Semantics-Based Composition of EMBOSS Services with Bio-jETI

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

[1] Technical University Dortmund, Chair for Programming Systems, Dortmund, D-44227, Germany
{anna-lena.lamprecht|stefan.naujokat|bernhard.steffen}@cs.tu-dortmund.de
[2] University Potsdam, Chair for Service and Software Engineering, Potsdam, D-14482, Germany
tiziana.margaria@cs.uni-potsdam.de

Abstract. Bio-jETI is a framework for model-based, graphical design, execution and management of bioinformatics analysis processes. Formal methodology like automatic service composition extends the framework and, in particular, allows for semantically aware workflow development. In this study we apply the workflow synthesis methodology to the EMBOSS suite of sequence analysis tools. As neither the tool suite itself nor its various interfaces provide ready-to-use semantic annotations, we set up a domain model that uses a high-level, semantically meaningful type nomenclature to describe the input/output behavior of the single EMBOSS tools. Based on this domain model, we demonstrate how working with the large, heterogeneous, and hence manually intractable EMBOSS collection is simplified by our service composition methodology.

1 Introduction

Research projects in modern molecular biology rely on increasingly complex combinations of computational methods to handle the data that is produced in the life science laboratories. A variety of bioinformatics databases, algorithms and tools is available for specific analysis tasks. Their combination to solve a specific biological question defines more or less complex analysis *workflows* or *processes*. Software systems that facilitate their systematic development and automation have found a great popularity in the community.

More than in other domains the heterogeneous services world in bioinformatics demands for a methodology to classify and relate resources in a both human and machine accessible manner. The Semantic Web, which is meant to address exactly this challenge, is currently one of the most ambitious projects in computer science. Collective efforts for modeling the bioinformatics domain have already led to a basis of standards for semantic service descriptions and meta-information. More concretely, the corresponding state of the art can be characterized as follows:

- *Domain Modeling:* Has started in particular in the BioMoby project [1], where a number of services has been prepared mainly for supporting semantics-based retrieval.
- *Components and Interfaces:* A popular example is the EMBOSS suite [2], a large collection of diverse tools for a specific field of bioinformatics (sequence analysis) that is already integrated into a common technical interface. That is, the components are 'wrapped' in order to simplify their use: they work seamlessly for a number of different formats and types, and therefore free the user from caring about compatibility and type conflicts.
- *Design Methodology:* There are different tools for the graphical development of analysis processes [3–7], most of them data-flow based and without connection to semantic modeling. An exception is Bio-jETI [8, 9], which supports the incorporation of semantically modeled domain information for control-flow oriented process construction.
- *Validation*: Bio-jETI is unique in supporting domain-modeling-based verification of processes.

In this paper, we present an extension of the Bio-jETI platform that simplifies the process development phase (item 3) in order to even reach biologists without programming background by

- Extending the currently available domain modeling to comprise the EMBOSS suite [2].
- Achieving type compatibility beyond predefined 'compatibility wrappers' by dynamic mediator synthesis. This allows us to cover also third party components without any programming effort.
- Generalizing Bio-jETI's synthesis technology to support a flexible kind of loose process programming: loosely specified components and partially defined connectors are concretized by ontology-based synthesis.
- Applying model checking to check global properties of complex (partially synthesized) processes.

The paper is structured as follows. Section 2 describes the workflow synthesis technology that is available in Bio-jETI from a user's perspective. In Section 3 we present the setup of the EMBOSS domain. As neither the tool suite itself nor its various interfaces provide ready-to-use semantic annotations, we extracted the relevant user-level semantic information from the tool descriptions and built a domain model that uses a high-level, semantically meaningful type nomenclature to describe the input/output behavior of the single EMBOSS tools. Based on this domain model, we demonstrate in Section 4 how working with the large, heterogeneous, and hence manually intractable EMBOSS collection is simplified by our service composition methodology. The paper ends with a conclusion and perspectives for future work in Section 5.

2 Semantics-Based Service Composition in Bio-jETI

Bio-jETI [8][3] is a framework for model-based, graphical design, execution and management of bioinformatics analysis processes. It has been used in a number of different bioinformatics projects [10–13] and is continuously evolving as new service libraries and service and software technologies become established.

Technically, Bio-jETI is based on the jABC modeling framework [14] as an intuitive, graphical user interface and the jETI electronic tool integration platform [15] for dealing with remote services. Using the jABC technology, 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. SLGs are directly executable by an interpreter component, and they can be compiled into a variety of target languages via the Genesys code generation framework [16].

