Supporting Process Development in Bio-jETI by Model Checking and Synthesis

Anna-Lena Lamprecht[1], Tiziana Margaria[2], and Bernhard Steffen[1]

[1] Chair for Programming Systems, Dortmund University of Technology,
44227 Dortmund, Germany,
`anna-lena.lamprecht@cs.tu-dortmund.de, steffen@cs.tu-dortmund.de`

[2] Chair for Service and Software Engineering, Potsdam University,
14482 Potsdam, Germany,
`margaria@cs.uni-potsdam.de`

Abstract. Bio-jETI is a platform for the intuitive graphical design and execution of bioinformatics workflows composed from heterogeneous remote services. In this paper we use a simple phylogenetic analysis process to show how formal approaches like model checking and process synthesis can be applied to further support the workflow development in Bio-jETI. To unfold their full potential these methods need a comprehensive knowledge base about the domain, containing semantic information about the single services as well as ontological classifications of the used terms. We outline how to systematically integrate these semantic web concepts into our framework and discuss the implications on checking and synthesis.

Key words: bioinformatics workflows, process verification, compliance, model checking, workflow synthesis

1 Introduction

More than in other domains the heterogeneous services world in bioinformatics demands for a good methodology to classify and relate resources in a both human and machine accessible manner. The development of semantically annotated services and comprehensive service and data ontologies in the domain is already very promising. Yet software is needed that fully utilizes the available semantic information in order to provide helpful tools to the *in silico* researcher.

In the last two years we introduced Bio-jETI [1] as an environment for service integration, service orchestration, process execution, and process deployment in the bioinformatics domain. Technically, Bio-jETI builds on the jABC modeling framework [2] that has been used so far in several different domains of science, education, and industry. We have, however, not yet exploited some techniques that are available in the jABC framework and would enable Bio-jETI to support the development of processes in terms of service semantics.

For instance, the jABC provides model checking techniques [3] that can be used for reasoning about properties of process models, and workflow synthesis methodology [4–6] that allows for the automatic creation of workflows according to formal specifications. Considering a simple phylogenetic analysis workflow, we show an example for problem detection via model checking (section 2.1) and for subsequent service insertion by means of synthesis (section 2.2).

It is essential for our methods to have proper semantic information available. On the one hand, we need predicates characterizing the single services, i.e. their function and their input/output behaviour. On the other hand, taxonomies or ontologies are required which provide the domain knowledge against which the services (their predicates) are classified. We discuss the open issues and future work regarding the integration of semantic information about web services, and implications on model checking and synthesis in section 3. Finally, section 4 draws a short conclusion.

2 Example: Developing a Phylogenetic Analysis Workflow

Using the jABC technology, Bio-jETI provides a framework for the graphical orchestration of bioinformatics processes. Process models, called *Service Logic Graphs* (*SLGs*) are constructed graphically by placing process building blocks, called *Service Independent Building Blocks* (*SIBs*), on a canvas and connecting them according to the flow of control. These SLGs are directly executable by an interpreter component, and they can be transformed into stand-alone applications via the GeneSys code generation framework [7], or compiled and deployed as a new web service using jETI [8]. Furthermore, they are accessible to other techniques that exploit the structure of the control-flow process models, such as model checking [3] and workflow synthesis [4–6].

For our example we assume a small SIB collection as presented in figure 1, comprising various remote and local services. It contains data retrieval services provided by the European Bioinformatics Institute, sequence analysis algorithms offered by the Bielefeld Bioinformatics Server, homology search services hosted by the DNA Data Bank of Japan, and some tools of the `EMBOSS` suite, as well as some locally available services, such as a specialized visualizer for phylogenetic trees or a workflow component realizing user interaction.

