Keyword Search in Workflow Repositories with Access Control

Susan Davidson, Soohyun M. Lee, and Julia Stoyanovich
{susan,soohyunl,jstoy}@cis.upenn.edu

Computer and Information Science Department
University of Pennsylvania, Philadelphia, PA, USA

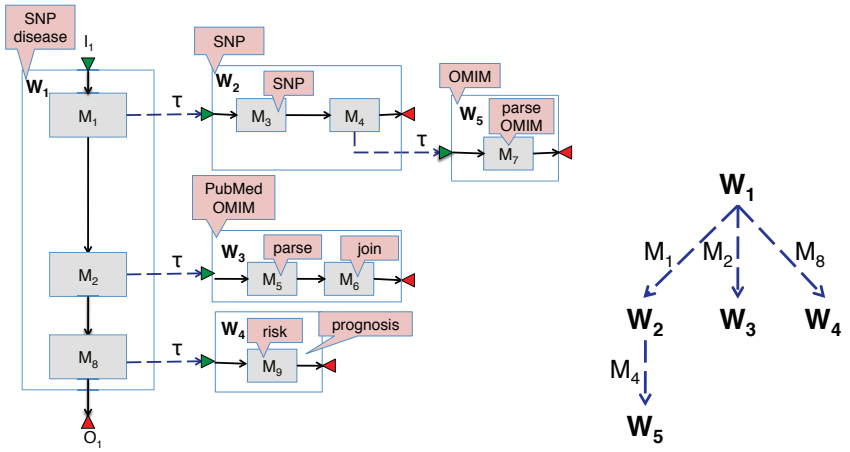
Abstract. A number of on-line repositories of scientific workflows are emerging. These repositories enable sharing and reuse of workflows, and aid in the design of new workflows. The growing size of these repositories, the complex hierarchical structure of the workflows, and the need to incorporate access control mechanisms make information discovery in these repositories an interesting challenge.
This paper formalizes keyword search in repositories of hierarchical workflows. We start by defining what it means for a single hierarchical workflow to match a keyword query while accounting for access control, and discuss options for displaying the resulting matches within the workflow. We extend this to search over workflow repositories, by proposing various ranking semantics that build on techniques from XML search and from information retrieval, and adapting them to our setting.

1 Introduction

Scientific workflows are increasingly important within the life sciences, where users deal with large amounts of data and perform sophisticated analyses. Workflows are an attractive paradigm for scientists because they make analyses repeatable and re-usable. A workflow is typically represented as a directed graph whose nodes represent analysis steps or *modules* and edges represent potential *dataflow* between modules. A module implements a certain functionality by transforming its data inputs into data outputs.

Workflow development platforms typically come equipped with libraries of pre-defined modules, which workflow designers may combine to implement a new analysis. Since workflows may be complex, sometimes consisting of hundreds of modules, it is natural to allow a sub-graph of the workflow graph – a *sub-workflow* – to be encapsulated in a composite module. A *composite module* implements the functionality of the sub-workflow, but reveals implementation details only when expanded. This naturally leads to a *hierarchical workflow* structure.

As an example, consider the hierarchical workflow in Fig. 1(a). This workflow, labeled W_1, estimates individual susceptibility to genetic disorders based on genome-wide SNP array data. The input to W_1, indicated by node I_1, is a set of single nucleotide polymorphisms (SNPs), ethnicity, lifestyle, family history, and symptoms. The output of W_1 is a prognosis, indicated by node O_1. The first



(a) Hierarchical workflow. (b) Expansion hierarchy.

Fig. 1. Running example: workflow for estimating genetic disease susceptibility.

module in W_1, named M_1, determines a set of disorders the patient is genetically susceptible to based on the input SNPs and ethnicity information. The second module, M_2, refines the set of disorders for which the patient is at risk, based on lifestyle, family history, and symptoms. Module M_8 computes the prognosis. Modules M_1, M_2, M_4, and M_8 are composite, and expand to sub-workflows that implement the required functionality (indicated by τ edges in Fig. 1(a)). The presence of such composite modules gives rise to an *expansion hierarchy*. The expansion hierarchy for W_1 is presented in Fig. 1(b).

On-line repositories of scientific workflows, such as *myExperiment.org* [11], are emerging in support of the sharing and re-use of workflows. To enable discovery of workflows of interest, several of these repositories support *tagging* — a workflow may be tagged by its author as well as by other users of the system. Workflow W_1 in our running example is tagged with "SNP" and "disease". Fig. 1(a) shows tags associated with workflows and modules as call-outs.

