Program Synthesis from Workflow-Driven Ontologies

Leonardo Salayandia
leonardo@utep.edu

Steve Roach
sroach@utep.edu

Ann Q. Gates
agates@utep.edu

Department of Computer Science
University of Texas at El Paso
El Paso, Texas 79902, USA

***Abstract* – An approach that results in the development of Workflow-Driven Ontologies (WDO) (called the WDO approach) allows domain scientists to capture process knowledge in the form of concepts as well as relations between concepts. Program synthesis techniques can be employed to generate algorithmic solutions by transforming the process knowledge expressed in the WDO as concepts and relations to variables and functions and computing unknown variables from known ones based on the process knowledge documented by the domain scientist. Furthermore, the algorithmic solutions that are generated by program synthesis potentially can support the composition of services, which result in the creation of executable scientific workflows.**

The ultimate goal of this work is to provide an end-to-end solution for scientists beginning with modeling the processes for creating work products in terminology from the scientist's own domains and ending with component-based applications that can be used to automate processes that can advance their scientific endeavors. These applications can exploit distributed components that are supported by current cyber-infrastructure efforts. This paper discusses extensions to the WDO approach that support program synthesis. To elucidate this scenario, an example related to earth sciences is presented.

I. INTRODUCTION

Recent efforts on cyber-infrastructure (CI) development have resulted in increased availability of resources for scientists to conduct research in new ways [1]. According to the National Science Foundation CI Council, cyber-infrastructure refers to the infrastructure that integrates "hardware computing, data and networks, digitally-enabled sensors, observatories and experimental facilities, and an interoperable suite of software and middleware services and tools." As a consequence, scientists now need assistance with integrating these new technologies into their practices. Salayandia et al. in [2] describe the Workflow-Driven Ontologies (WDO) approach that has been developed with the intention of enabling domain scientists to create and customize their own applications by composing resources that are accessible over CI. A WDO consists of an upper-level ontology that can be extended by domain scientists to capture process knowledge or knowledge about how scientific tasks are conducted. The structure of the WDO upper-level ontology serves as guidance to the scientist in documenting process knowledge that is amenable to automatic workflow specification generation. The workflow specifications that are generated from WDOs are referred to as *model-based workflows* (MBW) [3]. The MBW specifications produced from the WDO approach have a graphical representation that is an effective tool for communicating process knowledge among domain scientists [4]. Furthermore, MBW specifications can also be used as a communication tool between scientists and technologists to create component-based applications that implement process knowledge from computational components that are openly available for scientific use over CI.

To summarize, the WDO approach supports the scientist's ability to create and manipulate models (or ontologies) about the processes of their domain using domain-specific terms and to subsequently extract workflow specifications from these models. This paper describes the application of program synthesis techniques to generate MBW specifications from the process knowledge captured in workflow-driven ontologies.

The visionary goal of the National Science Foundation and other international agencies is to enable new ways of conducting science through CI [5]. The ultimate goal of the work described in this paper is to provide an end-to-end solution for the scientist where the result of the scientist's efforts to create a model of a process is not a specification that a technologist must implement, but rather, an application that can be generated and executed on the fly and that can be customized according to the scientist's specific needs. This scenario is elucidated through an example from the Earth sciences community.

The rest of the paper is organized as follows. Section 2 provides an overview of the WDO approach. Section 3 describes the program synthesis technique that has been chosen for this work. Section 4 explains how probabilistic interval and fuzzy uncertainty can be propagated via synthesized data processing programs. Section 5 describes the generation of workflow specifications by utilizing program synthesis on process knowledge captured through the WDO approach. Section 6 presents a scenario related to earth science to elucidate an end-to-end solution from capturing process knowledge to executing scientific workflows, and Section 7 presents conclusions and future work.

