Small Is Beautiful

DOI:
10.1109/BigData.Congress.2013.49

Document Version
Accepted author manuscript

Small Is Beautiful: Summarizing Scientific Workflows Using Semantic Annotations

Pinar Alper, Khalid Belhajjame, Carole Goble
School of Computer Science
University of Manchester, Manchester, UK
name.surname@cs.manchester.ac.uk

Pinar Karagoz
Department of Computer Engineering
METU, Ankara, TURKEY
karagoz@ceng.metu.edu.tr

***Abstract*—Scientific Workflows have become the workhorse of BigData analytics for scientists. As well as being repeatable and optimizable pipelines that bring together datasets and analysis tools, workflows make-up an important part of the *provenance* of data generated from their execution. By faithfully capturing all stages in the analysis, workflows play a critical part in building up the audit-trail (a.k.a. provenance) metadata for derived datasets and contributes to the veracity of results. Provenance is essential for reporting results, reporting the method followed, and adapting to changes in the datasets or tools. These functions, however, are hampered by the *complexity of workflows* and consequently the complexity of data-trails generated from their instrumented execution. In this paper we propose the generation of *workflow description summaries* in order to tackle workflow complexity. We elaborate reduction primitives for summarizing workflows, and show how primitives, as building blocks, can be used in conjunction with semantic workflow annotations to encode different summarization strategies. We report on the effectiveness of the method through experimental evaluation using real-world workflows from the Taverna system.**

***Keywords*-Scientific Workflow; Annotation; Rule-Based Summarization; Motif**

I. Introduction

In this age of data-intensive science we're witnessing the unprecedented generation and sharing of large scientific datasets, where the pace of data generation has far surpassed the pace of conducting analysis over the data. Scientific Workflows [6] are a recent but very popular method for task automation and resource integration. Using workflows, scientists are able to systematically weave datasets and analytical tools into pipelines, represented as networks of data processing operations connected with dataflow links. (Figure 1 illustrates a workflow from genomics, which "from a given set of gene ids, retrieves corresponding enzyme ids and finds the biological pathways involving them, then for each pathway retrieves its diagram with a designated coloring scheme").

As well as being automation pipelines, workflows are of paramount importance for the *provenance* of data generated from their execution [6]. Provenance refers to data's derivation history starting from the original sources, namely its lineage. By explicitly outlining the analysis process, workflow descriptions make-up a prospective part of provenance. The instrumented execution of the workflow generates a veracious and elaborate data lineage trail outlining the derivation relations among resultant, intermediary and initial data artifacts, which makes up the retrospective part of provenance. The systematic workflow-based organization of analysis steps and the faithful collection of provenance is important for activities such as **1) Reporting:** Scientists are often expected to report on the methods they followed and their resultant datasets and the base/origin data they used in academic papers describing findings. **2) Handling Change:** By having an explicit specification of the analysis process and data dependencies, scientists could inquire on the impact that a change of a particular input or tool will have on results and which parts of the experiment would need to be re-run. These potential ways of exploitation, however, are hampered by the complexity of Scientific Workflows.

Workflows are typically complex artifacts containing several processing steps and dataflow links. Complexity is characterized by the environment in which workflows operate. Workflows that collate datasets and tools from 3rd-party autonomous providers contain several adapter steps for handling format and protocol heterogeneities [9][8]. Meanwhile workflows that perform analytics over local(ized) data typically contain several steps dedicated to organization and pre-processing of data. We have shown in a previous study [8] that these mundane operations account for up to 60% of the steps that compose workflows. These steps dealing with the technical detail of resource access or data organization obfuscate the scientifically significant/critical operations of the workflow. Obfuscation degrades reportability of the scientific intent embedded in the workflow. Complex workflows combined with the faithful collection of execution provenance for each step results in even more complex provenance traces containing long data trails, that overwhelm users who explore or query them [3], [1].

Lack of effective simplification solutions have led scientists to devise their own ways to cope with complexity. These manual abstraction solutions have the disadvantage of being hard-coded into the workflow design and may not

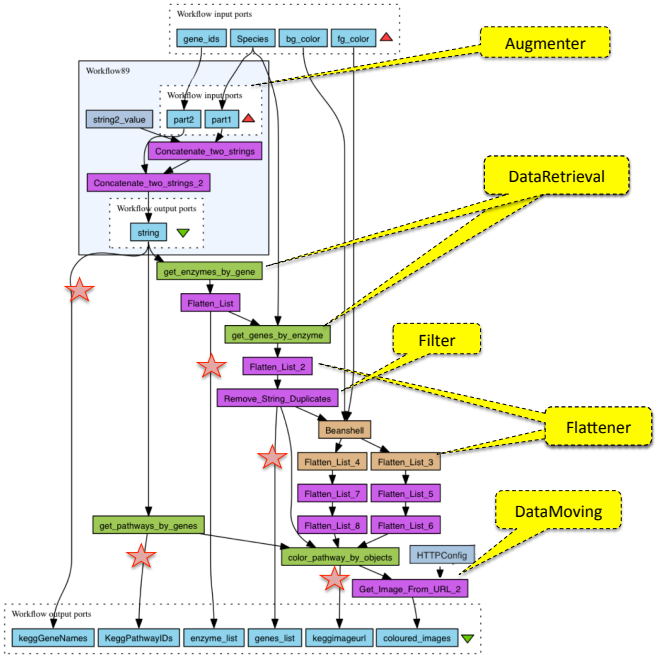

Figure 1: A Sample workflow from the Genomics Domain.

