Possible Hyperworlds: representing Kripke models in RDF

Christopher Leturc [a,*], Fabien Gandon [b] and Maxime Lefrançois [c]

[a] *MDSC, Université Côte d'Azur, CNRS, I3S, France*
E-mail: christopher.leturc@univ-cotedazur.fr
[b] *Wimmics, Inria, Université Côte d'Azur, CNRS, I3S, France*
E-mail: fabien.gandon@inria.fr
[c] *Mines Saint-Étienne, Univ. Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023, Saint-Étienne, France*
E-mail: maxime.lefrancois@emse.fr

Abstract. Modal logic extends propositional logic by adding modal operators, and enables us to reason with concepts such as necessity, possibility, knowledge, belief, obligation, or consequences of actions. The most common way to interpret modal logic formulas is to apply the possible worlds semantics and to formally represent it with a Kripke model. Kripke models consist of a set of possible worlds, accessibility relations between possible worlds, and a valuation function that assigns to propositional atoms a truth value relative to those worlds. Existing work explored the connection between modal logic and Description Logics, including Modal Description Logics, and consider the inclusion of a Kripke model in their semantics. We are interested in considering a novel perspective, where we aim to represent in RDF a Kripke model, formulas from a modal logic language, and in which possible worlds such formulas are verified. Such a representation could have several applications: 1. contributing to a standard exchange format for modal logic applications, including reasoners; 2. facilitating the Open Science initiative in the modal logic research community; 3. supporting the representation and management of uncertainty or other modalities; and 4. facilitating interoperability between heterogeneous systems. As a starting point, our intuition is that a set of triples could be translated to a formula, and one could explicit in what possible worlds a formula is verified using metadata about this set of triples. Following that intuition, the main contribution of this article is to provide an RDF representation of normal Kripke models, applying the linked data principles and standards, focusing on identifying possible worlds and expressing the (N+1)-ary accessibility relations between them. We call *possible hyperworld* the result of this hypermedia approach to representing possible worlds. We discuss the different modeling alternatives for linking sets of triples to possible worlds, and for representing accessibility relations. We motivate one modeling choice, and compile our design rationale in a formal vocabulary for describing Kripke models: the *Possible Worlds and Kripke Structures Ontology* (PWKSO). Adopting a simple and generic usage pattern, we illustrate PWKSO with examples from epistemic multi-modal logic, dyadic deontic logic, and the dynamic logic of mental attitudes and joint actions (DL-MA). Perspectives and future work include 1. defining a translation, in the context of an RDF dataset, of sets of triples to modal logic formulas; 2. querying, instantiating, reasoning with possible hyperworlds; 3. nesting possible worlds or Kripke structures as a way to model Dynamic Epistemic Logics (DEL) or Propositional Dynamic Logics (PDL); 4. potential applications that leverage existing modal logic formalisms for example to merge knowledge graphs, represent the knowledge or beliefs of agents, represent the effects of actions or norms, or develop model checking applications.

Keywords: Modal Logic, Possible Worlds Semantics, Kripke Models, Ontology, RDF annotation, N-ary relations

*Corresponding author. E-mail: christopher.leturc@univ-cotedazur.fr.

1. Introduction

Modal logic is a branch of philosophical logic that extends propositional logic and adds modal operators, such as for describing necessity, possibility, knowledge, belief, consequences of actions, or obligations [1]. The most widely applied semantics for interpreting modal logic formulas is the possible world semantics [2]. A possible world is an abstract representation of a state in which the real world could be. For instance, if we do not know if a statement S is true, we can represent that situation with two possible worlds: one world where S is true and another where S is false. To represent the fact that we do not know about S, we declare a binary relation between the two possible worlds representing the fact that we cannot distinguish between these two possible worlds. Such a combination of possible worlds and relations between them is called a *Kripke frame* [2]. A Kripke frame combined with a valuation function that assigns a truth value to propositional atoms is called a *Kripke model*.

Kripke frames have been widely used in areas such as formal verification, artificial intelligence, and philosophy, where they provide a powerful tool for reasoning about uncertainty, knowledge, and belief. They can be used to model a wide range of scenarios, from the behavior of complex systems to the interpretation of natural language statements. These models can be used, e.g., to reason about the consequences of actions, such as determining what effects an action will have on the possible worlds that are accessible from a given starting point. Kripke models are used to formalize many different concepts in logic and computer science, such as modal logic, epistemic logic i.e., reasoning about knowledge of agents [3], temporal logic, and dynamic logics, i.e., reasoning about actions and their consequences [4], or deontic logics, i.e., reasoning about norms and laws [5]. They are also useful for reasoning about knowledge and beliefs in multi-agent systems and, in this context, the accessibility relation $\mathcal{R}$ they provide is used to capture what each agent knows or believes about the world.

One of the main limitations of modal logic is that the models that are used to interpret formulas must be given a priori. This requires a lot of effort and expertise to create those models, and it can be difficult to ensure that the models are consistent, or adequately represent what was aimed. Additionally, because models must be given a priori, they are not always able to capture the full range of possibilities that might be relevant to a given problem, which can limit the applicability of modal logic to certain situations. In this context, the Semantic Web could provide an interoperable and standard way to capture and share Kripke frames, facilitating their reuse across different applications and domains. This has the potential to contribute to a standard exchange format for modal logic applications, including modal logic reasoners, and to facilitate the Open Science initiative in the modal logic research community.

In recent years, there has been a growing interest in integrating modal operators into the Semantic Web [6–9]. Let us consider the case of inconsistency that may arise when merging multiple knowledge graphs. In that case, possible worlds can be used to separate two pieces of contradictory information into two distinct possible worlds. This approach not only helps to prevent inconsistencies when merging knowledge but also opens up new avenues for representing knowledge of agents, beliefs, norms, etc. on the Semantic Web. In a context where more and more systems are exposed on the Web and use or combine different AI techniques, the possible world semantics can support the representation and management of uncertainty or other modalities, and facilitate interoperability between heterogeneous systems.

The main contribution of this article is to propose an RDF representation of Kripke frames following the linked data principles and standards. We call *possible hyperworld* the result of this hypermedia approach to representing possible world. We show that one can use several alternatives to represent possible worlds and accessibility relations between them, but we recommend a specific pattern and we provide the formal vocabulary for describing such Kripke frames. As a summary, our contribution aims at enabling different families of motivating scenarios:

- contribute to a standard exchange format for modal logic applications, including reasoners;
- facilitate the Open Science initiative in the modal logic research community;
- support the representation and management of uncertainty or other modalities;
- facilitate interoperability between heterogeneous systems.

The rest of this article is organised as follows. Section 2 presents a running example for the whole article, taken in the third family of motivating scenarios. Section 3 presents the relevant state of the art : we introduce the Kripke models, discuss existing approaches that introduce modal operators in the Semantic Web, and present different approaches for RDF metadata representation. Section 4 evaluates the alternative RDF metadata representations for possible worlds as linked data and Section 5 does the same for representing accessibility relations. Section 6 compiles the main contributions of the paper: the resulting design pattern and ontology, which we refer to as the *Possible Worlds and Kripke Structures Ontology* (PWKSO). Section 7 presents a discussion of the proposal and suggests direct future work, possible extensions, and applications. Section 8 concludes the article.