II. WORKFLOW-DRIVEN ONTOLOGIES

Our previous work has resulted in the development of the Workflow-Driven Ontologies (WDO) approach [2]. The intention of WDO is to facilitate the capture of process knowledge by scientists directly with terms from their own domains of expertise. Having well documented process knowledge has several benefits:

- It helps domain experts communicate and share operational process knowledge (i.e., how-to knowledge) with colleagues and students;
- it helps domain experts explain their requirements to colleagues from other fields, e.g., computer programmers, who can help build tools automate processes; and
- it helps domain experts find commonalities between their processes and the processes of other domains that can lead to improved inter-disciplinary collaboration.

In the WDO approach, process knowledge is captured as ontologies expressed in the Web Ontology Language (OWL). OWL provides a formal framework to document ontologies based on Description Logics [6]. Given the formalisms provided by OWL, the process knowledge documented through the WDO approach can be leveraged towards the automatic generation of workflow specifications that can potentially be translated to executable scientific workflows.

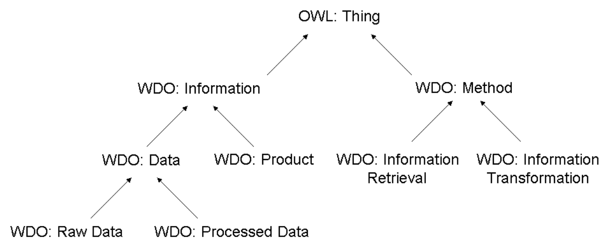

Fig. 1 Class hierarchy defined in the WDO-upper-level ontology

The WDO approach is based on the definition of an upper-level ontology that can be extended by domain scientists to capture process knowledge. Part of the upper-level ontology, referred to as the *WDO-upper level ontology*, is presented in Figure 1. This ontology presents a basic categorization of concepts into *Information* and *Method*. *Information* refers to the data and dataset concepts that are specific to the domain being addressed. *Method* refers to the algorithms, functionality, or actions that are available in the domain to transform or manipulate *Information*. In addition to guiding scientists through the process of identifying and categorizing information and method concepts, scientists are guided to identify input and output relations between concepts. As a result, process knowledge in the WDO approach takes the form of *Information* being input into *Methods*, and the *Methods* outputting some transformed *Information*. Figure 2 presents a compounded workflow specification where some data concept *is input to* a method concept and this method *outputs* some transformed data represented by a second data concept, and this basic workflow pattern is repeated until some desired target data is reached.

As a side effect of applying the WDO approach, the *Method* concepts defined in the resulting ontology have a unique input and output signature. The application of the WDO approach to capture process knowledge results in concept and relation descriptions of the form:

Information » *Method* > *Information*

The » sign in this pattern means that a *Method* can have multiple *Information* inputs, and the > sign means that a *Method* has a single *Information* output.

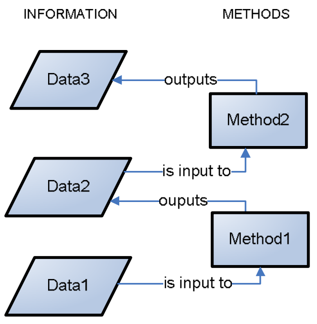

Fig. 2 Basic WDO relations used to document process knowledge

This concept and relation form simplifies the process of automating workflow generation by uniquely identifying the *Information* concepts related to a given *Method* on a workflow specification. The automatic generation of workflow specifications from WDO ontologies was initially explored in [2], and it is formalized here through the application of established program synthesis techniques.

III. Program Synthesis

As presented in [7-11], program synthesis tries to solve problems of the following general form:

- there are some known values $x_1, \ldots, x_n$, e.g., values that have been measured;
- there are some unknown values $y_1, \ldots, y_m$ that are of interest; and
- there is a known relation between the known values and the unknown values, e.g., $F(x, y) = 0$, where F is some known expression.

The question is: given this knowledge, i.e., the known values and the known relations, is it possible to compute the unknown values, and if possible, how?

To illustrate these types of problems, the *triangle* example is presented as discussed in [7, 8].