simultaneously cater for different aspects of complexity, as we shall analyze in this paper. A handful of researchers have proposed solutions to abstract workflows using user-defined views [3], or user-controlled manual redaction of provenance [4], [7]. These solutions expect the user to manually designate significant steps or directly control the editing process. Moreover, none of them take into account the nature of data-processing performed by the steps in the workflow, e.g., whether it is utility operation or whether it performs a scientifically significant function.

In this paper we propose the generation of *workflow description summaries* to tackle complexity in experiment reporting scenarios. Our approach has the following characteristics and **contributions**: We focus on data-driven workflows, which is the paradigm supported by most scientific workflow systems. We provide **a graph based model for representing data-flows**. We use graph re-writing as the method of summarization. We re-write workflow graphs with rules built on 1) semantic annotations that depict the data-oriented function of operations within workflows, which we call *motifs*, and 2) **a set of workflow reduction *primitives*** that we introduce. Rules associate motifs, and combinations of thereof, with reduction actions that specify the abstraction operation to be performed. Re-write rules capture scientist-specified strategies on how summarization should occur. Based on observations on workflows from the largest public repository of workflows, we elaborate **two sample strategies** that aim to abstract out "mundane" data adapter and organizer steps in workflows. In order to highlight the characteristic of primitives and asses our approach, we conducted an experiment where we summarized 30 real-world scientific workflows. Summarization is done by using the two sample strategies and we compare the results against summaries that are manually generated by the workflow designers as the ground truth.

The paper is organized as follows: we take a closer look at complexity and the manual ways of dealing with it in Sec-II. We then introduce scientific workflow motifs in Sec-III-A, and the model that we use to formally represent workflows in Sec-III-B. We present the re-write primitives in Sec-IV-B. We detail the two sample summarization strategies in Sec-V. We report on the results of the experimental evaluation in Sec-VI. We analyze and compare existing works to ours in Sec-VII, and conclude the paper in Sec-VIII.

II. Manual ways to deal with complexity

We observe that there are two dimensions along which workflows can be abstracted, 1) *process-wise*, where a simplified view of the process encoded in the workflow is given and 2) *data-wise*, where significant data products (intermediary or final) that a workflow generate is to generate are pinpointed. Scientists perform workflow abstractions in the following ways:

• *Grouping operations into subworkflows*. Subworkflows are a design construct for developing modular workflows, which is an established best practice in workflow development [8]. The major driver for sub-workflows is to create modules of significant or re-usable function within or across workflows. In practice, however, subworkflows can also be used for purposes other than functional modularity. For example, the workflow illustrated in Figure 1 contains a subworkflow that simply concatenates two parameters. Such subworkflows are created for aesthetic purposes to simplify the design, as opposed to abstracting significant functionality.

• *Sinking ports expected to hold significant intermediary data artifacts to workflow outputs.* The links tagged with stars in Figure 1 correspond to such sinking activity. We refer to these as "bookmarked data links" in the rest of the paper. Bookmarking is done when the user wants to report significant side-products alongside the intended outputs of the workflow, and encodes this as part of the workflow design. Hard-coded bookmarking is a poor practice for workflow design, as it clutters the output signature of the workflow. Moreover, it creates more complexity process-wise while trying to provide abstract data-wise.

• *Using libraries of components engineered to work together.* Several workflow systems provide libraries of components that have well-defined and interoperable-by-design invocation interfaces. By solely using modules from such libraries, compact workflows with little/no data adapter operations can be devised. This approach, however, is highly dependent on those curated libraries, and as soon as one tries to incorporate resources from other contexts or libraries, then one needs data adaptation and organization.

III. A Closer Look At Scientific Workflows

A. Scientific Workflow Motifs

At the heart of our approach lies the exploitation of information on the data processing nature of activities in workflows. We propose that this information is provided through semantic annotations, which we call *motifs*. We use a light-weight ontology, which we developed in previous work [8] based on an analysis of 200+ workflows (Accessible: http://purl.org/net/wf-motifs). The motif ontology characterizes operations wrt their **Data-Oriented** functional nature and the **Resource-Oriented** implementation nature. We refer the reader to [8] for the details, here we briefly introduce some of the motifs in light of our running example.