2. Running example

In this section we introduce a running example for this article, illustrated in Figure 2. This example is related to the management of uncertainty, and more specifically to source reconciliation, and it involves two agents, Bob and Alice. Bob asks Alice about the list of current presidents. Alice has access to two huge databases (ex. DBpedia and Wikidata) to provide Bob with an up-to-date answer. However, Alice encounters a problem: in DBpedia, it is said that `{<x> ex:isPresidentOf <France>}` (triple t_1), while in Wikidata, it is said that `{<x> ex:isPresidentOf <England>}` (triple t_2). If we assume that `ex:isPresidentOf` is functional and that `<France>` and `<England>` are different individuals, then this leads to a contradiction. To resolve this issue, Alice adopts an approach based on the possible world semantics by separating the two inconsistent pieces of information into two distinct worlds: w_1 containing t_1, and w_2 containing t_2. This way, Alice can communicate her beliefs to Bob without making any decisions or assumptions about which triple should be taken as true.

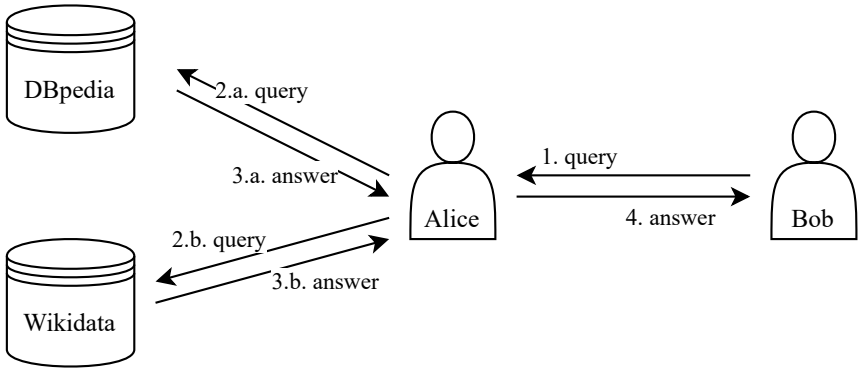

Fig. 1. Running example: a simple yet representative data integration situation.

This situation could also lead to a cooperation between Alice and Bob. Bob may inform Alice that there is no president in England, and she would therefore lean towards w_1. This example illustrates the interest of being able to represent and exchange knowledge and beliefs accurately. If Alice had chosen to remove triple `{<x> ex:isPresidentOf <France>}`, then she would have spread false information to Bob. Instead, by transferring both possible worlds, Alice defers to Bob the trustworthiness assessment of the information she transmits. Bob may confront the facts with other knowledge using some reasoning means to identify that world w_2 is impossible, and remove it from the information he received. Bob may further solicit other agents, potentially more knowledgeable or having better reasoning capabilities.

Therefore, to manage possibly contradicting information and allow differed processing, our proposed approach is to represent both possibilities using a Kripke model. Semantic Web technologies offers ways to identify, represent and exchange any volume of knowledge on the Web, and is therefore a good enabler for this scenario. More generally, we believe our contribution in this article will support a wide range of

scenarios from the different families mentioned in Section 1. In the next sections we will survey existing representation alternatives, and evaluate how appropriate they are to represent Kripke models.

3. State of the art

In this section, we first recall the fundamentals of the Kripke formalism to represent possible worlds. Then, we survey related approaches from the Semantic Web literature that considered modal logic and which, so far, mostly focused on Modal Description Logics. Finally, to prepare for possible worlds representation, we describe and compare five alternatives for annotating data in RDF: RDF reification, singleton properties, named graphs, RDF-star, and N3.

3.1. Kripke models

Kripke models [2] are used in modal logic to represent possible worlds and the relationships between them. As illustrated in Figure 2, each possible world of a Kripke model is represented by a node, and the relationships between worlds are represented by directed edges. These relationships represent accessibility relations (which worlds are accessible from other worlds), which can represent a variety of different concepts including temporal relations (which worlds come before or after another world), or epistemic relations (which worlds have a similar state of knowledge than in another world).

Formally, given a set of propositional atoms $\mathcal{P}$, a *basic Kripke model* [2] is a tuple $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ where:

- $\mathcal{W}$ is a non-empty set of possible worlds,
- $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$ is the accessibility relation on $\mathcal{W}$,
- $\mathcal{V} : \mathcal{P} \to 2^{\mathcal{W}}$ is a valuation function, i.e., $\mathcal{V}(p)$ is the set of possible worlds where p is true.

The pair $(\mathcal{W}, \mathcal{R})$ is called a *Kripke frame*. $(w, v) \in \mathcal{R}$ or $w\mathcal{R}v$ indicates that that world v is accessible from world w. Some work define $\mathcal{R}$ as a function from $\mathcal{W}$ to $\mathcal{W}$, giving the equally common notation $\mathcal{R}(w)$ to indicate the set of worlds accessible from w (also written: accessible worlds from w). The interpretation of the accessibility relation $\mathcal{R}$ depends on the application. For example in epistemic logic it is usually an equivalence relation noted $\mathcal{K}$ over the set of worlds, and is also called an *indistinguishability relation*.

The basic modal logic language $\mathcal{L}$ is generated by the following grammar given in Backus-Naur Form:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Box\phi$$

$\Box$ is a modal operator, and there are various interpretations of formula $\Box\phi$ depending on the application. For example it may mean "it is necessarily true that ϕ holds" in epistemic modal logic, or "it is believed that ϕ holds" in doxastic modal logic. The dual modality represented by $\Diamond$ refers to the possibility and is usually syntactically defined as: $\neg\Diamond\neg\phi \equiv \Box\phi$.

A modal logic language may define more than one modal operators such as K_i to represent that "agent i knows that ϕ is true" in epistemic multi-modal logic, or B_i to represent that "agent i believes that ϕ is true" in doxastic multi-modal logic. The corresponding Kripke frame holds more than one relations, such as $(\mathcal{W}, \{\mathcal{K}_i\}_{i \in I})$ for the epistemic multi-modal Kripke frame, where I is the set of agents.

A Kripke model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ is used to encode the truth value of a modal formula $\phi \in \mathcal{L}$ in a possible world $w \in \mathcal{W}$, noted $\mathcal{M}, w \models \phi$ if ϕ is true (or *verified*) in world w, or $\mathcal{M}, w \not\models \phi$ if ϕ is false in world w. For all $w \in \mathcal{W}$ and given any two formulas $\phi, \psi \in \mathcal{L}$ and any propositional atom $p \in \mathcal{P}$, $\mathcal{M}$ is such that :

1. $\mathcal{M}, w \not\models \bot$
2. $\mathcal{M}, w \models p$ iff $w \in \mathcal{V}(p)$
3. $\mathcal{M}, w \models \neg\phi$ iff $\mathcal{M}, w \not\models \phi$
4. $\mathcal{M}, w \models \phi \wedge \psi$ iff $\mathcal{M}, w \models \phi$ and $\mathcal{M}, w \models \psi$
5. $\mathcal{M}, w \models \Box\phi$ iff $\forall v \in \mathcal{W},$ if $w\mathcal{R}v$ then $\mathcal{M}, v \models \phi$, i.e., ϕ is true in all accessible worlds from w

Modality $\Diamond$ corresponds to an existential definition: $\mathcal{M}, w \models \Diamond\phi$ iff $\exists v \in \mathcal{W}$ s.t. $w\mathcal{R}v$ and $\mathcal{M}, v \models \phi$, i.e., there exists some world v accessible from w where ϕ is true.

