

Biologists often precede BLASTing a genetic sequence (*i.e.* running it through a BLAST service) with a RepeatMasker service to remove redundant structures and obtain a non-redundant sequence. We acknowledge the importance of this service combination for the biology domain by introducing a `BLASTNRSeq` workflow fragment in the T-Box. In general, concept expressions describing workflows are called *abstract workflows*.

$$\begin{aligned} \texttt{BLASTNRSeq} &\doteq \texttt{WF} \sqcap \exists \texttt{hp}.(\texttt{RepeatMasker} \sqcap \exists \texttt{hds}.\texttt{BLAST}) \\ \texttt{SeqAna} &\doteq \texttt{WF} \sqcap \exists \texttt{hp}.(\texttt{BLASTNRSeq} \sqcap \exists \texttt{hds}.\texttt{Mediator}) \end{aligned}$$

The use of the `hds` and `hp` roles allows to derive that, for instance, `SeqAna` is subsumed by

$$\texttt{myFragment} \doteq \exists \texttt{hp}.(\texttt{RemRedDNA} \sqcap \exists \texttt{hds}.\texttt{BLAST})$$

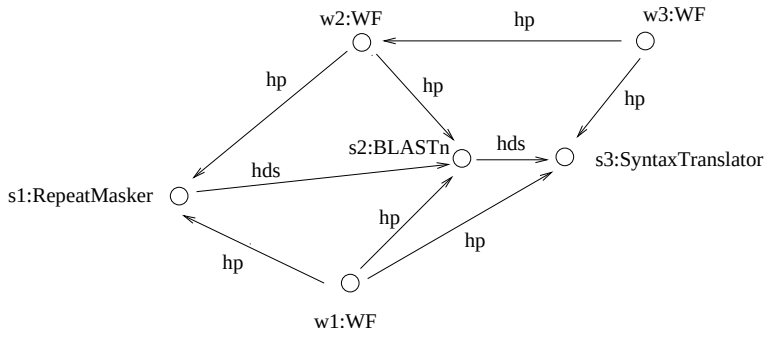

Figure 3: Contents of the A-Box without the hs roles, hp transitivity and inverses

We also wish to be able to deduce when one service (or piece of data) succeeds another even when a few services are in between (Q3). Introducing a role hierarchy has the desired effect. We define the *has successor* role `hs` to be transitive and a super role of `hds` by adding $\texttt{hds} \dot{\sqsubseteq} \texttt{hs}$ and $\texttt{Trans}(\texttt{hs})$ to the role hierarchy.

Moreover, it would be logical to expect that if a service, Service 1, is followed by a workflow fragment of which Service 2 is the first service, then Service 1 is succeeded by Service 2. To model such a derivation in a clean way, we would need to say that any *has successor* relationship between A and B followed by a *has part* relationship between B and C implies a *has successor* between A and C. This type of derivation can be achieved by means of a complex role inclusion axiom [8]. Unfortunately the role composition construct is unavailable in OWL Lite (and OWL DL). We approximate the desired inference by making the *has part* role `hp` as a sub role of has successor `hs`. It remains to be seen in how far this approximation yields undesired effects. Alternatively, because we know which roles will be combined before querying time, we could introduce a preprocessing step and enumerate all role assertions ourselves explicitly.

Finally, the *has direct precursor, has precursor*, and *is part of* roles are modelled as the respective inverse roles of `hds`, `hs` and `hp`.

A-Box *Service instances* and *concrete workflows* in the A-Box instantiate the service classes and abstract workflows from the T-Box. As a guiding principle, abstract workflows are used for *structuring* the ontology, whereas concrete workflows are used for *query answering*. Concrete workflows are used for the annotation of snippets of working code, and this is what the user is interested in. Figure 3 shows part of an A-Box containing three concrete workflows and three service instances, based on the T-Box defined earlier.

4 Querying workflows and workflow fragments

With the T-Box and A-Box described so far, we now demonstrate some of the queries (relating to Q1, Q2 and Q3) that can be answered based on the representation developed in the previous section. We have used Racer[1] to support retrieval of fragments and follow the syntax of [5]. We then consider the querying of incomplete workflows (Q4), as well as inexact fragment retrieval (Q3).

Example queries for workflow fragments

Find the workflows that analyse a non-redundant sequence for similarity and then manipulate the results. Both queries are acyclic conjunctive. They return w1 and w3.

$$\begin{array}{lll} (\mathtt{w}) & :- & \mathtt{WF}(\mathtt{w}), \mathtt{hp}(\mathtt{w}, \mathtt{w}'), \mathtt{BLASTNRSeq}(\mathtt{w}'), \mathtt{hds}(\mathtt{w}', \mathtt{s}), \mathtt{Mediator}(\mathtt{s}) \\ (\mathtt{w}) & :- & \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{y}), \mathtt{hds}(\mathtt{s}, \mathtt{t}), \mathtt{hds}(\mathtt{t}, \mathtt{y}), \\ & & \mathtt{RemRedDNA}(\mathtt{s}), \mathtt{SeqSimSearch}(\mathtt{t}), \mathtt{Mediator}(\mathtt{y}) \end{array}$$

Which workflows contain a service that does similarity search and a service that removes redundant information in DNA (the first query returns w1, w2 and w3)? *Which workflows connect these 2 services* (the next query yields w1, w2 and w3)?

$$\begin{array}{lll} (\mathtt{w}) & :- & \mathtt{WF}(\mathtt{w}), \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{t}), \mathtt{SeqSimSearch}(\mathtt{s}), \mathtt{RemRedDNA}(\mathtt{t}) \\ (\mathtt{w}) & :- & \mathtt{hp}(\mathtt{w}, \mathtt{s}), \mathtt{hp}(\mathtt{w}, \mathtt{t}), \mathtt{hs}(\mathtt{s}, \mathtt{t}), \mathtt{RemRedDNA}(\mathtt{s}), \mathtt{SeqSimSearch}(\mathtt{t}) \end{array}$$

Note the use of the hs role to indicate that intermediate links between s and t can be present. Neither Racer, the Manchester and Stanford OWL-QL implementations,[2] nor Pellet of U. Maryland[3] support the retrieval of the trace between role relations, *i.e.* to return the intermediate role relations linking two services or pieces of data. Postprocessing the returned role assertions with a shortest path algorithm solves the issue.

