1

Workflow Composition: Semantic Representations for Flexible Automation

Yolanda Gil

1.1 Introduction

Many different kinds of users may need to compose scientific workflows for different purposes. This chapter focuses on the requirements and challenges of scientific workflow composition. They are motivated by our work with two particular application domains: physics-based seismic hazard analysis (Chapter 10) and data-intensive natural language processing [1]. Our research on workflow creation spans fully automated workflow generation (Chapter 23) using artificial intelligence planning techniques for assisted workflow composition [2, 3] by combining semantic representations of workflow components with formal properties of correct workflows. Other projects have used similar techniques in different domains to support workflow composition through planning and automated reasoning [4, 5, 6] and semantic representations (Chapter 19). As workflow representations become more declarative and expressive, they enable significant improvements in automation and assistance for workflow composition and in general for managing and automating complex scientific processes. The chapter starts off motivating and describing important requirements to support the creation of workflows. Based on these requirements, we outline the approaches that we have found effective, including separating levels of abstraction in workflow descriptions, using semantic representations of workflows and their components, and supporting flexible automation through reuse and automatic completion of user specifications of partial workflows. These are all important areas in current and future research in workflow composition.

1.2 The Need for Assisted Workflow Composition

Scientific workflows typically comprise dozens of application components that process large data sets. The data sets are often sliced into smaller sets to be processed concurrently, often resulting in the execution of thousands of

jobs. Figure 1.1 shows a sketch of a partial workflow for machine translation. It illustrates how a data set is first divided into subsets, how these are processed in parallel by the same sequences of jobs, and how the final results are assembled in the final stage.

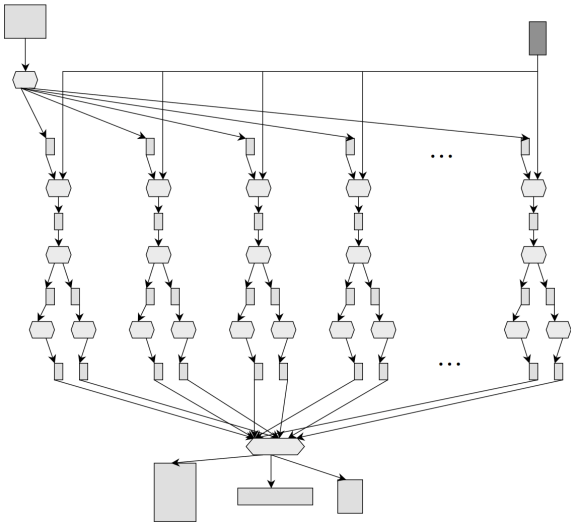

Fig. 1.1. Scientific workflows may be complex and often involve parallel processing of data sets. This figure shows an example where a data set is split up in the early stages, its subsets are processed concurrently, and final results are compiled in the later stages.

1.2.1 Unassisted Workflow Composition and Its Limitations

A common approach to creating workflows is to develop ad hoc scripts that handle the iterative nature of sets of jobs and can generate workflow variants through global variables. They also specify the data locations and execution locations necessary for each job. The data have to be moved to the locations specified, and the executables must be set up in the appropriate locations. The scripts also take care of generating the metadata associated with the workflow products, often using naming conventions to differentiate among alternative configurations and executions. As an alternative to scripts, workflows may be created by hand with a text editor and updated with a copy–edit process.

These approaches have severe limitations in terms of usability and scale. Workflows can only be created by users who are very familiar with the application components used in the workflow, the execution environment, and the scripting language. Errors abound, as with any manually managed process,

and users need to be able to understand error conditions and repair failures. Extending the size of the workflows to include new models has cascading effects that have to be managed manually, making it impractical unless the additions were anticipated in advance.

Usability and scale turn out to be crucial requirements for many scientific disciplines. We motivate the requirements for scientific workflow composition with two application domains that we have used in our work and are representative of the requirements we see in other disciplines.