In a recent work [12] we considered how tags may be used to enable browsing of workflow repositories such as *myExperiment.org*. In the *browsing* scenario, the user is not fully aware of his information need; he is exploring the repository to better formulate his information need and identify items of potential interest. In this paper we shift focus, and consider how tags may be used in *search* — a task performed by a user who is well-aware of his information need, and is looking for objects that satisfy that need precisely. For example, a user looking for workflows that make disease susceptibility predictions based on SNP information may pose a keyword query $Q_1 = \{$"SNP", "disease"$\}$ against the repository, expecting that workflows such as W_1 be retrieved. If the user's task is implemented fully by some workflow in the result, he may re-use the workflow without further modification.

However, users may also want to develop new functionality, possibly re-using existing workflows as building blocks. Suppose that the user wants to build a

workflow that combines SNP-based disease susceptibility hypotheses with information about potentially affected genes, and retrieves the pathways in which these genes participate. Realizing that part of his task deals with SNP look-ups in OMIM, he may issue the query $Q_2 = \{\text{“SNP”}, \text{“OMIM”}\}$.

Q_2 matches workflow W_2, and so W_2 should be returned as an answer. However, the user may also want to understand how to re-use W_2 in the scope of a larger workflow. Therefore, it is also valuable to return W_1 as a result for Q_2, and to show how W_2 is used in the context of W_1. Understanding this *re-use context* is important since it is often easier to learn by example, e.g., how to format the input parameters and what to expect as output.

Repositories of hierarchical workflows, while giving rise to exciting re-use opportunities, also bring about interesting challenges with respect to *security*. Security considerations arise naturally in domains where data objects are attributable to individual users (their authors) and have intrinsic intellectual or commercial value. We take the point of view that any information discovery approach should be mindful of these considerations, and develop a model of search in repositories of hierarchical workflows that incorporates access control.

Note that our approach leverages user-assigned tags to identify matching workflows and modules. Tags are thought to provide the emergent bottom-up semantics in many datasets on the Social Web, and have been used for search and data exploration in, e.g., *Delicious*, *Flickr*, and *myExperiment.org*, the largest public repository of scientific workflows. Note, however, that our approach is not specific to tags, and that it can also use the top-down semantics prescribed by ontologies or controlled vocabularies instead of, or in addition to, tags.

Related Work. Our work builds on the WISE search engine over repositories of hierarchical workflows presented in [9]. Following the intuition of [9], we return a portion of the workflow that is both *informative*, in that it shows the expansion and dataflow relationships necessary to understand a keyword match, and *concise*, in that only the modules and edges necessary to understand the match are returned. However, we extend the model of [9] in several important ways. First, we formalize the ideas, leveraging hierarchical workflows with τ expansion. Second, our characterizations of informativeness and conciseness are different, in that we always make the context of the match w.r.t. the top-level workflow explicit. Third, we discuss what the result is when a query matches a workflow in multiple ways. Fourth, we incorporate access control.

Our work is also related to keyword search over tree-structured data (e.g., XML) [1, 4, 6–8, 13], as well as XML access control [2, 3]. We compare and contrast our work to these in the following sections.

This paper makes the following contributions. We formalize keyword search over hierarchical workflows, and define what it means for a workflow to match a keyword query, while also accounting for access control specifications (Section 2). We discuss various options for identifying and displaying one or more matches per workflow (Section 3). We consider search over workflow repositories, and propose various ranking semantics that build on techniques from XML search and from information retrieval, and adapt them to our setting (Section 4).

2 Model

2.1 Workflows and Keyword Search

A workflow is *hierarchical*, in the sense that a module may be *composite* and expand to a sub-workflow. We start by defining a simple workflow in which composite modules are not expanded, and then extend to a system of workflow specifications that expand composite modules.

Definition 1. *A* simple workflow *W is a labeled directed acyclic graph $G = (V \cup \{I, O\}, E)$ whose set of nodes, V, models the* modules *of the workflow. The set of edges, E, models potential* dataflow *between modules $v \in V$ as well as input to and output from the workflow ($I \rightarrow v$, $v \rightarrow O$). Every $v \in V$ must be on a path from I to O and has a module name, $name(v)$.*

For example, the box in Fig. 1(a) named W_1 is a simple workflow. The input to W_1 is a node labeled I_1, and the output of W_1 is a node labeled O_1. The workflow consists of three modules, M_1, M_2, and M_8. Each of these modules is composite, and may be expanded.