We say that ϕ is *valid* in $\mathcal{M}$, noted $\mathcal{M} \models \phi$, if, and only if, it is verified in all possible worlds, i.e., $\forall w \in \mathcal{W},\ \mathcal{M}, w \models \phi$.

Example 3.1. *Let us consider the agents of our scenario from Section 2. Let t_1 and t_2 be two propositional atoms such that t_1 :=* `{<x> ex:isPresidentOf <France>}` *and t_2 :=* `{<x> ex:isPresidentOf <England>}`*. Let $\mathcal{W} = \{w_1, w_2\}$ be the set of possible worlds for Alice, and $\mathcal{V}$ be a valuation function such that $\mathcal{V}(t_1) = \{w_1\}$ and $\mathcal{V}(t_2) = \{w_2\}$. In this situation, w_1 represents* `{<x> ex:isPresidentOf <France>}`*, i.e., t_1 is true at w_1, and t_2 is false, i.e., $\mathcal{M}, w_1 \models t_1$, and $\mathcal{M}, w_1 \models \neg t_2$. Similarly, w_2 represents* `{<x> ex:isPresidentOf <England>}`*. Figure 2 illustrates the accessibility relation $\mathcal{K}_{Alice}$ that models the point of view of Alice.*

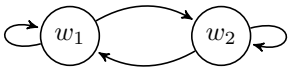

Fig. 2. The epistemic Kripke structure modeling the state of knowledge of Alice.

In Figure 2, both worlds w_1 and w_2 are accessible from themselves and from each other, and are therefore indistinguishable for Alice. This situation is formally described by the fact formulas $\neg\Box t_1$ (it is not necessarily true that t_1 holds) and $\neg\Box t_2$ (it is not necessarily true that t_2 holds) are valid in $\mathcal{M}$. Let us detail the first case. t_1 is false in w_2 and w_2 is accessible from w_1, so there exists a world $v \in \mathcal{K}_{Alice}(w_1)$ s.t. $\mathcal{M}, v \models \neg t_1$. Therefore, $\mathcal{M}, w_1 \models \neg\Box t_1$. Actually, w_2 is also accessible from w_2, so $\mathcal{M}, w_2 \models \neg\Box t_1$. Therefore, $\mathcal{M} \models \neg\Box t_1$. The same holds for t_2 (t_2 is false in w_1, and w_1 is accessible from w_1 and w_2).

In Example 3.1, the knowledge of Alice is expressed using a unary modal operator. The definition of basic Kripke structure has been generalized to N-ary modal operators. For example dyadic deontic logic [10] would be useful in our example to express: "when t_1, it ought to be $\neg t_2$" and "when t_2, it ought to be $\neg t_1$", i.e., someone cannot be the president of both countries. This would be described with the binary dyadic deontic modal operator O and two formulas: $O(t_1, \neg t_2)$ and $O(t_2, \neg t_1)$. The corresponding Kripke frame uses a ternary relation $\mathcal{T} \subseteq \mathcal{W}^3$. By abuse of notation, we either write $(v, w, z) \in \mathcal{T}$, or $(w, z) \in \mathcal{T}(v)$. Dyadic deontic logic imposes that for all possible worlds z, we have $\mathcal{M}, z \models O(\phi, \psi)$ if and only if $\forall (x, y) \in \mathcal{T}(z)$, if $\mathcal{M}, x \models \phi$ then $\mathcal{M}, y \models \psi$.

Example 3.2. *The Kripke frame representing the fact that someone cannot be the president of both France and England uses $\mathcal{P}$ and $\mathcal{W}$ from Example 3.1, and defines a ternary relation $\mathcal{T}$ such that $\mathcal{T}(w_1) = \mathcal{T}(w_2) = \{(w_1, w_1), (w_2, w_2)\}$. With this frame, we can prove that the formula $O(t_1, \neg t_2)$ (1) and $O(t_2, \neg t_1)$ (2) are valid in M. Let us sketch the demonstration for case (1). We need to prove that $\mathcal{M}, w_1 \models O(t_1, \neg t_2)$ (1.a), and $\mathcal{M}, w_2 \models O(t_1, \neg t_2)$ (1.b). To prove (1.a), we first consider $(w_1, w_1) \in \mathcal{T}(w_1)$ (1.a.a). $\mathcal{M}, w_1 \models t_1$ and $\mathcal{M}, w_1 \models \neg t_2$. Then we consider $(w_2, w_2) \in \mathcal{T}(w_1)$ (1.a.b) $\mathcal{M}, w_2 \not\models t_1$. Therefore it is true that $\forall (x, y) \in \mathcal{T}(w_1)$, if $\mathcal{M}, x \models t_1$ then $\mathcal{M}, y \models \neg t_2$, which proves (1.a). The rest of the demonstration follows.*

Extending to operators with higher arity may also be interesting in practice, for example to consider exceptions to a context in deontic applications. That is why Blackburn et. al. introduced a general definition for *normal modal logics* with N-ary modal operators [2]. A *normal Kripke model* is a tuple $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ and generalizes the notion of basic Kripke model as follows:

- $\mathcal{W}$ is a non-empty set of possible worlds,
- $\mathcal{R} \subseteq \mathcal{W}^{N+1}$ is a $(N+1)$-ary accessibility relation on $\mathcal{W}$, with $N \geqslant 0$.
- $\mathcal{V} : \mathcal{P} \to 2^{\mathcal{W}}$ is a valuation function.

$(w, v_1, \ldots, v_N) \in \mathcal{R}$ or $w\mathcal{R}(v_1, \ldots, v_N)$ denotes an $(N+1)$-ary edge in $\mathcal{R}$. Some work define $\mathcal{R}$ as a function from $\mathcal{W}$ to $\mathcal{W}^N$, giving the equally common notation $\mathcal{R}(w)$ to indicate the set of $(N+1)$-ary edges having w as the first argument.

The normal modal logic language $\mathcal{L}$ is generated by the following grammar given in Backus-Naur Form:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \nabla(\phi_1, \ldots, \phi_N)$$

Where ∇ is a N-ary modal operator that generalizes $\Box$. Point 5 in the caracterization of $\mathcal{M}$ above is generalized as follows. Given $w \in \mathcal{W}$ and given formulas $\phi_1, \ldots, \phi_N \in \mathcal{L}$:

$$5'.\ \mathcal{M}, w \models \nabla(\phi_1, \ldots, \phi_N) \text{ iff } \forall (v_1, \ldots, v_N) \in \mathcal{W}^N, \text{ if } w\mathcal{R}(v_1, \ldots, v_N) \text{ then } \exists k \in [\![1, N]\!],\ \mathcal{M}, v_k \models \phi_k$$

The dual N-ary modal operator $\triangle$ is defined syntactically by $\nabla(\phi_1, \ldots, \phi_N) := \neg\triangle(\neg\phi_1, \ldots, \neg\phi_N)$. It thus generalizes the operator $\Diamond$ and satisfies the following definition:

$$\mathcal{M}, w \models \triangle(\phi_1, \ldots, \phi_N) \text{ iff } \exists (v_1, \ldots, v_N) \in \mathcal{W}^N \text{ s.t. } w\mathcal{R}(v_1, \ldots, v_N) \text{ and } \forall k \in [\![1, N]\!],\ \mathcal{M}, v_k \models \phi_k$$

Let us notice that, as for basic modal logic, one may define normal modal logic formalisms with more than one modal operator, potentially with different arities, and intertwine their axiomatization [2]. Also, some formalisms define unary relations $\mathcal{R}$ which then correspond to nullary modal operators. The Dynamic Logic of Mental attitudes and joint Actions (DL-MA) [11, 12] is an example where a set of nullary modal operators represent actions. In this paper, we associate the following semantics: $\forall w \in \mathcal{W},\ \mathcal{M}, w \models \triangle_1$ iff $w \in \triangle_1$. This may be used in our example to describe the worlds where we have "One proclaims oneself President of France". Here, $\triangle_1$ denotes both the modal operator and the Kripke relation.

