SPFLOW : MAKE YOUR SCIENTIFIC WORKFLOWS EASIER TO USE

CHEN J / COHEN-BOULAKIA S / FROIDEVAUX C

Unité Mixte de Recherche 8623
CNRS-Université Paris Sud – LRI

07/2013

Rapport de Recherche N° 1564

SPFlow: Make your scientific workflows easier to use

[Demonstration Proposal]

Jiuqiang Chen
LRI, CNRS UMR 8623
Université Paris Sud, France
AMIB Group, INRIA Saclay
Lanzhou University, China
chenj@lri.fr

Sarah Cohen-Boulakia
LRI, CNRS UMR 8623
Université Paris Sud, France
AMIB Group, INRIA Saclay
cohen@lri.fr

Christine Froidevaux
LRI, CNRS UMR 8623
Université Paris Sud, France
AMIB Group, INRIA Saclay
christine.froidevaux@lri.fr

ABSTRACT

Bioinformatics experiments are usually represented using scientific workflows in which tasks are chained together forming very intricate and nested graph structures. In the meantime, the number of tools available to guide users in the design and execution of workflows has increased a lot. Such tools all perform intrinsically complex operations on workflow graphs making them difficult to use on intricate structures. While scientific workflows systems deal with general graph workflows (DAGs), several tools for workflow management have chosen to impose restrictions, namely considering *series-parallel* (SP) graphs to gain both in efficiency and usability. There is thus a crucial need to provide systems able to transform any DAG workflow into an SP workflow while preserving the meaning of the original workflow: This is the goal of the SPFlow approach that we demonstrate here.
Availability: SPFlow is available for use at
http://www.lri.fr/∼chenj/SPFlow

1. MOTIVATION

Scientific workflows management systems (e.g., [10, 7]) are increasingly used to specify and manage bioinformatics experiments represented by workflows that can be shared and reused. A *workflow specification* is a framework for the execution, specifying the set of tasks to be performed and the order of tasks executions. According to the input data given to the workflow, different *workflow runs* are obtained. Both workflows and runs are represented as graphs. In this work, our assumptions are based on the behavior of most of the scientific workflow systems: runs have the same structure as the workflow specifications; there is no cycle in the structures; every workflow graph has a source s and a target t, such that every vertex is on a path from s to t; tasks are deterministic: given a set of inputs a given task will always produce the same outputs. Figure 1 provides (a) an example of workflow specification from Taverna [10], (b) its representation as a graph and (c) an example of run.

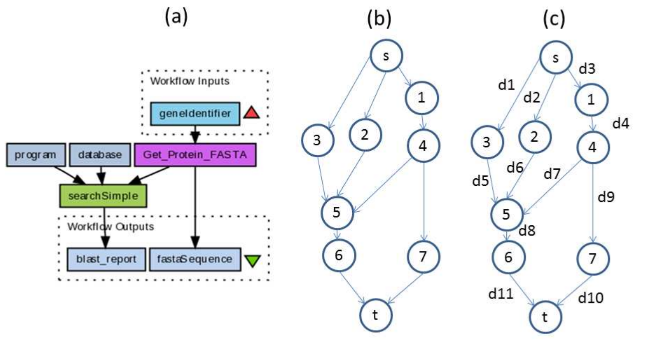


Figure 1: (a) Taverna workflow; (b) specification graph; (c) run graph

A significant number of tools have been developed to assist the design of workflows, search for the workflows available in repositories, help in the process of storing, querying and visualizing provenance information, and (re)schedule executions. These tools all make intrinsically complex operations on graph structures (e.g., search for (sub)graphs, comparing graphs) and their use may become particularly difficult when the workflow structure taken as input is too intricate. From a more formal point of view, such complex graph operations, if carried out on Directed Acyclic Graphs (DAGs) lead to NP-hard problems. Instead, these problems can be solved in polynomial time when specific restrictions are imposed, such as considering *series-parallel* (SP) structures [12].

