

Workflow Discovery Through Semantic Constraints: A Geovisualization Case Study

Vedran Kasalica(✉) and Anna-Lena Lamprecht(✉)

Department of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, Netherlands
{v.kasalica,a.l.lamprecht}@uu.nl

Abstract. The construction of computational pipelines, for example automated cartographic workflows for the construction of thematic maps, typically requires detailed knowledge about the available tools for the individual steps and the technicalities of their composition. It is a time-consuming process and comes with the risk of missing meaningful workflows because many possible pipelines are never taken into account. Automated workflow composition techniques can facilitate comprehensive workflow discovery based on semantic constraints: The users express their intention about the workflows by means of high-level constraints, and receive possible workflows that meet their request. The successful application of such methods essentially depends on the availability and quality of semantic domain models that describe the tools and data types in the domain. In this paper, we present an exemplary domain model for a geovisualization use case, and show how it enables the abstract specification and automated composition of a complex cartographic workflow.

Keywords: Scientific workflows · Automated workflow composition · Workflow discovery · Workflow synthesis · Semantic domain modeling · Geoscience · Geovisualization

1 Introduction

The combination of diverse computational tools into purpose-specific pipelines is a frequent challenge for 21st century scientists. One example from the field of geovisualization is the creation of automated cartographic workflows for the construction of thematic maps, which usually comprise several computational steps [6,35]. Whether the workflows are implemented in a conventional scripting or programming language or with a dedicated scientific workflow management system [3,7,28], their construction typically requires the users to know which components to connect, and which connections are possible with regard to the compatibility of input and output data types and formats. The idea of semantics-based automated component discovery and composition is to assist the user in this aspect of workflow development [9,20,24,30]. Ideally, users would only

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11621, pp. 473–488, 2019.
https://doi.org/10.1007/978-3-030-24302-9_34

need to state their intents about the workflow at an abstract, conceptual level, and the system would automatically translate this specification into a concrete executable workflow. The PROPHETS [31] and WINGS [15] frameworks are recent exemplary implementations of such approaches.

In practice the success of semantics-based automated workflow composition depends on the availability of rich tool annotations and controlled vocabularies formalizing the domain's specific terminology. However, this information is usually not available in the required form. The EDAM ontology [16] is a prominent example of a controlled vocabulary for describing operations, data types and formats in the bioinformatics domain, which is used for the comprehensive annotation of tools in the bio.tools [17] registry and thus facilitates semantics-based workflow discovery [32]. To the best of our knowledge, none of the existing ontologies and tool repositories in the geospatial domain are comparable to EDAM and bio.tools regarding scope and comprehensiveness. There has been considerable progress in the last years regarding the development of ontologies for the formalization of geospatial knowledge (e.g. [5,10,26,33]), but these are mostly focused on the classification of data, observation and measurements, rather than on technical entities of workflows (like operations, data types and formats) as in EDAM. Similarly, community standards like the Geospatial Data Abstraction Library (GDAL) [14] and the Open Geospatial Consortium (OGC) [1] aim to provide detailed classifications of the entities from the domain in order to facilitate the smooth interoperability of systems. Hence, classification and annotation endeavors such as those described in [13,27], are a crucial step towards a general and uniform classification and annotation of the technical entities of the domain.

In this paper, we show for a geovisualization example (the automated creation of a topographic map of the Netherlands that depicts waterbird movements) what automated workflow composition can accomplish once suitable technical vocabularies and tool annotations are available. We define taxonomies that provide the terminology for referring to operations and data types in the example domain. Then we add semantic annotations using the terms from these taxonomies to selected tools from the GMT (Generic Mapping Tools) [38]. Finally, we show how a workflow synthesis[1] framework can be used for the high-level specification, automated composition and refinement of executable workflows.

We use our new implementation of a scientific workflow discovery framework (provisionally called APE, for Automatic Pipeline Explorer) that aims to provide a flexible, practical and intuitive tool for automated workflow composition. It has been inspired by the PROPHETS [23,25,31] loose programming framework and uses the same Semantic Linear-Time Logic (SLTL) synthesis method [36]. The new implementation is however independent from a concrete workflow environment, and uses standard formats for the representation of ontologies and tool annotations as well as for the created workflows, in order to ease the integration with existing software environments.

[1] In this context, synthesis refers to the automatic creation of a concrete sequence of semantically described components according to an abstract specification [8].

The paper is structured as follows. Section 2 describes the application example (creation of a map that presents animal tracking data and topographical features of the area) that we address in this paper. Then, Sect. 3 describes the domain model that we set up for the use case, before Sect. 4 goes through an exemplary incremental synthesis process. Section 5 discusses achievements and limitations of the current approach, and concludes the paper with a summary and outline of future work.

2 Use Case: Creation of a Topographic Map with Depiction of Animal Movement Patterns

Cartographic workflows are implemented by sequences of tools that perform the individual elementary operations of the map creation process. There are many different Geographic Information System (GIS) tools available that can be used for map creation, such as the tools provided within ArcGIS [18], the Geospatial Data Abstraction Library (GDAL) [14], the CSISS Geospatial Web Services [2] or the already mentioned Generic Mapping Tools (GMT). For the purpose of this paper, we focus on the GMT, but the same approach can in principle be applied to any other of the aforementioned tools. One of the main reasons for choosing the GMT for this case study are its support for different data input formats and the concise and well-structured documentation of the tools, which provides a rich source of information as basis for the semantic domain modeling. In addition, the modularity of the GMT makes it highly suitable for workflow composition, as the individually accessible elementary operations allow firm control over the entire plotting process.