Querying for incomplete workflows

The open world semantics of description logics allows to query for incomplete workflows (Q5). Suppose that, in the example in Figure 2, a developer of a workflow has decided that a mediator service will be used, but has yet to decide where to put this service relative to the other services in the workflow. Even though the description is incomplete, it would still be of interest to other developers who are interested in the same type of mediation and thus it is useful to be able to publish and query such incomplete knowledge.

[1] Web site: www.sts.tu-harburg.de/~r.f.moeller/racer/

[2] Web sites: www.cs.man.ac.uk/~glimmbx/ and onto.stanford.edu:8080/owql/FrontEnd

[3] Web site: www.mindswap.org/2003/pellet/index.shtml

Querying for similar workflow fragments

So far, the queries involved the exact retrieval of fragments based on A-Box retrieval. One would also like to retrieve fragments that are largely relevant to a user (Q3) but happen to fall outside a strict subsumption relationship, *e.g.* the structure of two fragments is the same, except there are two services which are not in a subsumption relationship. A mechanism is needed to measure (dis-)similarity between fragments, calculating for instance how many services are to be moved, removed, added, replaced, merged or split to relate different fragments.

DL role-based approaches and implementations relying on structural algorithms have been developed for $\mathcal{FL}^-$ in [3], which uses shared roles and role values for inexact matching, and in [1], which counts shared parent concepts. In [4] a structural algorithm based on abduction and contraction is presented for a fragment of $\mathcal{ALC}$. A tableaux algorithm for abduction and contraction based matching in $\mathcal{ALN}$ is proven in [6]. Directly applying role-based approaches on the workflow ontology and the ^{my}Grid domain ontology has not proved possible, given the constructs used in the workflow and domain ontology. In case no abduction algorithm for OWL Lite is devised, approximation [2] might offer a way out by simplifying the ontology in a non trivial way to the level of expressivity the abduction algorithm can handle. Another option is to stay within OWL Lite and devise query relaxation strategies for a query manager.

5 Outlook

Capturing data flow and more complex control flow The current representation largely ignores data flow. In future we plan to introduce data objects that have input and output relationships in the A-Box. For control flow, due to the lack of expressive power of OWL, we must abstract from the structure of the concrete workflows when they are described in the T-Box. So far, we only use `hds` to represent control flow. For the fragment discovery purposes of scientists, this is probably what one wants to do anyway: these users are not interested in intricate control flow details. There may be cases where more complex control mechanisms such as loop, conditionals or concurrency constraints need to be modelled. We can easily capture part of a conditional in the T-Box by introducing *e.g. has possible successor* roles. With respect to loops, one cannot define and query for *loops* in the abstract workflows. One is still able to query for loops over concrete workflows in the A-Box.

A hybrid approach to repurposing It is clear that, when building workflows, we are often confronted with various problems that can (and should) be solved "syntactically" such as versioning support (Q6 and Q7). Moreover, enacting workflows generates data points which people query for. It seems sensible

to represent such objects not in the A-Box but in a database, as in effect one has generated everything there is to know about a particular enacted workflow and no extra inferences can be drawn.

References

[1] S. Bechhofer and C. Goble. Classification Based Navigation and Retrieval for Picture Archives. In *IFIP WG2.6 Conference on Data Semantics, DS8*, 1999.

[2] S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *KR2002*, pages 203–214, San Francisco, USA, 2002.

[3] J. Bullock and C. Goble. Tourist: the application of a description logic based semantic hypermedia system for tourism. In *9th ACM conference on Hypertext and Hypermedia*, pages 132–141, 1998.

[4] A. Cali, D. Calvanese, S. Colucci, T. Di Noia, and F. M. Donini. A description logic based approach for matching user profiles. In *DL2004*, 2004.

[5] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *17th ACM SIGACT SIGMOD SIGART Symp PODS*, pages 149–158, 1998.

[6] S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. A uniform tableaux-based approach to concept abduction and contraction in ALN. In *DL2004*, 2004.

[7] I. Horrocks, P. Patel-Schneider, and F. van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1), 2003.

[8] I. Horrocks and U. Sattler. Decidability of SHIQ with Complex Role Inclusion Axioms. In *IJCAI-03*, 2003.

[9] U. Keller, R. Lara, A. Polleres, et al. Wsmo web service discovery. WSML Working Draft D5.1 v0.1, University of Innsbruck, 12 November 2004.

[10] R.D. Stevens, H.J. Tipney, C.J. Wroe, T.M. Oinn, M. Senger, P.W. Lord, C.A. Goble, A. Brass, and M. Tassabehji. Exploring Williams Beuren Syndrome Using ^{my}Grid. *Bioinformatics*, 20:303–310, 2004.

[11] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the WWW*, 1(1):27–46, 2003.

[12] C. Wroe, R. Stevens, C. Goble, A. Roberts, and M. Greenwood. A suite of daml+oil ontologies to describe bioinformatics web services and data. *Intl. J. of Cooperative Information Systems*, 12(2):197–224, 2003.