Several approaches [3, 8] have shown that using SP workflows allows to design more user-friendly workflows and provide more efficient execution settings. Others [1, 5], in particular in the domain of provenance information management, have even chosen to restrict workflow graphs to SP structures. However, workflows obtained using popular workflow systems are DAGs with any structure. Developing an approach able to rewrite any workflow DAG into an SP graph would allow to better exploit all the workflow management tools. The rewriting should preserve the *meaning* of the workflow which is captured by the provenance of the outputs of its execution: Given some input data, the original workflow and the rewritten workflow should provide the same intermediate and final data. This is achieved by the SPFlow approach introduced in [6] that we will demonstrate in the present paper.

After a brief introduction of the workflow and provenance models used in SPFlow, we will present the architecture of the system and give a sketch for the demonstration.

2. SP WORKFLOWS AND PROVENANCE

2.1 Workflow Model

As stated in the previous section, a workflow specification is a directed acyclic labeled multigraph (dag) with one single source and one single target (st-dag). Vertices represent the workflow tasks and edges represent the *data flow* between tasks. Formal definitions are provided here.

A **workflow specification** is an st-dag $G_{spec} = (V_{spec}, E_{spec})$ where V_{spec} is a set of labeled vertices and E_{spec} is a multiset of labeled edges that are ordered pairs of vertices of V_{spec}.

A **workflow run** is an st-dag $G_{run} = (V_{run}, E_{run})$ whose source is $s(G_{run})$ and target is $t(G_{run})$. Vertices and edges are labeled using the functions $L_{vr} : V_{run} \rightarrow L_{VR}$, where L_{VR} is a set of labels for vertices, and $L_{er} : E_{run} \rightarrow L_{ER}$, where L_{ER} is a set of labels for edges. We will note $\tilde{\mathbf{x}}$ the label of vertex x, *i.e.* $L_{vr}(x) = \tilde{\mathrm{x}}$ and $\mathbf{d}_i$ the label of edge e_i, *i.e.* $L_{er}(e_i) = d_i$.

Figure 1 (b) (resp. (c)) is the graph corresponding to the workflow specification (resp. run) of Figure 1 (a).

Intuitively, SP-graphs can be easily decomposed into series and parallel components.

The class of **series-parallel graphs (SP-graphs)** is recursively defined as follows:
(i) The st-dag that contains two vertices s and t joined by a single edge is an SP-graph (**Basic SP graph**);
(ii) **Series and Parallel Composition**: let G_1 (source s_1 and target t_1) and G_2 (source s_2 and target t_2) be two SP-graphs: G obtained by identifying $s_2 = t_1$ is an SP-graph with source s_1 and target t_2 (**Series Composition**); G obtained by identifying $s_1 = s_2$, $t_1 = t_2$ is an SP-graph with source s_1 and target t_1 (**Parallel Composition**).

Determining whether a graph has an SP structure can be performed in linear time [12]. Moreover, it has been proven that an st-dag is series-parallel if and only if it does not contain a subgraph homeomorphic to the **forbidden pattern** of Figure 2 (a). Presence of vertex u in the forbidden pattern prevents from ranking the vertices within series and parallel order, making the forbidden pattern being non SP. Such a critical vertex is called a **reduction node** [2].

In Figure 1 (b, c), vertex #4 is a reduction node: such graphs are thus non SP.

2.2 Provenance model

In this work, we are interested in the meaning of the workflow as given by the provenance of its execution outputs. Two workflows have the same meaning if, given some input data, they both produce the same intermediate and final data i.e. they are *provenance-equivalent*. In this section, we will present the underlying provenance model of SPFlow.