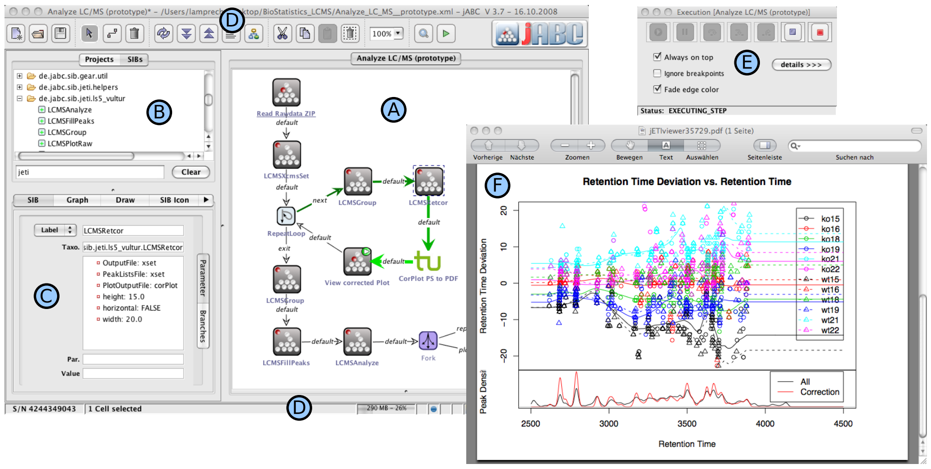


Fig. 1. Graphical User Interface of Bio-jETI

The Bio-jETI Graphical User Interface (GUI) is structured as follows (cf. Figure 1): The major part of the interface is used for the canvas where the SIBs are placed and connected to form the SLG (A). The SIB library (B) shows the available SIBs, whereas the Inspector Pane (C) is used for various GUI elements, such as global model configurations, SIB parameter editing, but also task specific elements for model-checking, local checking, synthesis, etc. Common structures like status bar and menu (D) complete the interface. The modeling with the Bio-jETI framework usually consists of the following steps:

[3] http://biojeti.cs.tu-dortmund.de/

1. drag & drop SIBs from the SIB library to the canvas,
2. connect SIBs with edges,
3. assign branch names,
4. define one start SIB, the entry point for execution, and
5. directly execute the model using an interpreter plugin (E) (window (F) shows a window that is opened by the currently executed SIB) or generate executable code with the Genesys code generation framework.

In [17, 9], we presented our approach to semantics-based service composition in the Bio-jETI platform. By integration of automatic service composition functionality into an intuitive, graphical process management framework, we maintain the usability of the latter for semantically aware workflow development. Furthermore, we can integrate services and domain knowledge from any kind of heterogeneous resource at any location, and are not restricted to any semantically annotated services of a particular platform.

We now present PROPHETS[4], an extension to the Bio-jETI framework that seamlessly integrates automatic service composition into the jABC. It enhances the previous approaches by including more formal methodology, but with less of it being required for the user to know, thus enabling the system to be used by a wider range of users. These enhancement are in particular:

– visualized/graphical semantic domain modeling,
– loose specification within the process model,
– non-formal specification of constraints using natural language templates, and
– automatic generation of model checking formulas.

Two roles are designed for using this extension. The *domain expert* provides information on available services and semantic classification over these services and their input and output types. The *application expert* is the one who uses the available services to model the processes. The following subsections deal with one of those roles, respectively, starting with the domain expert.

2.1 Domain Modeling

The domain model essentially consists of service definitions and their semantic classifications. The service definition enhances the SIBs by meta-information regarding their input/output behavior. Throughout our framework, types are represented by symbolic names, thus abstracting from concrete implementations. So a service is characterized by two subsets of the set of all symbolic type names, namely *input types* and *output types*. The SIBs meta-information is stored in a separate file within the project directory.

Furthermore, the services and types can be classified using taxonomies. These taxonomies are expressed as ontologies in OWL format. Although we also provide a seamlessly integrated graphical editor for these OWL files (see Figure 2), the domain expert may use any OWL tool according to his personal liking.

[4] Process Realization and Optimization Platform using a Human-readable Expression of Temporal-logic Synthesis