• **Data-Oriented nature.** Certain activities in workflows, such as $DataRetrieval$, $Analysis$ or $Visualizations$, perform the scientific heavy lifting in a workflow. The gene annotation pipeline in Figure 1) collects data from various resources through several retrieval steps (see "get_enzymes_by_gene" , "get_genes_by_enzyme" operations). The data retrieval steps are pipelined to each other through use of adapter steps, which are categorized with the $DataPreparation$ motif. $Augmentation$ is a sub-category of preparation operations. Augmentation decorates data with resource/protocol specific padding or formatting. (The sub-workflow "Workflow89" in our example is an augmenter that builds up a well formed query request out of given parameters.) $Extraction$ activities perform the inverse of augmentation, they extract data from raw results returned from analyses or retrievals (e.g. SOAP XML messages). $Splitting$ and $Merging$ are also a sub-category of data preparation, they consume and produce the same data but change its cardinality (see the "Flatten_List" steps in our example). Another general category is $DataOrganization$ activities, which perform querying and organization functions over data such as, $Filtering$, $Joining$ and $Grouping$. The "Remove_String_Duplicates" method in our example is an instance of the filtering motif. Another frequent motif is $DataMoving$. In order to make data accessible to external analysis tools data may need to be moved in and out of the workflow execution environment such activities fall into this category (e.g. upload to web, write to file etc). The "Get_Image_From_URL_2" operation in our genomics workflow is an instance of this motif.

• **Resource-Oriented nature**, which outlines the implementation aspects and reflects the characteristic of resources that underpin the operation. Classifications in this category are: $Human-Interactions$ vs entirely $Computational$ steps, $Statefull$ vs $Stateless$ Invocations and using $Internal$ (e.g. local scripts, sub-workflows) vs $External$ (e.g. web service, or external command line tools) computational artifacts. In our example, all data retrieval operations are realized by external, stateless resources (web services), whereas all data preparation operations are realized with internal, stateless resources (local scripts).

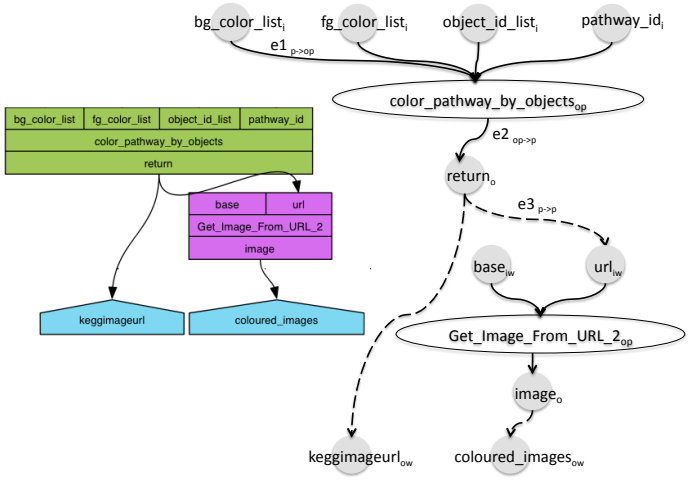

Figure 2: The graph based representation of operations and their ports.

In our evaluation we annotated the workflow corpus manually using the motif ontology. The effort that goes into the design of workflows can be significant (up to several days for a workflow). When compared to design, the cost of annotation is minor (as it amounts to single attribute setup per operation). Annotation can be (semi)automated through application of mining techniques to workflows and execution traces. It is also possible to pool annotations in service registries or module libraries and automatically propagate annotations to operations that rare re-used from the registry. Instead of the motif ontology it is also possible to use other ontologies [10] to characterize operations in workflows.

B. Graph Model for Scientific DataFlows

We address "pure" data flows, i.e. those, in which the order of execution is determined entirely by the availability of data rather than any explicit control-flow constructs (e.g. loops, conditionals).

Definition 1. We define a data-driven workflow as a directed acyclic graph (DAG) using the following pair $W = \langle N, E \rangle$, where N is a set of nodes representing the operations that constitute the workflow and the input and output ports of such operations: $N = (N_{OP} \cup N_P)$. N_{OP} nodes represent analysis operations in the workflow.

The set of ports N_P is composed of four sets, depending on whether the port is an input or output and whether it belongs to an operation or is a source/sink port of the entire workflow: $N_P = (N_I \cup N_O \cup N_{IW} \cup N_{OW})$.

Nodes are connected using three types of edges: $E = (E_{P->OP} \cup E_{OP->P} \cup E_{P->P})$, where $E_{P->OP}$ connect input ports to the operations they belong to, $E_{OP->P}$ connect operations to their output ports, and $E_{P->P}$ connect the output port of an operation to the input port of a following operation.

In Figure 2 we provide a mapping from the fragment of our genomics workflow to the graph representation, depicting operation nodes (white ellipses), port nodes (gray circles). (Not depicted in this figure, is an inferred influence relationship from each input of an operation to each output the operation. This relation is not utilized in graph re-writing but is used to query workflow descriptions and their summaries).

Definition 2. A data-flow decorated with motif annotations is 3-tuple: $W = \langle N, E, motifs \rangle$, where $motifs$ is a function that maps each graph node $n \in N$, be it an operation or port, to its motifs. Operation nodes could have motif annotations from *operation*, *resource* perspectives, and port nodes can have annotations from the *data* perspective (omitted in this paper due to space limitations).