The provenance of a data item labeling an edge is the ordered sequence of tasks performed to produce this data, and

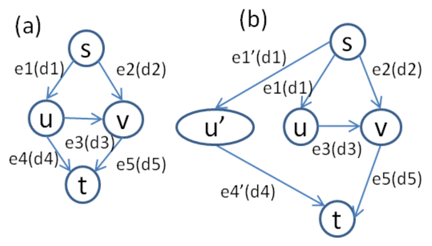


Figure 2: Forbidden pattern

input data to each task. Deep provenance describes the entire sequence of steps that produced the data [3].

Formally, let $G_{run} = (V_{run}, E_{run})$ be a run, with its sets of labels for vertices and edges. We consider regular expressions built on $L_{VR} \cup L_{ER}$, using operations "+" and "$\cdot$". Both operations are associative, "+" is commutative and "$\cdot$" is distributive over "+". Operation "$\cdot$" allows to track the succession of the tasks, while "+" denotes the alternative data paths reaching a task. We consider here the semiring $(2^{(L_{VR} \cup L_{ER})*}, \cdot, +)$ rather than the polynomial semiring [9] because the execution order must be taken into account in our context [6] (in our regular expressions, the "$\cdot$" operation is not commutative).

Let $u \in V_{run}$, $u \neq s(G_{run})$, with $L_{vr}(u) = \tilde{u}$; $f \in E_{run}$ one outgoing edge of u with $L_{er}(f) = d$; $e_i \in E_{run}$, $1 \leq i \leq p$ the incoming edges of u, with $L_{er}(e_i) = d_i$.
The **Deep Provenance** of f in G_{run} is recursively defined by the regular expression:
$DProv(f) = \tilde{u} \cdot (d_1 \cdot DProv(e_1) + \ldots + d_p \cdot DProv(e_p))$
The base case occurs when $u = s(G_{run})$ and f is an outgoing edge of s: $DProv(f) = \tilde{s}$.

Note that given a run G_{run} and a vertex u, all the outgoing edges of u have the same provenance, as they are all outputs of the same task.

Consider the graph G_r of Figure 2 (a). $DProv(e_5) = \tilde{v} \cdot (d_2 \cdot DProv(e_2) + d_3 \cdot DProv(e_3)) = ... = \tilde{v} \cdot (d_2 \cdot \tilde{s} + d_3 \cdot \tilde{u} \cdot d_1 \cdot \tilde{s})$.

Output Provenance of a Run. Given a run G_{run}, its **output provenance**, noted by $OutProv(G_r)$, is defined by the sum of the deep provenances of all the incoming edges of the target.

Continuing with Figure 2 (a), the output provenance of G_r is the sum of the provenances of e_4 and e_5. Using associativity of "$\cdot$" and "+" we get $OutProv(G_r) = (d_4 \cdot \tilde{u} \cdot d_1 \cdot \tilde{s}) + (d_5 \cdot \tilde{v} \cdot (d_3 \cdot \tilde{u} \cdot d_1 \cdot \tilde{s} + d_2 \cdot \tilde{s}))$.

We want to transform a non SP workflow run into another that has an SP structure and the same meaning as the original workflow run. We thus require the two graphs to have the same output provenance.

Provenance-equivalence Let G_{r1}, G_{r2} be two runs. G_{r1} and G_{r2} are **provenance-equivalent**, noted $G_{r1} \overset{prov}{\Leftrightarrow} G_{r2}$, iff $OutProv(G_{r1}) = OutProv(G_{r2})$.

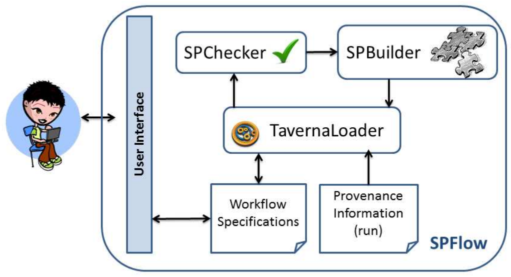


Figure 3: Architecture of SPFlow

Consider graphs G_r (a) and G'_r (b) of Figure 2 where G'_r (b) is obtained by using SPFlow on G_r (a). In G'_r: u is duplicated into u' with the same label ($\tilde{u}= \tilde{u'}$), edge e_1 is duplicated into e'_1 with the same label ($L_{er}(e_1) = L_{er}(e'_1) = d_1$), and similarly for e_4 and e'_4.

