A Geological Case Study on Semantically Triggered Processes

Yuanwei Qu(✉), Eduard Kamburjan, and Martin Giese

SIRIUS Center, University of Oslo, Oslo, Norway
{quy,eduard,martingi}@ifi.uio.no

Abstract. We present an approach to connect semantic descriptions of situations to program-based descriptions of processes. The main mechanism is a semantically formalised *trigger* that initiates a process. We demonstrate the viability of the approach by modelling scenarios and processes in petroleum geoscience.

1 Introduction

Semantic technologies are designed to build graph-based models to represent and reason about static relationships between entities and their properties, but not to represent dynamic behavior and changes. Although there is research focusing on formalisation of the concept of change [6] and top-level ontology frameworks to describe processes [1], there is still limited support for exploiting these models, e.g. in simulations, to build conditionals and loops to model the scenario that is described by the knowledge representations.

The distinction between utilizing semantic technologies to represent knowledge of dynamic processes and programming languages to implement the dynamics remains pronounced. The current work of [3] introduces a core programming language called 'Semantic Micro Object Language' (SMOL) to map the dynamic expression of an object-oriented programming language to the static description of semantic knowledge models, which demonstrates a structural approach for closing the gap between descriptive and computational modeling languages. Combined with the notion of programs as behavioural specifications, this enables hybrid semantic and behavioural models.

In this work, we propose an architecture to connect the modeling of processes in ontologies and the implementation of these processes in a simulator program. To illustrate our approach, we explore petroleum geological process modeling, which often involves complex reasoning, and for which various ontologies and knowledge graphs have been constructed to represent knowledge [4].

Traditional quantitative geological process simulation produces results that sometimes do not follow the geological knowledge and require experts to interpret the results. Such challenges can be avoided in knowledge-based qualitative simulation. Previous efforts [5] have shown that it is possible but still challenging to simulate geological processes based on formal domain knowledge. Geological

https://doi.org/10.1007/978-3-031-43458-7_18

processes usually take place when a set of conditions are met and lead to certain consequences. For each geological process, the corresponding conditions can be summarized as a trigger. Therefore, in addition to modeling the domain knowledge of the process, it is also important to model the condition that triggers the process and the entities related to the process. The program can thus utilize the trigger to query entities when a process starts.

Through our case study, we show how the notion of triggers is used to close the gap between semantic technologies and programming languages. This approach is not only easy to apply to other domains but also provides a clear structure for describing processes and implementing their simulation.

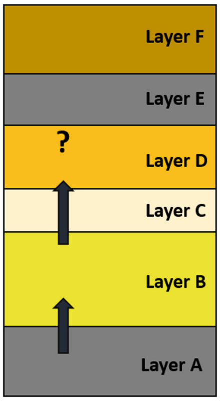
2 Architecture and Case Study

Our architecture is based on a split between so-called *event triggers* in the domain knowledge, and *event handlers* in the program. A process on some entity e is described two-fold. The ontology contains (a) the domain knowledge about e, (b) the domain knowledge when a process is triggered on e and (c) the domain knowledge of how this process relates to other processes. The program contains (1) the computational knowledge about e and (2) the computational knowledge of how the processes affect e.

The trigger bridges the gap between computation and description: while described in the domain, it can be used to query *inside the program* all entities where a process is about to start.

Domain Knowledge. We first introduce some basic geological terms. The geothermal maturation and migration of organic matter is a pivotal geological process for the energy industry, which makes it a fitting use case for our study. This process involves the transformation of subsurface kerogen into oil and/or gas, as a result of geothermal maturation, followed by its migration through stratigraphic layers. The kerogen is dehydrated organic matter and compacted by overlying rocks. The oil transformation window of kerogen is roughly around 90 to 150 °C, while the gas transformation window is mainly above 150 °C [2]. We molded these processes triggers, namely cooking and burning. The oil and gas may eventually be trapped and stopped by an impermeable layer or escape to reach the surface. Temperature and permeability are the two key factors in the process, as oil and gas need the sufficient temperature to be generated, and rock needs to be permeable to allow the petroleum to flow through.

In this use case, we are modeling the process of organic matter in the rock transferring into oil or gas and migrating upward toward the surface. As a proof of concept, we consider that the migration path is vertically upwards through the stratigraphic layers. Each stratigraphic layer is homogeneous as it is constituted by only one type of rock. The only barrier to stop the migration is the impermeable stratigraphic layer that petroleum can not flow through.



```
domain:SiliciclasticRock SubClassOf domain:Rock
domain:Shale SubClassOf domain:SiliciclasticRock
domain:Sandstone SubClassOf domain:SiliciclasticRock
domain:StratigraphicLayer SubClassOf domain:constitutedBy exactly 1 domain:Rock
domain:StratigraphicLayer SubClassOf domain:constitutedBy only domain:SiliciclasticRock
domain:CookingTrigger EquivalentTo
        (domain:constitutedBy some (domain:contains some domain:Kerogen))
  and (domain:temperature some xsd:integer[≥ 60 and ≤ 150])
domain:CookingTrigger SubClassOf domain:Trigger
```

```
//0 = none, 1 = deposit, 2 = kerogen, 3 = volatile
class Shale extends GeoLayer(hidden Int kerogenStatus)
  models(this.kerogenStatus == 1 || this.kerogenStatus == 2)
    "a domain:Stratigraphic_Layer;
     domain:constitutedBy [a domain:Shale];
     domain:constitutedBy [domain:contains [a domain:Kerogen]].";
  models "a domain:Stratigraphic_Layer;
          domain:constitutedBy [a domain:Shale].";
  Unit cook() this.kerogenStatus = this.kerogenStatus +1; end
  Unit burn() this.kerogenStatus = 3; end
end
class Sand extends GeoLayer(hidden Int porosity,
                            hidden Boolean hasHC)
  models(this.hasHC)
    "a domain:Stratigraphic_Layer;
     domain:constitutedBy [a domain:Sandstone];
     domain:constitutedBy [domain:contains [a domain:Kerogen]].";
  models "a domain:Stratigraphic_Layer;
          domain:constitutedBy [rdf:type domain:Sandstone].";
  Unit compact()
    if(this.porosity<3) then this.porosity = this.porosity +1; end
  end
  override Boolean canPropagate() return this.porosity < 3; end
end
```