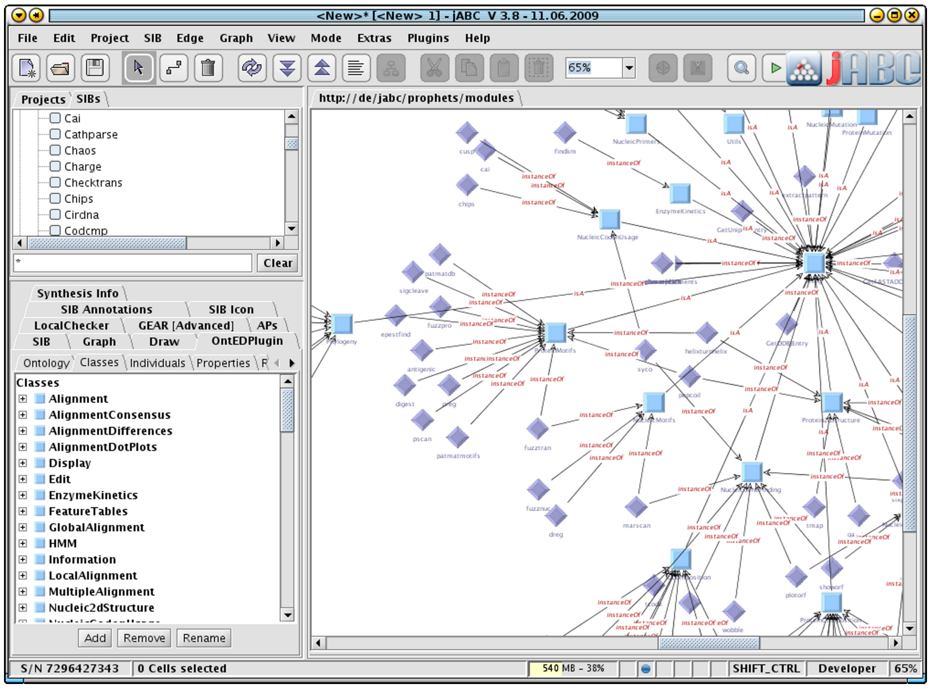


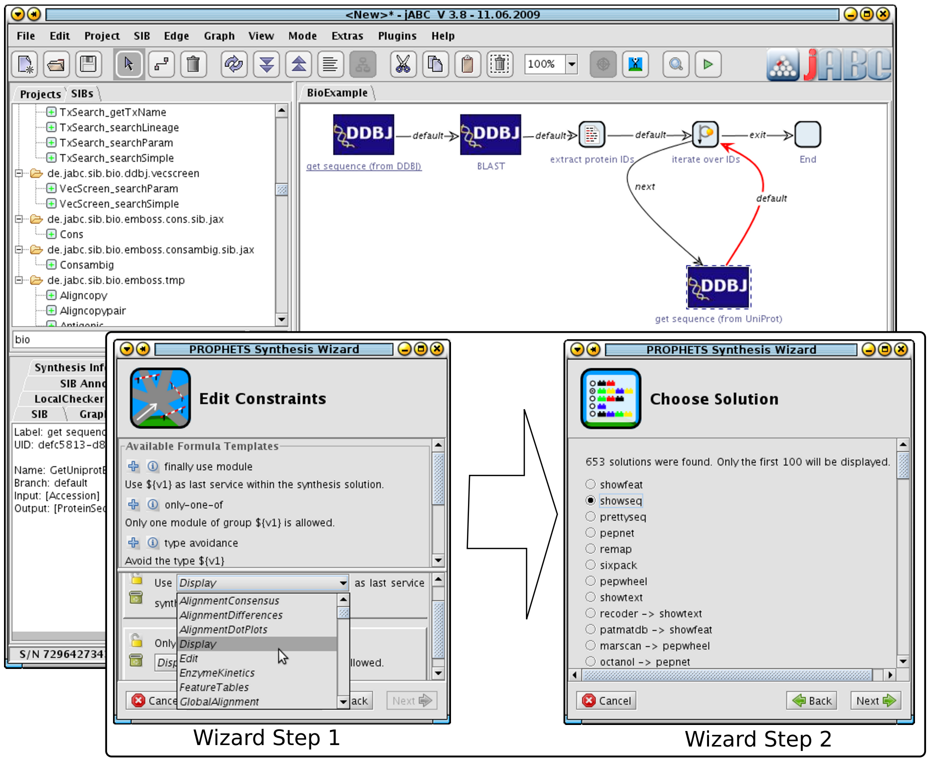
Fig. 2. Bio-jETI's integrated ontology editor

Finally, there might be domain specific knowledge like ordering constraints on services or general compatibility information. This knowledge must be formalized by the domain expert. Basically there are two possible options to do so: Either he expresses model checking formulas that must hold for every SLG within the project or he defines global constraints that are used for every process synthesis.

2.2 Process Design

After a domain has been set up by the domain expert, it can be used by the process expert for workflow design. He does not need to provide fully specified processes, because model parts that are marked as incomplete can be automatically synthesized by the framework. As part of the seamless integration into the jABC, the new extension concentrates on the usability for non-technical users. It facilitates incomplete specification of processes in an easily accessible way by introducing *loosely specified branches*, which the synthesis replaces by concrete solutions. Figure 3 (background) shows an example model using a loosely specified branch (colored red to be distinguishable from normal branches).

Behind the scenes the algorithm requires formal specifications of the synthesis problem using a configuration universe and a formula in the temporal logic SLTL (see [18] for details). Our goal with the here presented approach is to hide this



Label: get sequenc
UID: defc5813-d8

Name: GetUniprot
Branch: default
Input: [Accession]
Output: [ProteinSeq

Use ${v1} as last service within the synthesis solution.

Only one module of group ${v1} is allowed.

Avoid the type ${v1}

653 solutions were found. Only the first 100 will be displayed.

( ) showfeat
(●) showseq
( ) prettyseq
( ) pepnet
( ) remap
( ) sixpack
( ) pepwheel
( ) showtext
( ) recoder -> showtext
( ) patmatdb -> showfeat
( ) marscan -> pepwheel
( ) octanol -> pepnet

Fig. 3. Loosely specified model and synthesis execution wizard

formal complexity from the user and replace it by intuitive (graphical) modeling concepts. Furthermore, the actual execution of the synthesis is presented to the user as a set of wizard windows where he finally can choose the favored solution from the list of all possible solutions (”Wizard Step 2” in Figure 3).