Therefore $OutProv(G'_r) = (d_4\cdot\tilde{u}\cdot d_1\cdot\tilde{s}) + (d_5\cdot\tilde{v}\cdot(d_3\cdot\tilde{u}\cdot d_1\cdot\tilde{s}+ d_2\cdot\tilde{s}))$, which is exactly $OutProv(G_r)$. Thus: $G_r \overset{prov}{\Leftrightarrow} G'_r$.

3. SPFLOW ARCHITECTURE

SPFlow transforms any workflow having a non SP structure into a provenance-equivalent SP structured workflow. The architecture of SPFlow is provided on Figure 3 and described here after. SPFlow makes use of *Workflow Specifications* and *Provenance Information* provided by users or workflow systems. The current version of SPFlow is able to rewrite real workflows from the Taverna system (other systems are under consideration). The *TavernaLoader* module is thus responsible for loading the workflow into the SPFlow internal graph structure. *SPChecker* then determines whether or not the workflow taken in has an SP structure and provides a report with graph features, including the identification of reduction nodes (if any). If the workflow is not SP, it is sent to *SPBuilder* which then creates a new provenance-equivalent workflow graph having some duplicated vertices compared to the original workflow, following the process described in [6]. Finally, the *TavernaLoader* module produces the rewritten workflow into the Taverna XML format and makes it available to the user.

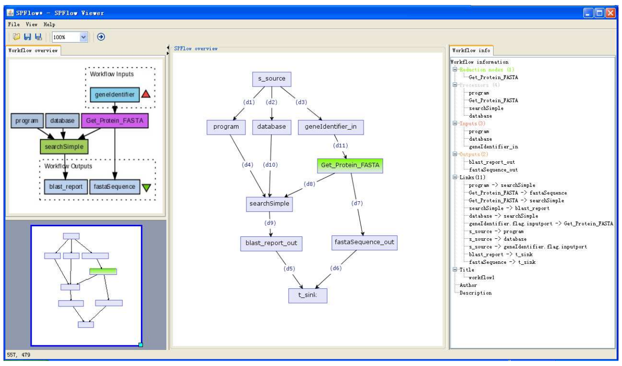
Users communicate with the system by loading and interacting with original and rewritten workflows.

4. DEMONSTRATION

Our demonstration will highlight the following features.

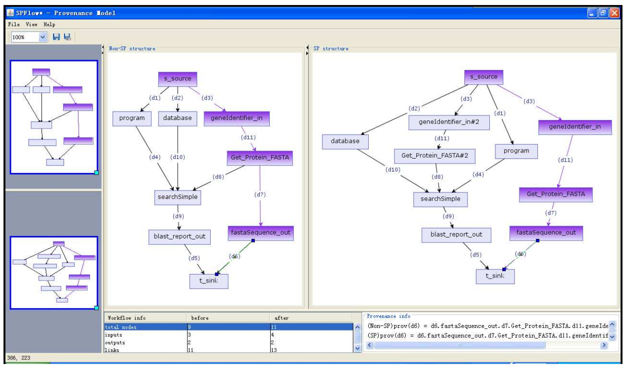
Loading Data: Users may load a workflow specification into the system (see Figure 4). SPFlow will display the original picture of the workflow from myExperiment [11] if available (left panel), determine (using SPChecker) the reduction nodes (if any) and highlight them (central panel). A report on graph features is produced (metadata on the workflow, right panel).