In order to simulate potential earthquakes, a workflow for seismic hazard analysis combines physics-based models including stress models that hypothesize the distribution of cumulated stress over fault systems given continental drift and the stress in other faults, fault rupture models that forecast potential earthquake sources in a fault system, wave propagation models that simulate the propagation of a seismic wave in a 3D Earth volume, site response models that predict how a seismic wave will be amplified at a certain location based on its soil type, and structure deformation models that simulate the effect of seismic waves in a man-made structure such as a building or a bridge. These models can be used today by the scientists who developed them, but ideally the users would include other scientists who want to use, extend, or validate the aggregate models. In addition, the models should be accessible to a wider range of users, such as engineers designing structures supposed to withstand ground motion to a reasonable degree, graduate research assistants doing advanced projects on the sensitivity of the models to certain controlled variations, and scientists in related disciplines.

Natural language researchers are developing data-intensive statistical training techniques to create language models useful for automatic summarization, machine translation, and document Indexing. A wide variety of models can be found to address different aspects of language processing, such as lexical analyzers, stemmers, part-of-speech taggers, syntax-based parsers, semantic parsers, translation rules, and so on. To put together a machine translation system requires assembling an entire suite of such models to process and parse the original language sentence, map it to the target language, and smooth out the output to make it as fluent as possible. Each of the models has to be trained, perhaps on a different body of text, depending on the topic of translation, before the actual translation is done on the original sentence. New models are developed constantly by different groups around the world, and variations of combinations of models are explored by different research groups for different purposes. Because better performance is invariably obtained with larger sets of training data, there is increased interest in workflow environments that exploit high-end computing and large data and storage management facilities. Sharing of data and models is often done informally across research groups. Flexible workflow composition and execution frameworks would support the rapid development and validation of novel approaches and models.

In summary, although it is possible to create and manage workflows of considerable size in an unassisted manner, there are severe challenges and

practical limitations in terms of usability and scalability that can only be addressed by end-to-end workflow systems that assist users with the creation, execution, and management of workflows.

1.2.2 Workflow Composition Scenarios

The following are representative scenarios for workflow composition illustrated in these two application domains. These scenarios motivate the requirements for workflow composition discussed in the next subsection.

Running a Common Kind of Analysis with a New Data Set

A common kind of wave propagation simulation takes a fault rupture and a model of the corresponding Earth volume's characteristics and runs a physics-based anelastic wave propagation model over that volume to generate 2D or 3D seismograms. This kind of analysis is done routinely by Southern California Earthquake Center (SCEC) scientists well versed in such physics-based wave propagation models, but a scientist in Seattle may want to apply the same analysis to data for the Pacific Northwest area. The workflow structure is essentially the same, but the input data to be used are different. The Seattle scientist will not be able to compose the workflow from scratch but could reuse the basic workflow structure. In a machine translation project, the same workflow can be tried out with a new body of text or a new language.

Creating a Variant of a Type of Analysis

A scientist in Santa Barbara who creates in her research a new model of a fault in Southern California would want to test this model with typical wave propagation simulation codes, except replacing the usual Earth volume model by one that incorporates hers. In this case, the scientist from Santa Barbara does not need to compose a workflow from scratch but instead could reuse the commonly used workflow and modify it slightly by substituting one of the components. In a machine translation project, a scientist may try out a new parser her group has developed and investigate its effect on the final translation quality.

Specifying only Critical Aspects of the Analysis

A scientist in Boston may be interested in simulating wave propagation using finite-difference models, but any of the finite-difference models would be acceptable. This illustrates that it is possible to describe categories of workflows based on abstract classes of models. The new workflow would not be composed from scratch, but by selecting one of the instances of the abstract class of models mentioned in the workflow. A machine translation researcher working on improving the fluency of the output will run workflows with a part-of-speech tagger but may have no preference regarding the kind used.

Running a Complex Analysis Composed of Common, Simpler Ones

An engineer in Palo Alto would like to simulate the effect of certain fault ruptures on his design of a freeway overpass at a location close to the San Andreas fault. This may require composing a workflow by combining two workflows: one designed to simulate the effect of certain ground motions on the overpass structure and another one designed to simulate the wave propagation from the fault ruptures to the site of the overpass. In a machine translation project, the output of translation may be used for document summarization, where the overall processing would be obtained by combining the two respective workflows.