As an example, we illustrate below two motif annotations of two operations in our running example. It is worth mentioning that a given node can be associated with multiple motifs.

$motifs(color\_pathway\_by\_objects) = \{\langle m_1 : Augmenter \rangle\}$
$motifs(Get\_Image\_From\_URL\_2) = \{\langle m_2 : DataRetrieval \rangle\}$

IV. Workflow Summarization

A. Overview of Summarization Rules

Motifs are used in conjunction with reduction primitives to build summarization rules. The objective of rules is to determine how to reduce the workflow when certain motifs or combinations of motifs are encountered. We propose graph transformation rules of the form $L \rightsquigarrow P$, where L specifies a pattern to be matched in the host graph specified in terms of a workflow-specific graph model and motif decorations and P specifies a workflow reduction primitive to be enacted upon the node(s) identified by the left-hand-side L. Primitive P captures both the replacement graph pattern and how the replacement is to be embedded into the host graph. Our approach is a more controlled primitive based re-writing mechanism, rather than a fully declarative one (as in traditional graph re-writing). This allows us to: *1)* Preserve data causality relations among operations together with the constraints that such relations have (e.g. acyclicity for Scientific Datalows) *2)* We can allow the user to specify high-level reduction policies by creating simple <motif, primitive> pairs. It would be a big undertaking to expect the users, who are not computer scientists or software engineers, to specify re-writes in a fully declarative manner (i.e. a matching pattern, a replacement pattern and embedding info).

B. Reduction Primitives

Reduction primitives are the second building block of rules, here we outline these primitives namely *Composition, Collapse* and *Elimination*. For each of these reduction primitives, we specify how the motif annotations are propagated when applying the primitive (which we call *motif behavior*),

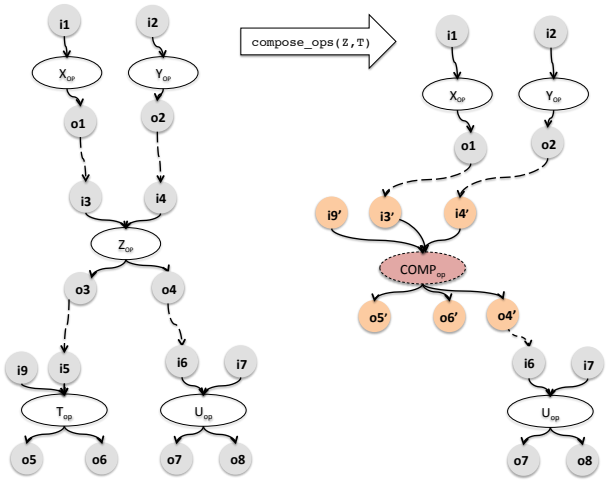

Figure 3: The graphical depiction of the composition primitive.

we specify the constraints that must be satisfied, and discuss the characteristics of the data flow generated as a result of the application of the primitive.

1) Composition: We define the function $compose\_ops : \langle W, op1, op2 \rangle \rightarrow \langle W', M \rangle$ as one, which takes an annotated workflow graph, and replaces operations $op1$ and $op2$ with their composite $op3$ and returns the updated workflow graph. The primitive also returns a set of mappings between the ports of the resulting workflow and the input workflow.

Motif Behavior: The composite $op3$ inherits motifs of both of its constituents. Operation nodes are identified by their labels, in the composition primitive the label for the composite is a newly generated identifier.

Constraint Checks: Operations that have a direct datalink among each other can be subject to composition. To prevent the creation of cyclic summaries, the two operations to be composed shall not co-exist on another indirect data flow path other than the direct link(s) connecting them.

Dataflow Characteristic: The composition primitive may result in loss of information on negative influence relations among artifacts. For example, in Figure 3, in the original workflow the input $i9$ is influential in the creation of outputs $o5$ and $o6$ but not in the creation of $o4$. When operations Z and T are composed however, we will have the information stating that $i9'$ which maps to $i9$ is influential in the creations of not only $o5'$ $o6'$ but also of $o4'$.

2) Elimination: We define the function $eliminate\_op : \langle W, op \rangle \rightarrow \langle W', M \rangle$ as one, which takes an annotated workflow definition, and eliminates a designated operation node from it. Elimination results in a set of **indirect** data links connecting the predecessors of the deleted operation to its successors. The source and target ports of these new indirect links are the cartesian product of the to-be-deleted operation's inlinks' sources and its outlinks' targets. As with other primitives elimination also returns a set of mappings between the ports of the resulting workflow and the input

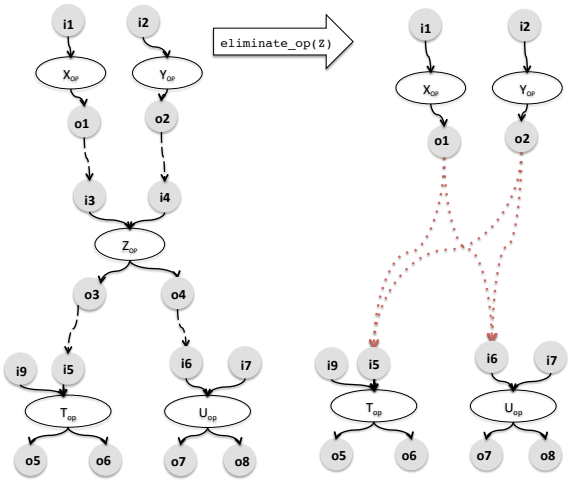

Figure 4: The graphical depiction of the elimination primitive.