In Section 4 onward, we will specifically consider how to represent normal Kripke models in RDF.

3.2. Related work on modal logic and the semantic web, and positioning

Description Logics (DLs) serve as a foundational framework for representing and reasoning about knowledge, providing the formal basis for the Web Ontology Language (OWL) [13]. While our article introduces a novel perspective, it's worth presenting and positioning existing work that explored the connection between modal logic and DLs, including Modal Description Logics.

Very early on, it has been observed that there is a strong connection between modal logic and DLs when mapping roles to accessibility relations. Using a transformation function f from a $\mathcal{ALC}$ concept to a K-formula such that $f(\forall R.C) = K_R f(C)$, Schild showed that the $\mathcal{ALC}$ Description logic can be considered as notational variant of the multi-modal logic K, and that the satisfiability of an $\mathcal{ALC}$-concept aligns with the satisfiability of the corresponding K-formula [14]. This realization highlights the close relationship between the two formalisms and underlines the significance of roles in capturing notions of accessibility and connectivity within the logical framework.

Baader and Laux introduced modal operators explicitly in the description logic $\mathcal{ALC}_M$ to extend expressiveness and provide a formalism for reasoning about concepts and roles in a more fine-grained manner [9]. This first combination of DLs with modal logics was followed by various modal extensions of DLs, which all provide a richer language for knowledge representation and reasoning. For instance, [8] proposes modalized DLs by modilizing roles. Modalized DLs extend the standard DLs by adding modal operators to the language. The modal operators provide a way to reason about concepts and roles in terms of their modal properties, such as possibility and necessity. Modalized DLs have been applied to the representation of knowledge of agents in multi-agent systems [15, 16], to reason about actions and changes [17], and temporality [18].

The use of hybrid logics is another way of introducing DLs into modal logics [19]. Hybrid logics extend modal logics by allowing for direct reference to individuals in the model, rather than only referring to

them indirectly through modal operators. Nominals in hybrid logic enable assertional ABox reasoning, in addition to standard terminological TBox reasoning, which closely links hybrid logic and description logic. This approach allows for a more direct way to reason about individuals, and has been applied to DLs to provide a more fine-grained way to express and reason about concepts and roles.

Positioning. As a summary, previous existing works focus on Modal DLs and consider in their semantics a Kripke frame. However, we claim that our running example does not need such expressive extension to DLs, and that RDF would be sufficient to represent the Kripke structures we require. Furthermore, some use cases from the families of motivating scenarios listed in Section 1 demand an explicit representation of a Kripke frame. In this article, we are therefore interested in considering a different perspective where we start from RDF and its minimal system [20] to represent Kripke frames, focusing on identifying possible worlds and expressing the relations between them. To the best of our knowledge, this perspective is yet unexplored.

Our intuition is that, in the context of an RDF dataset, a Kripke structure $(\mathcal{W}, \mathcal{R})$, formulas from a modal logic language $\mathcal{L}$, and in which possible worlds such formulas are verified, could be represented all at once. A set of triples $g \in \mathcal{G}$ could be translated to a formula $f(g) \in \mathcal{L}$ using a function f, and one could explicit in what possible worlds a formula $f(g)$ is verified using metadata about this set of triples g. A propositional atom $p \in \mathcal{P}$ could be a set of RDF triples. The valuation function $\mathcal{V}$ of a Kripke model, i.e., the truth of p w.r.t. the possible worlds, would then be represented by RDF metadata that would link sets of triples to possible worlds. All in all, one should therefore be able to represent and exchange a model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$ and formulas from $\mathcal{L}$, and reason about them.

As a first increment in this new direction, in this article we focus on a simple version of f where triples map to propositional atoms. This article thus only provides an RDF representation of Kripke models, but it is sufficient to support many use cases from the families of motivating scenarios listed in Section 1. In the next section, we recall a number of alternatives found in the literature to support RDF statements about RDF statements. Then in Section 4 and 5 we propose to evaluate and select one representation among the possible alternatives, and summarize our choices in Section 6. We open the discussion on the desired characteristics of a general function f in Section 7.1.

3.3. RDF about RDF: approaches for RDF metadata representation

To the best of our knowledge there are typically five ways to represent RDF metadata: the native RDF 1.0 reification syntax, the singleton property non standard extension, the RDF 1.1 named graph standard extension, the historical non standard N3 quoted graphs, and the RDF* quoted graphs currently under consideration for standardization as part of RDF 1.2 [21]. In addition, we also include the N-ary relation design patterns to go beyond the binary model of relations in RDF in particular to qualify a instance of a relation or its value. We review these existing options here as an input to Sections 4 and 5 where we consider means to represent possible worlds and (N+1)-ary accessibility relations in RDF.

RDF 1.0 reification syntax. Reification is an historical standard technique in the Semantic Web that allows to create metadata about a triple by turning it into a resource [21, 22]. One of the pros of using reification is that it can be used in native RDF since its version 1.0. However, one of the disadvantages of this technique is that it turns one triple into four triples: the `rdf:Statement` instance declaration, and three triples linking the statement to its subject, predicate, and object. This can be problematic depending on the storage used. Additionally, there is no direct link between the reified triple and the original triple, as the former is represented as four separate triples. Finally, different individual `rdf:Statement` instances may have the same values for their `rdf:predicate`, `rdf:subject` and `rdf:object` properties, yet RDF doesn't prescribe these instances to be equivalent.

N-ary relation design patterns. The N-ary relation design patterns [23] were introduced in a group note produced by the W3C Semantic Web Best Practices and Deployment Working Group for when we may want to represent properties of a relation beyond the imposed RDF binary relation linking two individuals

or an individual and a value. These patterns can be used when the relation needs to be among multiple individuals or when there is a need to identify and qualify a relation instance (e.g., uncertainty, units, weight, timestamp, etc.).

The first proposed pattern consists in introducing a new class for a relation. This pattern covers three use cases: 1. additional attributes describing a relation, where there may be multiple instances of the same relation linking different individuals, and one participant holds and links to the relation instance; 2. different aspects of the same relation, where different instances of the same relation with the same arguments are equivalent, and are recommended to be represented as blank nodes; 3. N-ary relation with no distinguished participant, where the relation instance only has outgoing links. The second pattern is to use lists for arguments in a relation, defining a dedicated vocabulary for representing linked lists.