Specifying a New Type of Analysis

A scientist may create a new model for wave propagation that runs very efficiently if coupled with certain types of models of an Earth volume. This scientist would have to compose a completely new workflow out of a new set of models by specifying step by step what models are to be used and how they need to be combined. A machine translation researcher may create models that represent a new approach to word-by-word translation and use them to create a new kind of workflow.

1.2.3 Requirements for Workflow Composition

The scenarios above illustrate that workflows have many users and uses that need assistance in creating workflows. From graduate students to experienced scientists, scientists with varied needs and expertise may need to conduct workflow analyses using the same underlying models and data. Engineers or scientists in other disciplines may benefit from using the same models if they are made accessible and easy to use within their own analysis process. The degree of freedom and the amount of assistance and automation required during workflow creation will be very different in each case. But ideally the same underlying mechanisms should be used to manage the workflow composition process.

Some of the scenarios above describe scientific exploration tasks. In those cases, the scientist will always want to specify some aspects of the analysis that are critical to their investigation, leaving it to the system to figure out the rest automatically. The initial specification may include partial descriptions of desired results, application components to be used in the analysis, input data to be used in the computation, or all of the above. This requires an expressive language that supports flexible descriptions of models and data. This may require assisting users to provide a complete and valid initial specification to ensure that all the pieces provided are mutually consistent and that it is possible to create a full workflow from them. Once the initial user specification is provided, it can then be automatically extended to form a complete workflow

that can be executed. This requires a flexible workflow completion mechanism since it will need to work from results back to what is required to generate them, from input data down to typical ways to process them, or from models and their requirements that need to be generated by adding other models to the workflow and so on.

Most of the scenarios above do not require creating workflows from scratch. Although in some cases step-by-step assembly of new workflows from individual components is needed, workflows can often be created by reusing existing workflows with minimal adaptations. This is not surprising, given that scientific exploration often involves repeated analysis with small variants or local modifications. Workflow reuse also encourages the practice of well-established methodologies captured in particular workflow specifications. The more common steps in the workflows used by different scientists to do similar analyses, the more comparable their results will be. This argues for reusing workflow structures as much as possible across research groups and across experiments or analyses. Results that are obtained using well-established methodologies should essentially be the products of well-known and easily identifiable workflows.

Workflow reuse involves two major aspects: retrieval and adaptation. Retrieval involves finding appropriate workflows in a library, which requires that workflow repositories be organized and indexed thematically and hierarchically. Adaptation of workflows has a wide range of complexity. The less sophisticated a user is, the more he or she is likely to reuse entire workflow structures. More advanced users will be familiar with details of the models and may venture to create variants of a previous workflow by adding or replacing components. The simplest kinds of adaptation involve simple substitutions of input data. The workflow composition system should ensure that the new data set is appropriate for the models included in the workflow. This requires that each workflow be described in terms of the types of data for which it is appropriate. More complex kinds of reuse involve substitutions of specific components together with the addition of steps to generate the data needed by the new components. Other steps needed by the old model may no longer be necessary and need to be removed. Supporting this adaptation process requires representing the characteristics and constraints of each model and the ability to use those representations to check the overall consistency of workflows. Checking the consistency and validity of workflows is even more necessary when several existing workflows at a time are reused to create a new workflow.

These scenarios also illustrate the need for describing workflows in terms that are critical for the experiment while ignoring necessary but irrelevant execution details. This necessary detail is needed to execute the workflow and includes components that prepare and transform the data into formats required by the models, move data to the locations where they need to be processed or stored, and perform conversions to different metric or reference systems. A workflow composition system should present workflows to users

at an appropriate level of abstraction. In addition, it should automatically manage any steps in the workflow that do not involve experiment-critical components.

In summary, there are three key requirements for assisting users in workflow composition. First, workflows must be described at different levels of abstraction that support varying degrees of reuse and adaptation. Second, expressive descriptions of workflow components are needed to enable workflow systems to reason about how alternative components are related, the data requirements and products for each component, and any interacting constraints among them. Third, flexible workflow composition approaches are needed that accept partial workflow specifications from users and automatically complete them into executable workflows. The next three sections discuss each of these three requirements in turn.

