Event Modeling for Reasoning of Consequences

Haroldo R. S. Silva[a], Fabrício H. Rodrigues[b] and Mara Abel[c]
Informatics Institute, Federal University of Rio Grande do Sul (INF-UFRGS),
Av. Bento Gonçalves, 9090 - Agronomia, Porto Alegre - RS, 91540-000, Brazil
{*hrssilva, fabricio.rodrigues, marabel*}*@inf.ufrgs.br*

Keywords: Ontology, Conceptual Modeling, Events, Reasoning, Semantic Web, OWL, SHACL.

Abstract: The modeling of events is crucial in several domains in which the temporal evolution of data supports decision-making, but the representation limitations in the state of the art in conceptual modeling are still a barrier to software application development. Current solutions fail to reconcile behavior expressiveness, reuse, and technological compatibility. This work considers event modeling under the approach of ontologies and focuses on the reasoning behind inferring the consequences of events. We propose the use of rule description languages to improve traditional ontology reasoning with interpretation capabilities of specific semantics, preserving the utility of current technologies (by not depending on non-analyzable descriptions, either by representational, modeling, or technological choice) while inferring in ways that are not possible with conventional axioms. During this work, we explore solutions compatible with the Semantic Web to represent the behavior of events, resulting in an OWL representation of an event model supported by SHACL-SPARQL inference and consistency check. We demonstrate our proposition by importing the resulting model to a domain ontology of the O&G industry and showing how the event consequences inferred affect a query over the oil flow.

1 INTRODUCTION

Ontologies are extremely useful tools for complex domains where ambiguous concepts and implicit semantics exist. We can apply ontologies as computational artifacts (referred to during the remaining of this work simply as "ontologies") for various purposes, be it interoperability of databases, semantic search, or information discovery through logical inference (part of a toolset of ontologies, together with consistency check, under the name of "reasoning"). Among the computational representations capable of reasoning, OWL (Web Ontology Language) is one of the most widely used languages for ontology representation. However, despite its expressiveness, not everything in the world of ontologies can be represented through it. Some representation limitations of OWL are discussed in (Keet, 2020), where other languages are explored to address these deficiencies, although such solutions leave the *semantic web* environment.

1.1 Discussion's Approach

The utility of event reasoning spans multiple aspects, be it extracting what events have occurred and which are their participants, which can, for example, help autonomous driving in its decision-making process, mainly in such a domain where these elements are implicitly contained in the driving environment and cannot be directly observed (Xue et al., 2018); or be it predicting future events, which can be done by reasoning over the causal relationships of prior events (Lei et al., 2019). In most cases, this reasoning is done through specific software implementation or by utilizing temporal logic. We will discuss a method compatible with ontologies without requiring external computational tools or domain-specific modeling.

This work discusses a solution to address the shortcomings of OWL's reasoning of temporal information, specifically events (also called Occurrents, Perdurants, or Processes). For this, we use the inference capability of SHACL (Shapes Constraint Language) for event reasoning. The presented solution focuses on inferring the consequences of event occurrences by proposing rules that allow OWL models to describe how events affect their participants and the concerned objects of the world. To define such rules, we base our modeling on upper-level types of events

[a] https://orcid.org/0009-0003-0594-4852
[b] https://orcid.org/0000-0002-0615-8306
[c] https://orcid.org/0000-0002-9589-2616

Silva, H. R. S., Rodrigues, F. H. and Abel, M.
Event Modeling for Reasoning of Consequences.
DOI: 10.5220/0013440400003929

In *Proceedings of the 27th International Conference on Enterprise Information Systems (ICEIS 2025) - Volume 2*, pages 71-82
ISBN: 978-989-758-749-8; ISSN: 2184-4992

based on ontological conservation (Rodrigues et al., 2020) to not create undesired logical consequences.

2 BACKGROUND

This study requires an understanding of what an ontology is, how ontologies are computationally treated, and how to represent events in an ontology. For Computer Science, an ontology is a formal specification of a shared conceptualization of reality (Studer et al., 1998). Computationally, this formalism is translated into conceptual models, traditionally represented in RDF/XML (Resource Description Framework) format, but with a wide range of alternative formats, most of which are standardized by the W3C.

2.1 Events in Ontology

Ontologies can represent both entities that exist entirely at each moment they exist, called *Continuants* or *Endurants*, and entities that exist in temporal parts, not being entirely present at any single moment in time, known as *Events*, *Occurrents*, or *Perdurants* (Guizzardi et al., 2013). This view also culminates in events having *different* temporal parts at different times, such that, at present, some of their proper parts can be missing (Masolo et al., 2003).

Another important concept for events is *participation*: events are entities that involve continuants as participants (Rodrigues and Abel, 2019; Bennett, 2002; Davidson, 1969), besides being entities directly related to time, they derive their spatial characteristics from their participants (Quinton, 1979).

We will refer to *things that happen in time with the participation of continuants* as *Events*. Although this definition is unclear as to what it is for something to *happen*, formalizing this definition is outside the scope of this work. For the sake of the reasoning approach we present, the notion about what can be said when an event happens is more interesting, as such, we will lean towards the definition presented in (Guizzardi et al., 2013), where events are transitions between situations that transform a portion of reality, although we will be very lenient on the need of representing such situations directly. Still following (Guizzardi et al., 2013)'s definitions, events *existentially depend* on objects and can be either *atomic*, and directly depend on an object, or *complex*, and directly depend on their proper parts (and indirectly on the objects of those parts).

2.2 Ontologies and the Semantic Web

The Semantic Web is an extension of the World Wide Web where information is given well-defined meaning (Berners-Lee et al., 2001), in this sense, the role of ontologies on semantic interoperability is of great interest in the semantic web. Given this circumstance, various representation languages, such as Terse RDF Triple Language (Turtle) and Web Ontology Language (OWL), were created and formalized for use in the Semantic Web. Currently, the OWL format is widely used in applications. Its variations that support Description Logic are important as they allow reasoning for inferences in the specified model (Horrocks et al., 2003). In the Semantic Web, RDF, RDFS, and OWL represent an evolutionary trend (culminating in OWL) of a simple graph reference model; a simple vocabulary and axioms for object-oriented modeling; and knowledge-based oriented constructs and axioms (Ding et al., 2007).

