UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

HAROLDO ROJAS DE SOUZA SILVA

Event Modeling for Supporting Reasoning of Consequences

Work presented in partial fulfillment of the requirements for the degree of Bachelor in Computer Science

Advisor: Prof. Dr. Mara Abel
Co-advisor: Dr. Fabrício Henrique Rodrigues

Porto Alegre
August 2024

ACKNOWLEDGMENTS

There are many people to thank, not only for the existence of this work but also for this whole journey, now reaching a new chapter. First of all, I would like to thank my family and my close friends, for providing me with the support to go through with it these five years, which were riddled with bumps, but you all were there to lift me back up (even if you haven't realized it). Thank you, Mother, for everything you have given me, so much that I will never be able to pay back. I can't thank enough Rafael Trevisan, for introducing me to the world of computer science, I would not be writing these words if it weren't for you.

I can't disregard my colleagues from PeTWIN, who served as an infinite fountain of inspiration. My co-advisor, Fabrício, thank you for the endless sessions of discussions, if this work has any meaningful contribution, it is thanks to you. Finally, I can't finish these acknowledgments without mentioning my advisor, Mara Abel, who, besides this very work, has, through countless opportunities, helped me shape the path that took me here.

ABSTRACT

The modeling of events is extremely important 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 to infer 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 flow of oil.

Keywords: Ontology. Conceptual modelling. Events. Reasoning. OWL. SHACL. SPARQL.

Modelagem de Eventos Para Suporte a Raciocínio de Consequências

RESUMO

A modelagem de eventos é extremamente importante em vários domínios nos quais a evolução temporal dos dados apoia a tomada de decisões, mas as limitações de representação no estado da arte em modelagem conceitual ainda são uma barreira para o desenvolvimento de aplicações de software. As soluções atuais falham em reconciliar expressividade comportamental, reutilização e compatibilidade tecnológica. Este trabalho considera a modelagem de eventos sob a abordagem de ontologias e foca no raciocínio para inferir as consequências dos eventos. Propomos o uso de linguagens de descrição de regras para melhorar o raciocínio tradicional de ontologias com capacidades de interpretação de semânticas específicas, preservando a utilidade das tecnologias atuais (ao não depender de descrições não analisáveis, seja por escolha representacional, de modelagem ou tecnológica) enquanto permitindo inferência de maneiras que não são possíveis com axiomas convencionais. Durante este trabalho, exploramos soluções compatíveis com a Web Semântica para representar o comportamento dos eventos, resultando em uma representação OWL de um modelo de eventos suportada por inferência e verificação de consistência usando SHACL-SPARQL. Demonstramos nossa proposta importando o modelo resultante para uma ontologia de domínio da indústria de óleo e gás e mostrando como as consequências inferidas dos eventos afetam uma consulta sobre o fluxo de petróleo.

Palavras-chave: Ontologias. Modelagem conceitual. Eventos. Raciocínio. OWL. SHACL. SPARQL.

LIST OF ABBREVIATIONS AND ACRONYMS

BFO	Basic Formal Ontology
DEC	Discrete Event Calculus
DEDP	Dynamic Engineering Design Process
O3PO	Offshore Petroleum Production Plant Ontology
RDF	Resource Description Framework
SWRL	Semantic Web Rule Language
SHACL	Shapes Constraint Language
UFO	Unified Foundational Ontology
OWL	Web Ontology Language
W3C	World Wide Web Consortium

LIST OF FIGURES

LIST OF TABLES

CONTENTS

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.

The utility of event reasoning spans multiple aspects, be it extracting what events have occurred and what 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; Fang; Zhang, 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 sort of reasoning is either done through specific software implementation or by utilizing temporal logic, but we will discuss a method that is compatible with ontologies without requiring external computational tools or domain-specific modeling.

This work discusses a solution to address the shortcomings of OWL in representing 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 proposed solution focuses on inferring the consequences of event occurrences by proposing a set of 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 based on ontological conservation (Rodrigues; Carbonera; Abel, 2020) to not create undesired logical consequences.

Although we are dealing with temporal entities (events), as already stated, we will specifically discuss the effects events have in the surrounding objects, leaving aside other important aspects of this domain. in this work, we will not discuss how events relate to time and to other events. In this sense, we will not define causal relationships or

implement the twelve temporal relations of Allen (Allen, 1983).

The following of this work is divided in this way: chapter 2 introduces the various base concepts of ontologies, events, and reasoning we lean on; chapter 3 explores how the reasoning of events has been approached in other works; chapter 4 explains the decisions we have taken in our approach of modeling, as well as reasoning, including examples of usage; chapter 5 demonstrates our approach in a real-world use case of the O&G industry; and chapter 6 discusses what we achieved, current limitations and what can still be explored.

2 THEORETICAL BACKGROUND

This study requires to understand of what an ontology is, how ontologies are computationally treated, and how to represent events in an ontology. Ontology is a logical theory for conceptual modeling that aims to represent entities in the world based on their intrinsic nature and structure, independent of the observer (Guarino; Daniel; Steffen, 2009). The representation of an ontology seeks to constrain the possible interpretations of terms in a language to bring the set of models allowed by the language closer to the set of models intended by the modeler and user (Guarino, 1995). In computer science, we can consider an ontology as an explicit specification of a conceptualization (Gruber, 1993), meaning that an ontology is an unambiguous representation of a conceived idea about reality. This notion was expanded by Studer (1998) to represent that an ontology is a formal specification of a shared conceptualization of reality. 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, 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 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.'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; Hendler; Lassila, 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). 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; Patel-Schneider; van Harmelen, 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 the adoption of specific technologies that would conflict with the environment created by the Semantic Web, or the need to use an incompatible modeling language. In fact, 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 WORK

This chapter 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; Beverley; Ruttenberg, 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 3.1 summarizes the comparison made below, 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, or that proposes another; *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 that 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 Compromise:** How specific is the ontological compromise 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 or not. This criterion is closely related to **Event coverage**.

The work "A Core Ontology for Business Process Analysis" (Pedrinaci; Domingue; Medeiros, 2008) models, in Operational Conceptual Modeling Language (OCML), concepts related to Business Process Management (BPM), specifically for tasks of Business Process Analysis (BPA). To that effort, this work presents 3 ontologies: Core Ontology for Business pRocess Analysis (COBRA), the main ontology discussed; a reference Events Ontology (EVO) to describe business processes; and an Event Analysis Ontology that describes relations to keep track and reason about processes, activities, actors, and roles. Although COBRA bases definitions on other ontologies, including top ontologies like DOLCE, the authors classify it as a Core Ontology, signifying that it represents a robust worldview but still leaves some space to apply it to other top-level ontologies.

Most of the reasoning on Pedrinaci; Domingue; Medeiros's work comes from the Event Analysis Ontology, describing not only relations but also OCML rules to support reasoning and inference. This use is outdated and may not be supported by current Semantic Web standards. These rules allow for "Activity Monitoring Events to update the current state of activity realizations, generate Life-Cycle Period instances, and contrast the transitions with the given state model." (Pedrinaci; Domingue; Medeiros, 2008), also providing a general relation that links a Business Activity Realization, a Time Instant and a Business Activity State.

The work "A Method of Emergent Event Evolution Reasoning Based on Ontology Cluster and Bayesian Network" (Li; Chen; Liu, 2019) presents a reasoning method to deal with emergency situation events (such as landslides and floods) based on probabilities. To do so, it utilizes a Bayesian Network for probability computation and extends SWRL rules to represent how events relate to each other and evolve. Although Ding, Peng and Pan (2006) presented the idea of extending ontological description languages with Bayesian Networks, through the *BayesOWL* framework, the focus on reasoning of events and on a rule language creates more similarities with the work we will be proposing.

To represent how emergent events evolve and support decision-making from policymakers, Li; Chen; Liu extends SWRL to associate probabilities to the rules, allowing a Bayesian Network to utilize them for deducing the probability of the occurrence of ref-

erence scenarios. In this way, the antecedents influence mathematically the consequent, enabling the representation of how events transition from one another. The Li and colleagues work presents an example of the usage of the method through a model based on the NFP 1600 standard for crisis management capabilities composed of four elements: an event knowledge base, a situation knowledge repository, a resource knowledge base (for countermeasure resources) and an association knowledge base (containing how the different knowledge bases relate). Even though there is a description of such a model, it is not clear how the authors implement this.