Definition 2. *A workflow S is a pair $(\mathcal{W}, \tau)$, where $\mathcal{W}$ is a finite set of simple workflows and τ is a (partial) expansion function that maps some of the module names in the workflows in $\mathcal{W}$ to simple workflows in $\mathcal{W}$.*

Note that if a module is used multiple times in a workflow, i.e., $v_1 \neq v_2$ but $name(v_1) = name(v_2)$, its τ expansion will be the same in each occurrence. From here on, we will use N to denote either a workflow or a module, M to denote a module name, $modules(S)$ to refer to the set of module names in the workflows in $\mathcal{W}$ for a workflow S, and $wf(N)$ to refer to the workflow in which N appears (if N is a module) or the workflow itself (if N is a workflow). Returning to Fig. 1(a), $\tau(M_1) = W_2$, $\tau(M_2) = W_3$, $\tau(M_4) = W_5$, and $\tau(M_8) = W_4$; $wf(W_1)=W_1$, $wf(M_1)=W_1$, and $wf(M_4)=W_2$.

Definition 2 allows a simple workflow that is part of two different workflows to have different expansion functions defined over its module names. For example, W_2 in the workflow W_1 has $\tau_1(M_4) = W_5$. However, in a different top level workflow W_0 it could be that $\tau_0(M_4)$ is not defined, or maps to a different sub-workflow. In a *repository* of workflows, we will disallow such inconsistencies.

Definition 3. *A* repository *of workflows is a set of workflows such that, for any two workflows W_1 and W_2, if $M \in$ modules$(W_1) \cap$ modules(W_2) then either (1) $\tau_1(M)$ and $\tau_2(M)$ are both undefined; or (2) $\tau_1(M) = \tau_2(M)$.*

We now define the notion of an *expansion hierarchy* for a workflow.

Definition 4. *Given a workflow $S = (\mathcal{W}, \tau)$, if M is a module name in a workflow W' and $\tau'(M) = W''$, we say that W'' is a sub-workflow of W'. The expansion hierarchy of S is the directed graph denoted $H(S)$ whose nodes are the workflows in $\mathcal{W}$, and in which there is an edge from $W' \in \mathcal{W}$ to $W'' \in \mathcal{W}$*

if and only if W'' is a sub-workflow of W'. Each edge is labeled with the module name M whose expansion creates the sub-workflow relationship. A workflow is well-formed if its expansion hierarchy is a tree.

The expansion hierarchy for our running example is shown in Fig. 1(b). Since the hierarchy is a tree, W_1 is well-formed. To simplify the discussion, we assume in the rest of this paper that all workflows are well-formed. Our results can be easily extended to expansion hierarchies that are DAGs. Extending our work to the (less common) recursive workflow definitions is an interesting open problem.

A module or workflow N is associated with a set of keywords, denoted *keywords*(N), e.g., *keywords*(W_1)= {"SNP", "disease"}, *keywords*(M_6)= {"join"}. Since the expansion of a module is its sub-workflow definition, we require that *keywords*(M)= *keywords*$(\tau(M))$ whenever $\tau(M)$ is defined, e.g., *keywords*(M_1)= *keywords*(W_2)= {"SNP"}. To reduce clutter, we omit the keywords associated with modules when they expand to a sub-workflow, and show only the keywords associated with the sub-workflow.

A keyword query Q is a set of keywords $\{k_1, ...k_n\}$. Informally, a query matches a workflow S if each k_i can be found somewhere in S.

Definition 5. *A keyword k* matches *a workflow S iff $k \in$* keywords*(S) or $k \in$* keywords*(M) for some $M \in$* modules*(S). A query $Q = \{k_1, ...k_n\}$* matches *S iff each k_i matches S.*

2.2 Access Control and Search

Inspired by conventions in Unix file systems, a workflow S has two actions associated with it: *read*, which allows an authorized user to perform a match against *keywords*(S); and *expand*, which allows an authorized user to see the structure of S, i.e., the module names and the dataflow links between the modules. Similarly, a module named M has two actions associated with it: *read*, which allows an authorized user to perform a match against *keywords*(M); and *expand*, which allows an authorized user to expand its definition to a sub-workflow.

It is helpful to think of the following analogy between workflows and modules, and Unix file system objects. A workflow corresponds to a directory, and having *expand* permissions on a workflow is analogous to having permission to enter the directory. A module corresponds to a file, and *read* access on a module is analogous to read permissions on a file. A module with a defined τ expansion corresponds to a symbolic link to a Unix directory, and having *expand* permissions on such a module is analogous to having permission to enter the directory.