We will focus this work's proposition on technological compatibility, given the importance of the Semantic Web and, proportionally, OWL in it. As such, we explore possible implementations that can be applied without specific technologies that would conflict with the environment created by the Semantic Web. We will abstain from utilizing technologies not standardized by the W3C. With this in mind, the proposed solution to the problem we tackle during this work (reasoning applied to event consequences) will require that we model our events and implement our reasoning approach through some preexisting technological solution of the Semantic Web.

3 RELATED WORKS

This section compares several technological solutions that deal with events from a reasoning standpoint. In the ontology state-of-the-art, the modeling of events is receiving plentiful attention, with widely adopted ontologies like UFO-B (Guizzardi et al., 2013) and BFO (Otte et al., 2022) offering constructs for occurent representations. However, it is rare to find technological approaches that implement reasoning solutions to complement standard OWL capabilities for event information extraction through inference. Table 1 summarizes the comparison made, given the following criteria:

- **Technology Compatibility.** The ability of the solution to integrate with the already widely adopted Semantic Web technological stack. This criterion can assume some value between *None*, describing a solution that does not utilize the technological

stack; *Partial*, where the solution adopts the technological stack but increments it in some way; and *Full*, where the proposed solution only utilizes technologies available for the Semantic Web. This also means we cannot categorize software solutions as *Full* under this criteria.

- **Event Coverage.** The capacity of a solution to represent (or not represent) a broad selection of events. This criterion can assume some value between *Specific*, where the solution works only for the set of events demonstrated by the authors; *Strict*, where the solution works for a set of events bound by strict rules; and *Broad*, where the solution covers a wide set of events, generally bound by an undefined upper limit. Since this criterion is subjective, we will adopt the following rule as the separation line between *Strict* and *Broad*: if the solution can only describe events from its domain, it is *Strict*, if we can extrapolate the solution to other domains, or it is domain agnostic, the solution is *Broad*.
- **Ontological Commitment.** How specific is the ontological commitment of the proposed solution, that is, how committed the solution is to a specific worldview. To define this, we adopt a minimum set of requirements that describe an event: an event can have Continuants as participants, an event exists in time, and a description of events does not make additional assumptions about the role of Time in the model. This criterion can assume some value between *Loose*, where the solution is loosely committed, and we can apply it to different worldviews that adopt our minimum criteria; *Half*, describing a solution that enforces its worldview to some aspect of the world, exceeding our minimum criteria; and *Full*, describing a solution that only works when fully enforcing its worldview, which, in most cases, imply the adoption of a well-founded top-level Ontology.
- **Domain.** What domain does the solution apply to, or whether it is agnostic. This criterion is closely related to **Event coverage**.

3.1 Creation, Destruction and Modification Events

Events regarding the creation, destruction and modification of entities are topics of discussion in ontology modeling. UFO explores such concepts through the specialization of *Participation* (as subclasses of event) in *object creation*, *object destruction* and *object change* (Benevides et al., 2019; Guizzardi et al., 2016). These definitions brace themselves in the reification of situations, in which creation events require that a created object is not present in the initial situation of the event and must be present in the final situation of the event; destruction events are the opposite, where the object is present in the initial situation of the event, but not in its final; and modification events require that the object is present throughout the situations, but the properties of the object have changed.

Earlier versions of BFO also contemplated these types of events, in this case, by specializations of the inverse relation to *participation*, *involvement*. The relations that contributed to describing such events were *creation*, when an event created a continuant; *destruction*, when an event destroyed a continuant; *sustaining in being*, when the event collaborated to the continued existence of the continuant; and *degradation*, when the event collaborated for the continuants eventual destruction (Smith and Grenon, 2005).

3.2 Current Limitations

Although the use of rule-based reasoning proves itself useful to events (Padilla-Cuevas et al., 2021; Mepham and Gardner, 2009; Zhong et al., 2012; Anicic et al., 2011; Li et al., 2019; Pedrinaci et al., 2008), some works end up, either extending semantic web available engines (Mepham and Gardner, 2009; Li et al., 2019), utilizing non-standard (again, for the semantic web) rule languages (Anicic et al., 2011; Pedrinaci et al., 2008) or implementing a specific reasoner (Li et al., 2020). When the proposed solution is fully compatible with the semantic web (Padilla-Cuevas et al., 2021), it does not propose a general model that works in different domains.

The model by (Ermolayev et al., 2008) is a solid event-focused ontology that relies solely on semantic web technologies and OWL axiomatization, lacking rule-based reasoning capabilities. Similarly, (Valadares Vieira et al., 2025) models events using upper-level types related to geological processes but also lacks a reasoning approach. Currently, no proposals effectively operationalize ontological event descriptions without additional tools or specific extensions.

This work will focus on the issue of event reasoning inside the *semantic web* toolkit, specifically for reasoning on event consequences. We will borrow some concepts regarding *event patterns* from (Anicic et al., 2011) to define types of events with shared behavior between all their individuals. Additionally, inspired by the use of SQWRL to supplement SWRL's shortcomings (Mepham and Gardner, 2009), we will utilize SHACL, as a rule-based language, together with SPARQL, through SHACL-SPARQL advanced

Table 1: Related Works Comparison.

Title	Technology Compatibility	Event Coverage	Ontological Commitment	Domain
A Core Ontology for Business Process Analysis	Partial	Broad	Half/Full	BPM
A Method of Emergent Event Evolution Reasoning Based on Ontology Cluster and Bayesian Network	None	Specific	Loose	Emergency Scenarios
An Ontology of Environments, Events, and Happenings	Full	Strict	Full	DEDP
ETALIS: Rule-Based Reasoning in Event Processing	None	Broad	Loose	Agnostic
Event ontology reasoning based on event class influence factors	None	Broad	Half	Emergency Scenarios
Implementing discrete event calculus with semantic web technologies	Partial	Broad	Half	Web Services
Ontology-Based Context Event Representation, Reasoning, and Enhancing in Academic Environments	Full	Specific	Half/Full	Academic
Time event ontology (TEO): to support semantic representation and reasoning of complex temporal relations of clinical events	Partial	Strict	Half	Medical
Portions of Matter and Their Existential Events: An Ontology-Based Conceptual Model	None	Specific	Full	Geology

features (Knublauch et al., 2017), making it unnecessary to implement a software solution to tie both of them.