1.3 From Reusable Templates to Fully Specified Executable Workflows

Representing workflows at appropriate levels of abstraction is key to support reuse and to manage the complexity and details involved in creating scientific workflows. In our work we consider three stages of creation of workflows, illustrated in Figures 1.2, 1.3, and 1.4. Each stage corresponds to a different type of information being added to the workflow, namely:

1. Defining *workflow templates* that are data- and execution-independent specifications of computations. Workflow templates identify the types of components to be invoked and the data flow among them. The nature of the components constrains the type of data that the workflow is designed to process, but the specific data to be used are not described in the template. In this sense, a workflow template is parameterized where its variables are data holders that will be bound to specific data in later stages of the workflow creation process. A workflow template should be shared and reused among users performing the same type of analysis.
2. Creating *workflow instances* that are execution-independent. Workflow instances specify the input data needed for an analysis in addition to the application components to be used and the data flow among them. A workflow instance can be created by selecting a workflow template that describes the desired type of analysis and binding its data descriptions to specific data to be used. While a workflow instance logically identifies the full analysis, it does not include execution details such as the physical replicas or locations to be used. That is, the same workflow instance can be mapped into different executable workflows that generate exactly the same results but use different resources available in alternative execution environments.

3. Creating *executable workflows*. Executable workflows are created by taking workflow instances and assigning actual resources that exist in the execution environment and reassigning them dynamically as the execution unfolds. Executable workflows fully specify the resources available in the execution environment (e.g., physical replicas, sites and hosts, and service instances) that should be used for execution. This mapping process can be automated and ideally is incremental and dynamic. In an incremental mapping scheme, only the initial workflow steps might be assigned to resources, while later steps can wait until the execution of the initial steps is finalized. The mapping should be dynamic so that when an execution failure occurs, the assignment can be reconsidered.

The template shown in Figure 1.2 depicts a rule-pruning workflow for machine translation. This template corresponds to the workflow shown in Figure 1.1. Templates should specify the types of input data that they are able to process. In this case, the template takes as input a plain text corpus and a roster of kernel rules (generated by a different workflow). An instance can be created simply by binding these two inputs to specific data. In Figure 1.3, the input is specified as WSJ-2001 and KR-09-05. The specification of the workflow instance can be quite compact since it consists of a template identifier and a set of bindings for its inputs. This compact specification can be turned into a fully expanded instance, shown on the right-hand side of Figure 1.3 and corresponding to the workflow in Figure 1.1. Executable workflows fully specify what needs to be executed, where, and in what order. They also include data movement steps as required by the computations. In the example shown in the figure, the initial and final stages may be performed in local machines, while the most computationally intensive stages could be executed in a shared resource (e.g., a cluster). The final results as well as some intermediate results may be recorded in shared data repositories. An executable workflow, shown in Figure 1.4, can be automatically generated from the workflow instance by analyzing the execution requirements of each step and mapping them into the particular data storage and computation resources available at the time of execution.

While these different stages make useful distinctions, they are not meant to be rigid. For example, a workflow template may be partially instantiated in that some of its data type placeholders may already be assigned to existing data. A partial instantiation can be used to specify an analysis of a data set against a standard invariant data set. It could also be used to specify parameter settings for some of the components of the analysis. The workflow creation process needs to be flexible across these stages.

Reuse is greatly facilitated by distinguishing these different levels of abstraction, from generic reusable workflow templates, to specific data-processing workflow instances, to execution-dependent workflows. Users can simply retrieve templates that are appropriate to their needs and specify the data to