The work "An Ontology of Environments, Events, and Happenings" (Ermolayev; Keberle; Matzke, 2008) proposes a model implemented in OWL-DL and inspired by Discrete Event Calculus (DEC), including the specificity of distinguishing between events as objective manifestations of phenomena and happenings as the perception of an event by an observer. Furthermore, the authors emphasize the concept of environment - the context surrounding events and objects - where the objects it contains and the events (contained or not) can influence environments and utilize the concept of fuzzy time to represent the time intervals on which the phenomena occur. Regarding the definition of time intervals, the proposed model differentiates between events and atomic actions, which can only occur instantly, extending such differences by classifying happenings as atomic actions.

Since the model implemented by Ermolayev; Keberle; Matzke utilizes OWL-DL only, it provides reasoning capabilities native to the OWL language, and it is possible to utilize it with any application stack supporting this language, mainly the Semantic Web stack. Contextually, this work addresses the use case of the PSI[1], including extending its ontology (the PSI Meta Ontology), in the domain of dynamic engineering design processes (DEDP), but, by demonstration, applies to other domains where there are observers. By extension, as denoted by the authors, it fails to correctly represent events without observers. Given that the PSI Meta Ontology utilizes DOLCE as a top-level ontology and SUMO as a commonsense reference ontology, there is a full ontological compromise to the worldview defined in such ontologies.

The work "ETALIS: Rule-Based Reasoning in Event Processing" (Anicic et al., 2011) describes an approach for dealing with Complex Event Processing (CEP) through a proposed rule-based logical language. The ETALIS *Language for Events* aims to enable *reasoning* and *inferencing* of complex events by distinguishing them from atomic events, where atomic events are instantaneous and logically interpreted as *facts*. At the same time,

[1]Performance Simulation Initiative (PSI) is the research and development project of Cadence Design Systems, GmbH.

complex events have a time interval in which they happen and are logically interpreted as *deductive rules*.

To describe complex events, the ETALIS language works with *event patterns* that describe complex events through pattern matching of simpler events (ultimately atomic events). This allows the reasoner to detect the occurrence of complex events using operators related to time, when an event starts, when it ends, and how events temporally relate to other events. The language also supports *Event-Driven Backward Chaining Rules* (EDBCR) to describe how events are causally related, supporting reasoning.

The work "Event ontology reasoning based on event class influence factors" (Zhong et al., 2012) defines a series of inference rules for an event model that describes an *Event* as a five-tuple $e = \langle A, S, T, O, L \rangle$ composed of an action (A), a subject (S), an object (O) a time (T) and a location (L).

Parting from this central definition, the notion that an *Event Class* is the set of events that share the same behavior (e.g., the elements of the tuple denote the same intention), the event class adjacency matrix w_ij (indicating that there is a relation between two event classes) and the probability of the occurrence of an event class, Zhong et al. describe three inference rules. These rules allow for events to trigger the occurrence of other events, but the authors are not clear on how they implement the proposed inference (especially considering that probability comes into play, which is a complicated factor and the main focus of the software implementations already discussed in (Li; Chen; Liu, 2019; Ding; Peng; Pan, 2006)).

The work "Implementing discrete event calculus with semantic web technologies" (Mepham; Gardner, 2009) describes an implementation, utilizing OWL and SWRL, of a model based on Discrete Event Calculus (DEC) to allow for reasoning regarding events. To do so, the model represents Event, Fluent, and Timepoint from the DEC as OWL classes and axiomatizes the relations between these classes with SWRL rules. Although these tools allow for easy application in a semantic web environment, the authors clarify that it was not possible to represent all axiomatizations present in the DEC with SWRL, developing a software solution to make up for the missing axioms with SQWRL.

To contextualize the work, Mepham; Gardner brings forth the web services domain, explaining their interest in applying their solution to the Semantic Web. Nevertheless, the proposed model is not inherently tied to such domain and can adapt itself to any circumstance where the DEC could represent, as demonstrated by using this model to solve the *Hanks-McDermott problem* (also known as the *Yale shooting problem*, this

formal situational logic problem, related to the classical *frame problem*, highlights how the formalization of inertia is not enough to represent causal relationships between happenings). Having this said, there is some significant ontological compromise inherent to this representation regarding time, as worldviews do not always represent time directly, as is the case with *Timepoint*.

The work "Ontology-Based Context Event Representation, Reasoning, and Enhancing in Academic Environments" (Padilla-Cuevas; Reyes-Ortiz; Bravo, 2021) aims to implement Ambient Intelligence in an academic environment. To do so, the authors present ontological models for people, locations, computer networks, time and events, while utilizing SQWRL to query the models and SWRL to infer over the models in a way that when an event occurs, there is a network node, with some location, responsible for registering the event, as well as what people are present and at what time it has happened.

In Padilla-Cuevas; Reyes-Ortiz; Bravo's work, the reasoning focus for events is the context that surrounds the event. By ingraining knowledge about the system at work (an Ambient Intelligence system where network devices measure and trigger events) in SWRL rules, it is possible to infer information about the event, including participants. The proposed models are specifically created to attend to the system in question, not creating ontologically grounded definitions that generalize the concepts explored to other domains or systems.

The work "Time event ontology (TEO): to support semantic representation and reasoning of complex temporal relations of clinical events" (Li et al., 2020) describes an OWL ontology that represents temporal information, including the two main classes of *Time* and *Event*. The focus of this representation is the different types of time relations and intervals, allowing great expressivity of temporal data, categorizing events on when they happened, their duration and the granularity of the temporal representation. For reasoning support to this ontology, the authors present an implementation expanding on the HermiT OWL reasoner, allowing for temporal reasoning to not only create a timeline of events but also to query the uncertain relationship between events with insufficient information.

The model provided for TEO is specifically designed to model and reason clinical events, describing events like *Clinical Intervention* and *Diagnosis*. Even though the core of the temporal representation seems general enough to model other domains, there is no example of this corroborated by the usage of a specific reasoner for a clinical setting.

Table 3.1 – Related Work Comparison

Title	Technology Compatibility	Event Coverage	Ontological Compromise	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

3.1 Creation, Destruction and Modification Events

Events regarding the creation, destruction and modification of entities are a topic of discussion in ontology modeling. UFO explores such concepts through the specialization of *Participation* (a subclass of event) in *object creation*, *object destruction* and *object change* (Benevides et al., 2019; Guizzardi; Guarino; Almeida, 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; Grenon, 2005).

3.2 Current Limitations

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

The model proposed by (Ermolayev; Keberle; Matzke, 2008) is a well-founded ontology focused on events and with support to reasoning depending only on semantic web technologies, but, by utilizing only OWL axiomatization, it does not have the inference capabilities of rule-based reasoning, failing to be able to represent behavior in

inference. No current proposition succeeds in operationalizing ontological event descriptions without extra tools or specific extensions of current tools.

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; Gardner, 2009), we will utilize SHACL, as a rule-based language, together with SPARQL, through SHACL-SPARQL advanced features (Knublauch; Allemang; Steyskal, 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. 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 world-views, the proposed modeling approach tries to make the minimum amount of compromises with how to model the surrounding context. The basis of the modeling approach is the Aristotelian Ontological Square, where we categorize *Events* and their types as *Substantial Universals* and we describe their behavior through a series of *Accidental Universals*. Another 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*, where, for this work, events are occurrents, and their participants are continuants.

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; Carbonera; Abel, 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, by being more general, than the usual existential dependence between events and their objects, but, nevertheless, 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 that we will describe 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 the OWL Working Group 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 some 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:

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 a 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*, subclass of *CreationEvent*, and an *ObjectType*.