Our workflow use case is about discovering a computational pipeline that creates a topographic map depicting waterbird movement patterns in the Netherlands (see Fig. 1). Wildlife tracking is an important process for biologists and environmental scientists to improve their understanding of animal behavior. Movement behavior gives insight to the ecology of animals, their interaction with other organisms and effect on the ecosystem dynamics. It can help to predict how environmental changes can affect their role in the ecosystem. The use case combines tools from the GMT collection with data from the Movebank [21] online database of animal tracking data. It is used to help animal tracking researchers to manage, share, protect, analyze, and archive their data. The database supports multiple sources of data, among others, integration with the Argos system (http://www.argos-system.org/), which is probably the largest source of wildlife tracking data worldwide. This type of sources provide diverse, robust and high-quality data, which is however often difficult to exploit and plot on a map, especially for ornithologists and other field researchers who are not very familiar with GIS tools (cf. [11,12]). The data we focus on in our example was used to find correspondences between movement patterns of a keystone waterbird species and the landscape configuration [19]. The data represents mallard movement patterns in the Netherlands [4].

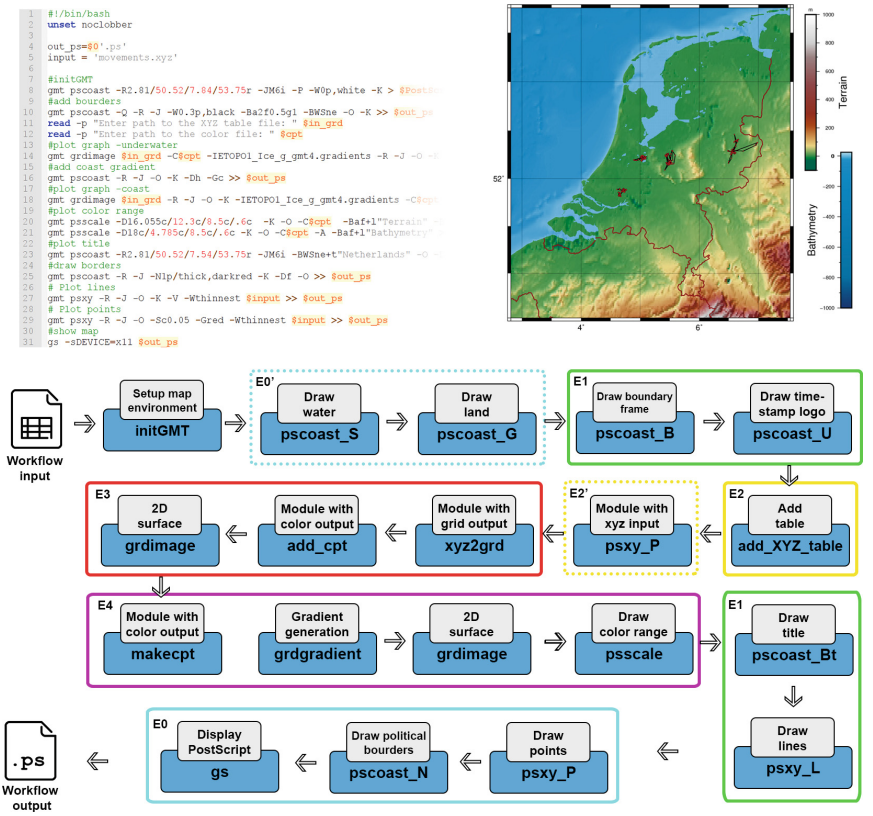

Fig. 1. Incremental development of the workflow (extension steps E0-E4), the corresponding generated GMT script, and the map created by its execution.

3 Domain Modeling

In this section we describe the domain model that we set up for the use case scenario. It comprises a tool and a type taxonomy for defining a controlled vocabulary for the operations and data types in the domain. In addition, it incorporates tool definitions using the terms provided by the taxonomies and a set of so-called domain constraints that express additional relevant knowledge. Overall, the GMT comprises over 100 different tools that can be parameterized on an elementary level, resulting in even more possible operations that would have to be defined as tools. For simplicity and conciseness, we have hence limited the domain model to the GMT tools relevant to the example scenario.

Figures 2 and 3 show the type and the tool taxonomy, respectively. The classification of tools and types in these taxonomies is essential for the effective usage of the synthesis algorithm. It provides the domain-specific, abstract terminology that the workflow developer can use to specify workflows without needing to have knowledge about the concrete tools and data types available (demonstrated in