In an access control specification for a workflow, *read* and *expand* privileges are associated with a set of user *groups* for each module and sub-workflow N; the special group *world* includes all users. A user u is *authorized* for an action on N iff u is in a group associated with N for that action.

We denote the set of users who are authorized to read (expand) N as *readers*(N) (*expanders*(N)). Authorizations associated with a sub-workflow are inherited by the modules that expand to it, i.e., *readers*(M)= *readers*$(\tau(M))$ and *expanders*(M)= *expanders*$(\tau(M))$, whenever $\tau(M)$ is defined.

Definition 6. *An* access control specification for a workflow S *is an assignment of* readers*(*N*),* expanders*(*N*) for each* $N \in S \cup$ modules*(*S*).*

In a repository of workflows we require that the access control specification be consistent across workflows.

Definition 7. *A repository* R *of workflows, together with their access control specifications, is* access control-consistent *iff for any user* u *and for any two workflows* W_1 *and* W_2 *in* R*, if* $M \in$ modules*(*W_1*)* $\cap$ modules*(*W_2*), then (1)* $u \in$ readers*(*M*) in both* W_1 *and* W_2 *or in neither* W_1 *nor* W_2*, and (2)* $u \in$ expanders*(*M*) in both* W_1 *and* W_2 *or in neither* W_1 *nor* W_2*.*

We also revise the notion of a *match* to take access control into account:

Definition 8. *A keyword* k matches a workflow S under access control specification A for user u *iff*

1. $k \in$ keywords*(*S*) and* $u \in$ readers*(*S*); or*
2. $k \in$ keywords*(*M*) for some* $M \in$ modules*(*S*),* $u \in$ readers*(*M*), and* $u \in$ expanders*(*W*) for every sub-workflow* W *on the path from* S *to* wf*(*M*) in the expansion hierarchy* $H(S)$*.*

A query $Q = \{k_1, ...k_n\}$ matches S *iff each* k_i *matches* S*.*

Consider again the example in Fig. 1(a), and suppose that user u has the following access privileges: $u \in$ *readers*$(W_1) \cap$ *expanders*(W_1), $u \in$ *expanders*(W_3), $u \in$ *readers*$(W_5) \cap$ *expanders*(W_5), and $u \in$ *readers*(M_5). The query $Q_1 = \{$"SNP", "parse"$\}$ matches W_1 for u, because "SNP" $\in$ *keywords*(W_1) and "parse" $\in$ *keywords*(M_5). Both keywords are accessible to u: *keywords*(W_1) is accessible because $u \in$ *readers*(W_1), and *keywords*(M_5) is accessible because $u \in$ *expanders*(W_1) $\cap$ *expanders*$(W_3) \cap$ *readers*(M_5). However, $Q_2 = \{$"OMIM"$\}$ does not match W_1 for u because $u \notin$ *readers*(W_3), preventing a match on *keywords*(W_3). Further, while $u \in$ *readers*(W_5), he cannot reach W_5 from W_1, because $u \notin$ *expanders*(W_2).

3 Result Presentation for a Single Workflow

Given a workflow that matches the query, the goal of this section is to define a query result that is *concise*, *informative*, and *non-redundant*. First, we formalize the notion of a *match assignment*, which maps query keywords to nodes in the expansion hierarchy. Next, we define a *projection* over the expansion hierarchy, which contains only the relevant portion of the hierarchy, and is a concise and informative representation of the match assignment. We then eliminate redundant projections from the set of projections for each match assignment. Finally, we show how non-redundant projections can be used to produce a query result.

Definition 9. *A* match assignment*,* μ*, is a function mapping each keyword* k_i *to the workflow or module name* $N \in S \cup$ modules*(*S*) such that* $k_i \in$ keywords*(*N*).*

Returning to our example, W_1 matches the query $Q = \{$"SNP", "OMIM"$\}$ since "SNP" $\in$ *keywords*(W_1) and "OMIM" $\in$ *keywords*(M_2), yielding a match assignment μ_1("SNP")$= W_1$ and μ_1("OMIM")$= M_2$. Importantly, keyword matches occur only at the top-level workflow (W_1 in our case) or at a module. In our example, the keyword "OMIM" is in *keywords*(W_3) as well as in *keywords*(M_2), but only M_2 will be used in a match assignment; the match for W_3 will be "caught" at M_2 since *keywords*(M_2)= *keywords*$(\tau(M_2))$ = *keywords*(W_3).

