ISSN: 2456-1134 www.isjcresm.com
Vol-09 Issue-01 Mar 2024

Distilling structure in Taverna scientific workflows: a refactoring approach

Vanapamula Veerabrahmachari , Chevula Rekha, Arekatla Madhava Reddy, Gudipati Mohan Singh Yadav

Associate Professor[1], Assistant Professor[2,3,4]

vveerabrahmachari@gmail.com[1] ,
rekhavenkat16@gmail.com[2],
amreddy2008@gmail.com[3],
gudipatimohan20@gmail.com[4]

Department of CSE, A.M. Reddy Memorial College of Engineering and Technology, Petlurivaripalem, Narasaraopet, Andhra Pradesh

Abstract :

Specifying and managing bioinformatics studies is becoming more commonplace via the use of scientific workflow management systems. Bioinformaticians like their programming paradigm because it allows them to quickly construct elaborate data processing pipelines. A graph structure forms the basis of this kind of model, with nodes standing in for individual bioinformatics activities and connections for the flow of information between them. There may be consequences for the reusability of scientific operations when the complexity of such network structures grows over time. In this paper, we advocate for the Taverna model as a means to efficiently design workflows. We contend that "anti-patterns," a word often used in program design, are a major cause of the problems associated with reuse since they imply the usage of idiomatic forms that result in too intricate design. This work's key contribution is a mechanism for automatically identifying such anti-patterns and replacing them with alternative patterns that reduce the structural complexity of the process. This approach to rewriting routines will improve operational efficiency while also improving the user experience (via simpler design and maintenance). (Easier to manage, and sometimes to exploit the latent parallelism amongst the tasks).

Introduction

Scientific workflows management systems [1-5] are increasingly used to specify and manage bioinformatics experiments. Their simple programming model appeals to bioinformaticians, who can use them to easily specify complex data processing pipelines. However, as stated by recent studies [6-8], while the number of available scientific workflows is increasing along with their popularity, workflows are not (re)used and shared as much as they could be. In this work, we have focused specifically on the Taverna workflow management system [9], which for the past ten years has been popular within the bio-Informastics community [1]. Despite the fact that hundreds of Taverna workflows have been available for years through my Experiment public workflow repository [10], their reuse by scientists other than the original author is generally limited.

Some of the causes for the limited reuse have been identified in the sheer difficulty to preserve a workflow's functionality vis-a-vis the evolution of the services it depends on [11]. In addition to this, another factor that limits reuse is the complexity of workflow structure, that involves the number of nodes and links but is also related to intricate workflow structure feetires. Several factors may explain such a structural comlaxity including the fact that the bioinformatics process to be implemented is intrinsically complex, or the workflow system may not provide appropriate expressivity, forcing users to design arbitrary complex workflows. In the present work, the system considered is Taverna. Our approach aims at automatically detecting parts of the workflow structure which can be simplified by removing

explicit redundancy and proposing a possible workflow rewriting. Our preliminary analysis of the structure of 1,400 scientific workflows collected from my Experiments reveals that, in numerous cases, such a complexity is due mainly to redundancy, which is in turn an indication of over-complicated design, and thus there is a chance for a

ISSN: 2456-1134 www.isjcresm.com
Vol-09 Issue-01 Mar 2024

reduction in complexity which does not alter the workflow semantics. Our main contention in this paper is that such a reduction in complexity can be performed automatically, and that it will be beneficial both in terms of user expertonce (easier design and maintenance), and in terms of operational efficiency (easier to manage, and sometimes to exploit the latent parallelism amongst the tasks).