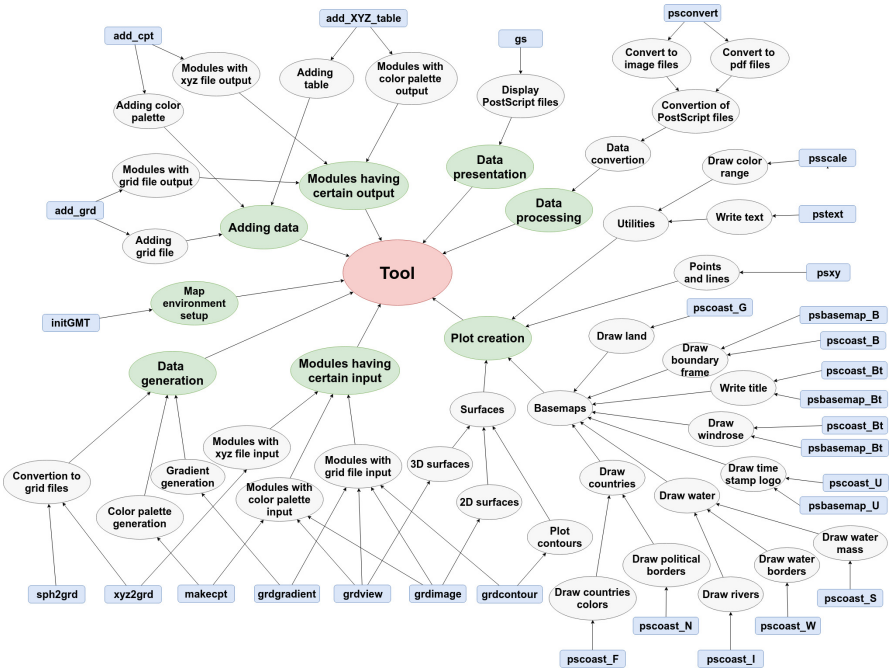

Fig. 2. Tool taxonomy

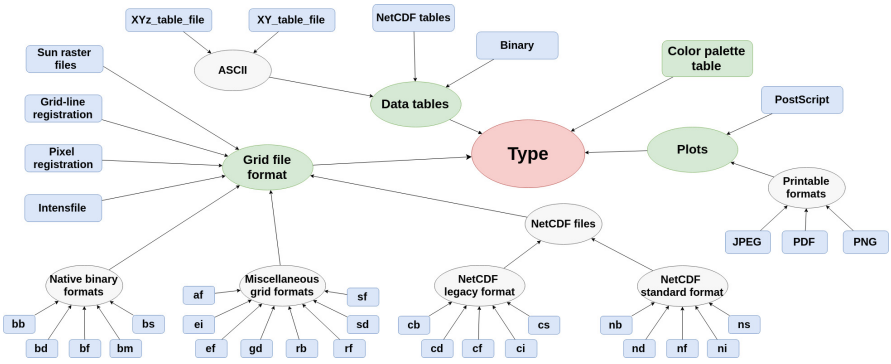

Fig. 3. Type taxonomy

the next section). To the best of our knowledge, there is no similarly structured classification model available for the GMT (or any other GIS tool set), so we defined new taxonomies for this purpose. The data types used by the GMT tools are well documented [39] and at the same time not overly comprehensive, hence the type taxonomy follows straightforward from the documentation and covers all of the data types mentioned in the documentation. The tool taxonomy, on the other hand, focuses on classifying just a part of the GMT in a detailed manner.

The idea was to classify the tools occurring in our scenario in a way that would allow the workflow developer a simplified workflow specification, using the newly introduced abstract classes.

For example, the tool *pscoast_s* in the lower right corner of the tool taxonomy (Fig. 2) is an implementation of the GMT command used to automatically color water surfaces on the map. The tool taxonomy allows us to abstract from the concrete tool and refer to it simply as a *Draw water mass* tool. Another abstract class groups the tools that are used to *Draw water*. As we go up the taxonomy tree we can see that our initial tool belongs to the class of tools used to create *Basemaps*, eventually characterizing all the tools used for *Plot creation*. Finally, every instance of the tool taxonomy is, naturally, considered to be a *Tool*. Every tool and data type in the taxonomies is classified in a similar manner.

Table 1 lists the tools defined for the domain model, each with its name, function description and its (possibly empty) sets of input and output types. The idea is to use the tools as elemental components of our workflows, that can be easily annotated and classified. In some cases, several tools refer to the same underlying GMT operation (*polymorphism*). For example *pscoast_B*, *pscoast_Bt*, *pscoast_U* and *pscoast_Td* all call the *pscoast* operation, but with different parameters, causing it to perform different functions.

In addition to the taxonomies and tool annotations, the domain model can comprise a set of SLTL formulas to express general knowledge that the synthesis algorithm also takes into account. These *domain constraints* are typically used to avoid obtaining workflows that are ambiguous, redundant, or not relevant to the domain. The SLTL formulas are evaluated over sequences of states (identified by a set of data types from the domain model) that are connected by actions (corresponding to the tools in the domain model) and thus represent a workflow. SLTL is a semantic extension of the widely known Linear-Time Logic (LTL). Like LTL, it comprises propositional variables and the classical Boolean connectives ($|$, & and $\neg$), and temporal modal operators G, F, U and X. In a nutshell, $G\phi$ requires the formula ϕ to hold *globally* (always), $F\phi$ specifies that ϕ must hold *finally* (eventually), $\phi_1 U \phi_2$ requires ϕ_1 to hold *until* ϕ_2 holds, and $X\phi$ expresses that ϕ must be valid in the *next state*. In addition, SLTL allows us to use concrete as well as abstract tools from the tool taxonomy to constrain the allowed transitions to a state, expressed as $\langle tool_c \rangle \phi$ (which requires ϕ to hold in a state reachable by $tool_c$), and concrete as well as abstract types from the type taxonomy for the propositions characterizing the states. For a full definition, please refer to [36].