        1.1.1. ConcernsCreationOf: 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 in 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* (represented with Turtle in listing 4.1), 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 (represented with Turtle in listing 4.2), with relation *ConcernsType*, and between individuals (represented with Turtle in listing 4.3), with relation *ParticipatesIn*. This is the minimum amount of metatypes needed to be able to represent all the different types of events we proposed to cover, but it is not the only approach we analyzed. Using only these general metatypes stops us from axiomatizing the restrictions on the classes in OWL, whose absence stops complete reasoning during the modeling phase of the ontology by the OWL reasoners included in ontology engineer-

ing tools. The solution to complement consistency checking, as well as the inference on event occurrence, is further detailed in chapter 4.2[1].

```
omice:EventType rdf:type owl:Class ;
rdfs:subClassOf omice:TypeOfTypes ;
owl:disjointWith omice:ObjectType ,
                 omice:QualityType ,
                 omice:RelationType .

omice:ObjectType rdf:type owl:Class ;
rdfs:subClassOf omice:TypeOfTypes ;
owl:disjointWith omice:QualityType ,
                omice:RelationType .

omice:QualityType rdf:type owl:Class ;
rdfs:subClassOf omice:TypeOfTypes ;
owl:disjointWith omice:RelationType .

omice:RelationType rdf:type owl:Class ;
                 rdfs:subClassOf omice:TypeOfTypes .

omice:TypeOfTypes rdf:type owl:Class .
```

Listing 4.1 – Triple definition for Metatype classes

```
:ConcernsCreationOf rdf:type owl:ObjectProperty ;
                    rdfs:subPropertyOf :ConcernsObject ;
                    rdfs:domain :EventType ;
                    rdfs:range :ObjectType .

:ConcernsDestructionOf rdf:type owl:ObjectProperty ;
                       rdfs:subPropertyOf :ConcernsObject ;
                       rdfs:domain :EventType ;
                       rdfs:range :ObjectType .

:ConcernsModificationAsObject rdf:type owl:ObjectProperty ;
                              rdfs:subPropertyOf :
   ConcernsModificationOf .

:ConcernsModificationAsSubject rdf:type owl:ObjectProperty ;
```

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

```
                                rdfs:subPropertyOf :
   ConcernsModificationOf .

:ConcernsModificationOf rdf:type owl:ObjectProperty ;
                        rdfs:subPropertyOf :ConcernsObject ;
                        rdfs:domain :EventType ;
                        rdfs:range :ObjectType .

:ConcernsObject rdf:type owl:ObjectProperty ;
                rdfs:subPropertyOf :ConcernsType ;
                rdfs:domain :EventType ;
                rdfs:range :ObjectType .

:ConcernsQuality rdf:type owl:ObjectProperty ;
                 rdfs:subPropertyOf :ConcernsType ;
                 rdfs:domain :EventType ;
                 rdfs:range :QualityType .

:ConcernsRelation rdf:type owl:ObjectProperty ;
                  rdfs:subPropertyOf :ConcernsType ;
                  rdfs:domain :EventType ;
                  rdfs:range :RelationType .

:ConcernsType rdf:type owl:ObjectProperty ;
              rdfs:domain :EventType ;
              rdfs:range :TypeOfTypes .

:GivesRelation rdf:type owl:ObjectProperty ;
               rdfs:subPropertyOf :ConcernsRelation ;
               rdfs:domain :EventType ;
               rdfs:range :RelationType .
```

Listing 4.2 – Triple definition for relations between types

```
:ModifiedBy rdf:type owl:ObjectProperty ;
            rdfs:subPropertyOf :ParticipatesIn ;
            rdfs:domain :TypeOfParticulars ;
            rdfs:range :ModificationEvent .

:BearerModifiedBy rdf:type owl:ObjectProperty ;
                  rdfs:subPropertyOf :ModifiedBy ;
                  rdfs:range :QualityModificationEvent .
```

```
9
10 :ObjectModifiedBy rdf:type owl:ObjectProperty ;
11                   rdfs:subPropertyOf :ModifiedBy ;
12                   rdfs:range :RelationModificationEvent .
13
14 :ParticipatesIn rdf:type owl:ObjectProperty ;
15                 rdfs:subPropertyOf owl:topObjectProperty ;
16                 rdfs:domain :TypeOfParticulars ;
17                 rdfs:range :Event .
18
19 :RemovesRelation rdf:type owl:ObjectProperty ;
20                  rdfs:subPropertyOf :ConcernsRelation ;
21                  rdfs:domain :EventType ;
22                  rdfs:range :RelationType .
23
24 :SubjectModifiedBy rdf:type owl:ObjectProperty ;
25                    rdfs:subPropertyOf :ModifiedBy ;
26                    rdfs:range :RelationModificationEvent .
27
28 :DestroyedBy rdf:type owl:ObjectProperty ;
29              rdfs:subPropertyOf :ParticipatesIn ;
30              rdfs:domain :TypeOfParticulars ;
31              rdfs:range [ rdf:type owl:Class ;
32                           owl:unionOf ( :DestructionEvent
33                                         :QualityModificationEvent
34                                       )
35                         ] .
```

Listing 4.3 – Triple definition for relations between individuals

4.2 The Reasoning

In order to extract the behavior from modeled events, we need to be able to define some inference rules for extracting the semantics implied 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 chapter 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 important to note that these definitions depend on SHACL advanced features (specifically for SHACL Rules and for the use of 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, which can be seen in listings 4.1 and 4.2 in comparison to listing 4.3, we define some rules to validate the correctness of the relation usage with classes placed in the taxonomy of events.

$$\begin{aligned} &\forall e, EventType(e) \land subClassOf(e, CreationEvent) \\ &\quad \rightarrow \exists o(ConcernsCreationOf(e, o) \land ObjectType(o)) \end{aligned} \tag{4.1}$$

$$\begin{aligned} &\forall e, EventType(e) \land subClassOf(e, DestructionEvent) \\ &\quad \rightarrow \exists o(ConcernsDestructionOf(e, o) \land ObjectType(e, o)) \end{aligned} \tag{4.2}$$

$$\begin{aligned} &\forall e, EventType(e) \land subClassOf(e, ModificationEvent) \\ &\quad \rightarrow \exists r(RelationType(r) \\ &\qquad \land (GivesRelation(e, r) \lor RemovesRelation(e, r))) \end{aligned} \tag{4.3}$$

$$\begin{aligned} &\forall e, EventType(e) \land subClassOf(e, QualityModificationEvent) \\ &\quad \rightarrow \exists o \exists q(ObjectType(o) \land QualityType(q) \\ &\qquad \land ConcernsModificationOf(e, o) \land ConcernsQuality(e, q)) \end{aligned} \tag{4.4}$$

$$\begin{aligned} &\forall e, EventType(e) \land subClassOf(e, RelationModificationEvent) \\ &\quad \rightarrow \exists o_1 \exists o_2(ObjectType(o_1) \land ObjectType(o_2) \\ &\qquad \land ConcernsModificationAsSubject(e, o_1) \\ &\qquad \land ConcernsModificationAsObject(e, o_2)) \end{aligned} \tag{4.5}$$

Axiom 4.1 can be expressed with SHACL-SPARQL through the use of a SHACL Shape with constraint represented by listing 4.4 and target represented by listing 4.5. 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 4.6 exemplifies how a Shape is described. Axioms 4.2, 4.3, 4.4 and 4.5 follow the same pattern of axiom 4.1.

```
: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 4.4 – Constraint on creation events.

```
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 4.5 – Target for creation events.

```
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 .
```

Listing 4.6 – Shape for type validation on creation events.

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 $C \equiv \neg B$.

There are also some implicit restrictions that 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 4.6.

$$\forall e_i, E(e_i) \wedge subClassOf(E, DestructionEvent) \wedge ConcernsDestructionOf(E, O) \rightarrow \exists o_i(O(o_i) \wedge DestroyedBy(o_i, e_i)) \tag{4.6}$$

- (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 4.7.
  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.

$$\forall e_i, E(e_i) \wedge subClassOf(E, ModificationEvent) \wedge ConcernsModificationOf(E, O) \rightarrow \exists o_i(O(o_i) \wedge ModifiedBy(o_i, e_i)) \tag{4.7}$$

- (iii) If an event removes a relation of inherence, the quality destroyed by the event should be of the concerned quality class: axiom 4.8.

$$\forall e_i, E(e_i) \land subClassOf(E, QualityModificationEvent) \land ConcernsQuality(E, Q) \rightarrow \exists q_i(Q(q_i) \land DestroyedBy(q_i, e_i)) \tag{4.8}$$

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 not available. 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 one with a single responsibility in constructing the new graph. Listing 4.7 describes *ReplicateUniversalsShape* shape that targets uninteresting entities (listing 4.11) and only replicates triples through its rule (listing 4.10), listing 4.8 describes *StepIncrementStateShape* shape that only deals with events that cannot cause destruction of entities (listing 4.12) and listing 4.9 describes StepDecrementStateShape shape that deals with events that may cause the destruction of entities (listing 4.13).

```
:ReplicateUniversalsShape rdf:type owl:NamedIndividual ,
                                    sh:shape ;
                          sh:rule :ReplicateTriplesRule ;
                          sh:target :UniversalsTarget .]
```