Each keyword query that matches workflow S (as per Definition 5) will generate a non-empty *set* of match assignments $\{\mu_1, ..., \mu_n\}$, since each keyword may occur in the workflow multiple times. In our running example, there are a total of nine match assignments, listed in Fig. 2(a): "SNP" is among the keywords for W_1, M_1 and M_3, and "OMIM" is among the keywords of M_2, M_4, and M_7.

Note that a match assignment may map multiple keywords to the same workflow or module N. For example, a valid match assignment for $Q = \{$"SNP", "disease"$\}$ is μ("SNP") $= W_1$ and μ("disease") $= W_1$.

In an access-controlled workflow, the set of match assignments for a user is restricted to those for which he has appropriate privileges (recall Definition 8).

Definition 10. *A match assignment* $\mu(k_1 = N_1), \ldots \mu(k_n = N_n)$ *is* valid *for user* u *in an access-controlled workflow* S *iff* $u \in$ *readers*(N_i) *and* $u \in$ *expanders*(W) *for every workflow* W *on the path from* S *to* wf(M) *in the expansion hierarchy* $H(S)$.

This notion of match is used in most approaches to querying under access control in XML, see, e.g., [3] and the strongest approach presented in [2].[1]

Having defined a match assignment in workflow S, we next need to transform it into a query result. We start by placing the match assignment into the context of the expansion hierarchy $H(S)$ (see Definition 4).

Definition 11. *Given a workflow* S *that matches query* Q*, and a corresponding match assignment* μ*, let* $R = \{$ wf(M) $\mid \exists k_i \in Q.\ \mu(k_i)=M$ $\}$ *be the* relevant *sub-workflows of* S*. Let* W_i *be the least common ancestor in* $H(S)$ *of nodes that appear in* R*. Then the* projection *of* $H(S)$ *over* R *is the tree* T *formed by the union of paths from* W_i *to* W_j *in* $H(S)$*, for all* $W_j \in R$.

Fig. 2 lists all match assignments for the query $Q = \{$"SNP", "OMIM"$\}$, and shows the corresponding projections of $H(W_1)$. Nodes and edges in the expansion hierarchy that are projected out are greyed out; nodes at which a keyword match occurred are presented in red font and denoted by a "*". Note that, while there are nine distinct match assignments for the query, there are only five distinct projections. To see why, consider the projection in Fig. 2(b) that corresponds to two match assignments μ_1 and μ_2: "OMIM" is matched by M_2 and maps to W_1 in the expansion hierarchy, and "SNP" is matched by either W_1 or M_1, both also mapping to W_1 in the expansion hierarchy.

[1] We do not consider the weaker (and nonstandard) notions discussed in [2], which allow matches to occur below "unexpandable" terms under certain conditions.

Some of the sub-workflows appearing as nodes in the projection T may not appear in R, but are necessary to show the connection between the match assignments, and so must be included to make the result *informative*. E.g., W_2 is not part of the match assignment, yet it is part of the projection in Fig. 2(e).

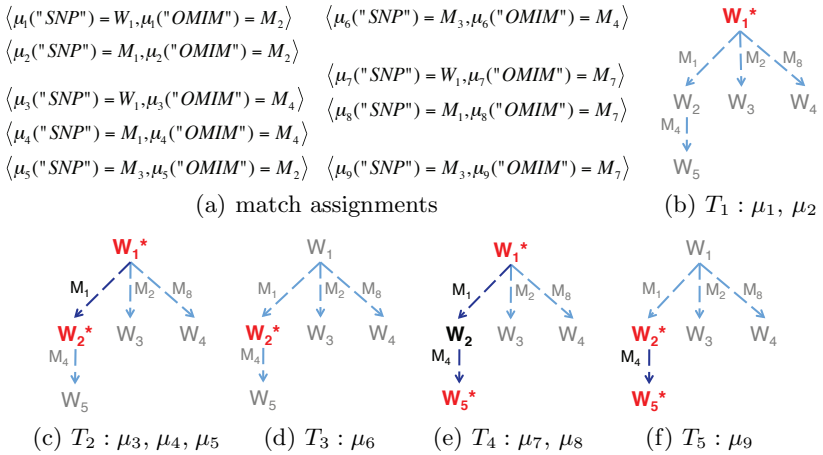


Fig. 2. Match assignments and projections of $H(W_1)$ for $Q = \{$"SNP", "OMIM"$\}$.