4 DEVELOPING A REASONING APPROACH

This work aims to propose a domain-agnostic method to infer the consequences of events while not requiring the modeler to utilize a description language external to the usual development stack for the semantic web. The approach is not intended for the final user of applications but for the modeler, or developer that connects data to the model. To be able to do so, we propose a modeling approach for events compatible with the triple-based description of ontologies.

To allow for reasoning on events as transitions to a broader selection of worldviews, the proposed modeling approach tries to make the minimum amount of compromises with how to model the surrounding context. An important distinction to keep in mind is between entities that have temporal parts, called *Occurents* or *Perdurants*, and entities that are wholly present in every instant, called *Continuants* or *Endurants*. *Events* are occurrents, and their participants are continuants. Furthermore, the basis of our modeling approach is the Aristotelian Ontological Square, which categorizes continuants into *Substantials*, which are existentially independent entities (e.g., a ball, a cat) and *Accidentals*, which are entities that depend on substantials and characterize the ways they are (e.g., color, size). Based on that, we characterize events and their types according to how the accidents of the participants vary during the event.

Considering that our interest in events in this work refers to the consequences of these events in the world, we will characterize events regarding their upper-level types. These types are the following: States, where the only change in situations is their temporal position; Simple Changes, where the qualities of participants change, but their identity is preserved; Transformations, where the qualities and identity of participants change; and Existential Occurrents, where the existence of participants change (Rodrigues et al., 2020).

To work with these upper-level types, we decided to break them down into simpler events that containerize their behavior on a "building blocks" approach. The identified event types are Creation Event, Destruction Event, Relation Modification Event, and Quality Modification Event. With these, we can represent *Existential Occurrents* with creation and destruction events, *Simple Changes* with relation and quality modification events, and *Transformations* with a combination of them all. This approach tries to minimize the intersection between the events while allowing for more granular control of expressivity.

Additionally, the behavior represented by such classes creates some dependencies with their participants, their types, and some relations. This dependence is slightly different, due to being more general, than the usual existential dependence between events and their objects, but represents existential dependence. As such, the following predicates emerge:

- Creating an individual by a *Creation Event* depends on the class of the created individual.
- The destruction of an individual by a *Destruction Event* depends on the destroyed individual.

- The gain and loss of a relation depends on the relation, the subject of the predicate, and the object of the predicate.
- The gain and loss of quality depend on the class of the quality, the individual that is (or will be) the bearer, and the relation of inherence.

Although it is possible to represent the *Quality Modification Event* as a combination of the *Relation Modification Event* and the creation and destruction events, due to the importance of the semantics regarding qualities in ontologies, we have decided to separate them into a special class of events as is described in the next section.

4.1 The Model

To address the event classes aforementioned, while containing their usage to OWL constructs, we face two interesting problems: *Creation Event* depends on a class (**i**), since the individual it would be able to reference does not exist yet; and modification events depend on relations (**ii**) since we need to make explicit *how* the participating individuals should be (or not be) related after the occurrence of the event. These lead to problems because OWL *ObjectProperties* are only allowed between individuals, creating a clear separation between the intensional and extensional models. This metamodeling problem is further discussed in (Motik, 2005), but (OWL Working Group, 2012) solves this problem with the introduction of "punning" in OWL 2, with the caveat that "To allow a more readable syntax, and for other technical reasons, OWL 2 DL requires that a name is not used for more than one property type (object, datatype or annotation property) nor can an IRI denote both a class and a datatype." (OWL Working Group, 2012).

With this addition, and while following the required restrictions for punning, it is possible to represent the necessary predicates in OWL 2 DL. To force some restrictions on how we utilize punning, it was necessary to define a set of metatypes that govern how our reasoning will interact with the defined classes. To do so, we take inspiration from how UFO deals with punning in their OWL implementation gUFO (Almeida et al., 2020). While gUFO's definitions are useful, our proposed model tries to be less compromising in regards to ontological commitment, resulting in a simplified subset of metatypes parting from the definitions of *gufo:EventType*, *gufo:EndurantType* and *gufo:RelationshipType*. The initial separation is between *TypeOfTypes* (*eg.* metatypes) and *TypeOfParticulars*, serving the same purpose of *gufo:Type* and *gufo:Individual*, which resulted in the following taxonomy, where enumeration represents its hierarchical level:

1. TypeOfTypes: Class of second-level types, whose individuals are other types. Equivalent to *gufo:Type*.
   1.1. EventType: The type of events.
   1.2. ObjectType: The type of independent continuants.
   1.3. RelationType: The type of binary predicates. Individuals of this type are object properties.
   1.4. QualityType: The type of existentially dependent continuants.
2. TypeOfParticulars: Class of particulars, whose individuals cannot be instantiated. Equivalent to *gufo:Individual*.
   2.1. Event: An Occurrent, something that happens in time.
      2.1.1. CreationEvent: An Event that concerns the creation of an object.
      2.1.2. DestructionEvent: An Event that concerns the destruction of an object. The occurrence of this event requires the participation of the object target of destruction.
      2.1.3. ModificationEvent: An Event that entails the gain or loss of some property.
         2.1.3.1. RelationModificationEvent: A Modification Event that concerns the gain or loss of a relation. The occurrence of this event requires the participation of a subject and an object in the relation.
         2.1.3.2. QualityModificationEvent: A Modification Event that concerns the gain or loss of quality. The occurrence of this event requires the participation of the object bearer of the quality.

To represent the class dependencies presented previously, we propose the following relations to complement the taxonomy above:

1. ConcernsType: A relation between an *EventType* and another *TypeOfTypes*. Indicates that there is some correlation between the two types that manifests in every occurrence of the individuals of the event class.
   1.1. ConcernsObject: A relation between an *EventType* and an *ObjectType*.
      1.1.1. ConcernsCreationOf: A relation between an *EventType*, subclass of *CreationEvent*, and an *ObjectType*. Indicates that the occurrence of an individual of the event class results in the creation of an individual of the object class.