Listing 4.7 – Shape that deals with individuals that will not be affected by inference.

```
:StepIncrementStateShape rdf:type owl:NamedIndividual ,
                                   sh:shape ;
                         sh:rule :CreationRule ,
                                 :DiscardLossObjectsRule ,
                                 :GainsQualityRule ,
                                 :GainsRelationRule ;
```

```
                          sh:target :UnnavoidableIndividualsTarget .
   ReplicateUniversalsShape rdf:type owl:NamedIndividual ,
                                    sh:shape ;
                           sh:rule :ReplicateTriplesRule ;
                           sh:target :UniversalsTarget .
```

Listing 4.8 – Shape that deals with individuals that do not participate in an event that removes triples.

```
:StepDecrementStateShape rdf:type owl:NamedIndividual ,
                                sh:shape ;
                       sh:rule :DiscardLossRelationsRule ,
                                :GainsRelationRule ;
                       sh:target :PossibleLossIndividualsTarget .
```

Listing 4.9 – Shape that deals with individuals that participate in an event that removes triples.

```
:ReplicateTriplesRule rdf:type owl:NamedIndividual ,
                             sh:SPARQLRule ;
                  sh:construct """
prefix : <https://hrssilva.github.io/ontology/omice.ttl#>
CONSTRUCT {
      $this ?r ?o .
}
WHERE {
      $this ?r ?o .
}
""" ;
                       sh:order 1 .
```

Listing 4.10 – Rule that only replicates all the triples from targets.

```
:UniversalsTarget rdf:type owl:NamedIndividual ,
                  sh:SPARQLTarget ;
                  rdfs:comment "Targets all universals, including
   individuals created through punning, annotation properties, object
   properties and data properties" ;
                  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 owl:Ontology .}
    UNION
    {?entity a owl:Class .}
    UNION
    {?entity a owl:AnnotationProperty .}
    UNION
    {?entity a owl:ObjectProperty .}
    UNION
    {?entity a owl:DataProperty .}
}
""" .
```

Listing 4.11 – Target for all entities that will not be affected by the inference.

```
omice:UnnavoidableIndividualsTarget rdf:type owl:NamedIndividual ,
                                    sh:SPARQLTarget ;
                                    rdfs:comment "Targets all
   individuals that do not lose properties or cease to exist by result
    of an 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 owl:NamedIndividual .
NOT EXISTS {?entity a owl:Class .}
NOT EXISTS {
    {
    ?entity :SubjectModifiedBy | :BearerModifiedBy ?event .
    ?event a ?ec .
    ?ec :RemovesRelation ?p .
    }
    UNION
    {
    ?entity :DestroyedBy ?event .
    }
}
}
""" .
```

Listing 4.12 – Target for all individuals that do not participate in an event that removes triples.

```
omice:PossibleLossIndividualsTarget rdf:type owl:NamedIndividual ,
                                    sh:SPARQLTarget ;
                                    rdfs:comment "Targets all
   individuals that may lose properties or cease to exist by result of
    an 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 owl:NamedIndividual .
NOT EXISTS {?entity a owl:Class .}
{
    {
        ?entity :SubjectModifiedBy | :BearerModifiedBy ?event .
        ?event a ?ec .
        ?ec :RemovesRelation ?p .
    }
    UNION
    {
        ?entity :DestroyedBy ?event .
    }
}
}
""" .
```

Listing 4.13 – Target for all individuals that participate in an event that removes triples.

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 the definitions for targets and rules. The targets for individuals (listings 4.12 and 4.13) 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. In a similar way,

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 are able to completely remove entities that were destroyed, 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.

```
:CreationRule rdf:type owl:NamedIndividual ,
sh:SPARQLRule ;
sh:construct """
prefix : <https://hrssilva.github.io/ontology/omice.ttl\#>
prefix owl: <http://www.w3.org/2002/07/owl\#>
CONSTRUCT {
  ?s a owl:NamedIndividual, ?c .
  }
  WHERE {
    $this a ?ec .
    ?ec :ConcernsCreationOf ?c .
    BIND(IRI(CONCAT(STR(?c), STR(NOW()))) as ?s) .
    }
    """ ;
```

- GainsQualityRule: Crates an individual of the concerned quality class and instantiates the concerned inherence relation between the new individual and the bearer.

```
:GainsQualityRule rdf:type owl:NamedIndividual ,
                          sh:SPARQLRule ;
                 sh:construct """
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\#>
CONSTRUCT {
            ?s a owl:NamedIndividual, ?qc .
            ?s ?r $this .
}
WHERE {
```

```
        $this :BearerModifiedBy ?e .
        ?e a ?ec .
        ?ec :GivesRelation ?r .
        ?ec :ConcernsQuality ?qc .
        BIND(IRI(CONCAT(STR(?qc), STR(NOW()))) as ?s) .
}
""" ;
```

- GainsRelationRule: Instantiates a relation between the subject and object-modified individuals.

```
:GainsRelationRule rdf:type owl:NamedIndividual ,
                            sh:SPARQLRule ;
                   sh:construct """
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\#>
CONSTRUCT {
            $this ?r ?o .
}
WHERE {
        $this :SubjectModifiedBy ?e .
        ?e a ?ec .
        ?ec :GivesRelation ?r .
        ?o :ObjectModifiedBy ?e .
}
""" ;
```

- DiscardLossObjectsRule: Replicates every triple for the target, except when an event destroyed the object of the triple.

```
:DiscardLossObjectsRule rdf:type owl:NamedIndividual ,
                                sh:SPARQLRule ;
                        sh:construct """
prefix : <https://hrssilva.github.io/ontology/omice.ttl\#>
CONSTRUCT {
            $this ?r ?o .
}
WHERE {
            $this ?r ?o
            NOT EXISTS {
            ?o :DestroyedBy ?event .
```

```
            }
}
""" ;
```

- DiscardLossRelationsRule: Replicates every triple for the target, except when an event removes the triple relation.

```
:DiscardLossRelationsRule rdf:type owl:NamedIndividual ,
                                    sh:SPARQLRule ;
                          sh:construct """
prefix : <https://hrssilva.github.io/ontology/omice.ttl\#>
CONSTRUCT {
            $this ?r ?o .
}
WHERE {
            $this ?r ?o .
            NOT EXISTS {
            $this :SubjectModifiedBy ?event .
            ?o :ObjectModifiedBy ?event .
            ?event :RemovesRelation ?r .
            }
            NOT EXISTS {
            $this :DestroyedBy ?event .
            }
}
""" ;
```

4.3 Demonstrations on Modeling for Inference

Using the proposed model and the defined inference rules, we can extract the behavior of events using SHACL inference. To validate this proposal, we utilize TopQuadrant's implementation of a SHACL API, built on top of Jena (TopQuadrant, 2017), for its convenience since most SHACL processors that implement SHACL-SPARQL are themselves a full-fledged triple store (which is the case with Jena itself). The following examples are simple models, importing only the proposed event model and demonstrating how to use these concepts.

To demonstrate how creation and destruction events can be utilized, we modeled an example using a simplified process of manufacturing natural coal. The manufactur-

ing consists of burning wood in a specific way to transform it into natural coal, as such we define the relevant object classes: *Wood* and *NaturalCoal*; and the event *ManufacturingOfNaturalCoal* with two event parts: *BurningOfWood* and *CreationOfCoal*. The event *BurningOfWood* concerns the destruction of the class *Wood* and the event *CreationOfCoal* concerns the creation of the class *NaturalCoal*.