A triangle is described by its angles A, B, and C, and its side lengths a, b, and c. The following relations are known:

(1) $A + B + C = \pi$, i.e., the sum of the angles is 180º, or π radians;

(2) $a^2 + b^2 - 2ab(\cos C) = c^2$, i.e., the cosine theorem, with similar equations for a^2 and b^2; and

(3) $a/(\sin A) = b/(\sin B) = c/(\sin C)$, i.e., the sine theorem.

Suppose that the side lengths a, and b and the angle A are known, and it is necessary to determine the length of side c. As a result, the knowledge in this example problem is represented by relations (1), (2), and (3) and the values of a, b, and A; and the unknown value is c.

A solution to these types of problems was proposed by Tyugu and Mints [8-11]. The first stage of their approach is an analysis to determine which quantities are directly computable. As an example, consider the triangle problem presented earlier; if A and B are known, then the value of C can be computed by applying equation (1), as $C = \pi - A - B$. This computability relation is represented by using the implication notation:

$$A, B \rightarrow C.$$

Similarly, if A and C are known, B can be computed, and if B and C are known, A can be computed. As a result, the additional computability relations are obtained: $A, C \rightarrow B$ and $B, C \rightarrow A$. Additional computability relations follow from equations (2) and (3).

The second stage of the approach is to determine whether a given variable of interest is directly or indirectly computable from known variables, and if so, compute it. The *wave algorithm* determines the computable closure of the variable set. The first step is to add all the variables bound to ground instances to the set of computable variables, **C**. Then, find all the computability equations that have all their condition variables in **C** chosen, and the conclusion variable of each is added to **C.** This procedure is repeated until nothing new is added to **C,** which means that nothing else can be computed from the initial set of computability relations and the initially known variables. Since there are finitely many variables, this process will eventually stop. If the variable of interest is in **C,** then it can be computed; otherwise, it cannot. Furthermore, keeping track of the order in which conclusion variables are added to **C** can be leveraged to determine the order in which the computability relations need to be applied to compute the variable of interest. Consider the triangle example; if a, b, and A are initially assigned values, then the first iteration of the wave algorithm will result in adding variable B to the **C** set through a computability relation resulting from equation (3), i.e., a, b, $A \rightarrow B$. For the second iteration, the computability relation $A, B \rightarrow C$ from equation (1) can be utilized to add variable C to the **C** set. Finally, the computability relation a, b, $C \rightarrow c$ from equation (2) will result in the addition of variable c to the **C** set, at which point there are no more variables to add to **C** and the wave algorithm stops.

As a result, a program can be synthesized to compute c from a, b, and A as follows: first a computability relation resulting from the sine theorem, i.e., equation (3), is applied to compute B from a, b, and A; then a computability relation obtained from equation (1) is applied to compute C from A and B, and finally; a computability relation obtained from the cosine theorem, i.e., equation (2), is applied to compute c from a, b, and C.

Tyugu and Mints also showed that the computability relations produced in the first stage can be reformulated in logical terms. For example, the relation: $A, B \rightarrow C$ can be interpreted as the propositional formula $A \,\&\, B \rightarrow C$, where A, B and C are Boolean variables meaning that the corresponding quantities are computable from the inputs. In the triangle example, the corresponding propositional formulae, along with the known variables represent the knowledge base, and the goal is to determine whether a given variable of interest is deducible from the knowledge base.

IV. Propagating Probabilistic, Interval, and Fuzzy Uncertainty

In practice, the values of the input quantities $x_1,\ldots,x_n$ come either from measurement, or from expert estimation. In both cases, the input quantities are known with uncertainty.

For measured values, the exact probability estimation of the measurement error is sometimes known. In such cases, the actual value is known with probabilistic uncertainty [12, 13].

In other cases, only the upper bound Δ on the measurement error is known. In such cases, based on the measurement result X, it can only be concluded that the actual value of the measured quantity belongs to the interval

$$[X - \Delta, X + \Delta].$$

In other words, in such cases, the value is known with interval uncertainty [14].