Projecting over the expansion hierarchy for a match assignment ultimately makes the query result *concise*, because T "trims" $H(S)$: T may be rooted at a sub-workflow of S rather than at S, trimming off top levels of $H(S)$ (e.g., Fig. 2(d) and 2(f)); subtrees of $H(S)$ rooted at leaves of T are also trimmed (e.g., all subtrees rooted at W_1 are trimmed in Fig. 2(b)).

Consider now all projections over $H(W_1)$ in Fig. 2, and note that some are more concise than others: Projection T_2 in Fig. 2(c) is rooted at W_1 but also contains W_2, while projection T_3 in Fig. 2(d) contains a complete match at W_2. In fact, T_3 is a subtree of T_2. Similarly, observe that T_5 is a subtree of T_4. We use these relationships to define *non-redundant* projections.

Definition 12. *Given query Q and workflow S, let $\mathcal{T}$ be the set of all projections of $H(S)$ for Q. A projection $T \in \mathcal{T}$ is* non-redundant *if there does not exist another projection $T' \in \mathcal{T}$, such that $T \neq T'$ and T' is a sub-tree of T.*

Non-redundant projections of $H(W_1)$ for Q are T_1, T_3, and T_5, in Fig. 2. Definition 12 is similar to that used in search and ranking over XML documents (e.g. [6, 7]), where it is referred to as "conciseness". We use "conciseness" w.r.t. a single projection, and "non-redundancy" w.r.t. a set of projections.

The node corresponding to the top-level workflow, W_1, is not always part of the projection; e.g., this is the case in Fig. 2(d). Since a user interested in re-using workflows will want to see how the relevant workflows are invoked, we need to provide *re-use context*. To improve informativeness of a result, we return a projection that is rooted at the top-level workflow.

Definition 13. *Let $T = (V, E)$ be a projection of $H(S)$ over R and let W be the root of T in $H(S)$. Then the* informative projection T' *is formed by adding to T the path in $H(S)$ from S to W.*

We are now ready to define the representation of a query result:

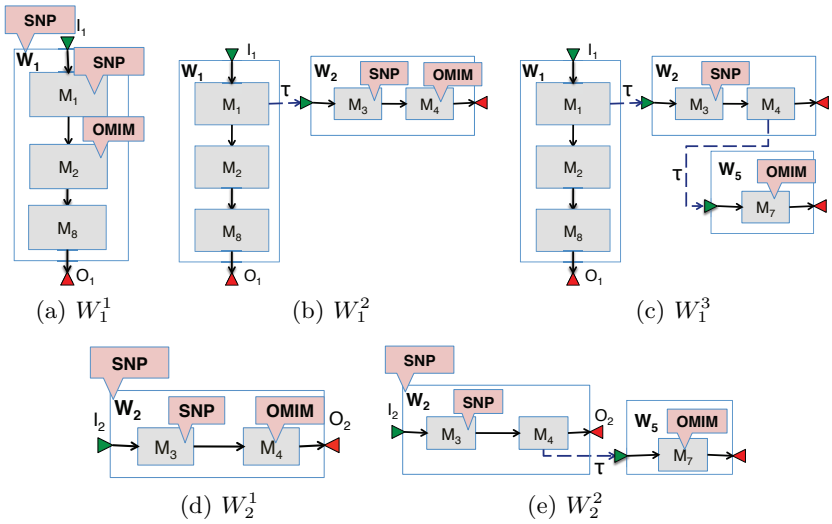
Definition 14. *Let $T = (V, E)$ be an informative projection of $H(S)$ over R for workflow $S = (\mathcal{W}, \tau)$. Then the* query result *is the specification $S' = (V, \tau')$, where $\tau'(M) = \tau(M)$ for all M that appear as labels on edges in E and undefined otherwise.*

To generate a result for a set of match assignments, we compute the set of projections $\mathcal{T}$, remove redundant projections from this set, resulting in a set of projections $\mathcal{T}'$, and then compute a query result for each $T \in \mathcal{T}'$. One option is to present these results to the user individually. For example, trees T_1, T_3 and T_5 in Fig. 2 are non-redundant, and the corresponding query results are shown in Fig. 3(a), 3(b), and 3(c). Another option is to first generate a union of the projections in $\mathcal{T}'$, and then produce a single query result for this union. Note that, because we require that all projections be informative (Definition 13), they will all have the same root — the top-level workflow S. The union of T_1, T_3 and T_5 is T_5, and the corresponding query result is shown in Fig. 4(a). Fig. 4(b) is the combined query result for W_2, which is a union of two results: one matching W_2 (top level), and another matching "SNP" at W_2 and "OMIM" at M_7.