Utilizing these SIBs, a workflow developer (a biologist, for example) can define the following simple process (see figure 2): show an input dialog where the

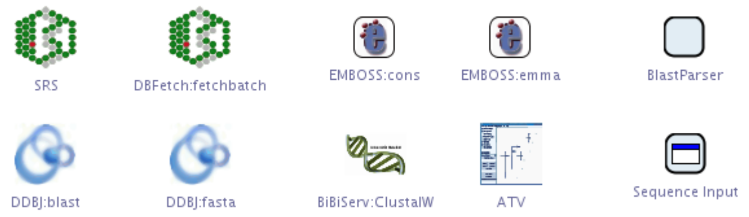


Fig. 1. Exemplary Bio-jETI SIB Collection.

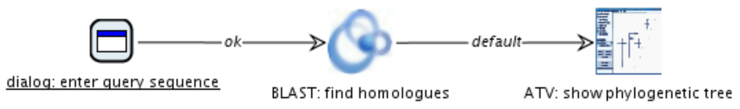


Fig. 2. Initial Process.

user enters a query sequence, then run a BLAST (Basic Local Alignment Search Tool) query with this sequence to find homologous sequences, and finally visualize the phylogenetic tree for this set of sequences with ATV (A Tree Viewer).

While an experienced bioinformatician might be aware of the problem within this process immediately (because of his familiarity with the involved tools), many users will only realize at runtime that there must be some undesired behaviour, since the execution comes to an unexpected halt. In the following we illustrate how to automatically detect and fix such issues already at design time using model checking and model synthesis.

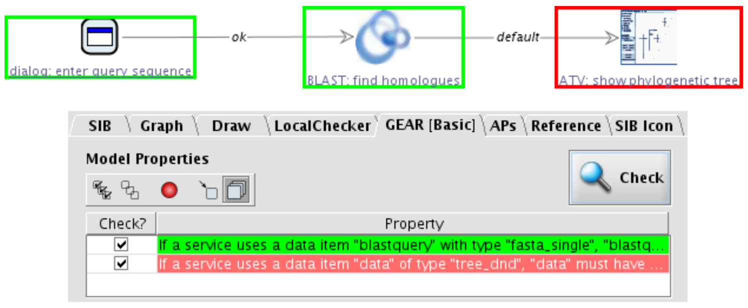
2.1 Using Model Checking to Detect Problems

In this phylogenetic analysis process two services use data: `BLAST` uses the blast query entered in the input dialog, and `ATV` at the very end uses the result of the preceding steps. We can now, e.g., check the type correctness of these uses:

If a service uses a data item x of type y, x must have been defined before with precisely this type, without having been overwritten since.

This property can be expressed in a temporal logic and given to a model checker. While the model checking detects no problems for the first use, it reveals a property violation for the second, as can be seen in figure 3: the rightmost SIB is marked by a red frame, indicating that the property is violated at that node. The cause is that the process does not provide the appropriate input type for the tree visualizer.

Once detected, there are different ways to fix the problem. One possible approach is to search for a sequence of additional services that resolve the type



Check?	Property
✓	If a service uses a data item "blastquery" with type "fasta_single", "blastq…
✓	If a service uses a data item "data" of type "tree_dnd", "data" must have …

Fig. 3. Error Detection via Model Checking.

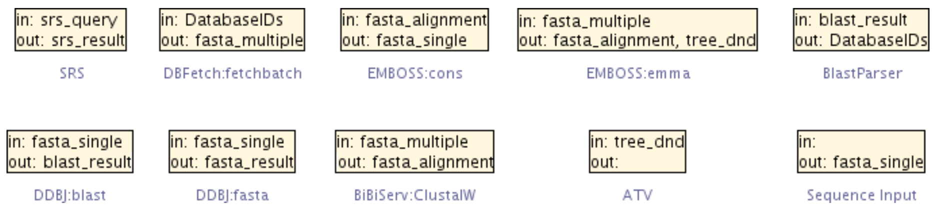


Fig. 4. Input and Output Characterization of the Services.

mismatch and insert them into the process. This data mediation sub-workflow is usually linear. It can consist of type conversions that simply adapt the involved data, or also of real computational services when they can not be related too easily. The next section shows how to use process synthesis to find an adequate sequence of services automatically.