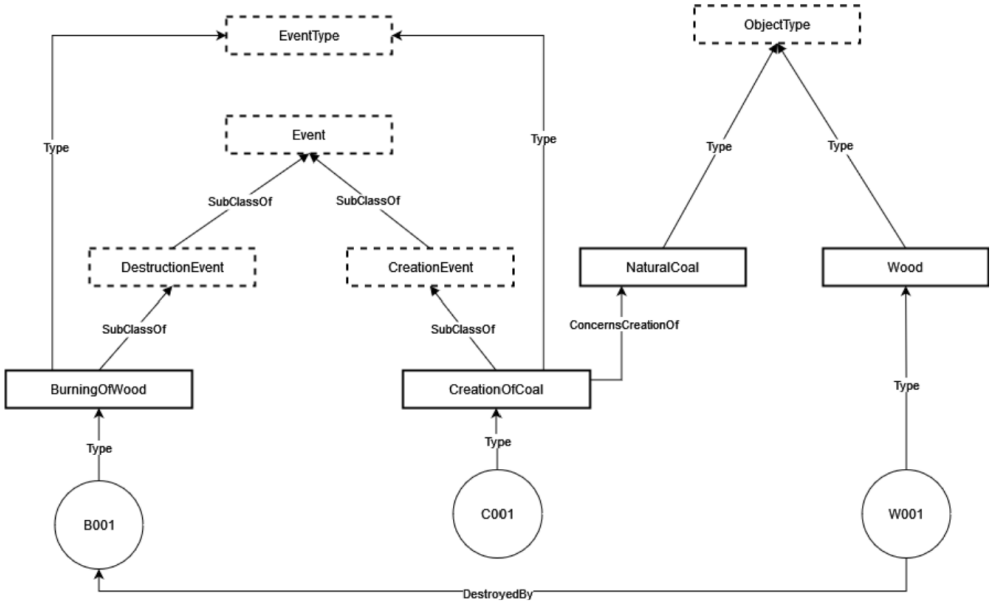

Figure 4.1 – Event model for creation of natural coal

The figure 4.1 shows the instantiation of this example's model with an individual of each event and an individual of *Wood* with the *DestroyedBy* participation relation with the individual of the *BurningOfWood* event, completing the model in the state where the manufacturing of natural coal is about to happen. Figure 4.2 shows all the triples that were included in the new model after inference, while Figure 4.3 shows all that were removed in the process[2]. By running the inference with OMICE as a *Shapes Graph* over the example as *Data Graph* we can see in the resulting model that the instance of the *Wood* class is no longer present, while there now exists a new individual of the *NaturalCoal* class.

To demonstrate how we can deal with quality modification events, we modeled a simple example of the painting of a vase. The painting consists of giving a new color to a vase, replacing its previous one. The object class *Vase* (with instance *V001*) and the quality class *Color* (with instance *C001*) are created to model the relevant continuants that will participate in the event, also defining the concerned inherence relation *InheresIn*

[2]The models, resulting models of inference and the graph difference image generation script can be found in the "examples" and "scripts" folders at <https://github.com/hrssilva/omice>

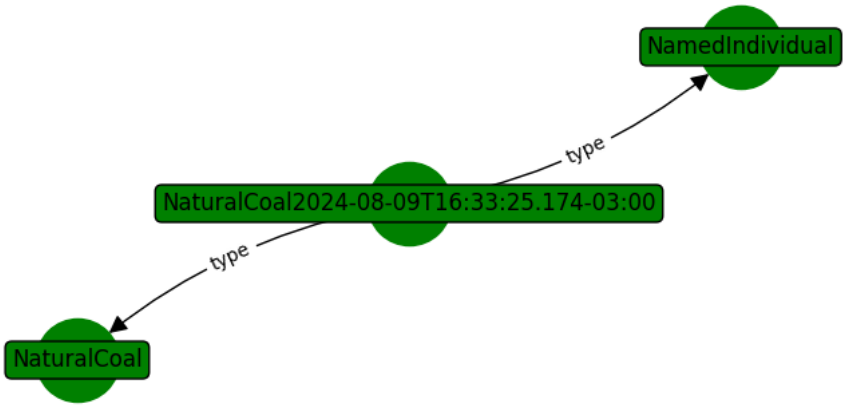

Figure 4.2 – New triples after Burning of Wood Event

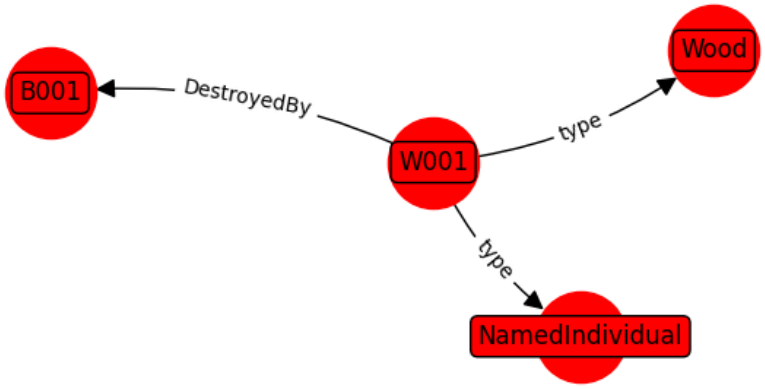

Figure 4.3 – Removed triples after Burning of Wood Event

and instantiating this relation between the individuals of *Color* and *Vase*. Two quality modification events are necessary: one to remove the previous color of the vase and one to give it a new color. As such, we create the *LossOfColor* event that removes the color quality of any object and the *PaintingOfVase* event that gives a color quality to a vase.

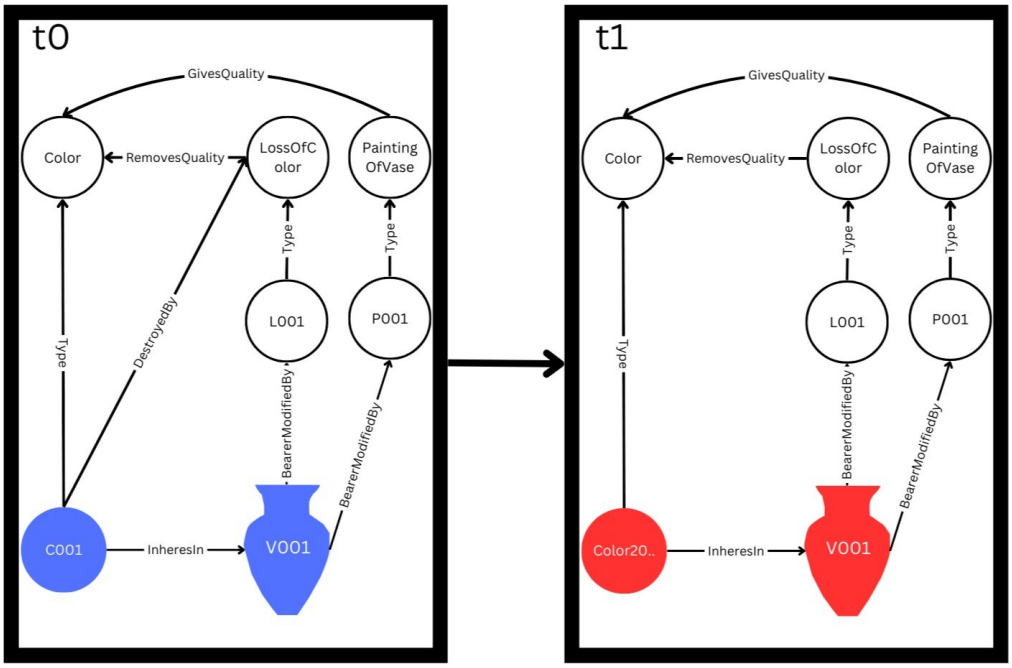

Figure 4.4 – The *Before* and *After* of the painting of a vase

The instantiation of the painting model with the events allows us to reason about the consequence of painting vase *V001*. Figure 4.4 shows that the triples related to color *C001* are no longer present in the model, and a new instance of *Color* has been created, inhering in vase *V001*.