Rewriting of the workflow: SPFlow (using SPBuilder) transforms any non SP workflow into an SP workflow (see Figure 5). Both workflows will be displayed and duplicated vertices highlighted.



```
Workflow information
  Reduction nodes (1)
    Get_Protein_FASTA
  Processors (4)
    program
    Get_Protein_FASTA
    searchSimple
    database
  Inputs (3)
    program
    database
    geneIdentifier_in
  Outputs (2)
    blast_report_out
    fastaSequence_out
  Links (11)
    program -> searchSimple
    Get_Protein_FASTA -> fastaSequence
    Get_Protein_FASTA -> searchSimple
    searchSimple -> blast_report
    database -> searchSimple
    geneIdentifier.flag.inputport -> Get_Protein_FASTA
    s_source -> program
    s_source -> database
    s_source -> geneIdentifier.flag.inputport
    blast_report -> t_sink
    fastaSequence -> t_sink
  Title
    workflow1
  Author
  Description
```

Figure 4: Loading a workflow in SPFlow



Workflow info	before	after
total nodes	9	11
inputs	3	4
outputs	2	2
links	11	13

```
Provenance info
(Non-SP)prov(d6) = d6.fastaSequence_out.d7.Get_Protein_FASTA.d11.geneIde
(SP)prov(d6) = d6.fastaSequence_out.d7.Get_Protein_FASTA.d11.geneIdentif
```

Figure 5: Provenance information in SPFlow

Provenance information: By clicking on an edge between two tasks, the user can visualize the provenance information (see Figure 5) of the data flowing on that edge not only on the initial workflow but also on the rewritten workflow (showing that both workflows are provenance-equivalent). The formal expression associated to provenance information is also displayed (bottom panel).

Running rewritten workflows: Any workflow rewritten by SPFlow can be opened in Taverna. We will show how it can be run and we will demonstrate that both workflow versions (non SP and SP) provide the same results for the same input (equivalence property).

On the benefit of using SP-workflows: We will take the example of the Zoom*userview system (ZOOM for short) [3] that takes in a workflow and a set of tasks of interest for the user (other tasks are usually formatting tasks) and provides a *user view*, that is, a view of the workflow composed of a set of composite tasks. Each composite task contains at most one significant task and takes its meaning. The difficulty for ZOOM lies in ensuring that no data dependencies between significant composite tasks is introduced or lost by the grouping process (*i.e.* consider two relevant tasks t_1 and t_2: t_1 consumes the data produced by t_2 if and only if the composite task containing t_1 consumes the data produced by the composite task containing t_2).

In Figure 6, the user has specified two tasks of interest to him (namely, blast-report and Fasta-sequence). Based

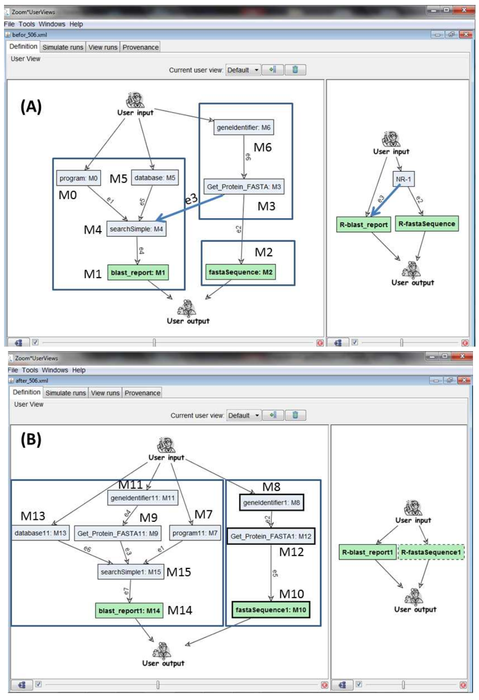


Figure 6: (A) non SP and (B) SP version of the workflow in ZOOM. User views are displayed on the right while full workflows are on the left.