1.1.2. ConcernsDestructionOf: A relation between an *EventType*, subclass of *DestructionEvent*, and an *ObjectType*. Indicates that the occurrence of an individual of the event class results in the destruction of an individual of the object class.

1.1.3. ConcernsModificationOf: A relation between an *EventType*, subclass of *ModificationEvent*, and an *ObjectType*. Indicates that the occurrence of an individual of the event class results in a modification of an individual of the object class.

1.1.3.1. ConcernsModificationAsSubject: A modification relation that makes explicit the role of the object as the predicate's subject in a relation. This relation is useful when an event concerns the modification of two objects, like in *RelationModificationEvent*.

1.1.3.2. ConcernsModificationAsObject: A modification relation that makes explicit the role of the object as the predicate's object in a relation. This relation is useful when an event concerns the modification of two objects, like in *RelationModificationEvent*.

1.2. ConcernsRelation: A relation between an *EventType*, subclass of *ModificationEvent*, and a *RelationType*. Indicates the concerned relation influences the state between occurrences of the event.

1.2.1. GivesRelation: A relation between an *EventType*, subclass of *ModificationEvent*, and a *RelationType*. Indicates that the occurrence of an individual of the event class results in an instantiation of the individual of *RelationType*.

1.2.2. RemovesRelation: A relation between an *EventType*, subclass of *ModificationEvent*, and a *RelationType*. Indicates that the occurrence of an individual of the event class results in removing an instance of the individual of *RelationType*.

1.3. ConcernsQuality: A relation between an *EventType*, subclass of *QualityModificationEvent*, and a *QualityType*. Indicates that the occurrence of an individual of the event class results in the creation or destruction of an individual of the quality class.

2. ParticipatesIn: A relation between an *owl:Thing*, of metatype *ObjectType*, and an *Event*. Indicates the participation of an object in an event.

2.1. DestroyedBy: A relation between an *owl:Thing*, of metatype *ObjectType* or *QualityType*, and an *Event*. Indicates that the event destroys the participant.

2.2. ModifiedBy: A relation between an *owl:Thing*, of metatype *ObjectType*, and a *ModificationEvent*. Indicates that the event will modify the participant in some way.

2.2.1. SubjectModifiedBy: A relation between an *owl:Thing*, of metatype *ObjectType*, and a *RelationModificationEvent*. It indicates that the participant is or will become subject to the relation concerned by the event.

2.2.2. ObjectModifiedBy: A relation between an *owl:Thing*, of metatype *ObjectType*, and a *RelationModificationEvent*. It indicates that the participant is or will become an object to the relation concerned by the event.

2.2.3. BearerModifiedBy: A relation between an *owl:Thing*, of metatype *ObjectType*, and a *QualityModificationEvent*. It indicates that the participant is or will be the bearer of some quality concerned by the event.

Note that this proposed model has four *Metatypes*, under the class *TypeOfTypes*, to represent the four main entities that we predicate about, as well as six types of metatype *EventType*, under the class *TypeOfIndividuals*. We also have delimited relations between types, with relation *ConcernsType*, and between individuals, with relation *ParticipatesIn*. This is the minimum amount of metatypes needed to represent all the different types of events we proposed to cover, but it is not the only approach we analyzed. These general metatypes introduce reasoning possibilities during the modeling phase of the ontology by the OWL reasoners included in ontology engineering tools. The solution to complement consistency checking, as well as the inference on event occurrence, is further detailed in section 4.2[1].

4.2 The Reasoning

To realize the behavior from modeled events, we need to be able to define some inference rules to interpret the semantics expressed by the defined relations. Additionally, we define some validation reasoning rules to ensure that all aspects of the model are correctly applied and the inference will not produce undesired results.

As already contemplated in section 3, OWL axioms are not expressive enough to represent dynamic behavior, so we opt for utilizing rule-based reasoning to validate and infer over events. We define the following reasoning rules with SHACL Shapes for validation and SHACL Rules for inferring, but it is impor-

[1]The developed model, including the reasoning rules, is available at https://github.com/hrssilva/omice.

tant to note that these definitions depend on SHACL advanced features (specifically for SHACL Rules and SHACL-SPARQL), depending on the support to such features to work.

4.2.1 Rule-Based Validation

Since the restrictions that can be represented with OWL are not specific enough for the relations between types, we define some rules to validate the correctness of the relation usage with classes in the taxonomy of events.

$$\forall e, EventType(e) \wedge subClassOf(e, CreationEvent) \rightarrow \exists o(ConcernsCreationOf(e,o) \wedge ObjectType(o)) \quad (1)$$

$$\forall e, EventType(e) \wedge subClassOf(e, DestructionEvent) \rightarrow \exists o(ConcernsDestructionOf(e,o) \wedge ObjectType(e,o)) \quad (2)$$

$$\forall e, EventType(e) \wedge subClassOf(e, ModificationEvent) \rightarrow \exists r(RelationType(r) \wedge (GivesRelation(e,r) \vee RemovesRelation(e,r))) \quad (3)$$

$$\forall e, EventType(e) \wedge subClassOf(e, QualityModificationEvent) \rightarrow \exists o \exists q(ObjectType(o) \wedge QualityType(q) \wedge ConcernsModificationOf(e,o) \wedge ConcernsQuality(e,q)) \quad (4)$$

$$\forall e, EventType(e) \wedge subClassOf(e, RelationModificationEvent) \rightarrow \exists o_1 \exists o_2(ObjectType(o_1) \wedge ObjectType(o_2) \wedge ConcernsModificationAsSubject(e,o_1) \wedge ConcernsModificationAsObject(e,o_2)) \quad (5)$$

Axiom 1 can be expressed with SHACL-SPARQL through the use of a SHACL Shape with constraint represented by listing 1 and target represented by listing 2. SHACL ties both of the listing definitions through a Shape, which it will use to validate the graph: if, by applying each *constraint* to each entity selected by *target*, constraint returns a triple, then the entity does not conform to shape (resulting in a violation). Listing 3 exemplifies how a Shape is described. Axioms 2, 3, 4 and 5 follow the same pattern of axiom 1.