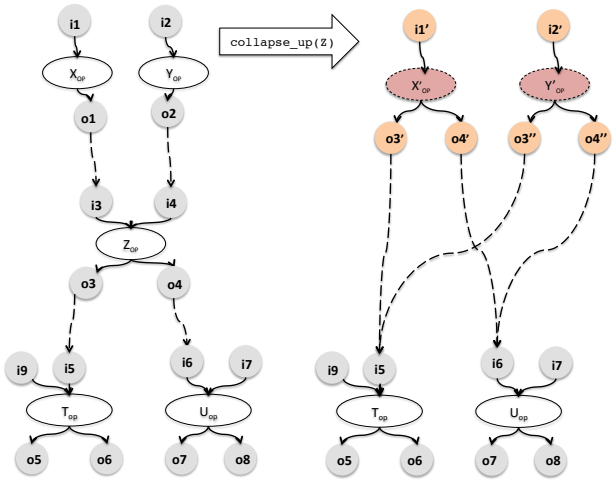

Figure 5: Graphical depiction of the Collapse-Up primitive.

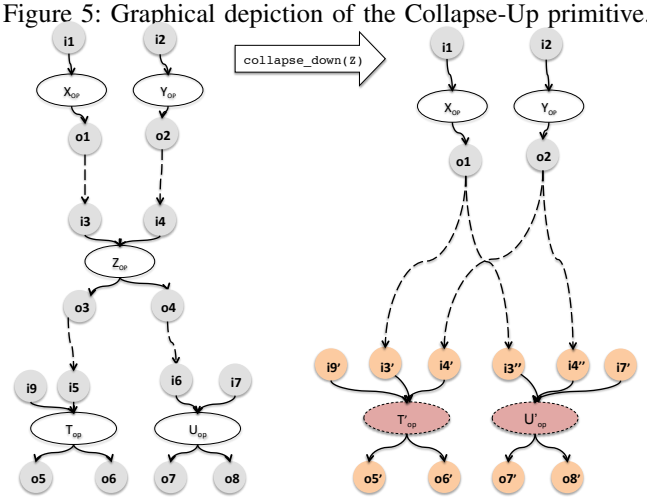

Figure 6: Graphical depiction of the Collapse-Down primitive.

workflow.

Motif Behavior: The designated operation is eliminated together with its motifs and these are not propagated to any other node in the workflow graph.

Constraint Checks: There are no particular constraint checks for the elimination primitive.

Dataflow Characteristic: Within the conventional definition of a workflow, ports are connected with direct datalinks, which correspond to data transfers i.e. the data artifact appearing at the source port is transferred to the target port as-is. In the indirect link case, however, there is no direct data transfer as the source and target artifacts are dissimilar, and some "eliminated" data processing occurs inbetween.

3) Collapse: The collapse primitive is a way to eliminate an operation from the workflow not by introducing indirect data links, but by introducing (sometimes redundant, therefore weak) composites. Collapse can be performed either upstream or downstream. We define the function $collapse\_op\_downstream : \langle W, op\rangle \rightarrow \langle W', M\rangle$ as one, which takes an annotated workflow definition, and collapses the designated operation *op* into all operations that immediately succeed it (a.k.a. collapse hosts) via a direct dataflow link. Collapse is realized by creating composites of *op* exclusively with each of its hosts (i.e. ignoring relations with other hosts). Upstream collapse is realized by creating composites of *op* with its immediate predecessors.

Motif Behavior: The operation that is collapsed onto another does not carry forward its motifs to the newly generated composite. Whereas the motifs of the operation onto which *op* is collapsed, i.e. the host propagates its motifs to the composite. Also, the composite inherits the identifier label of the host operation.

Constraint Checks: In order for an operation to be collapsed downstream it is required that it is succeeded by at least one operation. The reverse constraint is applicable when collapsing upstream. To prevent against creating cyclic summaries, the host operations on which an operation is to be collapsed shall not co-exist on a data-flow path.

Dataflow Characteristic: In case there are more than one host operations, on which collapse will occur the resulting composites are "weak" due to the exclusivity of compositions. Collapsing onto multiple hosts creates a redundancy of ports on generated composites and causes the "spreading" of influence relations, where a weak-composite does not provide the full picture of influence relations. For example, in Figure 5, the operation Z is collapsed upstream onto X and Y. The newly created composites X' and Y' are weak in the sense that they separately provide information on **some but not all of the inputs** that have influenced the generation of e.g. output $O3'$ and $O3''$ which both map to $O3$.