Compared to RDF reification, N-ary relations add additional arguments in the relation and do not usually characterize the statement but rather provide additional information about the relation instance itself. These patterns are purely RDF representations, but there are a few reasons why one might choose not to use the N-ary relation patterns. They are essentially the same as for the RDF reification (e.g., increased data volume, more complexity, etc.).

Singleton property non standard extension. The singleton property approach [24] is a non standard technique to represent reified statements. It essentially creates a dedicated sub-property for every triple that needs to be annotated. This method has the advantage of being usable in any version of RDF. However, it requires a dedicated vocabulary and a very specific implementation that is not found in any common or commercial implementation. Also, it generates a high number of unique property types, which causes the schema to grow proportionally to the number of reified triples.

RDF 1.1 named graphs. Since RDF 1.1, named graphs [25, 26] provide an elegant alternative to reification and singleton properties. They provide a way to give a specific name (URI) to an RDF graph, making it easier to modularize, reference, annotate and use sets of triples. Using named graphs can also help to manage different sources of information and in general provide and use metadata for provenance, traceability, etc. Named graphs do not have any standard semantics attached to them which makes them very versatile but also used for very different purposes and with very different ad-hoc semantics [27].

RDF, RDF-star and RDF 1.2.* RDF* provides explicitly integrates reification as part of RDF, providing users with means to write more complex and expressive knowledge graphs. The abstract syntax of RDF* is defined as an extension of that of RDF, by adding quoted triples to express statements about statements, i.e., reification. Thus, RDF* is a technique that offers simplicity in expressing statement-level metadata. At the time of writing this article, RDF* (renamed to RDF-star) [28, 29] is not yet a standard but has entered the W3C process for it and could be part of RDF 1.2. However, it currently requires specific RDF engines to be used, which can limit its applicability and adoption in certain contexts. Furthermore, only one triple can be annotated at a time.

N3, the historical non standard Notation 3. N3 (also known as Notation 3) [30] is a language that inspired the Turtle syntax and builds on top of RDF to extend its capabilities [31]. It allows for the representation of logical statements using a concise and easy-to-read textual syntax. In addition to the RDF data model, N3 includes features such as variables, logical implication, and functional predicates. It also introduces the concept of formulae, which are literals that represent subgraphs of RDF triples. This allows for the creation of nested and more complex structures than what is possible with RDF alone. N3 is designed to be a superset of RDF, which means that any valid RDF document can be interpreted as a valid N3 document, but the converse is not always true. While N3 provides a compact and readable way to represent reified statements, it is not widely adopted nor implemented outside its community.

Comparing alternative approaches for metadata. In addition to the alternatives mentioned in the previous sections, several other proposals were made. For example, the companion properties [32] extends singleton properties with a fixed naming scheme for these. The labeled k-partite graphs [33] extend RDF 1.1 to introduce an alternative graph data model with mapping to RDF and RDF^M [34] that integrates attributes

to the predicates. We do not discuss them because they don't have enough visibility at the moment of writing this article to be considered as usable alternatives for our purpose here. Table 1 lists the alternatives we consider in this article and some of their relevant features.

Table 1
Comparison of RDF metadata representation approaches

Alternative\features	standardized	size impact	large adoption	annotates
RDF Reification	yes	large graphs	yes	1 triple
N-ary patterns	no	large graphs	yes	1 N-ary relation
Singleton property	no	large schema	no	1 triple
Named graphs	yes	small	yes	sets of triples
RDF*	in progress	small	not yet	1 triple
N3	no	small	no	sets of triples

4. Possible Hyperworlds: representing possible worlds as linked data

In this section, we consider and discuss each of the approaches mentioned in the previous section to represent possible worlds from a normal Kripke model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$. We call *possible hyperworlds* these representations of possible worlds as linked data. We use Example 3.1 as a running example, with two possible worlds: w_1 represents {<x> ex:isPresidentOf <France>}, and w_2 represents {<x> ex:isPresidentOf <England>}. These statements are incompatible, and need therefore to be separated in two possible worlds.

As a starting point, we represent possible worlds in $\mathcal{W}$ as resources of type :PossibleWorld. For instance, we identify the possible worlds w_1 (resp. w_2) by <w1> (resp. <w2>), and represent them as follows:

```
<w1> a :PossibleWorld .
<w2> a :PossibleWorld .
```

The class :PossibleWorld is part of the PWKSO ontology. We will introduce other pieces of it as we progress in the analysis conducted this paper. The complete ontology is provided in Section 6 as a conclusion to all the discussions of this paper. Note that although we focus on RDF and approaches for annotating RDF, we do allow ourselves to use OWL constructs and axioms to enrich the PWKSO ontology itself whenever it may be appropriate.

4.1. Alternative representations for possible worlds

Our intuition is that in the context of an RDF dataset, some IRI $i \in \mathcal{I}$ or sets of triples $g \in \mathcal{G}$ could be translated to a formula $\phi \in \mathcal{L}$ using a function f. We will discuss this translation in Section 7.1, but for the sake of simplicity we assume in this article that f maps some statements to propositional atom. For example, $t_1 := f(\{$<x> ex:isPresidentOf <France>$\})$, and $t_2 := f(\{$<x> ex:isPresidentOf <England>$\})$.

We associate statements to each possible world where this statement is true using the property :verifiedIn. With our assumption, and given propositional atoms are atomic formulas, this property provides a direct representation of the valuation function $\mathcal{V}$. In our example, $\mathcal{M}, w_1 \models t_1$ and $\mathcal{M}, w_2 \models t_2$. Therefore the valuation function is such that $w_1 \in \mathcal{V}(t_1)$ and $w_2 \in \mathcal{V}(t_2)$.

To have Example 3.1 completely represented, we also need some sort of negation. In this section we make the closed-world assumption, which implies that $\mathcal{M}, w_1 \not\models t_2$ (resp. $\mathcal{M}, w_2 \not\models t_1$), therefore $\mathcal{M}, w_1 \models \neg t_2$ (resp. $\mathcal{M}, w_2 \models \neg t_1$), and the valuation function is such that $w_1 \notin \mathcal{V}(t_2)$ (resp. $w_2 \notin \mathcal{V}(t_1)$). We will discuss the open vs. closed-world assumption in Section 4.2.