2.2 Synthesis

The knowledge base needed for the process synthesis consists of input and output information for each service as well as service and type taxonomies that classify the services and types, respectively.

We assume a simple type taxonomy for our example, which has a generic type `type` at the root and refines the data types downwards. The basic input and output information for the services is defined in terms of the concrete data types, i.e. the leaves of the taxonomy tree. The corresponding information for our exemplary SIB collection is shown in figure 4.

Having all required knowledge available, the synthesis algorithm must now be provided with a specification for the desired service sequence. For instance,

Take a BLAST result as input and finally produce a phylogenetic tree.

is such a specification that can be formulated in a temporal logic.

Computing the shortest tool combination that satisfies the specification, the synthesis returns a sequence of three services that can be used to complete the erroneous process (figure 5, bottom): a `BlastParser` extracts the database

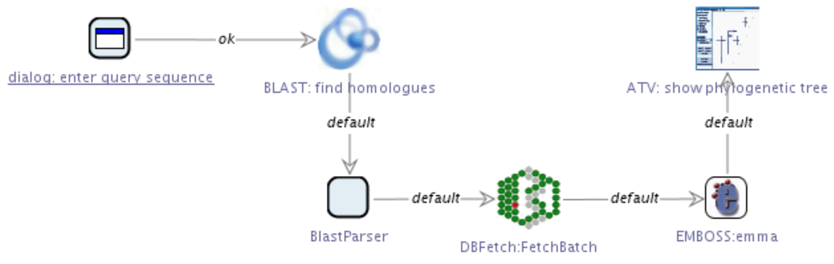


Fig. 5. Extension of the Initial Process.

IDs of the homologous sequences from the BLAST result, the corresponding sequences are fetched from a database using `WSDBFetch`, and `emma` (an interface to the ClustalW multiple sequence alignment algorithm) is invoked to obtain an alignment and the corresponding phylogenetic tree.

3 Discussion and Perspectives

The previous section demonstrated the model checking and synthesis methodology that is currently available in the jABC framework. We are now tailoring these techniques and underlying development methodologies to the bioinformatics application domain. This work spans three dimensions: domain modeling (section 3.1), model checking (section 3.2), and model synthesis (section 3.3).

3.1 Domain modeling

This dimension is the heart of making information technology available to biologists, as it enables them to formulate their problems in their own language terms - on the basis of adequately designed ontologies.

This raises the issue where the domain knowledge ideally comes from. It is, of course, possible for each user to define custom service and type taxonomies, allowing for exactly the generalization and refinement that is required for the special case. However, as the tools and algorithms that are used are mostly third-party services, it is desirable to retrieve domain information from public knowledge repositories as well. Therefore we plan to incorporate knowledge from different publicly available ontologies and integrate it into service and type taxonomies that can be used by our synthesis methodology.

Significative examples for a relevant and popular knowledge bases of bioinformatics data types and services are the constantly evolving namespace, object and service ontologies in BioMoby [9]. Originating from the early 2000s, the 1.0 BioMoby Semantic Web Service specification, however, does not adhere to the ontological standards that have been developed for the Semantic Web in the last years. Thus we follow with interest and hope to benefit from the development of the S(emantic)-Moby framework and also the SSWAP (Simple Semantic Web Architecture and Protocol) [10] project, which aims at providing this knowledge using standard RDF/OWL technology and plans, for instance, to provide the BioMoby domain knowledge accordingly.

It is, of course, also necessary that the services themselves are equipped with meta-information in terms of these ontologies. Again, we are looking at BioMoby with interest: numerous institutions have registered their web services at Moby Central, describing functionality and data types in pre-defined structures using a common terminology. Although BioMoby does not yet use standardized description formalisms like WSDL-S, it is already clear that there is semantic information available that we can use as predicates for automatic service classification.