V. Sample Summarization Strategies

In this section, we present two sample strategies for summarizing workflow graphs. Such strategies are dependent on the domain in which workflows exist, the characteristics of workflows and the user preferences on summary. Strategies

in this section are devised by observing Taverna workflow descriptions. These open-world workflows mainly collate datasets and tools from 3rd party autonomous resource providers. Observations on Taverna workflows and their textual narratives supplied by users pointed that:

• Scientists rarely report on data adapter steps within their brief textual summaries of workflows. They choose to mention steps that they deem significant, from a scientific perspective e.g. Data Retrieval, Analysis, Visualization etc. in their summaries.

• Scientists also report on the intermediary Inputs and Outputs of significant steps. This is done either by mentioning data artifacts in text summary or manually bookmarking them to workflow outputs. Bookmarking, however, is done at stages of data processing, where data is in its the most _reporting friendly_ form such as 1) Data with Reduced Cardinality and 2) Data that is stripped of Implementation Specific Payload. Consequently, we elaborate the following strategies:

1. Summary-By-Elimination: Requires minimal amount of annotation effort and operates with a single rule. We have only classified the scientifically significant operations to denote their motifs, and all remaining operations in the workflow are designated to be $DataPreparation$ steps. The following reduction rule is then used to summarize the workflow graph W.

If $\exists$ path p in W where $p = op_A$ and $motif(op_A)$ contains $< m1 : DataPreparation >$, then $eliminate\_op(W, op_A)$

2. Summary-By-Collapse: Requires annotation of workflow with motif classes having more specificity and a more fine-grained set of rules. Consequently annotations on data preparation steps are more specific designating what kind of preparation activity is occurring (e.g. $Merging$, $Splitting$, $Augmentation$ etc). We use these annotations to remove those data preparation steps from the workflow by the collapse primitive and collapsing in the direction so as to retain the ports that carry more reporting friendly artifacts. We collapse $Augmentation$, $Splitting$ and $ReferenceGeneration$ operations downstream as their outputs are less favored than their inputs, whereas we collapse $Extraction$, $Merging$, $ReferenceResolution$ and all $DataOrganization$ steps downstream as their outputs are preferred over inputs in a summary. Two rules in this approach are given below:

For example, In Figures 7 and 8 we provide the result of applying the two summarization rules to our sample genomics workflow. Both strategies result in a process-wise compact version of the workflow that contains only the Data Retrieval operations that have survived elimination after exhaustive execution of the rules. The summaries convey the causal ordering of these operations and their direct/indirect relations with the workflow inputs and outputs. The summaries highlight the characteristic of primitives.

If $\exists$ path p in W where $p = op_A$ and $motif(op_A)$ contains $< m1 : Augmentation >$, then $collapse\_op\_downstream(W, op_A)$
If $\exists$ path p in W where $p = op_A$ and $motif(op_A)$ contains $< m1 : Merging >$, then $collapse\_op\_upstream(W, op_A)$

Elimination causes indirect influence relationships to be generated (depicted with red links). Elimination of operations with multiple in/outlinks cause multiple indirect links (depicted with red arrows) to be generated. These indirect links convey the information that a particular input artifact (e.g. "Species"') influences the generation of another data artifact that is input to a certain operation through some hidden computation. Note that these indirect links connect two distinct data artifacts at each end. Elimination can also result in indirect links connecting workflow inputs to workflow outputs (See link connecting "Species" to "KeggGeneNames"). Elimination is fully-preserving of the fine-grained influence relationships among data artifacts, whereas collapse (and so does composition) results in the loss or scattering of this information. Consider, for example, the relations between the input "HTTPConfig" and outputs "coloured_images" and "keggimageurl". In the elimination case we know that this input has only influenced the generation of "coloured_images", whereas in the collapse case it appears to influence both of those outputs through some composite unit of computation.

Collapse results in composites (weak or regular) that embody multiple operations from the original workflow. In the case of collapsing onto multiple hosts, we get weak composites, where ports are redundantly copied over to multiple composites. Therefore tracing the data flow trail from just one of these redundant ports does not yield the complete set of data artifacts that are influenced by the artifact at this port. For example, the operation named "comp: get_enzymes_by_gene" contains two input ports that are copied of the ports of the operation named "Workflow89" in the original workflow. i.e. "Workflow89" is collapsed onto both "get_enzymes_by_gene" and "get_pathways_by_genes". Therefore, if one was using the summary to inquire about which data artifacts are influenced by the artifact that occurs at port "part1′ " of operation "get_enzymes_by_gene", then one needs to follow influence traces of all copies of that port including "part1′′" of "get_pathways_by_genes". This partial view of influence relations is more dramatic in the case of collapsing up to multiple hosts, where only a subset of inputs that have influenced the generation of an output is reported to us by a weak composite.