Listing 1: Creation Constraint.

```
:CreationEventTypeConstraint rdf:type
    owl:NamedIndividual ,
        sh:SPARQLConstraint ;
    sh:message "A CreationEvent class
        should concern the creation of
        an individual of ObjectType." ;
    sh:select """
    prefix : <https://hrssilva.github.
        io/ontology/omice.ttl\#>
    prefix rdfs: <http://www.w3.org
        /2000/01/rdf-schema\#>
    prefix owl: <http://www.w3.org
        /2002/07/owl\#>
    prefix rdf: <http://www.w3.org
        /1999/02/22-rdf-syntax-ns\#>
    SELECT  \$this ?path ?objectclass
    WHERE {
    BIND(:ConcernsCreationOf as ?path)
    NOT EXISTS {
    $this ?path ?objectclass .
    ?objectclass a :ObjectType .
    }}""" .
```

Listing 2: Creation Target.

```
omice:CreationEventTypeTarget rdf:type
    owl:NamedIndividual ,
        sh:SPARQLTarget ;
    rdfs:comment "Targets all
        subclasses of CreationEvent
        that are of type EventType" ;
    sh:select """
    prefix : <https://hrssilva.github.
        io/ontology/omice.ttl#>
    prefix rdfs: <http://www.w3.org
        /2000/01/rdf-schema#>
    prefix owl: <http://www.w3.org
        /2002/07/owl#>
    prefix rdf: <http://www.w3.org
        /1999/02/22-rdf-syntax-ns#>
    SELECT ?entity
    WHERE {
    ?entity a :EventType ;
    rdfs:subClassOf+ :CreationEvent .
    }""" .
```

Listing 3: Creation Shape.

```
omice:CreationEventTypeShape rdf:type
    owl:NamedIndividual ,
        sh:NodeShape ;
    rdfs:comment "Validates that all
        type restrictions on subclasses
         of CreationEvent are being
        correctly modeled" ;
    sh:sparql omice:
        CreationEventTypeConstraint ;
    sh:target omice:
        CreationEventTypeTarget .
```

This SHACL definitions relate to second-level logic axioms in the following way: an axiom of the shape

$A \rightarrow B$, representing a restriction over A, can be expressed by *Shape(T, C)*, where T is a SHACL Target, $T \equiv A$, C is a constraint and $C \equiv \neg B$.

Some implicit restrictions permeate across type relations and individual relations:

- (i) If an event concerns the destruction of an object class, then the individual destroyed by this event should be of the concerned object class: axiom 6.

$$\begin{aligned} &\forall e_i, E(e_i) \wedge subClassOf(E, DestructionEvent) \\ &\quad \wedge ConcernsDestructionOf(E, O) \\ &\quad\quad \rightarrow \exists o_i (O(o_i) \wedge DestroyedBy(o_i, e_i)) \end{aligned} \tag{6}$$

- (ii) If an event concerns the modification of an object class, then the individual modified by this event should be of the concerned object class: axiom 7.
  This topic unfolds into several cases in the proposed model, where the modification relation branches into bearer modification, subject modification, and object modification, but for simplicity's sake, we will consider that all those cases are included in the *ConcernsModification* relation and the *ModifiedBy* participation, generalizing this axiom for the *ModificationEvent* class.

$$\begin{aligned} &\forall e_i, E(e_i) \wedge subClassOf(E, ModificationEvent) \\ &\quad \wedge ConcernsModificationOf(E, O) \\ &\quad\quad \rightarrow \exists o_i (O(o_i) \wedge ModifiedBy(o_i, e_i)) \end{aligned} \tag{7}$$

- (iii) If an event removes a relation of inherence, the quality destroyed by the event should be of the concerned quality class: axiom 8.

$$\begin{aligned} &\forall e_i, E(e_i) \\ &\wedge subClassOf(E, QualityModificationEvent) \\ &\quad \wedge ConcernsQuality(E, Q) \rightarrow \exists q_i (Q(q_i) \\ &\quad\quad \wedge DestroyedBy(q_i, e_i)) \end{aligned} \tag{8}$$

4.2.2 Rule-Based Inference

To generate the desired inferences, we must be able to include and to **remove** triples from the model. The first is not a problem since SHACL processors create a new graph with the result of any inference, but for the same reason, removing triples directly from the model is unavailable. To allow for a decrement in the model, we apply the inference engine to replicate every triple we do not want to remove, resulting in a new inference-produced graph, the original graph where the undesired elements were not included.

For this purpose we created three shapes, each with a single responsibility in constructing the new graph. The *ReplicateUniversalsShape* targets uninteresting entities (entities that are not directly related to possible reasoning) and simply replicates those triples through its rule; the *StepIncrementStateShape* that only deals with events that cannot cause destruction of entities, by including new triples to the model; the *StepDecrementStateShape* that deals with events that may cause the destruction of entities, by selecting which triples will be replicated in the model and which will be ignored.

The assumption that a model going through inference rules is correctly modeled, because it is verified by the validation rules, allows us to be pragmatic in definitions of targets and rules. The targets for individuals ignore entities that are subjects of the relation *DestroyedBy*, effectively removing the destroyed entities from the resulting model.

The inference approach we propose builds a new model, based on the input model, with the inferred consequences of events ”baked in”, so, to deal with the removed relations and references to destroyed objects while replicating the maintained triples to the new model, the rule *DiscardLossObjectsRule* replicates all triples where the given entity is subject, while ignoring those whose object is being destroyed. Similarly, the *DiscardLossRelationsRule* has to ignore triples when the subject and object are being modified by a modification event, and this event class has a relation of *RemovesRelation* with the relation of the analyzed triple. In this sense, while applying the *DiscardLossObjectsRule*, the reasoning engine will look at all selected triples and try to match a negative pattern: for a triple *“T”* with shape *“a r b”*, if there is no triple with shape *“b DestroyedBy c”*, then *“T”* is included in the new model. By using these rules, we can completely remove destroyed entities, as well as any reference to them, from the newly generated model while including all other previous triples.