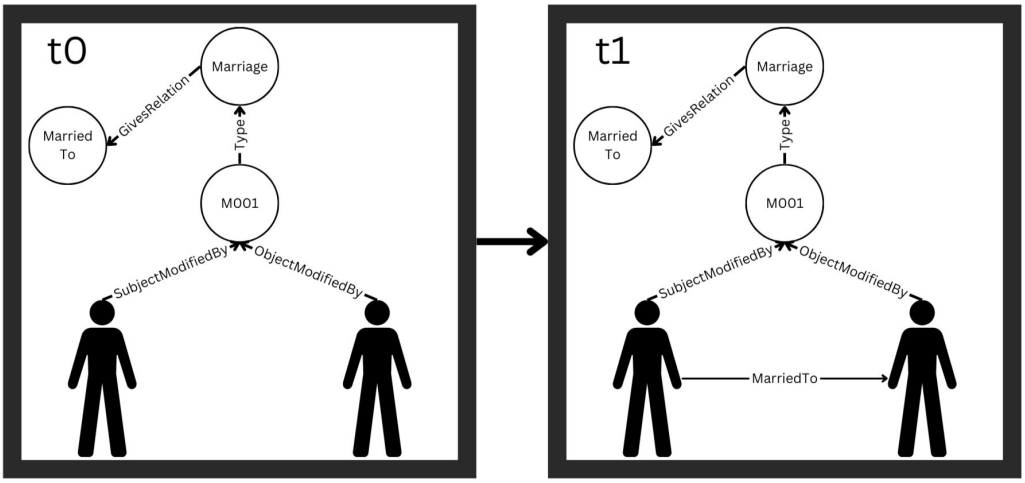

Figure 4.5 – The *Before* and *After* of the marriage event

We define a simple marriage event to demonstrate how relation modification events can be used. The *Marriage* event, a subclass of *RelationModificationEvent*, concerns the

class *Person* both as subject and object of a concerned (through the *GivesRelation* relation) symmetric relation *MarriedTo*.

The instantiation of this marriage model with two individuals of *Person* with relationships with an individual of *Marriage*, one through the *SubjectModifiedBy* relation, and another through *ObjectModifiedBy* relation, completes the model in the state where we can reason about the occurrence of the *Marriage* event individual. Figure 4.5 shows the resulting instantiation of the *MarriedTo* relation after inference.

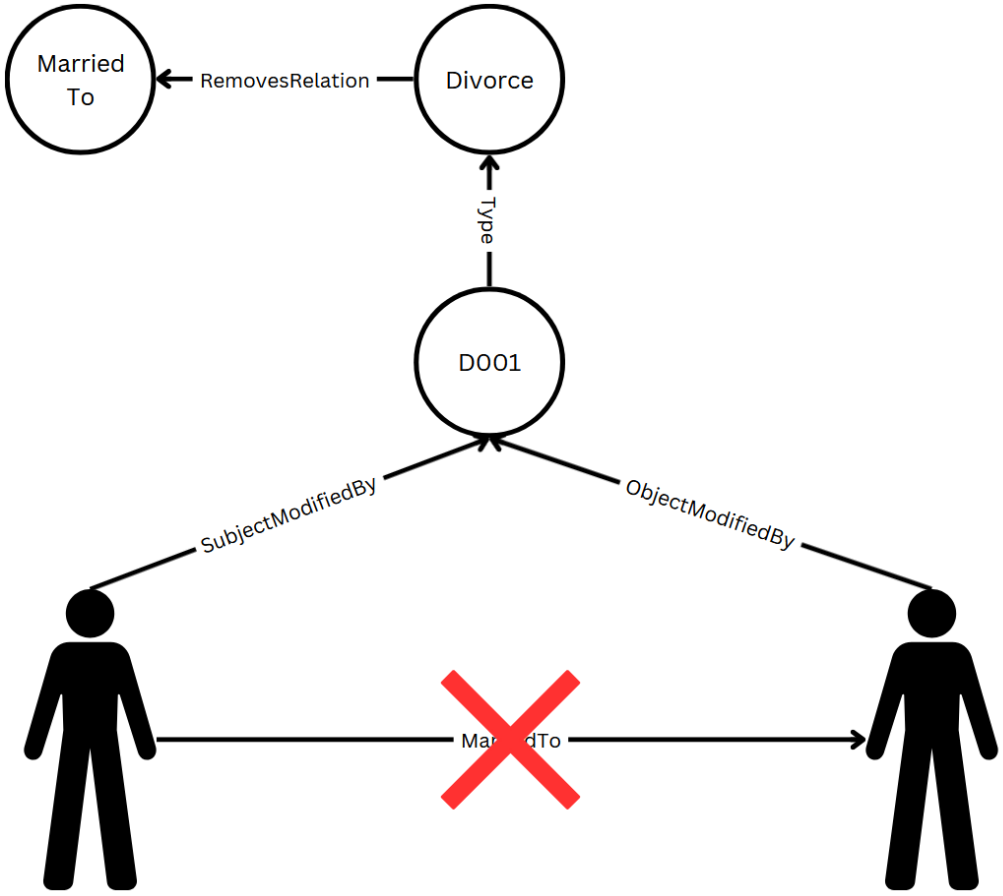

Figure 4.6 – Removed relation after divorce event

Figure 4.6 shows what happens if we extend this resulting model further with the class *Divorce* that concerns, through *RemovesRelation*, *MarriedTo* and change the participation of the two *Person* individuals to an instance of *Divorce*. We can see that after running the inference there is no more *MarriedTo* relation between the individuals.

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 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 to it. 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 to 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*.

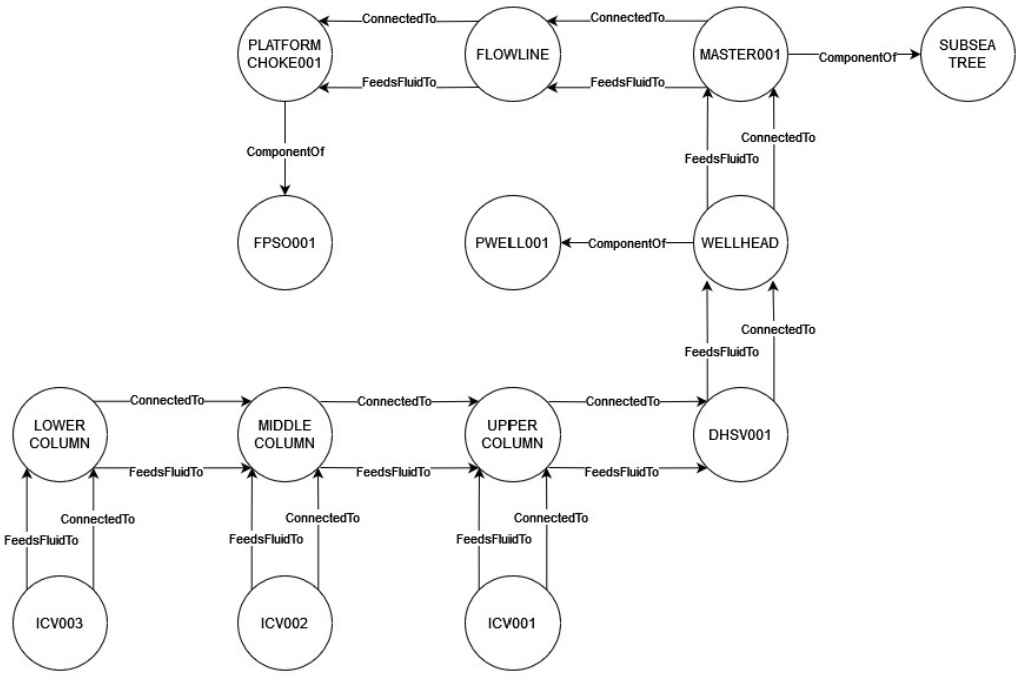

Figure 5.1 – Configuration of equipment for offshore oil extraction.

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 own instance of operational state quality (either *operational* or *not_operational*). The other relevant instances are: *Production_Well(PWELL001)* and *FPSO(FPSO001)*. Figure 5.1 shows the configuration of valves and equipment of this example.

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 5.2 illustrates how we have modeled the *ClosingOfValve* event, including individuals.