For the formulation of the domain constraints, knowledge of the SLTL syntax is not required, as our framework provides natural-language templates for several common workflow constraints, which can simply be instantiated by selecting the relevant terms from the taxonomies. Table 1 (**G1-G19**) lists the original SLTL and the natural-language representations of the domain constraints defined in this study. The constraints create a basic framework for map generation, thus simplifying the specification for the process developer. For example, constraints **G1**, **G2** and **G3** guarantee that generating/importing a certain file

Table 1. Tool annotations and domain constraints

Name	Description	Data type in	Data type out
add_cpt	Provide a color palette (.cpt) file		cpt_file
add_XYZ_table	Provide an xyz table file		XYZ_table_file
add_grd	Provide a grid file		NetCDF
grdgradient	Compute directional gradient	NetCDF	Intensfile
makecpt	Make color palette tables	cpt_file	cpt_file
xyz2grd	Convert an xyz table file to a 2-D grid file	XYZ_table_file	NetCDF
psconvert	Crop and convert PostScript files to PDF	PostScript	PDF
psconvert	Crop and convert PostScript files to PNG	PostScript	PNG
psconvert	Crop and convert PostScript files to JPEG	PostScript	JPEG
grdcontour	Contouring of 2-D gridded data sets	NetCDF	PostScript
grdview	3-D perspective imaging of 2-D gridded data	NetCDF, cpt_file	PostScript
grdview	3-D perspective imaging of 2-D gridded data	NetCDF, cpt_file, Intensfile	PostScript
grdimage	Produce color images from 2-D gridded data	NetCDF, cpt_file	PostScript
grdimage	Produce color images from 2-D gridded data	NetCDF, cpt_file, Intensfile	PostScript
psxy_L	Plot lines	XYZ_table_file	PostScript
psxy_P	Plot location points	XYZ_table_file	PostScript
pstext	Plot text strings on maps	XYZ_table_file	PostScript
psscale	Plot gray scale or color scale on maps	cpt_file	PostScript
pscoast_B	Drawing the map boundaries and grid		PostScript
psbasemap_B	Drawing the map boundaries and grid		PostScript
pscoast_Bt	Write the title of the map		PostScript
psbasemap_Bt	Write the title of the map		PostScript
pscoast_U	Draw the GMT logo and the time stamp		PostScript
psbasemap_U	Draw the GMT logo and the time stamp		PostScript
pscoast_Td	Draw the windrose		PostScript
psbasemap_Td	Draw the windrose		PostScript
pscoast_G	Coloring land surfaces		PostScript
pscoast_S	Coloring water mass		PostScript
pscoast_I	Draw rivers		PostScript
pscoast_N	Draw political borders		PostScript
pscoast_W	Draw water borders		PostScript
pscoast_F	Coloring countries for which the country codes were provided		PostScript
initGMT	Setup the GMT map environment		
gs	Tool used to display PostScript files	PostScript	

ID	Constraints in natural-language	Constraints in SLTL
G1	If **Tools with xyz output** is used, tool **Tools with xyz input** has to be used subsequently	G(¬<**Tools with xyz output**> true | X F <**Tools with xyz input**> true
G2	If **Tools with grid output** is used, tool **Tools with grid input** has to be used subsequently	G(¬<**Tools with grid output** true | X F <**Tools with grid input**> true
G3	If tool **Tools with color palette output** is used, tool **Tools with color palette input** has to be used subsequently	G(¬<**Tools with color palette output**> true | X F <**Tools with color palette input**> true
G4	If **Data processing** is used, tool **Data generation** cannot be used subsequently	G(¬<**Data processing**> true | X G ¬<**Data generation**> true
...		
G16	If **Data presentation** is used, tool **Plot creation** cannot be used subsequently	G(¬<**Data presentation**> true | X G ¬<**Plot creation**> true
G17	Do not use tool **3D surfaces**	G ¬ <**3D surfaces**> true
G18	Use the data type **Plots**	F **Plots**
G19	Use tool **Map environment setup**	F <**Map environment setup**> true

format requires a subsequent usage of that particular file. In other words, we made sure that the synthesis will not construct a program that loads or generates files that are not being used in the process. Similarly, using constraints **G5 - G16** we ensure proper ordering of the tools, to avoid overriding important annotation data and to prevent unnecessary permutations in the solutions. Furthermore, we target 2D and not 3D map representations and use the constraint **G17** to exclude the tools used for 3D plotting, which are represented by an abstract class *3D_surfaces* in the tool taxonomy. Finally, we want to use GMT tools and have a map as a product of each of our programs. This requirement can be fulfilled by using an abstract class from the type taxonomy, more specifically, by enforcing the usage of a data type *Plots* and the tool setup *initGMT*, as displayed in **G18** and **G19**. These 19 general constraints will be used in all synthesis runs.

4 Workflow Synthesis

In this section we focus on the workflow specification and synthesis phase and show how our framework can be applied to discover and compose a complex workflow without having to identify and connect the individual required tools manually. Our concrete case example is the creation of an executable workflow that annotates animal movement patterns on a topographic map of the Netherlands. Concretely, Sect. 4.1 explains the initial first synthesis step, and the following Sects. 4.2 to 4.5 discuss the increments that finally lead to the intended map creation workflow.

4.1 Initial Workflow

Synthesis in our framework starts with two sets of data types (possibly empty), that correspond to the input and output of the workflow. In addition, SLTL constraints can be added to express further intents of the user. In our scenario the initial data type is the table format provided by the use case, while the output data type corresponds to the common map representation format - PostScript. This would already provide enough specification to get a transition from our initial data into a PostScript file. However, we would like to add some context to the plotted locations, such as distinguishing land from water surface. Thus, we add constraints to express the first, trivial requirements that need to be fulfilled (see Table 2): Constraint **E0.1** ensures that a tool for drawing the sea will be used. *Draw_water* contains three subclasses (see Fig. 2), where each of them contains a different tool that can be used to satisfy the constraint. Moreover, constraints **E0.2** and **E0.3** follow the same logic for the drawing of the land and the political borders, respectively. Finally, constraint **E0.4** ensures that the workflow displays the generated map to the user (handy for immediate verification). These constraints are combined with the general constraints (**G1-G19**) to form the initial workflow specification.