For estimated values, experts often profile their uncertainty estimates by using imprecise words from natural language, e.g., “density is about 6.0.” A natural way to formalize this type of uncertainty is to use fuzzy techniques [15, 16].

Once the uncertainty in the inputs $x_1,\ldots,x_n$ is known, as well as the data processing algorithm transforming these inputs into the desired outputs $y_1,\ldots,y_m$, known uncertainty propagation techniques can be applied to derive uncertainty in the results $y_1,\ldots,y_m$ of data processing [12-18].

V. Applying Program Synthesis to WDO

As mentioned above, the application of the WDO approach to capture process knowledge results in concept and relation descriptions of the form:

$$\textit{Information} \gg \textit{Method} > \textit{Information}$$

More generally, this basic pattern can be mapped to relations of the form $F(A_1,\ldots,A_n) = 0$, where F is a function that corresponds to the *Method* concept and $A_1,\ldots,A_n$ correspond to the input and output *Information* concepts described in the basic pattern. From this mapping, the Tyugu and Mints program synthesis techniques can be applied.

Following the two-stage approach described previously, the first stage determines which quantities are computable from which. The WDO gives us this analysis directly. For example, consider the following WDO relation specification:

$$A, B \gg F > C$$

If *A* and *B* are known *Information* concepts, then *Information* concept *C* can be computed, and it is written as the computability relation: *A, B* → *C*. This transformation can be done for all the process knowledge captured as WDO relations.

The second stage of the approach is to determine whether a given variable of interest is computable from known variables, and if possible, how? This is done by applying the wave algorithm.

After a variable of interest is determined to be deducible from the knowledge base, and an ordering of formula applications has been determined by the wave algorithm, the formulae are mapped back to their original WDO relations form, and this ordering of WDO relations can be interpreted as an MBW for the *Information* concept of interest. Depending on the knowledge base used, a possibility is that there exist multiple orderings of formulae that will lead to the deduction of the variable of interest. Each of the alternative orderings will correspond to an alternative MBW specification. The wave algorithm may be extended to produce alternative solutions, and these alternative MBW specifications may offer important advantages when trying to implement executable workflows over existing components. For example, one alternative may not have all its corresponding components implemented or readily available.

VI. The Gravity Contour Map Scenario

Assume that a geoscientist wants to obtain a Contour Map of Bouguer Anomaly Gravity Data for a given region of interest, e.g., see [19]. The scientist starts by obtaining a WDO that documents process knowledge from the geophysics domain about how to generate such a map. A graphical representation of such a WDO is presented in Figure 3.

From this WDO, the following propositional formulae can be interpreted:

1. *SimpleBouguer* & *RegionOfInterest* → *Grid*
2. *Grid* → *ContourMap*

The scientist identifies *ContourMap* as the intended information concept desired, and identifies *RegionOfInterest* and *SimpleBouguer* anomaly data as the known information concepts. The application of the wave algorithm to this knowledge base will result in the conclusion that *ContourMap* indeed can be deduced. Furthermore, keeping track of the order in which conclusion variables are marked through the iterations of the wave algorithm will determine an ordering in which the propositional formulae should be applied to reach the *ContourMap* computation.

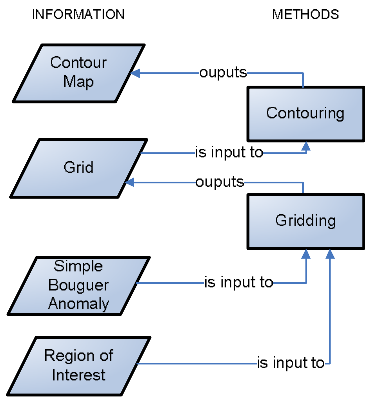

Fig. 3 WDO process knowledge to produce a Contour Map from Simple Bouguer Anomaly gravity data

In this case, there is only one possible ordering to compute the variable of interest, and so, the corresponding workflow specification will result in the following:

1. Apply the *Gridding* method using *SimpleBouguer* anomaly and *RegionOfInterest* information as input to produce *Grid* information;
2. Apply the *Contouring* method using the *Grid* information produced in the previous step as input to produce the target *ContourMap* information.