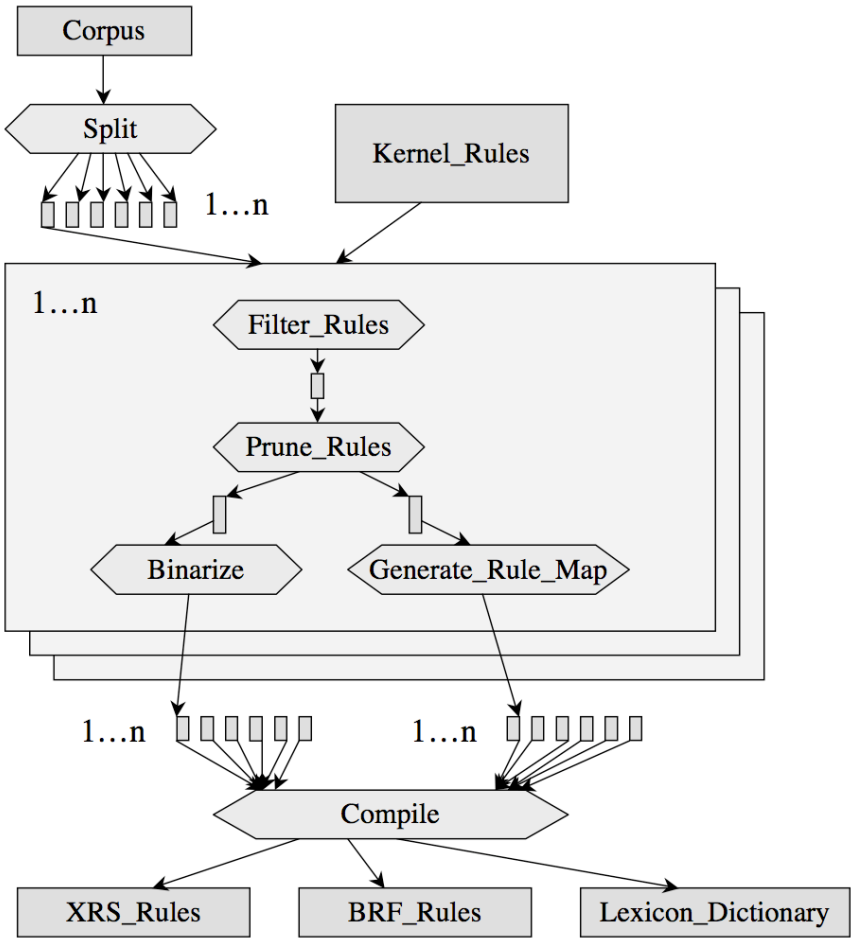

Fig. 1.2. A workflow template captures the structure of the workflow in a data-independent and execution-independent representation.

be processed. New types of workflows can be created by adapting or merging existing templates.

Validation is also facilitated through this three-stage process. Workflow templates can specify constraints on the types of input data that they can process. In creating workflow instances, users can be required (and guided!) to provide data that satisfy those constraints.

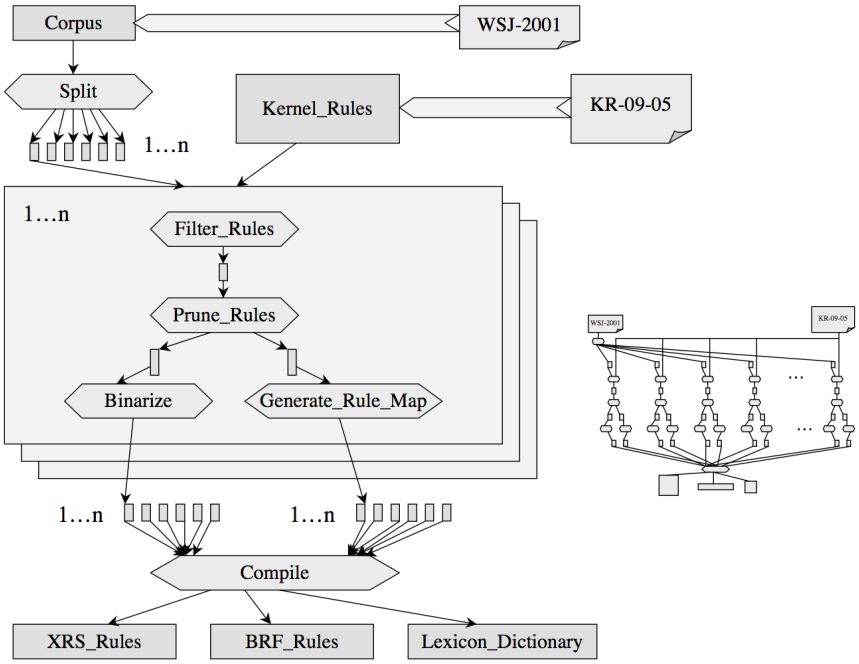

Fig. 1.3. A workflow instance specifies the data to be processed by a workflow template in an execution-independent representation.