The inference rules are as follows:

- CreationRule: Creates a new individual of the concerned class.
- GainsQualityRule: Crates an individual of the concerned quality class and instantiates the concerned inherence relation between the new individual and the bearer.
- GainsRelationRule: Instantiates a relation between the subject and object-modified individuals.
- DiscardLossObjectsRule: Replicates every triple for the target, except when an event destroyed the object of the triple.
- DiscardLossRelationsRule: Replicates every triple for the target, except when an event removes the triple relation.

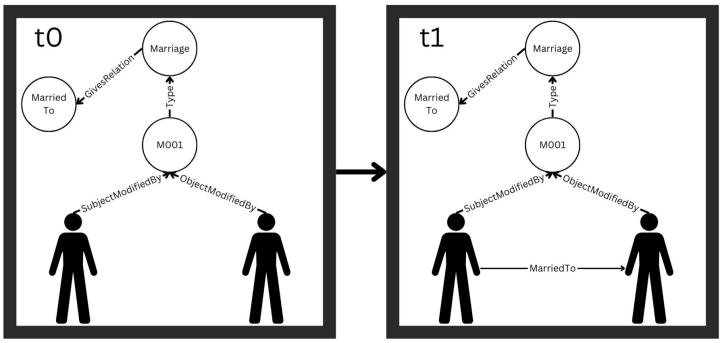

Figure 1: The *Before* and *After* of the marriage event.

Figure 1 exemplifies the behavior of the reasoner in the case of a marriage event, where, given the occurrence of the event, a pair of individuals of type *Person* will now be related through the *MarriedTo* relation. *t0* represents the original model, while *t1* is the resulting model after inference.

5 MODELING VALVE EVENTS FOR THE OIL & GAS INDUSTRY

We modeled a use case of the Oil & Gas (O&G) industry consisting of a Production Well to exemplify Rule-Based reasoning's real-world applicability. The complexity of the industry makes data analysis one of the biggest hurdles for O&G professionals, to the point where they can spend 80% of their time in data acquisition and conversion tasks (Brewer et al., 2019).

In this example, we utilize O3PO (Santos et al., 2024) as a domain ontology and, consequently, its top ontology (BFO), so we must integrate OMICE (the proposed solution, named Ontology Model for Inference on Consequences of Events) with it. This is not a problem since there are no incompatible definitions, and the only overlapping definitions are of *Event* and participation. Given that BFO does not define any subclasses for *Event* and OMICE has no conflicting axioms for events, it is enough to utilize OWL's *SameAs* between BFO and OMICE definitions of *Event* class and between both participation relations.

O3PO defines a *Production Well* as "an object aggregate that is used to produce hydrocarbons or inject fluids, and it is located in a wellbore" and allows the model to track the path of the Oil through the *feeds_fluid_to* relation. In this example, we demonstrate how to model the event of valve closing can be used to verify if closing a valve will stop the flow of Oil from a well to a platform.

We start by instantiating the *Floating Production Storage and Offloading* (FPSO) that we want the oil to reach as a component of a plant in a certain field. To determine that oil has reached this FPSO, we create a *choke valve* as a sort of entry point component. The path that needs to be followed to reach this entry point is modeled as a *flowline*, defined as "a pipeline that carries oil, gas or water that connects the wellhead to a manifold or a platform".

Looking at the other side of the oil flow, the source, we instantiate the *production well* and its parts: its *annular space* (the oil-filled space between the reservoir and the well tubing), its *borehole*, and its *wellhead*. The *production column* of the well receives the oil from the annular space. The *valves* and *tubing* build the column for transporting the oil to the *wellhead*. In this case, the *annular space* and the *production column* are separated into three parts by three *Inflow Control Valves* (ICV), which manage if the oil can pass from the respective part of the annular space to the *production column*. This means that if any ICV is open, oil flows through the *production column* and feeds to a *Downhole Safety Valve* (DHSV). From the DHSV, the oil goes to the *wellhead*, then leaves the *well* and goes to a *subsea tree* (specifically to a *master valve*, a component of the *subsea tree*), finally reaching the *flowline*.

From now on, in this section, we will use the syntax *A(a)* to say that *a* is an instance of class *A* and *R(a, b)* to say that *a* has relation *R* with *b*, so we can easily keep track of the different instances involved. In this example the valves created are: *icv(ICV001)*, *icv(ICV002)*, *icv(ICV003)*, *dhsv(DHSV001)*, *master_valve(MASTER001)* and, the destination of the oil, *choke(PLATFORMCHOKE001)*, each with their instance of operational state quality (either *operational* or *not_operational*). The other relevant instances are: *Production_Well(PWELL001)* and *FPSO(FPSO001)*. Figure 2 shows the configuration of valves and equipment of this example.

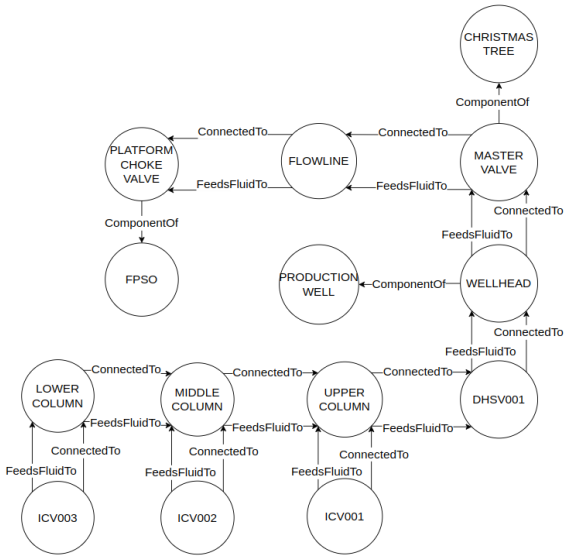

Figure 2: Configuration of equipment for offshore oil extraction.

To model the events, we create the *ClosingOfValve* and *OpeningOfValve* events (our main events) and the quality modification events *RemoveValveOperationalState*, *GiveValveClosedState* and *GiveValveOpenState*. Through OWL axioms, we force the *ClosingOfValve* instances to have exactly one instance of *RemoveValveOperationalState* and exactly one instance of *GiveValveClosedState* as parts, similarly, *OpeningOfValve* has parts *RemoveValveOperationalState* and *GiveValveOpenState*. Figure 3 show an excerpt of how we have modeled the *ClosingOfValve* events, including individuals.