Furthermore it will be interesting to consider the incorporation of more content-oriented ontologies like the Gene Ontology [11] or the OBO (Open Biomedical Ontologies) [12] into our process development framework. This would enable the software to not only support the process development on a technical level, but also in terms of the underlying biological and experimental questions. Additional sources of information, like the provenance ontologies of [13], could also be exploited by our synthesis and verification methods.

3.2 Model Checking

This dimension is meant to systematically and automatically provide biologists with the required IT knowledge in a seamless way, similar to a spell checker which hints at orthographical mistakes - perhaps already indicating a proposal for correction. Immediate concrete examples of detectable issues are (2.1):

- mismatching data types: a certain service is not able to work on the data format provided by its predecessor.
- missing resources: a process step is missing, where a required resource is fetched/produced.

However, this is only a first step. Based on adequate domain modeling, made explicit via ontologies/taxonomies, model checking can capture semantic properties to guarantee not only the executability of the biological analysis process but also a good deal of its purpose, and rules of best practice, like:

- all experimental data will eventually be stored in the project repository,
- unexpected analysis results will always lead to an alert, or
- chargeable services will not be called before permission is given by the user.

On a more technical side, model checking allows us to apply the mature process analysis methodology that has been established in programming language compilers in the last decades and has shown to be realizable via model checking [14]. Similar to the built-in code checks that most Integrated (Software) Development Environments provide, this would help Bio-jETI users to avoid the most common mistakes at process design time. In addition the list of verified properties is extendable by the user, and can thus be easily adapted to specific requirements of the application domain.

3.3 Model Synthesis

This dimension can be seen as a step beyond model checking: The biologist does not have to care about data types at all - the synthesis automatically makes the match by inserting required transformation programs (see section 2.2). This is similar to a spell checker which automatically corrects the text, thus freeing the writer from dealing with orthography at all.[3]

[3] In our model-based framework, things are well-founded, without the uncertainties of natural language. So please do not be put off by this example because of annoying experiences with spell checkers!

The potential of this technology goes even further: ultimately, the biologist will be able to specify her requests in a very sparse way, e.g. by just giving the essential corner stones, and the synthesis will complete this request to a running process. For instance, consider the following process description:

Having a single (genetic) sequence, I want to find similar sequences and get a hypothesis about their evolutionary relationship.

Typically there are many processes that solve such a request. In the example under consideration each solution would consist of a different set of similarity search services and tools for estimating phylogenies, as well as the required conversions, data retrievals etc. in between. Thus our synthesis algorithm provides the choice of producing a default solution according to a predefined heuristics, or proposes sets of alternative solutions for the biologist to select.

4 Conclusion

The Semantic Web is currently one of the most ambitious projects in computer science. Collective efforts have already lead to a basis of standards for service descriptions and meta-information. It is now mainly the service provider's task to extend the already available technology according to more specific requirements of their application domain and build up a semantics knowledge base.

The challenge for user-side software is to abstract from the underlying Semantic Web technology again and provide the achievements in an intuitive fashion. Some ontology-aware service browsers and clients are already available in the scope of the BioMoby project. With the MOBY-S Web Service Browser it is, for instance, possible to search for a matching next service, while in addition the sequence of actually executed tools is recorded and stored as a Taverna workflow [15]. However, these solutions demand quite some technical understanding from the user. This hampers the uptake by a larger biological community.

Our approach aims at lowering the required technical knowledge according to the "easy for the many, difficult for the few" paradigm. After an adequate domain modeling, including the definition of the semantic rules to be checked by the model checker or to be exploited during model synthesis, biologists should ultimatively be able to profitably and efficiently work with a world-wide distributed collection of tools and data, using their own domain language.

This goal differentiates us from other related projects like Taverna [16], Kepler [17] or Triana [18], which address a bioinformatics user, and not the biologist herself. We believe that our control-oriented approach has a much better potential to address non-IT personnel: it allows them to continue to think in 'Dos' and 'Don'ts', and steps and sequences of action in their own terms at their level of domain knowledge. In contrast, the three dataflow-oriented tools above require their users to change the perspective to a resource point of view, which, in fact, requires implicit (technical) knowlegde in order to profitably use them.