on the original workflow (figure 6 (A)), ZOOM designs the user view on the right, which is composed of three composite tasks, one focused on blast-report (R-blast_report which contains M_0, M_5, M_4 and M_1), another on fastaSequence (R-fastaSequence which contains only M_2) and unfortunately one task with no significance for the user (NR-1 which contains M_6 and M_3). Note that introducing the tasks of NR-1 into one of the two significant composite tasks would have introduced misleading data dependencies: e.g., if M_3 and M_6 were put into R-fastaSequence then from the user view perspective the edge e_3 would have been displayed from R-fastaSequence to R-blast_report, giving the feeling to the user that data provided by R-fastaSequence is used by R-blast_report while it was not the case in the original workflow. It has been proved in [4] that such a situation (having to introduce a composite task without any significance for the user to preserve provenance) can be avoided when SP structures are used while it is not possible for general DAGs.

In Figure 6 (B), the rewriting process of SPFlow has duplicated M_6 and M_3 from workflow (A) into M_{11} M_8 and M_9 M_{12} in workflow (B). As a consequence, the user view designed by ZOOM is only based on significant composite tasks (R-blast_report which contains M_{11}, M_{13}, M_9, M_7, M_{15} and M_{14}, and R-fastaSequence which contains M_8, M_12 and M_{10}). Such a workflow is then more user-friendly. In particular, each of the two composite tasks takes in now only user input and is then clearly easier to share and (re)use in another context.

Why would this demo be interesting for the scientific database community? Recent keynote talks, tutorials and research papers from the scientific database community show that (i) *scientific workflows* play a crucial role in data integration, (ii) data is increasingly *graph-structured*, and (iii) *provenance* is a particularly challenging topic.
Techniques to reduce the native complexity of classical operations on graph structures and improve the readability of the workflows are thus of increasing interest.
This demonstration is at the intersection of these topic areas, and provides a provenance-based rewriting technique that is theoretically and practically interesting to the scientific database community.

5. REFERENCES

[1] Z. Bao, S. Cohen-Boulakia, S. B. Davidson, A. Eyal, and S. Khanna. Differencing provenance in scientific workflows. In *Proc. of ICDE*, pages 808–819, 2009.

[2] W. W. Bein, J. Kamburowski, and M. F. M. Stallmann. Optimal reductions of two-terminal directed acyclic graphs. *SIAM J. Comput.*, 21(6):1112–1129, 1992.

[3] O. Biton, S. Cohen-Boulakia, S. B. Davidson, and C. S. Hara. Querying and managing provenance through user views in scientific workflows. In *Proc. of ICDE*, pages 1072–1081, 2008.

[4] O. Biton, S. B. Davidson, S. Khanna, and S. Roy. Optimizing user views for workflows. In *Proc. of ICDT*, pages 310–323, 2009.

[5] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Vistrails: visualization meets data management. In *Proc. of SIGMOD*, pages 745–747, 2006.

[6] S. Cohen-Boulakia, C. Froidevaux, and J. Chen. Scientific workflow rewriting while preserving provenance. In *Proc. of IEEE Int. Conf. on E-Science (e-Science)*, pages 1–9. IEEE Computer Society, 2012.

[7] J. Goecks, A. Nekrutenko, and J. Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. In *Genome Biology*, pages 438–462. 2011.

[8] A. González-Escribano, A. J. C. van Gemund, and V. Cardeñoso-Payo. Performance implications of synchronization structure in parallel programming. *Parallel Computing*, 35(8-9):455–474, 2009.

[9] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Proc. of PODS*, pages 31–40, 2007.

[10] D. Hull, K. Wolstencroft, R. Stevens, C. A. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(Web-Server-Issue):729–732, 2006.

[11] D. D. Roure, C. A. Goble, and R. Stevens. The design and realisation of the my$_{\text{experiment}}$ virtual research environment for social sharing of workflows. *Future Generation Comp. Syst.*, 25(5):561–567, 2009.

[12] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. In *STOC*, pages 1–12, 1979.