Fig. 1. Illustration of migration in rock layers (left); part of the domain ontology of the scenario (upper); and the simulation code of shale layers in SMOL (lower)

Domain Model. The concepts of the stratigraphic layers and their rock types as well as domain knowledge about their properties and process trigger conditions are modeled in an ontology which is partly illustrated in the upper part of Fig. 1 upper part. Note that the ontology formalises only process *triggers* but not models of the process themselves.

A *trigger* is equivalent to a physical entity that meets the required conditions to trigger some processes. In this use case, the physical entity is the *stratigraphic layer*. A *stratigraphic layer* is a layer that consists of one type of rock. We considered two petroleum-relevant rock types, namely *sandstone* and *shale*.

In the perspective of Petroleum Geoscience, instead of fresh organic matter, *oil* and *gas* is transferred from *kerogen*. This thermal transformation process is modeled as *cookingTrigger*.

Programming with Domain Knowledge. So far, entities are described in the knowledge graph. To program with them, we use *semantically lifted programming (SLP)*. SLP is based on the idea of lifting a program state *during* program execution into a knowledge graph, in our case, the geological knowledge, and query it from within the very same program to use the domain knowledge for computations. Our implementation is based the SMOL [3] implementation of SLP. Our simulator has a general setup with classes for different kinds of layers and rocks, and can simulate different actions, such as the deposition of material, or its erosion. Here we show only the relevant part of the class for shale layers.

The class declares a field `status` that models the state of the hydrocarbons within. In this field, two methods operate: `cook`, for cooking and `burn` for burning. Note that this modeling is optimized towards computations: it requires no knowledge about the exact nature of these processes, or even a concept of hydrocarbons. Instead, it can be manipulated with simple arithmetic operations.

Let us turn our attention to the **models** clause. It extends the semantic lifting by adding additional knowledge about the *program object*. If the status is between 0 and 3, then additional axioms are added, namely that the object is constituted by shale and contains kerogen, i.e., deposited and non-volatile hydrocarbons. In case the status is different, only the axiom to describe is constitution is added.

This allows us now to interpret a *program object* in terms of the domain. To facilitate an effective connection, the simulator is round-based, where each round models the progressing of time by a certain step. During this time, some action (erosion or deposition) is performed and the simulator state updated accordingly. After each time round, the simulator queries *itself*, i.e., its own lifted objects, for triggers. If the query for all entities that are triggering a cooking process, returns a non-empty set, then on all these objects the `cook` method is called. This is shown in Fig. 1, in the lower box. We stress again that this query is not performed on external data, but on internal objects with external knowledge.

Our model implements a separation of concerns between domain modeling of static structures, done in OWL, and modeling of dynamic behavior, done in a standard object-oriented programming language. This reduces the redundancy between code and ontology and provides a very clear interface between these two worlds: the modeling bridge that interprets an object in the domain. Note, however, that a SMOL object is *not* a geological layer, instead it is explicitly linked to one. Similarly, a trigger remains purely in the domain, it is not a concept of the programming language, but a technique to implement guards in our setting.

The case study is available at smolang.org. We performed two experiments, based on different sequences of deposition and erosion and can comprehend the resulting hydrocarbon migration in great detail and with more domain knowledge than the prior approach [5]. We note however that for now but we consider a simpler setup with only a single column of uniform layers.

3 Conclusion and Future Work

We presented an architecture to model processes by connecting knowledge and computation: the same concept is modeled once in a knowledge graph and once in a program, where the program entities can be connected with the knowledge graph using semantical lifting. A process is modeled as (a) a *trigger*, which is a domain description of entities where a process can start (e.g., cooking starts), and (b) a method, a program description of the effects of the process (cooking itself). An event handler is used to repeatedly query the program for triggered entities, on which the method is subsequently executed.

This is the first work on modeling and simulating a geological process by using semantic technologies and a programming language. We plan to extend this work to model more complex geological processes and apply this to other domains as a general method.

References

1. Arp, R., Smith, B., Spear, A.D.: Building Ontologies with Basic Formal Ontology. MIT Press, Cambridge (2015)
2. Bjørlykke, K.: Source rocks and petroleum geochemistry. In: Bjørlykke, K. (ed.) Petroleum Geoscience, pp. 361–372. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-642-34132-8_14
3. Kamburjan, E., Klungre, V.N., Schlatte, R., Johnsen, E.B., Giese, M.: Programming and debugging with semantically lifted states. In: ESWC, vol. 12731 (2021)
4. Ma, X.: Knowledge graph construction and application in geosciences: a review. Comput. Geosci. **161**, 105082 (2022)
5. Yu, I.C., et al.: Subsurface evaluation through multi-scenario reasoning. In: Patel, D. (ed.) Interactive Data Processing and 3D Visualization of the Solid Earth, pp. 325–355. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-90716-7_10
6. Zamborlini, V.C., Guizzardi, G.: An ontologically-founded reification approach for representing temporally changing information in OWL. In: COMMONSENSE (2013)