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

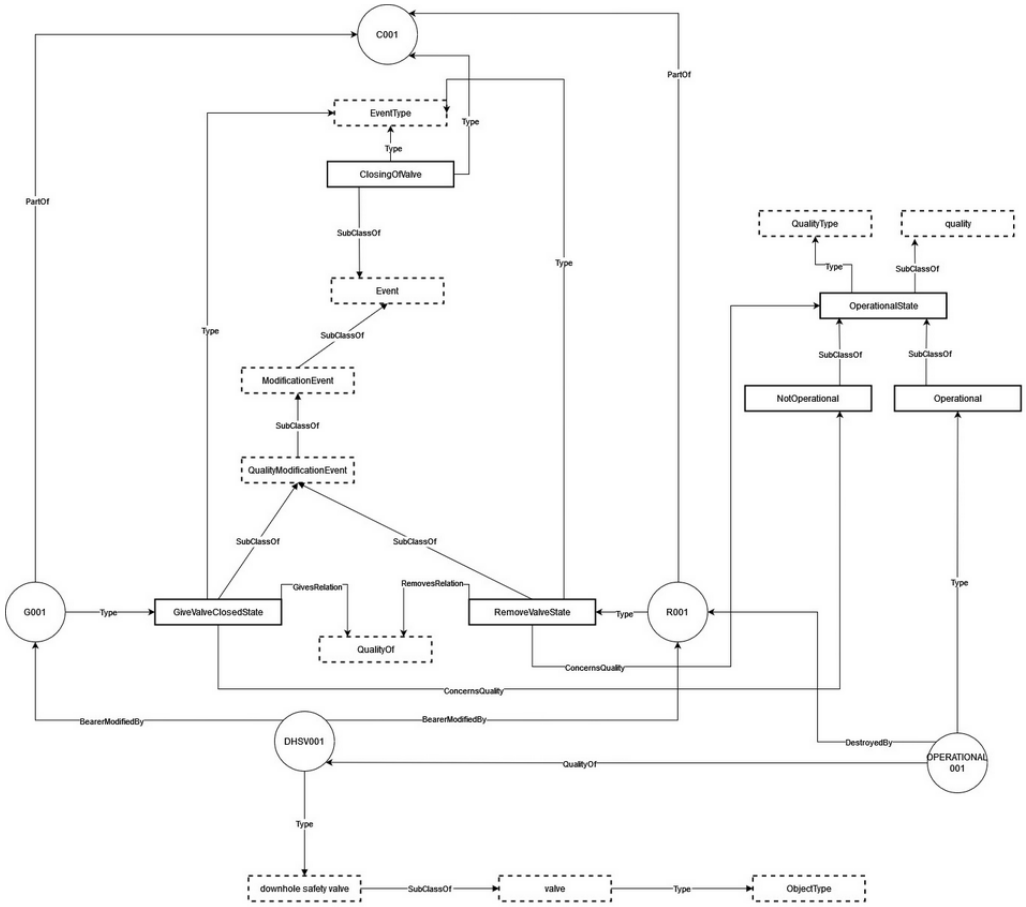

Figure 5.2 – *ClosingOfValve* example model.

not_operational inhering in it, resulting in the query in listing 5.1.

```
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 .
}
```

Listing 5.1 – Query to know if well is feeding oil to platform.

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

Listing 5.2 – Query result for the initial model state.

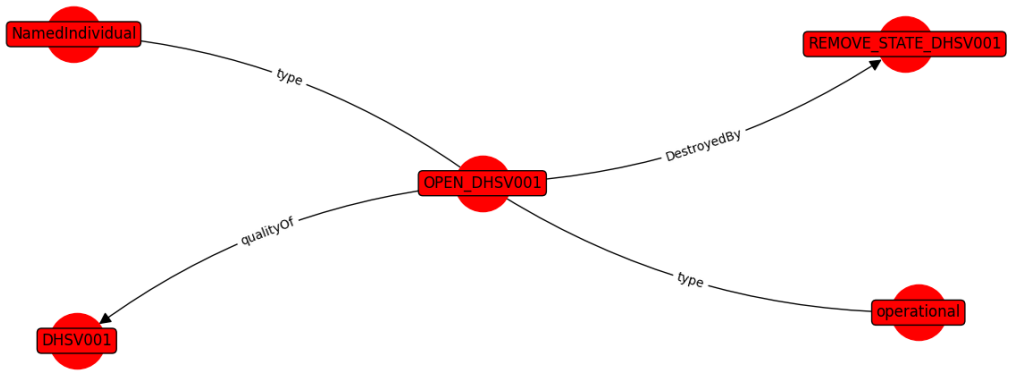

Figure 5.3 – Removed triples after *ClosingOfValve*

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

Listing 5.3 – Query result with a closed DHSV.

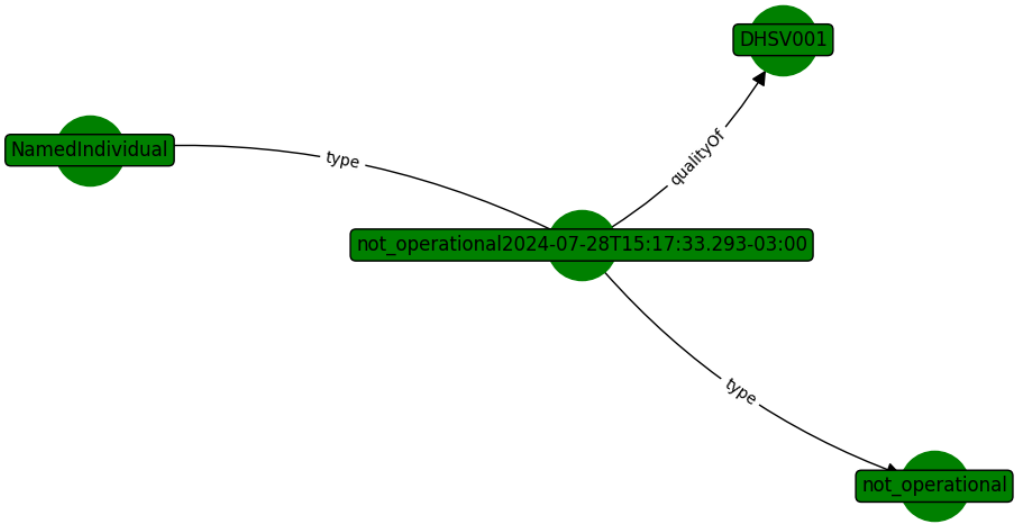

Figure 5.4 – New triples after *ClosingOfValve*

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 5.2. By instantiating a *ClosingOfValve* event, targeting *DHSV001*, and running the inference, we can see that the result, seen in Figure 5.3 and Figure 5.4, ends up resulting in no available path, since *DHSV001* closes the only path between the production column and the wellhead. When rerunning the query of Listing 5.1, we then receive *False*, as shown in Listing 5.3.

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 extracting 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 also by taking technological compatibility in high regard.

6.1 A Discussion On Different Approaches

Before reaching the proposed model, we have experimented with some taxonomy approaches to implement the described scenario. Besides the model presented in section 4.1, two other models proved sufficient to represent the event types: the first one included the same event classes already presented but described them as metatypes, while the second one extended the proposed model with subtypes of the *EventType* metatype inspired by the original upper-level types proposed by Rodrigues; Carbonera; Abel.

The first alternative model is the least compromising and the easiest to adopt since it only requires the usage of the proposed metatypes as types of the modeler's choice of event classes. This option allows different models to utilize the given relations to represent an event behavior with minor changes (or none at all) to the original taxonomy. Additionally, using descriptive metatypes allows for OWL axiomatization of the necessary restrictions (as opposed to the SHACL validation rules presented), facilitating development. Although there is axiomatization, it only supports restrictions based on classes, not on individuals, failing to properly represent restrictions on participation. Another, perhaps even bigger, flaw lies in the semantics implied by abstracting the descriptions to this higher level: the event *PaintingOfVase*'s type might describe it as concrete and as carrying identity, but not as a *QualityModificationEvent*, since it is, in fact, a specific type of this event.

The second alternative model fixes the issue of axiomatization by describing not only metatypes, but also types, having the same taxonomy of events as the final proposed model. At the same time, it alleviates the flaw in semantics by utilizing more general subtypes of *EventType*, but failing to, in fact, fix it. This model is better founded and covers all necessary restrictions with OWL axioms, but the importance of creating a consistent model without unclear semantics outweighs the importance of representing all restrictions with OWL. This brought us to the final proposed model, where there are no subtypes to *EventType*, and to the use of rule-based validation.

6.2 Current Limitations

Although we were successful in operationalizing 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 obviously clear in our examples when repainting a vase by destroying its color and 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.

6.3 Future Works

As section 6.2 made clear, there is still much room for growth in our current proposition, which we intend to do. Another interesting door this work opens 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.

REFERENCES

ALLEN, J. F. Maintaining knowledge about temporal intervals. **Communications of the ACM**, ACM New York, NY, USA, v. 26, n. 11, p. 832–843, 1983.

ALMEIDA, J. et al. gufo: A lightweight implementation of the unified foundational ontology (ufo). 01 2020.

ANICIC, D. et al. Etalis: Rule-based reasoning in event processing. In: ____. **Reasoning in Event-Based Distributed Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 99–124. ISBN 978-3-642-19724-6. Available from Internet: <https://doi.org/10.1007/978-3-642-19724-6_5>.