Workflows in Taverna

As mentioned earlier, this work is specific to the Taverna workflow model [1], which we briefly summersize here. Examples of Taverna workflows are given throughout the paper. Taverna combines a dataflow model of computation with a functional model that accounts for list data processing. A workflow consists of a set of processors, which represent software componets such as Web Services and may be connected to one another through data dependencies links. This can be viewed as a directed acyclic graph in which the nodes are processors, and the links specify the data flow. Processors have named input and output ports, and each link connects one output port of a processor to one input port of another processor. A workflow has itself a set of input and output ports, and thus it can be viewed as a processor within another workflow, leading to structural recursion. The workflow depicted in Figure 1 (I), for instance, has one input called Name and two outputs named respecttimely Average and Standard. In turn, processor GetStatistics output has one input port named input and five output ports named Average, Kurtosis, Skewness, Standard Deviation and Sums. We call the triple ⟨< workflow name >, < workflow inputs >, < workflow outputs >⟩ the signature of the workflow. Note that multiple outgoing links from processors or inputs are allowed, as is the case for the workflow input of Figure 1 (I) which is used by two processors. Also, not all output ports must be connected to downstream procensors (e.g., the value on output port attachment list in Get Statistics is not sent anywhere), and symmetrically, not all inputs are required to receive an input data (but input ports with no incoming links should have a default value, or else the processor will not be activated). Input ports are statically typed, according to a simple type system that includes just atomic types (strings, numbers, etc.) and lists, possibly recursively nested (i.e., the type of a list element may be a list, with the constrain that all sub-lists must have the same depth). The functional aspects of Taverna come into play when one or more list-value inputs are bound to processor's ports which have an atomic type (or, more generally, whose nesting level is less than the nesting level of the input value). In order to reconcile this mismatch in list depth, Taverna automatically applies a higher-order function, the cross product, to the inputs. The workflow designer may specify an alternative behaviour by using a dot product operator instead. This produces a sequence of input tuples, each consisting of values that match the expected type of their input port. The processor is then activated on each tuple in the list. There resulting "implicit iteration" effect can be defined formally in terms of recursive application of the map operator [12].

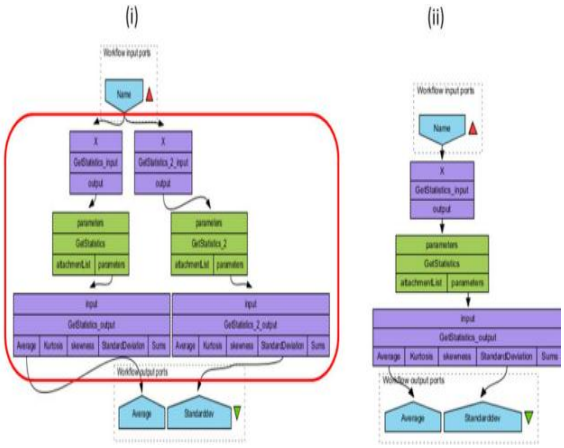

Figure 1 Example of workflow (my Experiment 2383). Example of a Taverna workflow extracted from my Experiment. On the left-hand side (numbered (I)) the original workflow is displayed and a red box highlights the part where redundancies occur. The workflow depicted on the right-hand side of the figure (numbered (ii)) is a semantically-equivalent workflow with no redundancies.

Methods

This section begins by illustrating the two main types of anti-patterns found by our workflow survey, by means of two use cases. The formalization of the anti-patterns and the Distil Flow algorithm will be then introduced. Use cases The first use case (Figure 1 (I)) involves the duplication of a linear chain of connected processors Get Statistics input, Get Statistics and Get Statistics output. The last processor in the chain reveals the rationale for this design, namely to use one output port from each copy of the processor. Clearly, this is unnecessary, and the version in Figure 1 (ii) achieves the same effect much more economically, by drawing both output values from the same copy of the processor. In the second use case (Figure 2 (I)), the workflow begins with three distinct processing steps on the same input sequence. We observe that the three steps

ISSN: 2456-1134 www.isjcresm.com
Vol-09 Issue-01 Mar 2024

that follow those are really all copies of a master Get_image_From_URL task. This suggests that their three inputs can be collected into a list, and the three occurrences can be factored into a single occurrence which consumes the list. By virtue of the Taverna list processing feature described earlier, the single occurrence will be activated three times, one for each element in the input list. Also, the outputs of the repeated calls of Get_image_From_URL will be in the same order as items in the list. Therefore, this new pattern achieves the same result as the original workflow. Note that collecting the three outputs into a list requires a new built-in merge node (the circle icon in Figure 2 (ii)). Similarly, a Split processsoar has been introduced to decompose the outputs (list of values) into three single outputs. These two examples are instances of the general patterns depicted in Figures 3 and 4 (left hand side). These are the anti-patterns we alluded to earlier, and our goal is to rewrite them into the new structures shown in the right-hand side of the figures. In the rest of this section, we describe this rewriting process in detail.

Anti-patterns and transformations

The transformations aim at reducing the complexity of the workflow by replacing several occurrences of the same processor with one single occurrence whenever possible. Although new processors are sometimes introduced in the process (i.e., merge and split operators), on balance we expect a cleaner design, better use of the functional feetures of Taverna (automated list processing) and lower redundancy, and thus fewer maintenance problems.

Assumptions

The following four assumptions must hold for processor instances to be candidates for the transformations described below.