With this specification as input, we start the actual workflow synthesis. Note that in this study, we perform the search for possible workflows until the first

Table 2. Initial workflow constraints

ID	Constraints
E0.1	Use tool **Draw_water**
	F <**Draw_water**> true
E0.2	Use tool **Draw_land**
	F <**Draw_land**> true
E0.3	Use tool **Draw_political_bourders**
	F <**Draw_political_bourders**> true
E0.4	Use **Display_PostScript** as last tool in the solution
	F(<**Display_PostScript**> true
	G(¬<**Display_PostScript**> true | ¬ X X true)



Fig. 4. Initial workflow output

depths where solutions are found (the search depth is the same as the length of the solutions). Usually, the shortest solutions are also the most relevant with respect to the specification, as they present the smallest number of steps necessary to satisfy the workflow specification, so this is a workable heuristic. For the initial loose specification, our synthesis tool finds 32 solutions of length 6. That is, even though only five constraints (*G19* and *E0.1* - *E0.4*) were used that enforce the use of certain tools, each solution contained at least one additional step. The reason for this lies in the general constraint in the domain model: the existence of the input data type *XYZ_table* requires subsequent usage of a tool that would implement the XYZ table (see **G1** in Table 1). Thus, the domain model ensures the usage of a tool that is required, but not explicitly requested by the process developer.

From the possible workflows suggested by the synthesis framework for our specification, we chose to use the workflow shown in Fig. 1 under labels **E0** and **E0'**, as it uses proper tools for coloring the land and the sea, draws the political borders and uses point locations to depict the animal movement patterns. The framework provides the workflow and its implementation as a shell script, which upon execution produces the map presented in Fig. 4.

The presented map is not yet the intended result. To add more information to the map, more steps need to be included into the workflow. One possibility is to also consider synthesis solutions of lengths greater than 6. For example, we might want to use both points and lines to depict mallard movement patterns. A workflow for that case would be found at depth 7, additionally including the tool *psxy_L*. If we expand our search correspondingly, we find that there are 2100 possible workflows of this length, and their evaluation is required in order to find the suitable one. The evaluation of this amount of workflows is usually not feasible. An alternative approach is to reduce the size of the search space and thus the number of possible solutions found. This is accomplished by adding further constraints, describing the actually desired workflow as precisely as possible, and starting the synthesis process again.

Table 3. Extension 1 constraints

ID	Constraints
E1.1	Use tool **Draw_lines**
	F <**Draw_lines**> true
E1.2	Use tool **Draw_points**
	F <**Draw_points**> true
E1.3	Use tool **Draw_boundary_frame**
	F <**Draw_boundary_frame**> true
E1.4	Use tool **Write_title**
	F <**Write_title**> true
E1.5	Use tool **Draw_time_stamp_logo**
	F <**Draw_time_stamp_logo**> true



Fig. 5. Extension 1 map

4.2 Extension 1: Annotations

As mentioned before, the first map clearly lacks further information, such as annotations (frame, grid, title etc.). In order to properly annotate the map, we have to define corresponding tool enforcement constraints. For example, we would like to have both annotations of the mallard locations as well as annotations of the path taken, in order to properly understand the movements patterns from the map (see **E1.1** and **E1.2** in Table 3). In addition, we would like the map to include, the coordinates and boarder frame of the map (see **E1.3**), the name of the area of interest (see **E1.4**), the time it was created, as well as the name of the tool set used in the process (see **E1.5**). Our goal is to enrich the annotation of the previously generated map, hence, extend the generated workflow with corresponding annotation tools.

Our new workflow specification consists of the constraints used before and the new set of constraints. For this specification, the synthesis returns over 8,000 (shortest) solutions of length 10. Although the number of candidate solutions is too big to evaluate, they were mostly different permutations of the tools, or they incorporated different versions of the same tools (e.g. some tools have *pscoast* and *psbasemap* version of it). Evaluation of the first 5 candidate solutions has shown that the optimal solution in this scenario corresponds to the workflow presented in Fig. 1 under labels **E0**, **E0'** and **E1**, where the label *E1* corresponds to the newly introduced annotation tools. The execution of the workflow produces a properly annotated version of the map (see Fig. 5).

4.3 Extension 2: Providing the Elevation Dataset

Even though the annotated map is self-explanatory and can be presented as such, there is still room for improvement. The figure lacks information on the characteristics of the area, as the land and the sea are depicted with simple plain colors. The study [19], that motivated us to choose the corresponding waterbird tracking data in our scenario, focuses on predicting animal movement based on the landscape configuration. Therefore, it is natural to assume that

Table 4. Extension 2 and 3 constraints

ID	Constraints
E2.1	Use tool **Add_table**
	F <**Add_table**> true
E3.1	Use tool **2D_surfaces**
	F <**2D_surfaces**> true

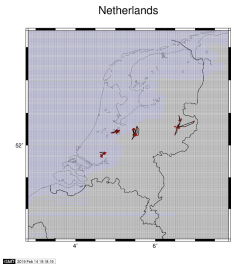

Fig. 6. Extension 2 map

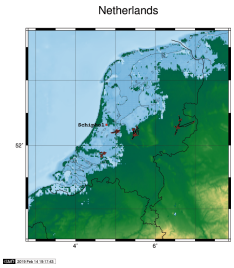

Fig. 7. Extension 3 map

the map should depict some of the topographic and bathymetric features of the Netherlands and its surroundings.