Before proceeding to the question of result generation from a set of matches, it is worth understanding the relationship between our notion of keyword search and that of keyword search in XML trees. Since the dataflow structure of a workflow is not used in matching, one can think of the workflow as an XML tree that is formed from the workflow expansion hierarchy and also includes atomic modules as children of the sub-workflow to which they belong. Each node in the XML tree is labeled with the concatenation of its keywords; a node matches a query if its label contains one or several query terms as a substring. Similarly to our scenario, in XML keyword search, the result returned is a subtree rooted at the least common ancestor of the keyword matches, whenever the result does not have a proper subtree which is also a keyword match (i.e., is non-redundant). In our scenario, the result is further trimmed by deleting subtrees that do not contain matches (i.e., is concise). Finally, our result is augmented with the path to the root from the least common ancestor (i.e., is informative).

4 Search and Ranking in Workflow Repositories

We now focus on search in workflow repositories, where queries will likely return many matches, motivating ranking. Suppose that a workflow repository is given, and consists of five access-controlled workflows (per Definition 10) W_1, W_2, W_3, W_4, and W_5. Suppose that user u issuing query $Q = \{$"SNP", "OMIM"$\}$ is among the readers and expanders of all workflows and modules in the repository. Workflows W_1 and W_2 match Q, and one option is to present a ranked list of



(a) W_1^1 (b) W_1^2 (c) W_1^3

(d) W_2^1 (e) W_2^2

Fig. 3. Query results for workflows W_1 and W_2 for $Q = \{$"SNP", "OMIM"$\}$.

query results in Fig. 3. Another presentation option is to return the union of all query results for each workflow, and to then rank these combined results. Fig. 4 presents combined query results for W_1 and W_2. We discuss both options below.

Consider again query results in Fig. 3, and note that there is a correspondence between some of these results. In particular, Fig. 3(b) and 3(d) refer to the same actual match that occurs in W_2, but in Fig. 3(b) this match is presented in the context of W_1. A similar relationship holds between results in Fig. 3(c) and 3(e). We argued in the introduction that both types of results are of potential interest to the user. However, it is certainly reasonable to prioritize one type of a result over another in a ranked list, using a measure of quality.

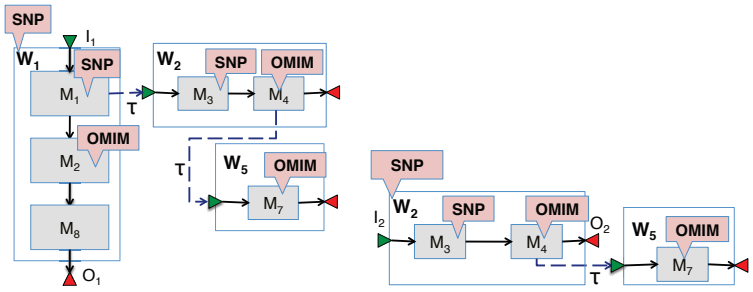
Given our conjunctive semantics of matching, all query results are exact matches, and are therefore *relevant*. However, larger results are more difficult for the user to understand, and so smaller results may be preferred. A natural way to quantify result quality is to consider the number of modules in the result.

Definition 15. *Given an access-controlled workflow S, a query Q, and a result S^i for Q in S, the* size *of S^i, denoted $size(S^i)$, is the number of modules in S^i.*

Returning to our example in Fig. 3, we have: $size(W_1^1) = 3$, $size(W_1^2) = 5$, $size(W_1^3) = 6$, $size(W_2^1) = 2$, and $size(W_2^2) = 3$. Using size as a measure of quality gives rise to the following ranking (rank position is recorded next to each entry): $(W_2^1, 1)$, $(W_2^2, 2)$, $(W_1^1, 2)$, $(W_1^2, 4)$,$(W_1^3, 5)$.

Another natural way to quantify result quality is by using hierarchical workflow structure, and we define one possible such measure below.

Definition 16. *Given an access-controlled workflow S, a query Q, and a query result S^i for Q in S, the* depth *of S^i, denoted $depth(S^i)$, is the number of τ expansions in S^i.*



(a) W_1' (b) W_2'

Fig. 4. Combined query results for workflows W_1 and W_2, for $Q = \{$"SNP", "OMIM"$\}$.