Notice that in this scenario, both *SimpleBouguer* and *RegionOfInterest* are necessary to produce *ContourMap*. If either of these concepts were unknown, the wave algorithm would determine that the *ContourMap* could not be produced from the given knowledge base.

This is a simplified example that is intended to demonstrate the use of the program synthesis techniques (originally proposed by Tyugu and Mints) in the WDO context. In general, the WDO approach is intended to be leveraged by scientists that are collaborating across multiple disciplines. Applying this program synthesis technique to WDOs that document process knowledge from multiple domains of expertise will hopefully result in the discovery of MBW specifications that are non-evident for an expert user of one particular domain.

VII. Conclusions and Future Work

The ultimate goal of this work is to enable new ways of conducting science through CI by providing an end-to-end solution for the scientist where the result of the scientist's efforts to create a model of a process includes the ability to generate an application that can be executed automatically. Although there is still a gap between the MBW specifications generated by this approach and the executable application that the scientist desires, this work makes a step towards separating the scientist user from the technical details of constructing the application program.

In addition, the work presented here provides the benefit of being able to deal with cross-disciplinary scenarios, where complex data-processing specifications can be generated using process knowledge from multiple scientific domains such as those supported by CI efforts.

General ontology specifications offer more sophisticated relations than the input/output WDO relations discussed here. For example, in OWL, a relation may be declared to be transitive, bi-directional, or to have a specific cardinality [20]. Furthermore, relations of the type *IS_A* offer a mechanism to model concept inheritance that is commonly supported in ontology specifications, including WDO ontologies. Some of these additional types of relations may prove useful in documenting domain process knowledge. Additional work is required to extend the program synthesis technique described here to take advantage of additional computability relations that may be extracted from these types of relations.

There are other program synthesis techniques that are more sophisticated and powerful. If all of the formulae for variables of interest can be oriented, rewriting approaches might be appropriate [21]. If the formulae are not all orientable, deductive synthesis using resolution [22] or tableau [23] can provide synthesis using the full expressiveness of first order logic. A common difficulty with first order logic arises from the search space explosion. Techniques for procedural attachment have been developed [24, 25], as have techniques for automatically generating domain-specific decision procedures for subsets of domain theories [26]. Applying these types of techniques to extract workflow specifications from WDO ontologies is left as future work along with investigation of more sophisticated ontological relations for use in the WDO approach. For now, the simplicity of the propositional-logic based approach proposed by Tyugu and Mints offers a clean, efficient implementation that maps nicely to the WDO approach.

Other work discussed in [27] addresses the need to provide feedback to the end-user, e.g., a scientist, with respect to the quality of the end-product resulting from a data processing application. By attaching provenance information to the end-product about the datasets and functional components used in its creation, the scientist can have access to additional information to assist in making an informed decision about the appropriateness of the end product. Since the program synthesis technique described here can result in the extraction of different MBW alternatives from the documented process knowledge in a WDO, integrating tools that aid the scientist in the decision process of selecting an alternative will become an essential component in the envisioned end-to-end solution to conduct science over CI.

ACKNOWLEDGMENT

This work is funded in part by the National Science Foundation (NSF) through the CyberShARE Center of Excellence, grant HRD-0734825. Additionally, the authors would like to thank Dr. Vladik Kreinovich for his advice in this work.

REFERENCES

[1] The Geosciences Network: Building Cyberinfrastructure for the Geosciences, http://www.geongrid.org, May 2006.

[2] L. Salayandia, P. Pinheiro, A.Q. Gates, and F. Salcedo, "Workflow-driven ontologies: an earth sciences case study," *Proc. of 2nd Intl. Conf. on e-Science and Grid Technologies (e-Science 2006)*, December 4-6, 2006, Amsterdam, Netherlands.