One of the ways to solve this problem is to introduce a file that contains elevation data of the region. The data would be used to plot the relief of the land and the sea. Our initial workflow uses two plotting tools that are not required in this scenario, *pscoast_S* and *pscoast_G* (see **E0'** labeled elements in Fig. 1), as the plain coloring will be overwritten by the elevation data. Therefore, we will extend the workflow obtained in the previous extension step without the two mentioned tools. In order to allow for an easier evaluation of our candidate solutions, we will keep the desired parts of the previously generated workflow and separately synthesize the following sub-workflow.

In order to accurately extend the workflow, we require the usage of the elevation data table. This is accomplished by enforcing the usage of the appropriate abstraction class from our taxonomy (see **E2.1** in Table 4). The synthesis finds two possible workflows at depth 2 that are aimed to extend the previous workflow. Both of the candidate solutions use a system tool **add_XYZ_table** and a tool for plotting lines and points. Based on the generated output the workflows did not differ, and thus we have chosen an arbitrary candidate. Our extended graph corresponds to the tools labeled with **E0**, **E1**, **E2** and **E2'** in Fig. 1. The output of our workflow is presented in is Fig. 6.

4.4 Extension 3: Plotting the Elevation Dataset

We can observe that the current workflow does not properly utilize the provided elevation data (see Fig. 6). The reason for this is the plotting tool (*psxy_P*) used in the process. The idea of the specified workflow extension was to introduce detailed elevation data and to use a tool that can depict that elevation on the map (i.e. using a rich coloring scale). However, the extended workflow only distinguishes between positive and negative elevations, plotting them as black and blue tiles, respectively. In order to solve this issue and draw a proper relief map, the part of the workflow using the mentioned tool needs to be redefined. We expect the usage of a tool, tailored for plotting surfaces based on a elevation data, and thus, our synthesis requires a constraint enforcing it (see **E3.1** in Table 4).

Table 5. Final workflow constraints

ID	Constraints
E4.1	Use tool **2D_surfaces**
	F <**2D_surfaces**> true
E4.2	Use tool **Gradient_generation**
	F <**Gradient_generation**> true
E4.3	Use tool **Tools_with_color_palette_output**
	F <**Tools_with_color_palette_output**> true
E4.4	Use tool **Draw_color_range**
	F <**Draw_color_range**> true

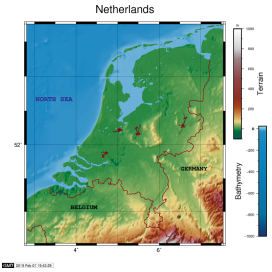

Fig. 8. Final map

The synthesis over the two constraints (see **E2.1** and **E3.1**) finds the two shortest solutions at depth 4. Note that similarly to the initial step, the synthesis extended our workflow with four new elements, even though we have introduced only two constraints. This time, however, the reason for it was the tool annotation part of the domain model, more specifically the dependency between input and output types of the workflow elements.

Considering that the first, out of the two solutions, requires a new grid file to be provided, whereas the second one uses the provided XYZ table to generate the required grid file, we chose the second for inclusion in the workflow (see **E3** labeled tools in Fig. 1). Our workflow has thus been extended to **E0**, **E1**, **E2** and **E3** labeled tools in Fig. 1. The generated map is presented in Fig. 7.

4.5 Final Extension: Topographical and Bathymetrical Features

The current map (Fig. 7) has some inconsistencies with the actual coastline of the Netherlands. We pinpointed Amsterdam's airport (Schiphol) on the map to illustrate the issue. The problem is the elevation of the country, as about one third of the Netherlands lies below sea level and our basic elevation color palette depicts all negative heights as blue. To solve this issue, we have to separate the plotting of the topographical and bathymetrical features, which requires a further extension of our workflow. The idea is to use the existing workflow for plotting the bathymetrical data and the annotations of the map, while a newly synthesized sub-workflow will be used to plot valid topographical features.

In order to ensure the desired behavior we have to specify the appropriate constraints. The proper plotting tool usage is covered by using the constraint used in the previous step (see **E4.1** in Table 5). Additionally, we would like to emphasize the topographical features of the Netherlands, by shading the generated map (see **E4.2**). However, these constraints do not guarantee the usage of a different color palette, which is crucial in order to distinguish negative sea and land elevations. As we need a new color palette, it is intuitive to use a constraint of the form **E4.3**. Finally we enforce drawing the elevation legend - color scale (see **E4.4**). Our goal is to extend the existing workflow with a new set of tools, and thus we perform the synthesis of the described sub-workflow.

The first two solutions are found at depth 4. Similar to the previous case, both of them are using the appropriate plotting tool, but the two solutions differ when it comes to the color palette data. The first solution requires the file to be imported from the system, while the second solution generates the color palette from a part of the already provided color palette file. As the second solution seems more intuitive and does not require us to manually generate another color palette file, we have selected it (see **E4** labeled sub-workflow in Fig. 1). The new extended workflow is presented in Fig. 1 with labels **E0**, **E1**, **E2**, **E3** and **E4**. The final product is the map in Fig. 8, created by a workflow that has been composed automatically based on a set of semantic constraints, which described the workflow developer's intentions free from the technicalities of the underlying GMT-based implementation.