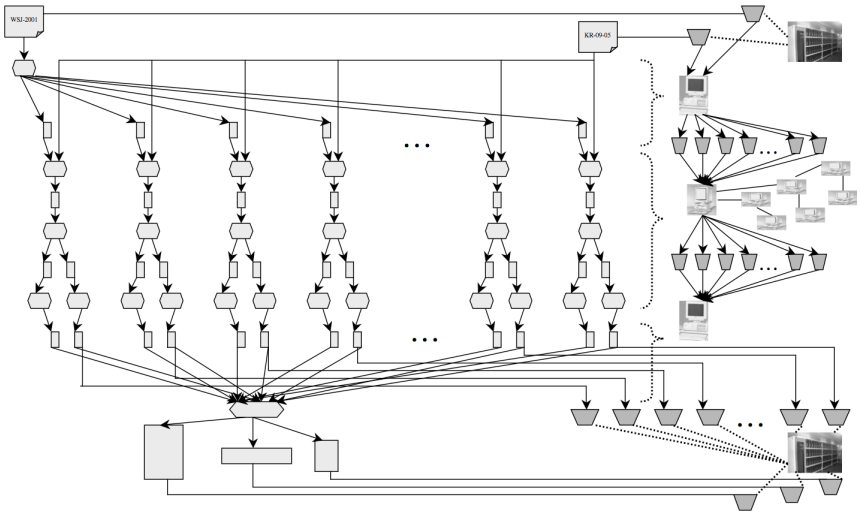

Fig. 1.4. An executable workflow specifies where the data are stored and processed and may include additional steps to move data as required for computation and storage.

1.4 Semantic Representations of Workflows to Support Assisted Composition

Figure 1.5 shows an overview of the kinds of ontologies that we use to represent workflows. We use the Web Ontology Language (OWL) [7], a W3C recommendation that now has several widely available editors and reasoners for efficient inference. Application-specific ontologies, shown in the middle of Figure 1.5, include definitions of terms and relations that are useful to describe different data properties in the domain at hand. These domain terms can be organized in classes described in terms of their relations to other classes as well as by their class-specific properties. For example, a class of data objects defined for the machine translation workflow is "Kernel_Rules." This can be a subclass of a more general class, "Translation_Rules." These domain terms are then used to define the classes of data required for each component, as well as the data they create. In our example, the definition of the component "Filter_Rules" would state that one of its inputs is a file of type "Kernel_Rules." Components can be organized by component types, also organized in classes. For example, a generic class "Process_Rules" may be defined as having at least one input that is of type "Translation_Rules." Given this definition, the component "Filter_Rules" belongs to that more general class. Workflow templates and instances can be specified by using these component classes and descriptions. Since the output of "Filter_Rules" is specified to be a set of "Translation_Rules," the file that results from the second component of the template is of that type. All these application-specific ontologies can be specializations of application-independent definitions, shown at the top of Figure 1.5. These generic ontologies can describe the relationships between components, data, and workflows, as well as their properties. At the bottom of the figure, external catalogs can be used as repositories of data, components, and executed workflows. All these catalogs can be indexed by the ontologies, where any metadata attributes would correspond to terms defined in the ontologies.

Semantic workflow representations support workflow composition in several ways. First, the ontology definitions of classes and properties can be used to check that newly created workflows are valid. In our running example, a workflow instance may be noted to be invalid if KR-09-05 is not recognized to be an object of type "Kernel_Rules," which may be either directly stated in or deduced from the metadata attributes of KR-09-05. The consistency of newly created workflow templates can also be checked against the definitions in the ontologies. The second use of semantic workflow representations is for retrieval from workflow libraries. Using ontology-based query languages, a workflow template can be retrieved by providing a description of the features sought. For example, one may search for workflows that take "Kernel_Rules" and include at least one component of type "Process_Rules." With this description, the template in Figure 1.2 would be found.