[3] L. Salayandia, P. Pinheiro, A.Q. Gates, A. Rebellon, "A model-based workflow approach for scientific applications," *Proc. of 6th OOPSLA Workshop on Domain-Specific Modeling*, October 2006, Portland, OR.

[4] Materials of the Summer Southwest Regional Cyberinfrastructure Workshop, August 10, 2007, El Paso, Texas, unpublished.

[5] Cyberinfrastructure Council 2007, *Cyberinfrastructure Vision for 21st Century Discovery*, http://www.nsf.gov/pubs/2007/nsf0728/nsf0728.pdf

[6] OWL Web Ontology Language, W3C Recommendation, February 2004, http://www.w3.org/TR/owl-features

[7] D.E. Cooke, V. Kreinovich, and S.A. Starks, "ALPS: A logic for program synthesis (motivated by fuzzy logic)," *Proc. of the FUZZ-IEEE'98 Intl. Conf. on Fuzzy Systems*, Anchorage, Alaska, May 4-9, 1998, Vol. 1, pp. 779-784.

[8] E. Tyugu, *Knowledge-based programming*, Addition-Wesley, Wokingham, England, 1988.

[9] E. Tyugu, *Algorithms and architectures of artificial intelligence*, IOS Press, Amsterdam, The Netherlands, 2007.

[10]G. Mints and E. Tyugu, "The programming system PRITZ," *Journal of Symbolic Computations*, 1988, Vol. 5, pp. 359-375.

[11]G.E. Mints and E.H. Tyugu, "Propositional logic programming and the PRITZ system," *Journal of Logic Programming*, 1990, Vol. 9, pp. 179-193.

[12]D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, Boca Raton, Florida, 2004.

[13]H. M. Wadsworth, (ed.), *Handbook of statistical methods for engineers and scientists*, McGraw-Hill Publishing Co., New York, 1990.

[14]L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001, ISBN 1-85233-219-0.

[15]G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, Prentice Hall, New Jersey, 1995.

[16]H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[17]W. Pedrycz, A. Skowron, and V. Kreinovich (eds.), *Handbook on Granular Computing*, Wiley (to appear).

[18]C. Hu, R.B. Kearfott, A. Korvin, and V. Kreinovich (eds.), *Knowledge Processing with Interval and Soft Computing*, Springer Verlag, Berlin-Heidelberg-New York (to appear).

[19]C.M.R. Fowler, *The Solid Earth, An Introduction to Global Geophysics*, Cambridge University Press, 2004.

[20]M.K. Smith, C. Welty, and D.L. McGuinness, (eds.), *OWL Web Ontology Language Guide*, W3C Recommendation, February 10, 2004, http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[21]F. Baader and T. Nipkow, *Term Rewriting and All*, That Cambridge University Press, 1999.

[22]M. Lowry and J. Van Baalen, "Synthesis of Efficient Domain-Specific Program Synthesis Systems," *Automated Software Engineering*, Volume 4, Number 2, Springer, April 1997 , pp. 199-241(42)

[23]Z. Manna and R. Waldinger, *The Deductive Foundations of Computer Programming*, Addison-Wesley Professional, 1993.

[24]M. Stickel, "Automated Deduction by Theory Resolution," *In Proc. of 9th Intl Joint Conf on Artificial Intelligence, IJCAI 85,* 1985.

[25]H. Bürckert, "A resolution principle for clauses with constraints," *Lecture Notes in Computer Science*, Volume 449, Springer, 1990.

[26]S. Roach and J. Van Baalen, "Automated Procedure Construction for Deductive Synthesis," *Journal of Automated Software Engineering*,

Springer Science and Business Media, Vol. 12, No. 4, pp 393-414, October, 2005.

[27] N. Del Rio and P. Pinheiro, "Probe-It! visualization support for provenance," *In Proc. of the 3rd Intl. Symposium on Visual Computing (ISVC 2007)*, Lake Tahoe, NV/CA, November 26-28, 2007.