5 Discussion and Conclusions

We demonstrated how workflow synthesis technology can simplify the discovery and creation of geographic data manipulation processes once an adequate domain model is available. A workflow developer, for example an ornithologist as in the applied scenario, can compose a purpose-specific workflow without having to be familiar with the concrete tools in the domain. Instead of trying to find out which tools can be combined and how, the they can use abstract terms from the domain ontology to describe the intended workflow in a declarative way. The synthesis algorithm translates the specification into concrete and executable candidate workflows, from which the developer can choose one, test it, and if required re-create or extend the pipeline under development.

We designed an exemplary workflow scenario along with the corresponding domain-specific vocabularies and formalized domain knowledge. To the best of our knowledge, no such taxonomies or ontologies and tool annotations are yet available in the domain of geographic information and data processing, thus we contribute an elemental example for the classification of tools and data types. Although it copes well with our use case, it is still not the ideal solution, in particular with regard to scalability to larger application scenarios. Ideally, such domain models would be standardized and defined by the corresponding scientific communities and provided in a structural way, analogously to the previously mentioned EDAM ontology and bio.tools registry in the bioinformatics domain. In the scope of future work we plan to explore if the core concepts for spatial information as proposed by Kuhn [22] can form the basis of an domain ontology that is more easily extensible to other use case scenarios.

The quality of the results obtained by any workflow discovery framework essentially depends on the quality of the domain model. Only an accurate and detailed formalization of the relevant technical domain knowledge enables the process developer to specify the intended workflow in a simple and precise manner, and the discovery framework to provide accurate and satisfying solutions. Furthermore, the domain model can provide the vocabulary for different levels of abstraction, allowing process developers with different levels of expertise

to use the framework transparently. As we have shown in the previous section, workflows can be specified using only abstract terms from the domain model, with no knowledge about the GMT being required for that step. In principle, the concrete tools could be replaced with corresponding ones from, e.g., the CSISS services [2] or the GDAL tools [14], and without changing the specification the synthesis would return different, but semantically equivalent workflows.

The constraint-driven synthesis approach used in APE allows for a very fine-granular and explicit formulation of the workflow developers' intentions. This is not only useful to prune the search space in order to filter out irrelevant results, but a decrease of the size of the search space usually also means a speedup of the synthesis process. It lies in the nature of synthesis problems that they suffer from state explosion effects, that is, the combinatorial blowup of the search space causing exponential runtime complexity of the algorithms [29,37]. APE relies on the efficiency of state-of-the-art SAT solving techniques for the implementation of the synthesis algorithm, which has led to significant improvements of the runtime performance compared to earlier implementations.

The comparison and ranking of the synthesis candidate solutions is open issue. As mentioned before, the current approach implements a simple heuristics of ranking the solutions by their length. Although this is a workable evaluation in most cases, quite often it is not sufficient on its own to filter out less desired solutions. For example, in Sect. 4.2, the synthesis has returned over 8,000 different solutions of length 10, which is not manageable without further support. The synthesis research community regards in particular domain-specific search heuristics (exploiting e.g. non-functional properties or additional knowledge about e.g. the preferred ordering of tools) as a promising approach to solve this issue [8,34]. Such heuristics would allow rankings between simple tool permutations or comparison between similar tools in order to pre-select and provide better candidate solutions. This is an open problem in the field and a crucial step towards effective and efficient workflow synthesis in practice.

In summary, we believe that there lies a great and not yet exploited potential in the use of workflow synthesis technology to support the semantics-based discovery of domain-specific computational pipelines. Future research needs to address the challenges of large-scale semantic domain modeling and state-explosion effects. Furthermore, to be adopted by the scientific community, the workflow synthesis frameworks need to be integrated with the scientists' accustomed software ecosystems in order to support not only their construction, but also their systematic benchmarking with real input data. These are some of the challenges that we are going to address with the future development of our semantics-based workflow discovery framework.

References

1. Abstract Specifications—OGC. https://www.opengeospatial.org/docs/as
2. CSISS/GMU Geospatial Web Services. http://cube.csiss.gmu.edu/grassweb/manuals/index.html. Accessed 14 Feb 2019