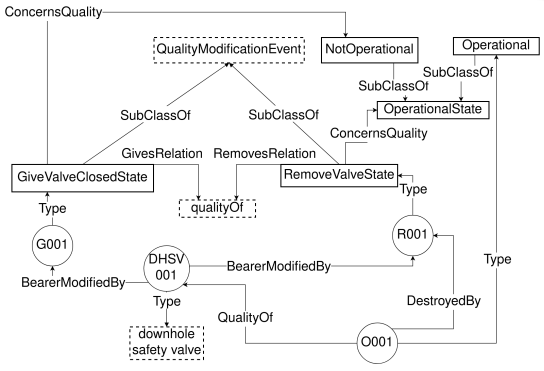

Figure 3: *ClosingOfValve* example model.

To know if a well is feeding oil to a platform, we will utilize a SPARQL ASK query to test if there is a path between any of the well's ICV and a platform component. Additionally, there must not exist any valve in the way that has an instance of *not_operational* inhering in it, resulting in the query in listing 4.

Listing 4: Query to know if well is feeding oil to platform.

```
PREFIX o3po: <https://www.petwin.org/
    o3po-resources/o3po#>
PREFIX rdfs: <http://www.w3.org
    /2000/01/rdf-schema#>
PREFIX core: <https://purl.
    industrialontologies.org/ontology
    /core/Core/>
PREFIX ex: <http://www.example.org/
    well.ttl#>
ASK
WHERE {
  ?bot a o3po:ICV ;
       o3po:feeds_fluid_to+ ?v1;
       o3po:connected_to*/o3po:
           component_of* ex:PWELL001
           .
  FILTER NOT EXISTS {?bot ^core:
      qualityOf [ a ex:
      not_operational ] } .
  ?v1 a/rdfs:subClassOf* o3po:valve ;
       o3po:feeds_fluid_to+ ?v2 .
  ?q1 core:qualityOf ?v1 ;
     a ex:operational .
  ?v2 a/rdfs:subClassOf* o3po:valve ;
      o3po:feeds_fluid_to* ?top .
  ?q2 core:qualityOf ?v2 ;
     a ex:operational .
  ?top o3po:component_of ex:FPSO001 .
}
```

In the initial state of the model, where no event has occurred, all valves are operational except for ICV002. In this scenario, the query returns *True*, shown in listing 5. By instantiating a *ClosingOfValve* event, targeting *DHSV001*, and running the inference, we can see that the resulting model excludes the triples shown in Equation 9 and introduces the triples in Equation 10, resulting in no available path, since *DHSV001* closes the only path between the production column and the wellhead. When rerunning the query of Listing 4, we then receive *False*, as shown in Listing 6.

$$\begin{aligned} &OPEN\_DHSV001 \quad \overrightarrow{type} \quad NamedIndividual \\ &OPEN\_DHSV001 \quad \overrightarrow{qualityOf} \quad DHSV001 \\ &OPEN\_DHSV001 \quad \overrightarrow{type} \quad operational \\ &OPEN\_DHSV001 \quad \overrightarrow{destroyedBy} \\ &\qquad REMOVE\_STATE\_DHSV001 \end{aligned} \tag{9}$$

$$\begin{aligned} &not\_operational202407... \quad \overrightarrow{type} \quad NamedIndividual \\ &not\_operational202407... \quad \overrightarrow{qualityOf} \quad DHSV001 \\ &not\_operational202407... \quad \overrightarrow{type} \quad not\_operational \end{aligned} \tag{10}$$

Listing 5: Query result for the initial model state.

```
{'head': {}, 'boolean': True}
```

Listing 6: Query result with a closed DHSV.

```
{'head': {}, 'boolean': False}
```

This example demonstrates how we can utilize the reasoning approach proposed to dynamically query over the state of the model, given the occurrence of some event. Through the SPARQL queries, we can attest to the consequence of a possible closing of a valve, demonstrating the effect of the reasoning. Nevertheless, there are some problems with the developed event model: the need for the classes *Operational* and *NotOperational* highlights the limitation imposed by not dealing with quality values. By ontological standards, a quality is inherent to its bearer and should not be destroyed when there is a change in quality value, but for implementation issues, dealing only with gain and loss of quality requires that we take such an approach in this model. To correctly represent this example, only the class *OperationalState*, whose individuals have the values "0" or "1", should exist, and the *ClosingOfValve* event should set this value to "0".

6 CONCLUSION

Through this work, we explored rule-based reasoning to infer the consequences of events, successfully keeping our solution inside the boundaries of the Semantic Web by utilizing SHACL-SPARQL as a rule description language. We also presented a model on types of events that implement relations capable of revealing the implicit qualities of event occurrence, *how* an event happens, and describing processable behavior of *creation*, *destruction*, and *modification* of objects. We also exemplified how this model can be utilized and successfully modeled a use case of the O&G industry by importing our proposed model in the domain ontology O3PO. Considering the gathered (in chapter 3) current solutions to deal with event reasoning in ontologies, our proposed solution takes a different direction by focusing on event consequences and taking technological compatibility in high regard.

6.1 Current Limitations

Although we successfully operationalized events, we only approached a marginal part of all that events represent. A big missing part for real-world applicability is the treatment of OWL *DataProperties*. This caused a discomforting peculiarity in our model when dealing with qualities, which was clear in our example when choosing how to model the operational state of valves.

Another deficit of our work is regarding how events relate to one another. When dealing with complex scenarios, it is common to have events that happen in a certain order or that affect the outcome of one another. By not representing this relationship, it becomes very difficult to model complex events when the intermediary steps of the event are of interest.

6.2 Future Works

As section 6.1 made clear, there is still much room for growth in our current proposition, which we intend to do. Another interesting path is to utilize the same reasoning approach to other aspects of events, like dealing with participant recognition and event triggers, or to implement temporal relations between events, such as those defined by (Allen, 1983).

ACKNOWLEDGEMENTS

The PeTwin project was financed by FINEP and the Libra Consortium (Petrobras, Shell Brasil, Total Energies, CNOOC, CNPC). The research group is supported also by CAPES Finance Code 001 and CNPq, the Brazilian Finance Council.