A processor must be deterministic: it should always produce the same output given the same input

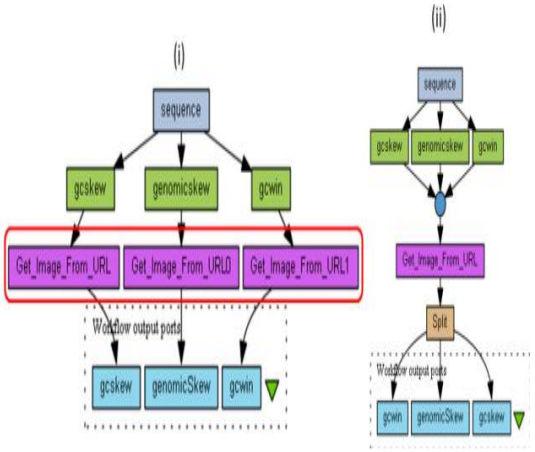

Figure 2 Example of workflow (my Experiment 778). Example of a Taverna workflow extracted from my Experiment. On the left-hand side (numbered (I)) the original workflow is displayed and a red box highlights the part where redundancies occur. The workflow depicted on the right-hand side of the figure (numbered (ii)) is a semantically-equivalent workflow with no redundancies. A merge node (circle) and a split node have been introduced.

Results

Anti-patterns in workflow sets We have applied the refactoring approach on two workflow sets: the public workflows from mixermints and the private workflows of the BioVel project (www.biovel.eu), a consortium of fifteen partners from nine countries which aims at developing a virtual elaboratory to facilitate research on biodiversity. BioVel promotes workflow sharing and aims at providing a library of workflows in the domain of biodiversity data analysis. Access to the repository to contributors, however, is restricted and controlled. Because of the restricted access and the focus on a specific domain of these workflows, they are broadly expected to be curated and thus of higher quality than the general my Experiment population.

For each workflow set, the total number of workflows, the number of workflows having at least one anti-pattern (of kind (A) or (B)) are provided in Table 1. Note that it is possible that the same workflow contains the two kinds of anti-pattern. Interestingly, 25.7% of the workflows of the myExperimint set contains at least one anti-pattern. Although anti-pattern A appears in only 5.5% of the total, it is particularly costly because it involves multiple executions of the same processor with the exact same input, therefore being able to remove it would be particularly beneficial. The prevalence of pattern B suggests that workflow designers may not know the list processing properties of Taverna (or functional languages). As for the BioVel private

ISSN: 2456-1134 www.isjcresm.com
Vol-09 Issue-01 Mar 2024

workflows, 40.8% include at least one anti-pattern, all of kind B and thus none contains any kind A. Additionally, we have observed that a workflow from BioVel contains, on average, fewer anti-patterns than, on average, a workflow from my Experiment.

Discussion Simpler structures

When all the anti-patterns can be removed by Distil Flow, the resulting workflow structures are particularly simpler, as illustrated in examples provided all along the paper, including the two use cases (Figures 1, 2). Figures 9 and 10 provide two additional examples. In Figure 9, we have highlighted the rewritten subgraph that is particularly simper compared to the same fragment of the workflow in the original setting. In Figure 10, the global structure is also simpler. Processors have been numbered so that the relationship between the two workflows (before and after the refactoring process) can be seen: in the original workflow pi denotes the I the occurrence of processor p and in the

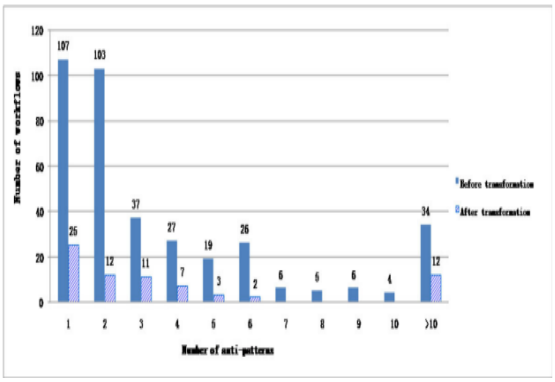

Figure 3 Distribution of anti-patterns in my Experiment. Distribution of number of anti-patterns among workflows in my Experiment, before and after applying Distil Flow.