The synthesis algorithm requires a set of start types as initial state within the configuration universe. Previously, these start types had to be specified manually. These are now determined automatically according to preceding SIBs using data-flow analysis methods. The types that are available independently from the execution path in the model are taken as start types. The input types of a loosely specified branch’s goal SIB form the goal types for the synthesis. Both are implicitly specified by the user by marking branches as loosely specified.

As stated above, the synthesis requires constraints that are expressed in the modal logic SLTL. As we won’t expect common process experts to deal with this formal specification, we provide means to express constraints using a system that is based on templates in natural language. The user chooses a restricting concept and then simply has to fill in a cloze text with prepared values (”Wizard Step 1” in Figure 3). The templates can easily be extended to the needs of the specific domain. The possible values for the cloze text fields are automatically extracted from the domain (i.e. module definition and semantic classification).

The previous subsection already mentioned that the domain expert can define global knowledge in means of model checking formulas to describe properties that must hold for any model in this domain. In addition to these manually defined formulas, our framework can automatically generate formulas that check the type consistency of the given model. The type usage is considered to be consistent, if there is no execution path possible that contains a SIB with an input type that has not been generated as output on this very path. This verification is done by a combination of data-flow analysis (the same as is used for start type determination) to annotate available types to SIBs and checking locally if every SIB has all required types (i.e. all input types) annotated that way.

3 Setting up the EMBOSS Domain

EMBOSS (European Molecular Biology Open Software Suite [2]) is a collection of freely available tools for the molecular biology user community. It contains a number of small and large programs for a wide range of tasks, such as sequence alignment, database searches, protein motif identification, nucleotide sequence pattern analysis, and codon usage analysis as well as the preparation of data for presentation and publication[5]. As of October 2009, EMBOSS (Release 6.1.0) consists of around 230 tools, some derived from originally standalone packages.

EMBOSS provides a common look and feel for the diverse tools that are contained in the suite. They can easily be run from the command line, or accessed from other programs. Thus, it is also suitable for being set up behind GUIs and web interfaces. What is more, EMBOSS automatically copes with data in a variety of formats, even allowing for transparent retrieval of sequence data from the web. This enables us to focus on the actual service semantics rather than on technical details of data compatibilities when setting up the domain.

Of the around 230 tools of the complete EMBOSS suite, 175 are currently integrated in our domain. For presentation in this paper we use a representable subset of this domain, consisting mainly of the HMMER [19] applications. HMMER is a software for biosequence analysis using Profile Hidden Markov Models [20]. It contributes 9 applications to EMBOSS, namely `ehmmalign`, `ehmmbuild`, `ehmmcalibrate`, `ehmmconvert`, `ehmmemit`, `ehmmfetch`, `ehmmindex`, `ehmmpfam`, and `ehmmsearch`. The prefix 'e' is used to distinguish the EMBOSS integration from the orginal HMMER programs. In addition to the HMMER tools, our domain contains the multiple sequence analysis tools `emma` and `edialign`, `makeprotseq` and `makenucseq` for the generation of random protein and nucleotide sequences, respectively, as well as some tools for the display of specific data. A complete list of the services in our domain subset is given in Table 1.

Additional structuring of the domain is provided by the classification of types and services in taxonomies, which are simple ontologies that relate entities in terms of *is-a* relations. Figure 4 shows the service taxonomy that we

[5] http://emboss.sourceforge.net/index.html

Table 1. Services in the HMMER subset of the domain.

Service	**Description**	
	Input Types	**Output Types**
edialign	Local multiple alignment of sequences.	
	MultipleSequence	Alignment
ehmmalign	Align sequences to an HMM profile.	
	HMM, Sequence	Alignment
ehmmbuild	Build a profile HMM from an alignment.	
	Alignment	HMM
ehmmcalibrate	Calibrate HMM search statistics.	
	HMM	HMM
ehmmconvert	Convert between profile HMM file formats.	
	HMM	HMM
ehmmemit	Generate sequences from a profile HMM.	
	HMM	EhmmemitOutput
ehmmfetch	Retrieve an HMM from an HMM database.	
	HMMDatabase	HMM
ehmmindex	Create a binary SSI index for an HMM database.	
	HMMDatabase	HMMDatabase
ehmmpfam	Search one or more sequences against an HMM database.	
	HMMDatabase, Sequence	EhmmpfamOutput
ehmmsearch	Search sequence database with a profile HMM.	
	HMM, SequenceDatabase	EhmmsearchOutput
emma	Global multiple alignment of sequences.	
	MultipleSequence	Alignment, Tree
makenucseq	Create random nucleotide sequences.	
	-	MultipleNucleotideSequence
makeprotseq	Create random protein sequences.	
	-	MultipleProteinSequence
showalign	Display a multiple sequence alignment in pretty format.	
	Alignment	-
showseq	Display sequences with features in pretty format.	
	Sequence	-
showtext	Display a text.	
	Text	-