3. Scientific workflow system. https://en.wikipedia.org/w/index.php?title=Scientific_workflow_system&oldid=877419140. Accessed 14 Feb 2019
4. Movement patterns of a keystone waterbird species are highly predictable from landscape configuration (2017). http://hdl.handle.net/10255/move.644
5. Albrecht, J., Derman, B., Ramasubramanian, L.: Geo-ontology tools: the missing link, **12**, 409–424 (2008)
6. Asche, H.: Process-oriented geoinformation systems and applications. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012. LNCS, vol. 7610, pp. 324–324. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34032-1_31
7. Atkinson, M., Gesing, S., Montagnat, J., Taylor, I.: Scientific workflows: past, present and future. Futur. Gener. Comput. Syst. **75**, 216–227 (2017)
8. Bodik, R., Jobstmann, B.: Algorithmic program synthesis: introduction. Int. J. Softw. Tools Technol. Transf. **15**(5), 397–411 (2013)
9. Chen, L., et al.: Towards a knowledge-based approach to semantic service composition. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 319–334. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39718-2_21
10. Cox, S.J.D.: An explicit owl representation of ISO/OGC observations and measurements. In: Proceedings of SSN 13, SSN 2013, vol. 1063, pp. 1–18. CEUR-WS.org, Aachen (2013)
11. Coyne, M.S., Godley, B.J.: Satellite tracking and analysis tool (STAT): an integrated system for archiving, analyzing and mapping animal tracking data. Mar. Ecol. Prog. Ser. **301**, 1–7 (2005). https://doi.org/10.3354/meps301001
12. Dodge, K.L., Galuardi, B., Miller, T.J., et al.: Leatherback turtle movements, dive behavior, and habitat characteristics in ecoregions of the Northwest Atlantic Ocean. PLoS ONE **9**(3), e91726 (2014). https://doi.org/10.1371/journal.pone.0091726
13. García-Chapeton, G.A., Ostermann, F.O., de By, R.A., Kraak, M.J.: Enabling collaborative GeoVisual analytics: systems, techniques, and research challenges. Trans. GIS **22**(3), 640–663 (2018)
14. GDAL/OGR contributors: GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation (2018). Accessed 14 Feb 2019
15. Gil, Y., Ratnakar, V., Kim, J., et al.: Wings: intelligent workflow-based design of computational experiments. IEEE Intell. Syst. **26**(1), 62–72 (2011)
16. Ison, J., Kalaš, M., Jonassen, I., et al.: EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. Bioinformatics **29**, 1325–1332 (2013)
17. Ison, J., Rapacki, K., Ménager, H., et al.: Tools and data services registry: a community effort to document bioinformatics resources. Nucleic Acids Res. **44**(D1), D38–D47 (2015)
18. Johnston, K., Ver Hoef, J.M., Krivoruchko, K., Lucas, N.: Using ArcGIS Geostatistical Analyst, vol. 380. Esri, Redlands (2001)
19. Kleyheeg, E., van Dijk, J.G.B., Tsopoglou-Gkina, D., et al.: Movement patterns of a keystone waterbird species are highly predictable from landscape configuration. Mov. Ecol. **5**(1), 2 (2017). https://doi.org/10.1186/s40462-016-0092-7
20. Kona, S., Bansal, A., Blake, M., Gupta, G.: Generalized semantics-based service composition. In: ICWS 2008, pp. 219–227. IEEE Computer Society (2008)
21. Kranstauber, B., et al.: The movebank data model for animal tracking. Environ. Model. Softw. **26**(6), 834–835 (2011)
22. Kuhn, W.: Core concepts of spatial information for transdisciplinary research. Int. J. Geogr. Inf. Sci. **26**(12), 2267–2276 (2012)

23. Lamprecht, A.L.: User-Level Workow Design - A Bioinformatics Perspective. Lecture Notes in Computer Science, vol. 8311. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45389-2
24. Lamprecht, A.L., Margaria, T., Steffen, B.: Bio-jETI: a framework for semantics-based service composition. BMC Bioinform. **10 Suppl 10**, S8 (2009)
25. Lamprecht, A.L., Naujokat, S., Margaria, T., Steffen, B.: Synthesis-based loose programming. In: Proceedings of QUATIC 2010, Portugal, pp. 262–267. IEEE (2010)
26. Lehmann, J., Athanasiou, S., Both, A., et al.: The GeoKnow handbook. Technical report (2015). http://jens-lehmann.org/files/2015/geoknow_handbook.pdf
27. Lemmens, R., Wytzisk, A., de By, R., Granell, C., Gould, M., Van Oosterom, P.: Integrating semantic and syntactic descriptions to chain geographic services. IEEE Internet Comput. **10**(5), 42–52 (2006)
28. Liu, J., Pacitti, E., Valduriez, P., et al.: A survey of data-intensive scientific workflow management. J. Grid Comput. **13**(4), 457–493 (2015)
29. Lustig, Y., Vardi, M.Y.: Synthesis from component libraries. Int. J. Softw. Tools Technol. Transf. **15**(5), 603–618 (2013)
30. Martin, D., et al.: Bringing semantics to web services: the OWL-S approach. In: Cardoso, J., Sheth, A. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30581-1_4
31. Naujokat, S., Lamprecht, A.-L., Steffen, B.: Loose programming with PROPHETS. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 94–98. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28872-2_7
32. Palmblad, M., Lamprecht, A.L., Ison, J., Schwämmle, V.: Automated workflow composition in mass spectrometry-based proteomics (2018)
33. Reza, K., Krzysztof, J., Femke, R., Luc, B., Ali, A.: Collaborative ontology development for the geosciences. Trans. GIS **18**(6), 834–851 (2014). https://doi.org/10.1111/tgis.12070
34. Schrijvers, T., Tack, G., Wuille, P., Samulowitz, H., Stuckey, P.J.: An introduction to search combinators. In: Albert, E. (ed.) LOPSTR 2012. LNCS, vol. 7844, pp. 2–16. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38197-3_2
35. Simon, M., Asche, H.: Automated spatial data processing and refining. In: Lamprecht, A.-L. (ed.) ISoLA 2012/2014. CCIS, vol. 683, pp. 38–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-51641-7_3
36. Steffen, B., Margaria, T., Freitag, B.: Module configuration by minimal model construction. Technical report, FMI, Universität Passau (1993)
37. Valmari, A.: The state explosion problem. In: Reisig, W., Rozenberg, G. (eds.) ACPN 1996. LNCS, vol. 1491, pp. 429–528. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-65306-6_21
38. Wessel, P., Smith, W.H.F.: Free software helps map and display data. EOS Trans. Am. Geophys. Union **72**(41), 441–446 (1991)
39. Wessel, P., Smith, W.H., Scharroo, R., et al.: Generic mapping tools: improved version released. EOS Trans. Am. Geophys. Union **94**(45), 409–410 (2013)