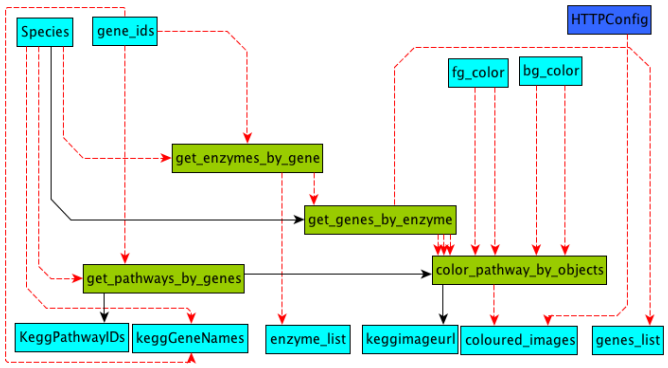

Figure 7: Results of summarization by elimination.

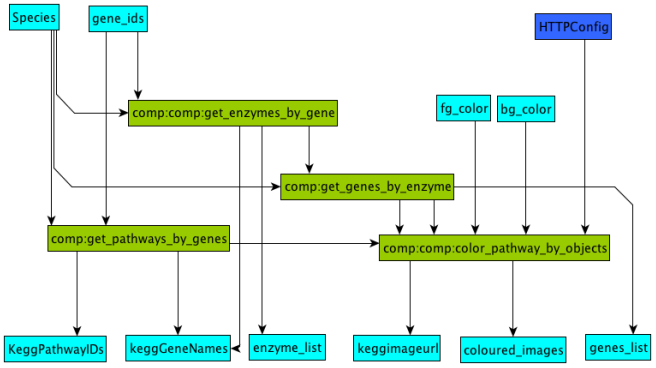

Figure 8: Results of summarization by collapse.

VI. Evaluation

Summarizing Taverna Scientific Workflows: Our test dataset is comprised of 30 Taverna workflows from various domains. We applied the two strategies present in the previous section. To assess the results, we analyzed the workflow summaries obtained from a mechanistic perspective, by just examining the graphs, and from a user-oriented perspective by taking the user summaries as ground truth. We shall reiterate that the two strategies *summary-by-elimination* and *summary-by-collapse* are intended to be exemplary rather than set-in-stone solutions.

• *Mechanistic Effect of Summarization* In terms of simplification of the workflow from a process perspective, as shown in Table I, and in parallel to our previous empirical findings more than 60 percent of operations are removable from the workflow description. Also a significant number (more than 30 percent) of data links are removed or replaced with composite links. We have also measured workflow depth. i.e. the longest path that connects a workflow input to an output. Both strategies yield significant reduction in depth (more than 50 percent), yet when we look at the number of distinct data artifacts (ports) along the longest data-flow path in a workflow we observe that the *summary-by-elimination* strategy cannot equally reduce the data-wise depth of the workflow.

• *Comparison Against User Specified Summary* The my-

	By-Elm.	**By-Col.**
Avg. Decrease in (Process-Wise)# of Operations	68%	63%
Avg. Decrease in # of Links (Process-Wise)	31%	37%
Avg. Decrease in Workflow Depth (Process-Wise)	62%	57%
Avg. Decrease in Workflow Depth (Data-Wise)	33%	57%

Table I: Table shows the reduction in workflow artifacts as a result of the summarization in two strategies.

Experiment repository provides users the ability to associate textual summaries with published workflows and most workflows bear these descriptions. Textual summaries used to describe the overall scientific intent of the workflow, its input and output artifacts. Of the workflows we analyzed 23 of them had some basic form of narrative text of the process followed and referred the significant operations in it. For these, we have recorded the mentioned operations as "significant" and expected to be included in a workflow summary. Among the dataset, 8 workflows had the link bookmarking in their design. We recorded the bookmarked ports as "significant" and expected to be included in a workflow summary.

Using these two sets of information as ground-truth, the precision and recall values of summaries are given in Table II. We observe that both the *summary-by-elimination* and *summary-by-collapse* strategies are quite effective in generating summaries of the workflow from the process-perspective. This is encouraging, especially for the case of *summary-by-elimination* as it shows how much summarization can be achieved with very basic classification of operations in workflows. These findings validate the observation we make regarding the process reporting preference of users, that within a summary they report on scientifically significant data minting operations.

For data-wise summarization, the automatic detection of bookmarks (for both strategies) is less effective. Even with the summary-by-collapse strategy where we provide fine-grained annotations we get (on average) 43% of the bookmarks right. This shows that our earlier assumptions are partially valid. Also, there are differences in the data reporting characteristics of different domains or users, e.g., while some prefer to bookmark exact XML message received from the web service (for debugging or audit purposes) others will bookmark the data artifacts extracted from the message (to be put into a paper or web page) and shows that the way we have encoded motifs into summarization rules do not fit all domains or all usage purposes. In fact the possibility to encode different summarization policies for different domains or for different users and usecases (e.g. debugging versus publishing) within the same domain is the main flexibility of our approach.