RDF reification for possible worlds. The RDF reification allows us to represent statements, and link them to the possible worlds where they are verified. Property `:verifiedIn` is then used to map a `rdf:Statement` to a `:PossibleWorld`. Using RDF reification, our example is written as follows:

```
## world w1
<t1> a rdf:Statement ;
    rdf:subject <x> ;
    rdf:predicate ex:isPresidentOf :
    rdf:object <France> .

<t1> :verifiedIn <w1> .

## world w2
<t2> a rdf:Statement ;
    rdf:subject <x> ;
    rdf:predicate ex:isPresidentOf ;
    rdf:object <England> .

<t2> :verifiedIn <w2> .
```

Singleton property for possible worlds. In this approach, a singleton property is defined as a unique property that can have only one value per individual, *i.e.*, one per statement we need to represent. Property `:verifiedIn` is then used to map a singleton property to a `:PossibleWorld`. Using singleton properties, our example is written as follows:

```
## world w1
<x> <isPresidentOf#t1> <France> .
<isPresidentOf#t1>  rdf:singletonPropertyOf   ex:isPresidentOf .

<isPresidentOf#t1> :verifiedIn <w1> .

## world w2
<x> <isPresidentOf#t2> <England> .
<isPresidentOf#t2>  rdf:singletonPropertyOf   ex:isPresidentOf .

<isPresidentOf#t2> :verifiedIn <w2> .
```

Named graphs for possible worlds. This approach uses RDF 1.1 named graphs to group statements. Property `:verifiedIn` is then used to map a named graph to a `:PossibleWorld`. Using named graphs, our example is written as follows:

```
## world w1
<t1> { <x> ex:isPresidentOf <France> . }
<t1> :verifiedIn <w1> .

## world w2
<t2> {  <x> ex:isPresidentOf <England> . }
<t2> :verifiedIn <w2> .
```

RDF for possible worlds.* The RDF* specification extends RDF with the ability to attach properties directly to quoted triples. Property `:verifiedIn` is then used to map a quoted triple to a `:PossibleWorld`. Using RDF*, our example is written as follows:

```
<< <x> ex:isPresidentOf <France> >> :verifiedIn <w1> .
<< <x> ex:isPresidentOf <England> >> :verifiedIn <w2> .
```

N3 for possible worlds. With the N3 notation, one can use curly braces to group statements. Property `:verifiedIn` is then used to map a group of statements to a `:PossibleWorld`. Using N3, our example is written as follows:

```
{ <x> ex:isPresidentOf <France> } :verifiedIn <w1> .
{ <x> ex:isPresidentOf <England> } :verifiedIn <w2> .
```

Comparison and choice of an alternative. All the approaches above can successfully represent Example 3.1. Some offer a more natural and concise representation than others, with the RDF reification approach being the least concise, and the singleton property arguably being the least natural. The RDF reification, singleton properties, and named graphs approaches allow to identify the propositions t_1 and t_2, and use this identifier as the subject of `:verifiedIn`. Also, we only dealt with a simple case where the function f transforms statements to propositional atoms. In the general case where the function f transforms sets of triples to formulas, property `:verifiedIn` should map sets of triples to a `:PossibleWorld`. This leaves only the named graphs and the N3 approaches as sufficiently generalizable. Finally, given that N3 is not standardized and not largely adopted, and given it may be useful to identify formulas, we choose to adopt the named graphs approach.

4.2. Closed-world assumption vs open-world assumption

In the previous section, we described explicitly what propositional atom (a triple) a possible world verifies, giving a direct representation of the valuation function $\mathcal{V}$. In our example, $\mathcal{M}, w_1 \models t_1$ and $\mathcal{M}, w_2 \models t_2$, therefore the valuation function is such that $w_1 \in \mathcal{V}(t_1)$ and $w_2 \in \mathcal{V}(t_2)$. However, the definition of the valuation function is not necessarily exhaustive: how to represent that $\mathcal{V}(t_1) = \{w_1\}$ and $\mathcal{V}(t_2) = \{w_2\}$? Exhaustivity comes naturally if we adopt the closed-world assumption: a principle in knowledge representation and reasoning where any statement or proposition that is not known to be true is considered false. In other words, in a closed-world assumption, the absence of evidence for a fact is treated as evidence against that fact. This means that the knowledge base is assumed to be complete, and any information that is not explicitly stated is considered false. The closed-world assumption is commonly used in database systems and some logic-based reasoning systems.

With the open-world assumption, any statement or proposition that is not known to be true is considered unknown or uncertain, rather than being considered false. This means that the knowledge base is assumed to be incomplete, and any information that is not explicitly stated is not assumed to be false but rather left open or unknown. The open-world assumption is commonly used in many logical and knowledge representation systems, including semantic web technologies and most real-world reasoning scenarios.

As an alternative to exhaustively describing the set of worlds where a propositional atom is verified, one may want to explicitly represent that $w_2 \notin \mathcal{V}(t_1)$ and $w_1 \notin \mathcal{V}(t_2)$. This is possible using property `:notVerifiedIn`, that associates statements to possible worlds where this statement is false. Using this property and the named graph approach, the complete Example 3.1 is written as follows:

```
## propositional atoms
<t1> { <x> ex:isPresidentOf <France> .  }
<t2> { <x> ex:isPresidentOf <England> . }

## world w1
<t1> :verifiedIn    <w1> .
<t2> :notVerifiedIn <w1> .

```

```
## world w2
<t1> :notVerifiedIn <w2> .
<t2> :verifiedIn    <w2> .
```

Using the open-world assumption, the truth value of a formula $f(g)$ where the set of triples g is not explicitly asserted to be verified or not verified in a world w cannot be determined solely based on the representation. It must then be determined using additional information, context, or inference mechanisms.

In the PWKSO ontology, an instance of `:PossibleWorld` can be explicitly typed with `:OpenPossibleWorld` or `:ClosedPossibleWorld` to express that one or the other assumption is expected to be used.

Section 7.1 discusses other ways of representing negation using the open-world assumption.

5. Representing accessibility relations with N-ary patterns

In this section we move from representing individual possible worlds to representing their Kripke relations. We will discuss the different ways to do so, illustrating the alternatives with our running example from Section 2 and the example Kripke models from Section 3.1. Section 5.1 focuses on binary relations, Section 5.2 on (N+1)-ary relations using lists for arguments, and Section 5.3 on (N+1)-ary relations using a new class for the relation instance.

5.1. Representing binary accessibility relations between worlds

This section uses Example 3.1 where Alice cannot distinguish between the two possible worlds w_1 and w_2, and Bob knows that `{<x> ex:isPresidentOf <France>}`.

Our objective is to represent the epistemic multi-modal Kripke frame $(\mathcal{W}, \{\mathcal{K}_{\text{Alice}}, \mathcal{K}_{\text{Bob}}\})$ where $\mathcal{W} = \{w_1, w_2\}$ is the set of possible worlds, $\mathcal{K}_{\text{Alice}}$ is such that $\mathcal{K}_{\text{Alice}}(w_1) = \mathcal{K}_{\text{Alice}}(w_2) = \{w_1, w_2\}$, and $\mathcal{K}_{\text{Bob}}$ is such that $\mathcal{K}_{\text{Bob}}(w_1) = \{w_1\}$, $\mathcal{K}_{\text{Bob}}(w_2) = \{\}$.