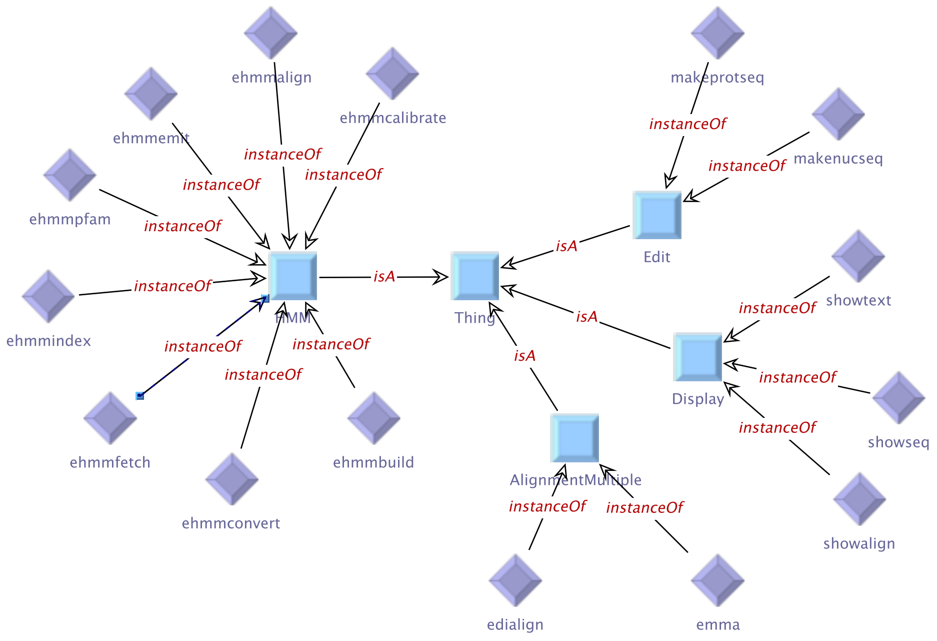


Fig. 4. Service Taxonomy for the HMMER subset of the domain.

defined for the HMMER subset of our domain. The generic type `Thing` (center) represents the root of the taxonomy, underneath which four abstract service groups are defined. The abstract group `Edit` has the services `makenucseq` and `makeprotseq` as instances, the services `showseq`, `showalign` and `showtext` are classified as `Display` by the taxonomy. `Edialign` and `emma` are abstractly described as `AlignmentMultiple`, the remaining tools belong to the `HMM` group.

The type taxonomy for the subset of the domain is shown in Figure 5. All services in this subset work on text-based data, thus all available types belong to the `Text` group. The different `Sequence` types are distinguished further into the groups `ProteinSequence`, `NucleotideSequence`, and `MultipleSequence`. Note that some types are instances of multiple groups: `MultipleNucleotideSequence`, for instance, is both a `MultipleSequence` and `NucleotideSequence`.

Currently the service taxonomy for our complete EMBOSS domain contains the 175 services and 42 abstract groups, which to the most part correspond to the groups that EMBOSS defines. The type taxonomy for the complete domain consists of 135 different data types and 11 abstract classifications. This large number of concrete data types is due to the fact that several tools that are integrated in EMBOSS produce specific tool outputs, often in addition to data that is formatted in a common format. Although these tool outputs can not directly be used as input to other tools, they are relevant to the domain, since it is possible to extract information from them that is suitable as input data.

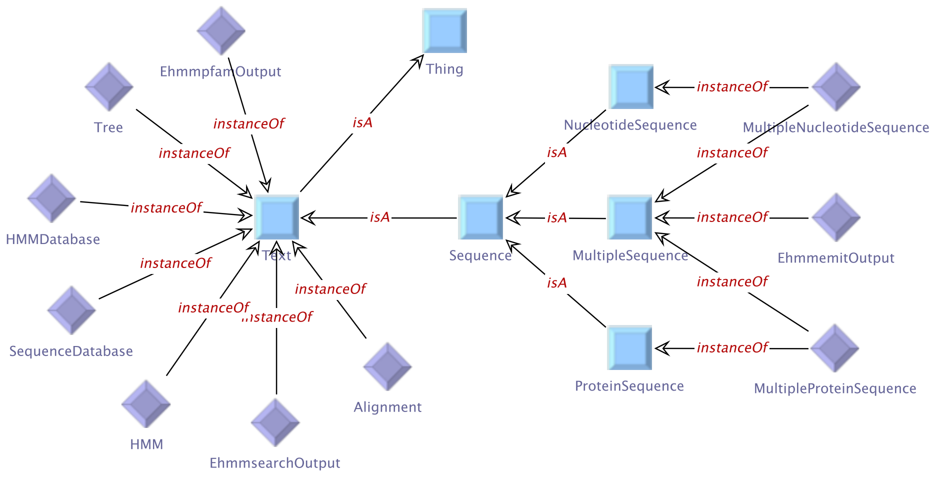


Fig. 5. Type Taxonomy for the HMMER subset of the domain.

4 Working with the EMBOSS Domain

In the previous section we described the setup of the EMBOSS domain, which is the task of the domain expert. In this section, we illustrate the work of the application expert, who designs the actual analysis processes dealing with particular biological questions.