Implementation: We implemented graph re-write primitives as a set of custom Java methods. We did not use

	By-Elm. Precision	**By-Elm.** Recall	**By-Col.** Precision	**By-Col.** Recall
Process-Wise	0.74	0.92	0.65	0.93
Data Wise	0.14	0.55	0.33	0.43

Table II: Precision Recall values of summaries generated by two summarization strategies against user summaries.

a declarative, graph re-write code-generator framework. The reasons being: 1) our primitives are defined over a typed/restricted graph model specific to data flows, 2) the scale we operate over, in terms of workflow graphs, is in the high hundreds as we perform summarization at the description level rather than execution trace level. For pattern matching, i.e. left hand side of rules we used SPARQL queries that are executed over an annotated and abstract representation of the workflow description using the wfdesc workflow model [2].

VII. Related Work

Abstracting workflow provenance has received attention in the recent years. One common approach to abstraction is the generation of views on top of the provenance graph [3] [5]. In [3] authors create views by automatically partitioning the workflow graph, where each partition is built by grouping multiple activities into composites. Grouping is centered around "user-specified" significant activities. The abstract workflow is then used to scale-down results of data lineage queries. View based systems typically provide drill-down/up capability to transition between the view and the actual underlying provenance graph.

In [7] a user-based provenance graph editing and publishing approach is described which takes motivation from data privacy preservation. Users are provided with a set of provenance graph customization primitives (e.g. hide, anonymize, abstract and retain nodes). Given that users have full control of editing, they could possibly violate data-dependency integrity of the provenance graph. The system encodes possible integrity violations as policies and reports and resolves (to the extent allowed by user) conflicts among customization actions, and policies. Also targeted for data privacy, in [4] authors describe user-driven redaction of provenance approach. A set of redaction primitives are provided such as edge and node contraction, node relabeling, and a graph rewriting approach is proposed that allows users to specify sub-graphs of interest (using a graph query language with regular expressions) and also the corresponding redaction action to be performed.

The existing approaches to provenance customization give the user the full responsibility of searching and editing the provenance graph. Given the complexity of workflows and provenance it is a big expectation from the users to perform this editing without assistance. Therefore, we took motivation for an indirect approach to summarization.

VIII. Conclusions and Future Work

In this paper, we presented a novel solution for creating workflow description summaries using a rule-based approach that acts on patterns of semantic annotations on workflow graphs and re-writes the workflow with well-defined primitives, namely Composition, Collapse and Elimination. We have encoded two sample summarization strategies, which are grounded onto empirical observation that we made by analyzing scientific workflows. Using a test set of Taverna workflows annotated with motifs the effectiveness of summarization strategies is shown against user summaries.

Summaries can find usage in several areas, one is the retrospective querying and browsing of workflow execution traces in a simplified user-friendly manner. Currently we're experimenting on the utility of summaries in provenance querying and experiment reporting (Using Taverna and other workflows system's provenance). As yet, we have not defined the inverses of the proposed primitives. Primitives such as Decompose, and Expand, can be devised and can allow for new ways of interactively browsing of workflows. Another, prospective use is to treat summaries as bookmarks for selective collection of execution traces, which is known to introduce serious overhead to workflow runs.

Acknowledgment

This work was supported by the myGrid Platform Grant (EPSRC EP/G026238/1, "myGrid: A platform for e-Biology Renewal").

References

[1] M. K. Anand, S. Bowers, et al. Provenance browser: Displaying and querying scientific workflow provenance graphs. In *ICDE*, pages 1201–1204, 2010.

[2] K. Belhajjame et al. Workflow-centric research objects: First class citizens in scholarly discourse. In *SEPUBLICA*, Crete, Greece, 2012.

[3] O. Biton, S. Cohen-Boulakia, et al. Querying and Managing Provenance through User Views in Scientific Workflows. *ICDE*, pages 1072–1081, Apr. 2008.

[4] T. Cadenhead et al. Transforming provenance using redaction. In *SACMAT '11*, pages 93–102. ACM, 2011.

[5] K. Cheung and J. Hunter. Provenance explorer; customized provenance views using semantic inferencing. In *ISWC*, pages 215–227, Berlin, Heidelberg, 2006. Springer.

[6] S. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD*, 2008.

[7] S. C. Dey, D. Zinn, et al. Propub: towards a declarative approach for publishing customized, policy-aware provenance. In *SSDBM*, pages 225–243. Springer, 2011.

[8] D. Garijo, P. Alper, et al. Common motifs in scientific workflows: An empirical analysis. In *eScience*. IEEE CS, 2012.

[9] D. Hull, R. Stevens, et al. Treating shimantic web syndrome with ontologies. In *AKT Workshop on Semantic Web Services*, 2004.

[10] M. Copeland, A. Brown, et al. The SWO Project: A Case Study for Applying Agile Ontology Engineering Methods for Community Driven Ontologies. In *ICBO*, 2012.