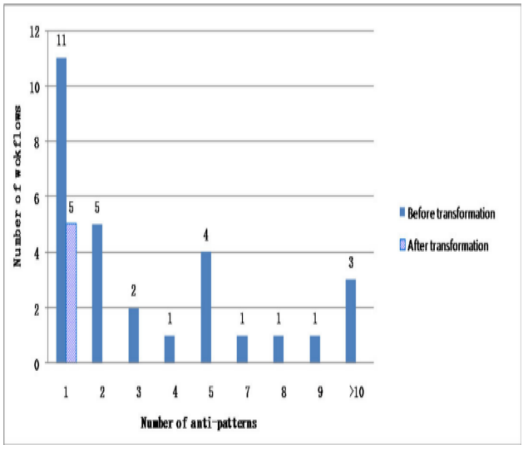

Figure 4 Distribution of anti-patterns in BioVel. Distribution of number of anti-patterns among workflows in BioVel, before and after applying DistillFlow (NB: no workflow in this set has 6 anti-patterns)

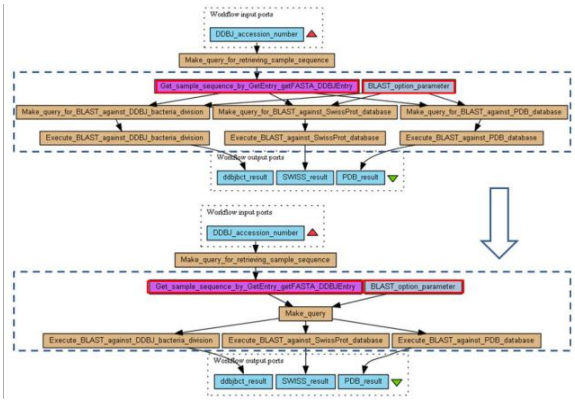

Figure 5 Example of transformation using Distil Flow. Example of transformation obtained using Distil Flow (original workflow at the top and rewritten workflow at the bottom).

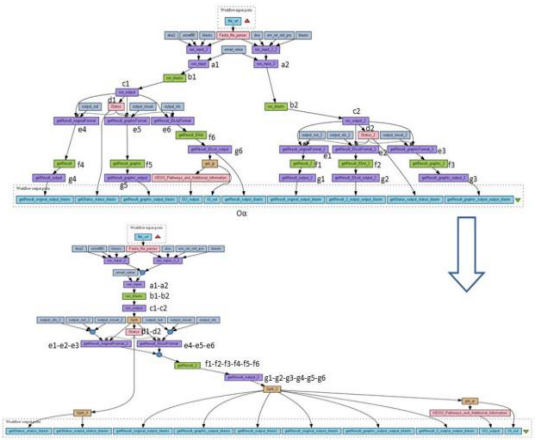

Figure 6Example of Non-SP to SP transformation. Example where the rewritten workflow becomes SP (original workflow at the top and rewritten workflow at the bottom).

rewritten workflow, pi − ... − pj denotes the node resulting of the merging of occurrences pi − ... − pj. For example, f1, f2, f3, f4, f5, f6 are all occurrences of the same processor which are replaced by one occurrence in the rewritten workflow (noted f1 − f2 − f3 − f4 − f5 − f6 in the rewritten workflow). As a result of the refactoring process on the workflow of Figure 10, three SPLIT processors have been introduced and 18 unnecessary duplications of processors have been removed.

SP structures

As explained in the previous sections, Distil Flow acts carefully on the workflow structures, by removing antipatterns (A) and (B) while never introducing new indricate structure as non-SP

structure may be. Removing anti-patterns may actually automatically transform a non-SP structure into an SP structure as illustrated in Figure 9 in which the original workflow has two reduction nodes underlined in the figure (namely, Get_sample_sequence_byGetEntry_getFASTA_D DBJEntry and BLAST_option_parameter). While these nodes have several input/output links in the original setting they have (at most) one input link and one output link in the transformed version and they are not reduction nodes anymore. More generally, in the myExperiment corpus, a total of 15 workflows had a non-SP structure before applying the refactoring algorithm and have an SP structure after. However, it may also be the case that anti-patterns cannot be removed because removal would imply merging nodes which would create a new reduction node, making the structure of the transformed workflows more intricate. The number of reduction nodes is actually a commonly used metric to measure how far from an SP structure a structure may be [17]. In that sense, merging such nodes would make the rewritten workflow being further from an SP structure compared to the original workflow structure.

Conclusion

In this study, we introduce the method Distil Flow, which refactors Taverna processes by removing explicit duplication, perhaps making them more usable and shareable by others. Distil Flow can recognize two types of anti-patterns and rewrite them as improved patterns that better demonstrate traits like maintainability, reusability, and perhaps resource efficiency. This is accomplished primarily by combining, given enabling circumstances, many instances of the same workflow processors into a single instance, while simultaneously compiling a list of inputs to each of the original instances. This modification is demonstrated to be faithful to the original workflow behaviour thanks to Taverna's functional approach to list processing. We tested Distil Flow on two different sets of workflows: the public my Experiment workflows and the private BioVel workflows. Interesting enough, the BioVel collection of workflows has a much less average number of anti-patterns per workflow and a smaller average number of duplicated nodes per anti-pattern than my Experiment set of workflows. Our research therefore confirms that the BioVel collection receives greater attention to curation and quality control than the more diverse my Experiment processes. We have shown that our method may still be useful for both data sets.