References

1. Margaria, T., Kubczak, C., Steffen, B.: Bio-jETI: a service integration, design, and provisioning platform for orchestrated bioinformatics processes. BMC Bioinformatics **9 Suppl 4** (2008) S12 PMID: 18460173.
2. Steffen, B., Margaria, T., Nagel, R., Jörges, S., Kubczak, C.: Model-Driven Development with the jABC. In: Hardware and Software, Verification and Testing. (2007) 92–108
3. Bakera, M., Margaria, T., Renner, C., Steffen, B.: Verification, Diagnosis and Adaptation: Tool-supported enhancement of the model-driven verification process. In: Revue des Nouvelles Technologies de l'Information (RNTI-SM-1). (December 2007) 85–98 (Journal version to appear in ISSE).
4. Steffen, B., Margaria, T., Beeck, M.: Automatic synthesis of linear process models from temporal constraints: An incremental approach. In ACM/SIGPLAN Int. Workshop on Automated Analysis of Software (AAS'97) (1997)
5. Margaria, T., Steffen, B.: Backtracking-Free Design Planning by Automatic Synthesis in METAFrame. In: Fundamental Approaches to Software Engineering. (1998) 188
6. Margaria, T., Steffen, B.: LTL Guided Planning: Revisiting Automatic Tool Composition in ETI. In: Proceedings of the 31st IEEE Software Engineering Workshop, IEEE Computer Society (2007) 214–226
7. Jörges, S., Margaria, T., Steffen, B.: Genesys: Service-Oriented Construction of Property Conform Code Generators. Innovations in System and Software Engineering - a NASA Journal (to appear)
8. Margaria, T., Nagel, R., Steffen, B.: jETI: A Tool for Remote Tool Integration. In: Tools and Algorithms for the Construction and Analysis of Systems. Volume 3440/2005 of LNCS., Springer Berlin/Heidelberg (2005) 557–562
9. Wilkinson, M.D., Links, M.: BioMOBY: an open source biological web services proposal. Briefings in Bioinformatics **3**(4) (December 2002) 331–41 PMID: 12511062.
10. Gessler, D.: SSWAP – Simple Semantic Web Architecture and Protocol. `http://sswap.info/docs/SSWAP.pdf`
11. Ashburner, M., Ball, C.A., Blake, J.A., et al.: Gene ontology: tool for the unification of biology. Nature Genetics **25**(1) (May 2000) 25–9 PMID: 10802651.
12. Smith, B., Ashburner, M., others, C.R.: The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nat Biotech **25**(11) (November 2007) 1251–1255
13. Sahoo, S.S., Sheth, A., Henson, C.: Semantic Provenance for eScience: Managing the Deluge of Scientific Data. IEEE Internet Computing **12(4)** (2008) 46–54
14. Steffen, B.: Data Flow Analysis as Model Checking. In: Theoretical Aspects of Computer Software, International Conference TACS '91, Sendai, Japan, September 24-27, 1991, Proceedings, London, UK, Springer-Verlag (1991) 346–365
15. Dibernardo, M., Pottinger, R., Wilkinson, M.: Semi-automatic web service composition for the life sciences using the BioMoby semantic web framework. Journal of Biomedical Informatics (March 2008) PMID: 18373957.
16. Oinn, T., Addis, M., Ferris, J., et al.: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17) (2004) 3045–3054
17. Altintas, I., Berkley, C., Jaeger, E., et al.: Kepler: An Extensible System for Design and Execution of Scientific Workflows. In SSDBM (2004) 21—23
18. Taylor, I., Shields, M., Wang, I., Harrison, A.: The Triana Workflow Environment: Architecture and Applications. In: Workflows for e-Science. Springer, New York, Secaucus, NJ, USA (2007) 320—339