As a first example we consider the small workflow in Figure 6 (A): it consists of the services `makeprotseq`[6] and `showalign`, which are connected by a loosely specified `default` branch. The synthesis problem that is defined by the loose branch is simply given by the output type of `makeprotseq`, providing the input type for the synthesized sequence, and the input type of `showalign`, which is the type that the synthesized sequence must finally produce. That is, the synthesis algorithm has to find a way from `MultipleProteinSequence` to `Alignment`. Obviously, this request can be met by inserting a single multiple sequence alignment service, for example `emma`. Figure 6 (B) shows the result.

A similar synthesis problem is defined by the process shown in Figure 6 (C), where the type `Sequence` must be produced. Obviously, the shortest solution is an empty service sequence, as `makeprotseq` already provides a suitable input for `showseq`. We might, however, have a process in mind that does some analysis on the initially generated sequences and produces another set of sequences, for instance via a Profile HMM. As already indicated in Section 2, additional constraints can be used in the workflow specification that is given to the synthesis algorithm. For expressing the sketched case, we can give an additional constraint

[6] For simplicity, we let our example processes begin with services that randomly generate sequences that can be processed further. Note that they can be easily exchanged by the retrieval of sequences from a public database, or by loading a sequence file.

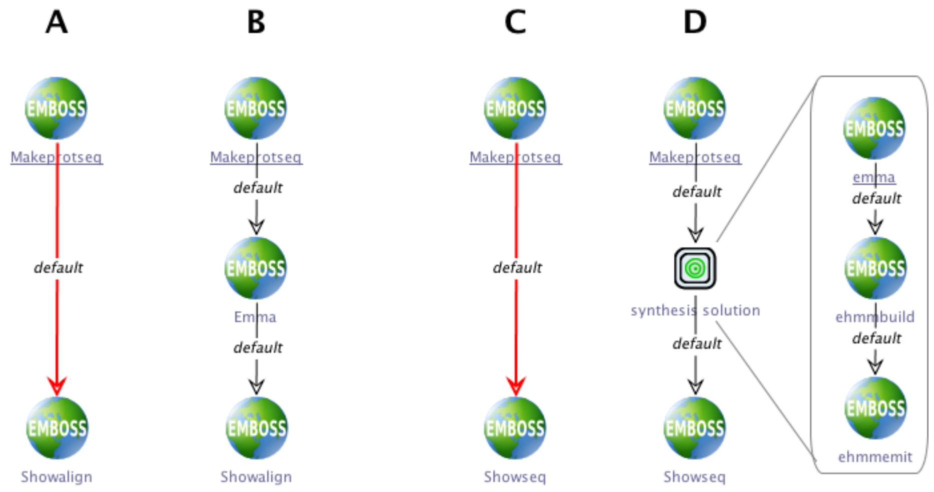


Fig. 6. A: Loose specification between `makeprotseq` and `showalign`. B: Result of simple synthesis. C: Loose specification between `makeprotseq` and `showseq`. D: Result of synthesis with additional constraint that enforces the use of `ehmmemit`.

to the synthesis algorithm that enforces the use of the service `ehmmemit`. One of the shortest thus possible processes is given in Figure 6 (D): the initial input sequences are converted into an `Alignment` by `emma`, which is then used by `ehmmbuild` to create a Profile HMM. `Ehmmemit` emits a set of sequences based on this HMM that are finally displayed by `showseq`.

In [17, 9] we showed how model checking techniques can be applied to monitor global properties of the process models, and used it preliminarily to detect mismatching data types. However, model checking can also validate higher-level constraints that are expressed in terms of application-level domain knowledge. For an example, consider the process in Figure 7 (A). It corresponds to the result from Figure 6 (D). Now, we might want to calibrate each built HMM before it actually emits sequences. Formally, this is expressed as

$$\mathit{ehmmbuild} \Rightarrow (\neg \mathit{ehmmemit}\ \mathbf{WU}\ \mathit{ehmmcalibrate})$$

denoting that the use of `ehmmbuild` implies that `ehmmemit` is not used before `ehmmcalibrate` has been executed. As Figure 7 (A) shows, this requirement is not met by the previously created process, because the `ehmmbuild` SIB does not fulfill the property (indicated by the red overlay icon in the lower right corner of the SIB). Inserting the `ehmmcalibrate` service into the workflow fixes this issue, as Figure 7 (B) shows: all SIBs are marked by a green icon. Naturally, and as (C) shows, this constraint is also fulfilled if the HMM is not built by the process, but fetched from an HMM database.

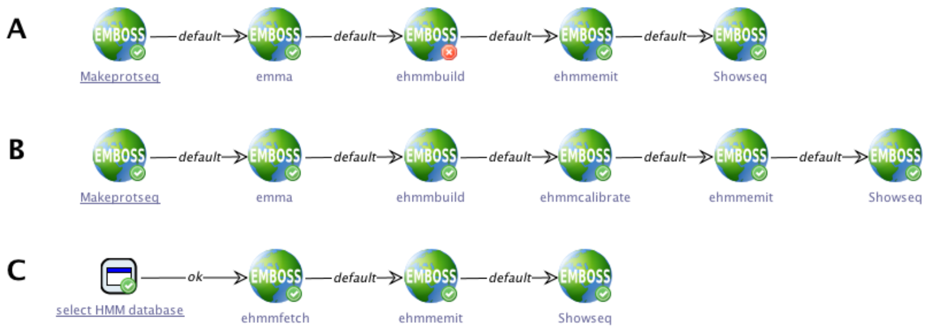


Fig. 7. Model checking of a formula expressing that each built HMM is calibrated before it actually emits sequences.