REFERENCES

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Almeida, J., Falbo, R., Guizzardi, G., and Prince Sales, T. (2020). gufo: A lightweight implementation of the unified foundational ontology (ufo).

Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., and Studer, R. (2011). *ETALIS: Rule-Based Reasoning in Event Processing*, pages 99–124. Springer Berlin Heidelberg, Berlin, Heidelberg.

Benevides, A., Bourguet, J.-R., Guizzardi, G., Peñaloza, R., and Almeida, J. (2019). Representing a reference foundational ontology of events in sroiq. *Applied Ontology*, 14:1–42.

Bennett, J. (2002). What events are. In Gale, R. M., editor, *The Blackwell Guide to Metaphysics*, page 43. Wiley-Blackwell.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43.

Brewer, T., Knight, D., Noiray, G., and Naik, H. (2019). Digital Twin Technology in the Field Reclaims Offshore Resources. volume Day 1 Mon, May 06, 2019 of *OTC Offshore Technology Conference*.

Davidson, D. (1969). *The Individuation of Events*, pages 216–234. Springer Netherlands, Dordrecht.

Ding, L., Kolari, P., Ding, Z., and Avancha, S. (2007). *Using Ontologies in the Semantic Web: A Survey*, pages 79–113. Springer US, Boston, MA.

Ermolayev, V., Keberle, N., and Matzke, W.-E. (2008). An ontology of environments, events, and happenings. In *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pages 539–546.

Guizzardi, G., Guarino, N., and Almeida, J. P. A. (2016). Ontological considerations about the representation of events and endurants in business models. In La Rosa, M., Loos, P., and Pastor, O., editors, *Business Process Management*, pages 20–36, Cham. Springer International Publishing.

Guizzardi, G., Wagner, G., Falbo, R., Guizzardi, R., and Almeida, J. (2013). Towards ontological foundations for the conceptual modeling of events. volume 8217, pages 327–341.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From shiq and rdf to owl: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26.

Keet, C. M. (2020). *An Introduction to Ontology Engineering*, chapter 10, pages 193 – 209. C. Maria Keet.

Knublauch, H., Allemang, D., and Steyskal, S. (2017). Shacl advanced features. https://www.w3.org/TR/2017/NOTE-shacl-af-20170608/. Accessed at: february 4th, 2024.

Lei, L., Ren, X., Franciscus, N., Wang, J., and Stantic, B. (2019). Event prediction based on causality reasoning. In Nguyen, N. T., Gaol, F. L., Hong, T.-P., and Trawiński, B., editors, *Intelligent Information and Database Systems*, pages 165–176, Cham. Springer International Publishing.

Li, F., Du, J., He, Y., Song, H.-Y., Madkour, M., Rao, G., Xiang, Y., Luo, Y., Chen, H. W., Liu, S., Wang, L., Liu, H., Xu, H., and Tao, C. (2020). Time event ontology (teo): to support semantic representation and reasoning of complex temporal relations of clinical events. *Journal of the American Medical Informatics Association*, 27(7):1046–1056.

Li, S., Chen, S., and Liu, Y. (2019). A method of emergent event evolution reasoning based on ontology cluster and bayesian network. *IEEE Access*, 7:15230–15238.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., and Oltramari, A. (2003). Wonderweb deliverable d18. http://wonderweb.man.ac.uk/deliverables/documents/D18.pdf.

Mepham, W. and Gardner, S. (2009). Implementing discrete event calculus with semantic web technologies. In *2009 Fifth International Conference on Next Generation Web Services Practices*, pages 90–93.

Motik, B. (2005). On the properties of metamodeling in owl. In Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., editors, *The Semantic Web – ISWC 2005*, pages 548–562, Berlin, Heidelberg. Springer Berlin Heidelberg.

Otte, J. N., Beverley, J., and Ruttenberg, A. (2022). Bfo: Basic formal ontology. *Applied ontology*, 17(1):17–43.

OWL Working Group (2012). Owl 2 web ontology language primer (second edition). https://www.w3.org/TR/owl2-primer/. Accessed at: july 4th, 2024.

Padilla-Cuevas, J., Reyes-Ortiz, J. A., and Bravo, M. (2021). Ontology-based context event representation, reasoning, and enhancing in academic environments. *Future Internet*, 13(6).

Pedrinaci, C., Domingue, J., and Alves de Medeiros, A. K. (2008). A core ontology for business process analysis. In Bechhofer, S., Hauswirth, M., Hoffmann, J., and Koubarakis, M., editors, *The Semantic Web: Research and Applications*, pages 49–64, Berlin, Heidelberg. Springer Berlin Heidelberg.

Quinton, A. (1979). Objects and events. *Mind*, 88(350):197–214.

Rodrigues, F., Carbonera, J., and Abel, M. (2020). *Upper-Level Types of Occurrent Based on the Principle of Ontological Conservation*, pages 353–363. Springer International Publishing.

Rodrigues, F. H. and Abel, M. (2019). What to consider about events: A survey on the ontology of occurrents. *Applied Ontology*, 14(4):343–378.

Santos, N. O., Rodrigues, F. H., Schmidt, D., Romeu, R. K., Nascimento, G., and Abel, M. (2024). O3PO: A domain ontology for offshore petroleum production plants. *Expert Systems with Applications*, 238:122104.

Smith, B. and Grenon, P. (2005). The cornucopia of formal-ontological relations. *Dialectica*, 58:279–296.

Studer, R., Benjamins, V., and Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1):161–197.

Valadares Vieira, L., Henrique Rodrigues, F., and Abel, M. (2025). Portions of matter and their existential events: An ontology-based conceptual model. In Maass, W., Han, H., Yasar, H., and Multari, N., editors, *Conceptual Modeling*, pages 152–169, Cham. Springer Nature Switzerland.

Xue, J.-R., Fang, J.-W., and Zhang, P. (2018). A survey of scene understanding by event reasoning in autonomous driving. *International Journal of Automation and Computing*, 15(3):249–266.

Zhong, Z., Liu, Z., Li, C., and Guan, Y. (2012). Event ontology reasoning based on event class influence factors. *International Journal of Machine Learning and Cybernetics*, 3(2):133–139.