The most straightforward option to represent a binary relation between possible worlds is to define a dedicated binary property in RDF. For instance a property `:hasAccessTo` could be defined as follows:

```
:hasAccessTo a owl:ObjectProperty  ;
    rdfs:domain :PossibleWorld  ;
    rdfs:range :PossibleWorld .
```

This property `:hasAccessTo` could then be used to represent relation $\mathcal{K}_{\text{Alice}}$:

```
<w1> :hasAccessTo <w1>, <w2> .
<w2> :hasAccessTo <w1>, <w2> .
```

This option offers a simple way to explicit frame conditions that hold on the relation using OWL, such as when it is reflexive (axiom **T**), reflexive and symmetric (axiom **B**), or reflexive and transitive (axiom **S4**). The Euclidian condition (necessary for axiom **S5**) could be modelled using a sub-property chain axiom. This way, one would be able to reason with accessibility relations using a OWL reasoner, provided the ontology has an expressivity the reasoner can manage. For example, property `:indistinguishableFrom` models the indistinguishable relation $\mathcal{K}$ in epistemic logic, which is an equivalence relation. It could be modeled as follows:

```
:indistinguishableFrom a owl:ObjectProperty ;
    a owl:ReflexiveProperty , owl:SymmetricProperty , owl:TransitiveProperty ;
    rdfs:subPropertyOf :hasAccessTo .
```

On the other hand, this representation does not allow us to separate the indistinguishable relations $\mathcal{K}_{\text{Alice}}$ and $\mathcal{K}_{\text{Bob}}$ of Alice and Bob in the case of epistemic multi-modal logic. We would then need to introduce one property per agent as for instance in:

```
## Agent dedicated properties
<aliceCannotDistinguish> a owl:ObjectProperty ;
    rdfs:subPropertyOf :indistinguishableFrom .

<bobCannotDistinguish> a owl:ObjectProperty ;
    rdfs:subPropertyOf :indistinguishableFrom .

## Agent views
<w1> <aliceCannotDistinguish> <w2>, <w1> .
<w2> <aliceCannotDistinguish> <w1>, <w2> .

<w1> <bobCannotDistinguish> <w1> .
```

Alternatively, we could annotate statements using one of the approaches presented in Section 3.3, to indicate according to which agent the worlds are indistinguishable. For instance using RDF-star:

```
<< <w1> :indistinguishableFrom <w1> >> :associatedToAgent <alice> .
<< <w1> :indistinguishableFrom <w2> >> :associatedToAgent <alice> .
<< <w2> :indistinguishableFrom <w1> >> :associatedToAgent <alice> .
<< <w2> :indistinguishableFrom <w2> >> :associatedToAgent <alice> .
<< <w1> :indistinguishableFrom <w1> >> :associatedToAgent <bob> .
```

Or using named graphs:

```
<#ngAlice> { <w1> :indistinguishableFrom <w2>, <w1> .
             <w2> :indistinguishableFrom <w1>, <w2> . }
<#ngAlice> :associatedToAgent <alice> .
<#ngBob>   { <w1> :indistinguishableFrom <w1> . }
<#ngBob> :associatedToAgent <bob> .
```

This need to qualify more precisely a Kripke relations may arise in many other situations. Although the binary relation approach is simple at first, it can become clumsy as the Kripke relations become more complex. As a result, we will prefer a more general N-ary relation pattern to capture all the arguments and indices that may be involved in asserting a Kripke relation between possible worlds. In the rest of this article, and in the ontology and design pattern we present in Section 6, we will not support the case of direct binary relations.

5.2. Representing (N+1)-ary accessibility relations using lists for arguments

As an alternative to using properties to model the binary accessibility relations, one may use the design pattern for representing N-ary relations using lists for arguments (See Section 3.3). In such representation, a set of accessible worlds could be represented as a special list:

```
:PossibleWorldList rdfs:subClassOf
    [ a owl:Restriction ;
      owl:onProperty rdf:first ;
      owl:cardinality 1 ] ,
    [ a owl:Restriction ;
      owl:onProperty rdf:first ;
      owl:allValuesFrom :PossibleWorld ] ,
    [ a owl:Restriction ;
      owl:onProperty rdf:rest ;
      owl:allValuesFrom :PossibleWorldList  ] .
```

The entry-point property could be defined as follows.

```
:hasRelationTo a owl:ObjectProperty  ;
    rdfs:domain :PossibleWorld  ;
    rdfs:range :PossibleWorldList .
```

When describing an edge $w\mathcal{R}(v_1, \ldots, v_N)$ with this approach we would follow the following pattern.

```
<w> :hasRelationTo (<v1> ... <vN>) .
```

For example to represent the dyadic deontic Kripke structure $(\mathcal{W}, \mathcal{T})$ from Example 3.2, one would write:

```
:hasDyadicDeonticRelationTo a owl:ObjectProperty ;
    rdfs:subPropertyOf :hasRelationTo .

<w1> :hasDyadicDeonticRelationTo (<w1> <w1>) , (<w2> <w2>) .
<w2> :hasDyadicDeonticRelationTo (<w1> <w1>) , (<w2> <w2>) .
```

However, this notation is not natural for binary relations, as for example representing the Kripke frame $(\mathcal{W}, \{\mathcal{K}_{\text{Alice}}, \mathcal{K}_{\text{Bob}}\})$ from Section 5.1 would write:

```
<aliceCannotDistinguish> a owl:ObjectProperty ;
    rdfs:subPropertyOf :hasRelationTo .

<w1> <aliceCannotDistinguish> (<w1>) , (<w2>) .
<w2> <aliceCannotDistinguish> (<w1>) , (<w2>) .
<w1> <bobCannotDistinguish>   (<w1>) .
```

Also, our definition of `:PossibleWorldList` employs `rdf:first` and `rdf:rest` to allow for concise list notations in Turtle, and uses them to identify object properties in OWL. This use of IRIs in the reserved vocabulary violates a condition on object properties from the OWL 2 DL profile specification.

Finally, another disadvantage of this modeling choice is that parameters inside the list cannot be named, although it may sometimes be desirable. For example in the case of the dyadic deontic relation from Example 3.2, it is meaningful to refer to the first parameter in the list with a dedicated property `:hasContext` (the condition), and to the second parameter with a dedicated property `:itOughtToBe` (the ideal world).

To overcome these drawbacks we propose in the next section to follow the second pattern for representing N-ary relations using a new class for relation instances. This next pattern will be our recommended modeling choice for representing (N+1)-ary accessibility relations.

5.3. Representing (N+1)-ary accessibility relations using a new class for relation instances

We propose here a general pattern to represent (N+1)-ary Kripke relations using the pattern "Introducing a new class for a relation" introduced in the W3C Working Group Note [23]. This pattern has three variants covering different use cases: 1. additional attributes describing a relation, where there may be multiple instances of the same relation linking different individuals, and one participant holds and links to the relation instance; 2. different aspects of the same relation, where different instances of the same relation with the same arguments are equivalent, and are recommended to be represented as blank nodes; 3. N-ary relation with no distinguished participant, where the relation instance only has outgoing links.

As we represent mathematical objects, we only consider variants 2 and 3. Variant 3 corresponds to N-ary relationships that link individuals that play different roles without any single individual standing out as the subject. Representing an (N+1)-ary edge $(w, v_1, \ldots, v_N) \in \mathcal{R}$ with this pattern would write:

```
[] a :KripkeRelation ; :param1 <w> ; :param2 <v1> ; ... ; :paramNPlus1 <vN> .
```

However, in [2], an accessible N-uplet is defined as *from* a possible world w, leading to the more common notations $w\mathcal{R}(v_1, \ldots, v_n)$. Some work define it as a function from $\mathcal{W}$ to $\mathcal{W}^N$, giving the equally common notation $(v_1, \ldots, v_n) \in \mathcal{R}(w)$. The relation is thus directed by nature: on one side we have the possible world w, and on the other side the set of accessible N-uplets of possible worlds. As a result, we adopt Variant 2, supporting a distinguished participant to preserve the direction. We introduce property `:hasRelation` to link the subject possible world to the N-ary relation instance:

```
:KripkeRelation a owl:Class .

:hasRelation a owl:ObjectProperty ;
    rdfs:domain :PossibleWorld ;
    rdfs:range :KripkeRelation .

## Pattern
<w> :hasRelation [ a :KripkeRelation ;
                   :param1 <v1> ;
                   ## ...
                   :paramN <vN> ] .
```

This patterns allows us to represent all accessibility relations simply and uniformly, disregarding their arity. One downside with respect to modeling binary accessibility relations using a properties as in Section 5.1, is that here we cannot use OWL axioms to represent standard frame conditions such as reflexivity (axiom **T**), symmetry (necessary for axiom **B**), or transitivity (necessary for axiom **S5**) of binary relations. However, class `:KripkeRelation` can be specialized and its sub-classes could be attached specific entailment rules or constraints to model these frame conditions. An example is given in Section 7.2.