As a third and final example in this paper, we discuss the process that we already showed in Figure 3 to illustrate the use of the synthesis plugin, which shows a process that does not (yet) contain any EMBOSS services. A (nucleotide) sequence is fetched from the DNA Data Bank of Japan, and used for a BLAST search against a protein database. The Uniprot IDs are extracted from the BLAST result and then processed in a loop that fetches the Uniprot entry for this ID. The remainder of the loop body is a loosely specified branch, to be concretized by an appropriate sequence of services. The synthesis plugin has access to both the EMBOSS and the DDBJ domain model and can transparently combine services from both sources.

For this example, we use the complete EMBOSS domain to find an appropriate sequence of services that does something with the protein sequence that is retrieved within the loop. If we start the synthesis with no further constraints, several thousand possible solutions are found, even if the length of the solution is limited. The reason lies in the nature of the EMBOSS domain: many tools work on the same input type (sequence), some again producing sequences, so that if the synthesis is only based on the type information, unfathomable many variations of solutions are possible. This shows that a adequate domain modeling requires to incorporate as much domain knowledge as possible, far beyond the mere technical aspects of the different types and services.

In order to get less, but more reasonable results, we can now formulate some additional constraints for the synthesis. For instance, we might want the sequence to end with a `Display` service, displaying the available data directly or applying some analysis to the sequence and then displaying the result of the analysis. This can be expressed using formula templates in the synthesis wizard (see Figure 3, Wizard Step 1). The list of solutions that is offered by the wizard is still long (100 out of 653 are displayed, see Figure 3, Wizard Step 2), but the proposed workflows now meet the intention of the process developer.

5 Conclusion

Bio-jETI is a framework for model-based, graphical design, execution and management of bioinformatics analysis processes. Formal methodology like automatic service composition extends the framework and, in particular, allows for semantically aware workflow development [9]. In this study we applied the workflow synthesis methodology to the EMBOSS suite of sequence analysis tools. As neither the tool suite itself nor its various interfaces provide ready-to-use semantic annotations, we set up a domain model that uses a high-level, semantically meaningful type nomenclature to describe the input/output behavior of the single EMBOSS tools. Based on this domain model, we demonstrated how working with the large, heterogeneous, and hence manually intractable EMBOSS collection is simplified by our service composition methodology.

The challenge of semantics-based service composition in the bioinformatics application domain has also been addressed by a number of other projects. For instance, the BioMoby project provides a composition functionality for its services. With the MOBY-S Web Service Browser [21] it is possible to search for an appropriate next service and store the sequence of executed tools as a Taverna workflow. Similarly, the REMORA web server [22] offers functionality for the discovery and step-by-step composition of BioMoby services and the DDBJ's Web API for biology provides next applicable services according to the outputs of previously executed services [23]. Another example is the approach taken by [24], who link meaningful terms from the text of a web page to executable web services, thereby automatically creating workflows that are suitable within the current context.

Bio-jETI is unique in its holistic perspective, which covers both the scope of the process modeling as well as the coverage of individual services and platforms:

- Process development is addressed from a goal-oriented global perspective. Our loose programming concept allows the user to describe the actually intended workflow as a whole, and the synthesis finds shortest solutions directly matching the global intent. In contrast, the automatic service-composition functionality of the approaches mentioned above is limited to small sub-workflows or even single steps of the analysis process, which come with the risk that users get stuck when stepwisely trying to construct the globally intended solution. Especially for collections like EMBOSS, where many tools work on the same data types, a mere local discovery of services is not productive with respect to the construction of a multi-step workflow.
- Due to the integration into the jABC framework and the decoupled specification of service descriptions, any kind of heterogeneous resource at any location can be integrated. There is no restriction to semantically annotated services of a particular platform. On the contrary, any service that is available as a jABC component can be enhanced by a semantic service description and will immediately be available for synthesis of Bio-jETI processes.

Furthermore, Bio-jETI scores with the seamless, user-friendly integration of the domain modeling and synthesis methodology into a graphical process management framework, which, due to the loose programming paradigm, enables application experts to describe their desires in a way that can be automatically transformed into running solutions. Other approaches to automatic service composition that we are aware of require their users to work on a far more technical level. For instance, [25] describe a framework for the composition of data workflows where a domain ontology is modeled in a first-order logic language, and relational data descriptions by formulas over concepts and relations of the ontology. A similar amount of familiarization is required for GOLOG [26], which extends the ALGOL programming language by elements of the Situation Calculus.

All approaches to (semi-) automatically dealing with the large number of distributed, heterogeneous services that are availablein the bioinformatics application domain share the difficult task of finding or defining semantically appropriate service and type descriptions [27]. Projects like the (my)Grid Ontology [28], BioCatalogue [29], BioMoby [1], and SSWAP [30] address this issue by providing knowledge bases that particularly capture bioinformatics data types and services. We plan to integrate their services and domain knowledge in the scope of future case studies. The resulting domains will contain far more heterogeneous services than the comparatively 'closed' EMBOSS domain that we used for the current study, creating new challenges for the client-side software, challenges that Bio-jETI is designed for.