In our example in Fig. 3, we have: $depth(W_1^1) = 0$, $depth(W_1^2) = 1$, $depth(W_1^3) = 2$, $depth(W_2^1) = 0$, and $depth(W_2^2) = 1$. This gives rise to the following ranking: $(W_1^1, 1)$, $(W_2^1, 1)$, $(W_1^2, 3)$, $(W_2^2, 3)$, $(W_1^3, 5)$. Size and depth may be combined into a single quality measure, using, e.g., a (weighted) average of the two scores.

Another presentation option is to return the union of all query results for each workflow, and to then rank these combined results. Fig. 4 presents combined query results for W_1 and W_2. One natural way to discriminate between multiple combined results is to consider what portion of each result is query-relevant. We refer to this measure as *specificity*, and define it below. Our definition is in-line with the specificity measure proposed by INEX and used in XML retrieval [5].

Definition 17. *Given an access-controlled workflow S, a query Q, and a combined query result S' for Q in S, the* specificity *of S', denoted $spec(S')$, is the proportion of query-relevant modules in S' to $size(S')$.*

For the combined results in Fig. 4, $spec(W_1') = \frac{5}{6}$ and $spec(W_2') = 1$, and so W_2' would rank higher than W_1'.

Specificity treats occurrences of different keywords equally. However, there are cases when it is natural to assign higher importance to certain keywords. For example, "SNP" may be a very common keyword in the repository, while "OMIM" may be uncommon. A combined results containing many uncommon keywords should receive a high score. This semantics may be represented using, e.g., the popular *tf-idf* weighting scheme from information retrieval [10].

Keyword search and ranking over XML has been considered, e.g., [1, 4]. In [1], XML documents have a *tree* structure. Queries consist of keywords appearing at leaf nodes and of labels appearing at intermediate tree nodes. Subtrees of the document are returned, ranked on a combination of *tf-idf*-style weights and subtree structure, e.g., its size and the ancestor-descendant relationships between query-relevant nodes. While the semantics of a match are somewhat different in our case, where keywords may be assigned to tree nodes other than leaves, the ranking options of [1] may be applicable. In [4], a weight of each keyword is computed using authority propagation in *hyperlinked* XML documents. Keyword proximity is taken into account when computing the rank of a result.

5 Conclusions and Future Directions

We formalized keyword search in access-controlled repositories of hierarchical workflows. We defined what it means for a single hierarchical workflow to match a keyword query, and discussed ranking semantics for workflow repositories.

Much interesting follow-up work remains to be done. The focus of this paper was on defining the formalism, and on making connections to search and ranking in XML and in information retrieval. A natural next step is to develop efficient query processing and ranking algorithms. This is non-trivial, because of the interesting interactions between query results and access control. For example, to support *tf-idf* ranking over a repository, one may pre-compute *tf-idf* weights for all *(keyword, workflow)* pairs. In an access-controlled setting, each user may potentially have a different view over the repository, requiring that *tf-idf* weights be computed per *(user, keyword, workflow)*. This data structure may quickly become too large to compute and maintain, motivating optimizations.

6 Acknowledgments

This work was supported in part by NSF grants IIS-0803524, IIS-0629846, IIS-0915438, CCF-0635084, IIS-0904314, and by grant 0937060 to the Computing Research Association for the CIFellows Project.

References

1. S. Cohen et al. XSEarch: A semantic search engine for XML. In *VLDB*, 2003.
2. E. Damiani et al. A fine-grained access control system for XML documents. *TISSEC*, 5, 2002.
3. W. Fan, C. Y. Chan, and M. N. Garofalakis. Secure XML querying with security views. In *SIGMOD*, 2004.
4. L. Guo et al. XRANK: Ranked keyword search over XML documents. In *SIGMOD*, 2003.
5. G. Kazai and M. Lalmas. Inex 2005 evaluation measures. In *INEX*, 2005.
6. G. Li et al. Effective keyword search for valuable LCAs over XML documents. In *CIKM*, 2007.
7. Y. Li, C. Yu, and H. V. Jagadish. Schema-free XQuery. In *VLDB*, 2004.
8. Z. Liu, J. Walker, and Y. Chen. XSeek: A semantic XML search engine using keywords. In *VLDB*, 2007.
9. Z. Lui, Q. Shao, and Y. Chen. WISE: Searching workflow hierarchies. *VLDB*, 2010.
10. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
11. D. D. Roure, C. A. Goble, and R. Stevens. The design and realisation of the myExperiment virtual research environment for social sharing of workflows. *FGCS*, 25(5), 2009.
12. J. Stoyanovich, B. Taskar, and S. Davidson. Exploring repositories of scientific workflows. In *WANDS*, 2010.
13. Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In *SIGMOD*, 2005.