References :

[1]. Hull D, Wolstencroft K, Stevens R, Goble CA, Pocock MR, Li P, Oinn T: Taverna: a tool for building and running workflows of services. Nucleic Acids Res 2006, 34(Web-Server):729-732.

[2]. Goecks J, Nekrutenko A, Taylor J: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol 2011, 11(8):438-462.

[3]. Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones MB, Lee EA, Tao J, Zhao Y: Scientific workflow management and the Kepler system. Concurr Comput 2006, 18(10):1039-1065.

[4]. Callahan SP, Freire J, Santos E, Scheidegger CE, Silva CT, Vo HT: VisTrails: visualization meets data management. Proceedings of the ACM-Sigmod International Conference on Management of Data 2006, 745-747.

[5]. The Knime workflow system. [http://www.knime.com/].

[6]. Starlinger J, Cohen-Boulakia S, Leser U: (Re)Use in Public Scientific Workflow Repositories. In Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM). Volume 7338. Lecture Notes in Computer Science; 2012:361-378.

[7]. Cohen-Boulakia S, Leser U: Search, adapt, and reuse: the future of scientific workflows. SIGMOD Record 2011, 40(2):6-16.

[8]. Littauer R, Ram K, Ludäscher B, Michener W, Koskela R: Trends in Use of Scientific Workflows: Insights from a Public Repository and Recommendations for Best Practice. Int Journal of Digital Curation 2012, 7(2):92-100.

[9]. Missier P, Soiland-Reyes S, Owen S, Tan W, Nenadic A, Dunlop I, Williams A, Oinn T, Goble C: Taverna, reloaded. In Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM). Volume 6187. Lecture Notes in Computer Science; 2010:471-481.

[10]. Roure DD, Goble CA, Stevens R: The design and realisation of the myExperiment Virtual Research Environment for social sharing of workflows. Future Gener Comput Syst 2009, 25(5):561-567.

[11]. Zhao J, Gómez-Pérez JM, Belhajjame K, Klyne G, García-Cuesta E, Garrido A, Hettne KM, Roos M, Roure DD, Goble CA: Why workflows break - Understanding and combating decay in Taverna workflows. Proceedings of the IEEE International Conference on E-Science (e-Science) IEEE Computer Society; 2012, 1-9.

[12]. Missier P, Paton NW, Belhajjame K: Fine-grained and efficient lineage querying of collection-based workflow provenance. Proceedings of the International Conference on Extending Database Technology (EDBT) 2010, 299-310.

[13]. Valdes J, Tarjan RE, Lawler EL: The recognition of Series Parallel digraphs. Proceedings of the ACM Symposium on Theory of Computing (STOC) 1979, 1-12.

[14]. Biton O, Cohen-Boulakia S, Davidson SB, Hara CS: Querying and Managing Provenance through User Views in Scientific Workflows. Procedings of the IEEE International Conference on Data Engineering (ICDE) 2008, 1072-1081.

[15]. Bao Z, Cohen-Boulakia S, Davidson SB, Eyal A, Khanna S: Differencing Provenance in Scientific Workflows. Proceedings of the IEEE International Conference on Data Engineering (ICDE) 2009, 808-819.

[16]. Cohen-Boulakia S, Froidevaux C, Chen J: Scientific workflow rewriting while preserving provenance. Proc of the IEEE International Conference on E-Science (e-Science) IEEE Computer Society; 2012, 1-9.

[17]. Bein WW, Kamburowski J, Stallmann MFM: Optimal Reductions of TwoTerminal Directed Acyclic Graphs. SIAM J Comput 1992, 21(6):1112-1129.

[18]. Cohen-Boulakia S, Froidevaux C, Goble C, Williams A, Chen J: Distilling scientific workflow structure. EMBNet.journal, Proceedings of the International Workshop on Network Tools and Applications in Biology, NETTAB 2012 2012, 18(Suppl B):100-102.

[19]. van der Aalst WMP, van Hee KM, ter Hofstede AHM, Sidorova N, Verbeek HMW, Voorhoeve M, Wynn MT: Soundness of workflow nets: classification, decidability, and analysis. Formal Aspects of Comput 2011, 23(3):333-363.

[20]. Trčka N, Aalst WM, Sidorova N: Data-Flow Anti-patterns: Discovering DataFlow Errors in Workflows. In P