References

1. Wilkinson, M.D., Links, M.: BioMOBY: an open source biological web services proposal. Briefings in Bioinformatics **3**(4) (December 2002) 331–341
2. Rice, P., Longden, I., Bleasby, A.: EMBOSS: the European Molecular Biology Open Software Suite. Trends in Genetics: TIG **16**(6) (June 2000) 276–267
3. Bausch, W., Pautasso, C., Alonso, G.: BioOpera: Cluster-aware Computing. In: Proceedings of the 4th IEEE International Conference on Cluster Computing (Cluster. (2002) 99–106
4. Eker, J., Janneck, J., Lee, E., et al.: Taming heterogeneity - the Ptolemy approach. Proceedings of the IEEE **91**(1) (2003) 127–144
5. Altintas, I., Berkley, C., Jaeger, E., et al.: Kepler: An Extensible System for Design and Execution of Scientific Workflows. In SSDBM (2004) 21–23
6. 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
7. 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
8. 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
9. Lamprecht, A., Margaria, T., Steffen, B.: Bio-jETI: a framework for semantics-based service composition. BMC Bioinformatics **10**(Suppl 10) (2009) S8
10. Margaria, T., Kubczak, C., Njoku, M., Steffen, B.: Model-based Design of Distributed Collaborative Bioinformatics Processes in the jABC. In: Proceedings of ICECCS, IEEE Computer Society (2006) 169–176

11. Kubczak, C., Margaria, T., Fritsch, A., Steffen, B.: Biological LC/MS Preprocessing and Analysis with jABC, jETI and xcms. In: Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006. (2006) 303–308
12. Lamprecht, A., Margaria, T., Steffen, B., et al.: GeneFisher-P: variations of GeneFisher as processes in Bio-jETI. BMC Bioinformatics **9 Suppl 4** (2008) S13
13. Lamprecht, A., Margaria, T., Steffen, B.: Seven Variations of an Alignment Workflow - An Illustration of Agile Process Design and Management in Bio-jETI. In: Bioinformatics Research and Applications. Volume 4983 of LNBI., Atlanta, Georgia, Springer (2008) 445–456
14. Steffen, B., Margaria, T., Nagel, R., et al.: Model-Driven Development with the jABC. In: Hardware and Software, Verification and Testing. (2007) 92–108
15. 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
16. Jörges, S., Margaria, T., Steffen, B.: Genesys: service-oriented construction of property conform code generators. Innovations in Systems and Software Engineering **4**(4) (December 2008) 361–384
17. Lamprecht, A., Margaria, T., Steffen, B.: Supporting Process Development in Bio-jETI by Model Checking and Synthesis. In: Proc. of 1st Workshop SWAT4LS08, Edinburgh, United Kingdom, CEUR Workshop Proceedings (November 2008)
18. Steffen, B., Margaria, T., Freitag, B.: Module configuration by minimal model construction. (1993)
19. Eddy, S.: Profile hidden Markov models. Bioinformatics (Oxford, England) **14**(9) (1998) 755–763
20. Eddy, S.: HMMER: biosequence analysis using profile hidden markov models. http://hmmer.janelia.org/
21. 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)
22. Carrere, S., Gouzy, J.: REMORA: a pilot in the ocean of BioMoby web-services. Bioinformatics (Oxford, England) **22**(7) (April 2006) 900–901 PMID: 16423924.
23. Kwon, Y., Shigemoto, Y., Kuwana, Y., Sugawara, H.: Web API for biology with a workflow navigation system. Nucl. Acids Res. **37**(suppl_2) (July 2009) W11–16
24. Sutherland, K., McLeod, K., Ferguson, G., Burger, A.: Knowledge-driven enhancements for task composition in bioinformatics. BMC Bioinformatics **10**(Suppl 10) (2009) S12
25. Ambite, J., Kapoor, D.: Automatically composing data workflows with relational descriptions and shim services. In: The Semantic Web. (2008) 15–29
26. Levesque, H.J., Reiter, R., Lespérance, Y., et al.: GOLOG: a logic programming language for dynamic domains. Journal of Logic Programming **31** (1997)
27. Lord, P., Bechhofer, S., Wilkinson, M.D., et al.: Applying Semantic Web Services to Bioinformatics: Experiences Gained, Lessons Learnt. In: The Semantic Web ISWC. (2004) 350–364
28. Wolstencroft, K., Alper, P., Hull, D., et al.: The (my)Grid ontology: bioinformatics service discovery. International Journal of Bioinformatics Research and Applications **3**(3) (2007) 303–325
29. Goble, C.A., Belhajjame, K., Tanoh, F., et al.: BioCatalogue: A Curated Web Service Registry For The Life Science Community (April 2009)
30. Gessler, D.: SSWAP - Simple Semantic Web Architecture and Protocol. http://sswap.info/docs/SSWAP.pdf