BENEVIDES, A. et al. Representing a reference foundational ontology of events in sroiq. **Applied Ontology**, v. 14, p. 1–42, 07 2019.

BENNETT, J. What events are. In: GALE, R. M. (Ed.). **The Blackwell Guide to Metaphysics**. [S.l.]: Wiley-Blackwell, 2002. p. 43.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. **Scientific American**, v. 284, n. 5, p. 34–43, may 2001. Available from Internet: <http://www.sciam.com/2001/0501issue/0501berners-lee.html>.

BREWER, T. et al. Digital Twin Technology in the Field Reclaims Offshore Resources. In: . [S.l.: s.n.], 2019. (OTC Offshore Technology Conference, Day 1 Mon, May 06, 2019).

DAVIDSON, D. The individuation of events. In: ____. **Essays in Honor of Carl G. Hempel: A Tribute on the Occasion of his Sixty-Fifth Birthday**. Dordrecht: Springer Netherlands, 1969. p. 216–234. ISBN 978-94-017-1466-2. Available from Internet: <https://doi.org/10.1007/978-94-017-1466-2_11>.

DING, L. et al. Using ontologies in the semantic web: A survey. In: ____. **Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems**. Boston, MA: Springer US, 2007. p. 79–113. ISBN 978-0-387-37022-4. Available from Internet: <https://doi.org/10.1007/978-0-387-37022-4_4>.

DING, Z.; PENG, Y.; PAN, R. Bayesowl: Uncertainty modeling in semantic web ontologies. In: ____. **Soft Computing in Ontologies and Semantic Web**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 3–29. ISBN 978-3-540-33473-6. Available from Internet: <https://doi.org/10.1007/978-3-540-33473-6_1>.

ERMOLAYEV, V.; KEBERLE, N.; MATZKE, W.-E. An ontology of environments, events, and happenings. In: **2008 32nd Annual IEEE International Computer Software and Applications Conference**. [S.l.: s.n.], 2008. p. 539–546.

GRUBER, T. R. A translation approach to portable ontology specifications. **Knowledge Acquisition**, v. 5, n. 2, p. 199–220, 1993. ISSN 1042-8143. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1042814383710083>.

GUARINO, N. Formal ontology, conceptual analysis and knowledge representation. **International Journal of Human-Computer Studies**, v. 43, n. 5, p. 625–640, 1995. ISSN 1071-5819. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S107158198571066X>.

GUARINO, N.; DANIEL, O.; STEFFEN, S. What is an ontology. In: ____. **Handbook on Ontologies**. Dordrecht: Springer Verlag, 2009. Available from Internet: <iaoa.org/isc2012/docs/Guarino2009_What_is_an_Ontology.pdf>.

GUIZZARDI, G.; GUARINO, N.; ALMEIDA, J. P. A. Ontological considerations about the representation of events and endurants in business models. In: ROSA, M. L.; LOOS, P.; PASTOR, O. (Ed.). **Business Process Management**. Cham: Springer International Publishing, 2016. p. 20–36. ISBN 978-3-319-45348-4.

GUIZZARDI, G. et al. Towards ontological foundations for the conceptual modeling of events. In: . [S.l.: s.n.], 2013. v. 8217, p. 327–341. ISBN 978-3-642-41923-2.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; van Harmelen, F. From shiq and rdf to owl: the making of a web ontology language. **Journal of Web Semantics**, v. 1, n. 1, p. 7–26, 2003. ISSN 1570-8268. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S1570826803000027>.

KEET, C. M. An introduction to ontology engineering. In: ____. [S.l.]: C. Maria Keet, 2020. chp. 10, p. 193 – 209.

KNUBLAUCH, H.; ALLEMANG, D.; STEYSKAL, S. **SHACL Advanced Features**. 2017. <https://www.w3.org/TR/2017/NOTE-shacl-af-20170608/>. Accessed at: february 4th, 2024.

LEI, L. et al. Event prediction based on causality reasoning. In: NGUYEN, N. T. et al. (Ed.). **Intelligent Information and Database Systems**. Cham: Springer International Publishing, 2019. p. 165–176. ISBN 978-3-030-14799-0.

LI, F. et al. Time event ontology (teo): to support semantic representation and reasoning of complex temporal relations of clinical events. **Journal of the American Medical Informatics Association**, v. 27, n. 7, p. 1046–1056, 07 2020. ISSN 1527-974x. Available from Internet: <https://doi.org/10.1093/jamia/ocaa058>.

LI, S.; CHEN, S.; LIU, Y. A method of emergent event evolution reasoning based on ontology cluster and bayesian network. **IEEE Access**, v. 7, p. 15230–15238, 2019.

MASOLO, C. et al. **WonderWeb Deliverable D18**. 2003. <http://wonderweb.man.ac.uk/deliverables/documents/D18.pdf>.

MEPHAM, W.; GARDNER, S. Implementing discrete event calculus with semantic web technologies. In: **2009 Fifth International Conference on Next Generation Web Services Practices**. [S.l.: s.n.], 2009. p. 90–93.

MOTIK, B. On the properties of metamodeling in owl. In: GIL, Y. et al. (Ed.). **The Semantic Web – ISWC 2005**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 548–562. ISBN 978-3-540-32082-1.

OTTE, J. N.; BEVERLEY, J.; RUTTENBERG, A. Bfo: Basic formal ontology. **Applied ontology**, v. 17, n. 1, p. 17–43, 2022.

OWL Working Group. **OWL 2 Web Ontology Language Primer (Second Edition)**. 2012. <https://www.w3.org/TR/owl2-primer/>. Accessed at: july 4th, 2024.

PADILLA-CUEVAS, J.; REYES-ORTIZ, J. A.; BRAVO, M. Ontology-based context event representation, reasoning, and enhancing in academic environments. **Future Internet**, v. 13, n. 6, 2021. ISSN 1999-5903. Available from Internet: <https://www.mdpi.com/1999-5903/13/6/151>.

PEDRINACI, C.; DOMINGUE, J.; MEDEIROS, A. K. Alves de. A core ontology for business process analysis. In: BECHHOFER, S. et al. (Ed.). **The Semantic Web: Research and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 49–64. ISBN 978-3-540-68234-9.

QUINTON, A. Objects and events. **Mind**, [Oxford University Press, Mind Association], v. 88, n. 350, p. 197–214, 1979. ISSN 00264423, 14602113. Available from Internet: <http://www.jstor.org/stable/2252963>.

RODRIGUES, F.; CARBONERA, J.; ABEL, M. Upper-level types of occurrent based on the principle of ontological conservation. In: ____. **Conceptual Modeling**. [S.l.]: Springer International Publishing, 2020. p. 353–363. ISBN 978-3-030-62521-4.

RODRIGUES, M. A. F. H. What to consider about events: A survey on the ontology of occurrents. **Applied Ontology**, v. 14, n. 4, p. 343–378, 11 2019.

SANTOS, N. O. et al. O3PO: A domain ontology for offshore petroleum production plants. **Expert Systems with Applications**, v. 238, p. 122104, 2024. ISSN 0957-4174.

SMITH, B.; GRENON, P. The cornucopia of formal-ontological relations. **Dialectica**, v. 58, p. 279–296, 2005. Available from Internet: <https://api.semanticscholar.org/CorpusID:12720558>.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: Principles and methods. **Data Knowledge Engineering**, v. 25, n. 1, p. 161–197, 1998. ISSN 0169-023X. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0169023X97000566>.

TOPQUADRANT. **TopBraid SHACL API**. 2017. <https://github.com/TopQuadrant/shacl>. Accessed at: february 4th, 2024.

XUE, J.-R.; FANG, J.-W.; ZHANG, P. A survey of scene understanding by event reasoning in autonomous driving. **International Journal of Automation and Computing**, v. 15, n. 3, p. 249–266, Jun 2018. ISSN 1751-8520. Available from Internet: <https://doi.org/10.1007/s11633-018-1126-y>.

ZHONG, Z. et al. Event ontology reasoning based on event class influence factors. **International Journal of Machine Learning and Cybernetics**, v. 3, n. 2, p. 133–139, Jun 2012. ISSN 1868-808X. Available from Internet: <https://doi.org/10.1007/s13042-011-0046-8>.