Semantic workflow representations allow a better integration of data management systems and workflow systems by supporting detailed descriptions of

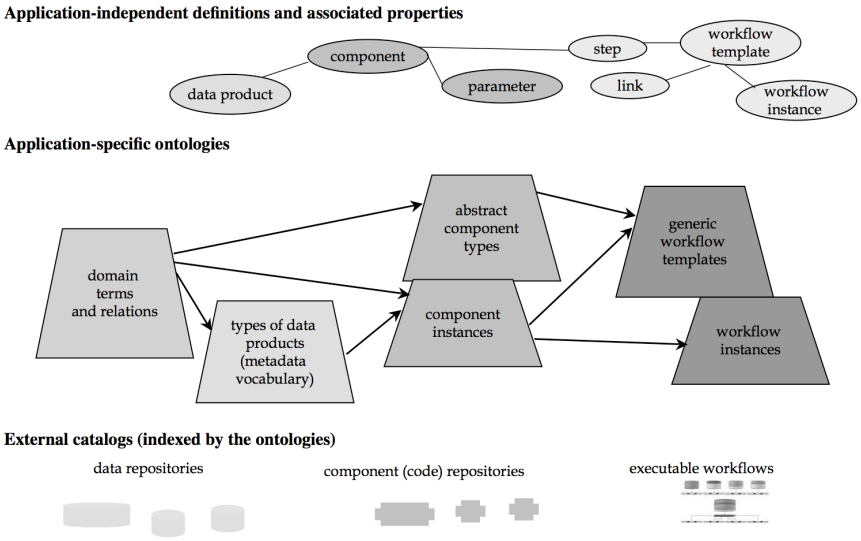

Fig. 1.5. Semantic workflow representations include application-specific ontologies organized as specializations of generic application-independent definitions. The ontologies can be used to index repositories of data, components, and executed workflows.

the requirements of a given workflow. Data can be described in a workflow using metadata attributes to more or less detail. A workflow specification could include intensional descriptions that describe the data required or extensional descriptions that refer to specific data sets to be used. These descriptions could be used to retrieve the data by matching the description against a catalog. The descriptions of the data could also prompt the automatic assembly or generation of the data, perhaps by firing off another workflow. Note that these representations can describe the data required or produced by a workflow at different levels of abstraction, including regarding its format and storage. The same data can be stored in alternative formats, such as a database, or perhaps in a set of files, each structured either as a table, an XML structure, or a labeled list. The same data may be replicated in different locations, perhaps with alternative file breakdowns or directory structures. In some cases, data formats have a major influence on the efficiency of the workflow, whereas in other cases data formats do not affect the logical analysis in the workflow and their handling and conversions may be completely abstracted away.

Rich metadata descriptions of intermediate or final workflow results can be automatically created based on the representations of the template and instance that generated them. This supports an important requirement of scientific domains in terms of documenting how any data are generated and

what their properties are. It also supports automatic workflow completion, as we explain in the next section.

In summary, semantic workflow representations can support workflow composition in several ways. The reasoners can use the ontologies and associated definitions to ensure the consistency of user-created workflows. During workflow creation and retrieval, ontology-based query languages can be used to find relevant workflows and components based on their properties.

1.5 Automatic Completion of Workflows

In our work, we have used search and planning algorithms from artificial intelligence to design workflow completion and automatic generation algorithms. The abstraction levels and the semantic representations discussed so far turn out to be useful in supporting a more flexible framework for automatic completion of workflows.

Complete automation is desirable in the stage of creating an executable workflow from a workflow instance. This is possible when the execution requirements of the workflow components are specified, and the system can query the execution environment to find what resources are available for execution. Important challenges include optimizing the completion time of any given workflow, considering resource assignment trade-offs across many workflows, and designing appropriate failure handling and recovery mechanisms.

Automation can also be used to complete underspecified workflow templates. Workflow templates are underspecified when they include abstract computation descriptions to be specialized during workflow instance creation. For example, a step in a workflow template may specify a class of component such as "Gridding." The specific gridding component to be used may depend on the nature of the data processed by the workflow. Workflow templates can also be underspecified in that they may be missing workflow components that perform data formatting, conversions, and other steps that are not considered critical to the analysis done in the workflow. As we mentioned, these are necessary ingredients of a workflow, yet the details of how these steps are performed may be irrelevant to the experimental design and to the scientist. Automatically adding these steps is possible when the format requirements for experiment-critical components are declaratively specified and when the component library includes appropriate components for doing the kinds of data processing required. The kinds of data processing needed may not be known until the data are specified and therefore would not typically be included in a workflow template. Once input data are specified, new data-processing steps can be added during workflow instance creation. Intermediate data products may also need to be converted for consumption of a subsequent step. In some cases, their format can be anticipated from the workflow template definitions and the new steps can be added during workflow instance creation. However, in other cases, the format of intermediate data products will only be known

once they are created during execution, and in those cases the insertion of data-processing steps will need to be interleaved with the execution process.

Full automation of the workflow composition process may be desirable for some kinds of workflows and application domains. Given a description of the desired data products, the task of creating a valid workflow from individual application components can require full-fledged automatic programming capabilities. Automatic workflow generation is manageable in domains where application components can be clearly encapsulated and described with detailed specifications of the component's outputs based on the properties of their input data. These specifications must include criteria for component selection and data selection when several alternatives are appropriate and where the quality of the workflow results may depend on mutually constraining choices. As an alternative to creating new workflows from scratch, fully automatic workflow generation can also be achieved by reusing workflow templates. This approach requires a library of workflow templates that reflects common analysis processes in the domain and is appropriately indexed according to the requirements that will be provided at workflow generation time, be they the desired results, the input data to be analyzed, or the types of components in the workflow.

1.6 Conclusions

The scale and complexity of scientific applications challenge current workflow representations and pose limitations on simple workflow composition tools such as graphical editors or authoring environments with limited user assistance. We have argued that workflow composition environments can greatly benefit from (1) semantic representations of workflows and their components, (2) workflow representations at different levels of abstraction, and (3) flexible automation in completing user-provided partial workflow descriptions. These novel capabilities can have broader implications beyond workflow composition in terms of increased automation and intelligent capabilities for workflow management and execution.

Acknowledgments

This research was supported in part by the National Science Foundation under grant EAR-0122464 and in part by an internal grant from the Information Sciences Institute (ISI) of the University Of Southern California. I am very grateful to Ewa Deelman for many fruitful discussions on the topics put forward by this chapter. I would like to thank members of the Southern California Earthquake Center and the ISI Machine Translation project for sharing with us their challenging workflow problems. I also thank other members of the Intelligent Systems Division and the Center for Grid Technologies at ISI.

References

1. K. Knight and D. Marcu. Machine Translation in the Year 2004. In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 965–968. IEEE Computer Society, New York, 2005.
2. J. Kim, M. Spraragen, and Y. Gil. An Intelligent Assistant for Interactive Workflow Composition. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 125–131. ACM Press, New York, January 2004.
3. P. Maechling, H. Chalupsky, M. Dougherty, E. Deelman, Y. Gil, S. Gullapalli, V. Gupta, C. Kesselman, J. Kim, G. Mehta, B. Mendenhall, T. Russ, G. Singh, M. Spraragen, G. Staples, and K. Vahi. Simplifying Construction of Complex Workflows for Non-Expert Users of the Southern California Earthquake Center Community Modeling Environment. *ACM SIGMOD Record*, 34(3):24–30, 2005.
4. S. Thakkar, J. L. Ambite, and C. A. Knoblock. Composing, Optimizing, and Executing Plans for Bioinformatics Web services. *VLDB Journal, Special Issue on Data Management, Analysis and Mining for Life Sciences*, 14(3):330–353, 2005.
5. D McDermott. Estimated-Regression Planning for Interactions with Web Services. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *6th International Conference on Artificial Intelligence Planning and Scheduling*. AAAI Press, Menlo Park, CA, 2002.
6. S. McIlraith and T Son. Adapting Golog for Programming in the Semantic Web. In *Fifth International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 195–202. In press, 2001.
7. OWL Web Ontology Language. `http://www.w3.org/TR/owl-features/`.