6. Summary and resulting ontology

In this section we draw together all the previous analysis we made on the RDF representation of possible worlds and Kripke structures and compile our design rationale in a corresponding ontology. We name this ontology *Possible Worlds and Kripke Structures Ontology* (PWKSO).

Section 6.1 overviews the PWKSO ontology, Section 6.2 provides a generic usage pattern and usage recommendations, and Section 6.3 applies this pattern to illustrate how the different example Kripke models from Section 3.1 can be represented.

6.1. The PWKSO ontology for representing possible (hyper)worlds and Kripke structures

Given a normal Kripke model $\mathcal{M} = (\mathcal{W}, \mathcal{R}, \mathcal{V})$, PWKSO allows us to represent possible worlds in $\mathcal{W}$ (`:PossibleWorld`) and assert that triples are verified (`:verifiedIn`) or not verified (`:notVerifiedIn`) in these worlds using different approaches for RDF metadata representation. We motivated in Section 4.1 the adoption of the named graph approach as it is standardized, largely adopted, and it annotates any number of triples. In PWKSO, possible hyperworlds can be declared as using the closed-world assumption (`:ClosedPossibleWorld`) or the open-world assumption (`:OpenPossibleWorld`) as suggested in Section 4.2.

As mentioned before, to support the general case of (N+1)-ary accessibility relations $\mathcal{R}$, PWKSO adopts the pattern for representing N-ary relations with a dedicated top class for Kripke relations (`:KripkeRelation`), and distinguished top properties to identify the parameters (`:param`) and indices (`:index`) of the relations. Following the W3C design pattern[1], it is recommended to use blank nodes to identify instances of a relation, and the instances of the same relation with the same arguments are regarded as equivalent.

[1] W3C pattern https://www.w3.org/TR/swbp-n-aryRelations/#anonvnamed

We assume a function f from IRIs and sets of triples to modal logic formulas in $\mathcal{L}$, and as a first increment we only consider in this article that f maps some statements to propositional atoms, providing a direct representation of the valuation function $\mathcal{V}$. PWKSO does not include specific constructs to explicitly express modal logic formulas in $\mathcal{L}$, but the named graph approach allows for future extensions as is discussed in Section 7.1.

The resulting PWKSO is a lightweight OWL 2 DL ontology that groups the terminology introduced in Sections 4 and for the (N+1)-ary pattern defined in Section 5.3. It contains 6 classes and 5 object properties. The HTML documentation and the Turtle code of the PWKSO ontology is published under license CC-BY 4.0 at its IRI http://ns.inria.fr/pwkso/ using content negotiation. The preferred prefix is `pwsko:` and is registered at prefix.cc.

```
@base <http://ns.inria.fr/pwkso/> .
@prefix : <> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix cc: <http://creativecommons.org/ns#> .
@prefix vann: <http://purl.org/vocab/vann/> .
@prefix dct: <http://purl.org/dc/terms/> .

: a owl:Ontology ;
    dct:title "Possible Worlds and Kripke Structures Ontology"@en ;
    dct:description  "an ontology for representing possible worlds and Kripke structures."@en ;
    cc:license <https://creativecommons.org/licenses/by/4.0/> ;
    vann:preferredNamespacePrefix "pwkso" ;
    vann:preferredNamespaceUri : .

:PossibleWorld a owl:Class ;
    rdfs:label "possible world"@en;
    rdfs:comment "a consistent representation of how the world is, could have been or would be"@en .

:verifiedIn a owl:ObjectProperty ;
    rdfs:label "verified in"@en;
    rdfs:comment "links triples to a possible world in which their associated formula is verified"@en;
    rdfs:range :PossibleWorld ;
    owl:propertyDisjointWith :notVerifiedIn .

:notVerifiedIn a owl:ObjectProperty ;
    rdfs:label "not verified in"@en;
    rdfs:comment "links triples to a possible world in which associated formula is not verified"@en;
    rdfs:range :PossibleWorld ;
    owl:propertyDisjointWith :verifiedIn .

## Closed vs open possible worlds

:ClosedPossibleWorld a owl:Class ;
    rdfs:subClassOf :PossibleWorld ;
    rdfs:label "closed possible world"@en;
    rdfs:comment "represents a possible world with the Closed-World Assumption"@en .

:OpenPossibleWorld a owl:Class ;
    rdfs:subClassOf :PossibleWorld ;
    rdfs:label "open possible world"@en;
    rdfs:comment "represents a possible world with the Open-World Assumption"@en .


## Representation using the (N+1)-ary pattern: Introducing a new class for a relation with a
↪  distinguished participant

```

```
:KripkeRelation a owl:Class ;
    rdfs:label "Kripke relation"@en;
    rdfs:comment "represents the Kripke relations between possible worlds. Instances of :KripkeRelation are recommended to be represented as blank nodes"@en .

:param a owl:ObjectProperty ;
    rdfs:label "parameter"@en;
    rdfs:comment "parameter of a Kripke relation"@en;
    rdfs:domain :KripkeRelation ;
    rdfs:range :PossibleWorld .

:index a owl:ObjectProperty ;
    rdfs:label "index"@en;
    rdfs:comment "index of a Kripke relation"@en;
    rdfs:domain :KripkeRelation .

:hasRelation a owl:ObjectProperty ;
    rdfs:label "has relation"@en;
    rdfs:comment "links a possible world to one of its N-ary relations"@en;
    rdfs:domain :PossibleWorld ;
    rdfs:range :KripkeRelation .

:BinaryKripkeRelation
    rdfs:label "binary Kripke relation"@en;
    rdfs:comment "relation that links a possible world with another to represent a unary modal operator"@en;
    rdfs:subClassOf :KripkeRelation,
     [ a owl:Restriction ;
       owl:cardinality 1 ;
       owl:onProperty :param ] .

:UnaryKripkeRelation
    rdfs:label "unary Kripke relation"@en;
    rdfs:comment "relation that links a possible world with another to represent a nullary modal operator"@en;
    rdfs:subClassOf :KripkeRelation,
     [ a owl:Restriction ;
       owl:cardinality 0 ;
       owl:onProperty :param ] .
```

6.2. Generic usage pattern for the PWKSO ontology

To facilitate the use of the ontology we provide a generic usage pattern below. This pattern takes the example of applying and extending PWKSO to define operators and use them to instantiate possible worlds:

```
## Generic usage pattern
@base <http://example.org/> .
@prefix : <> .
@prefix pwkso: <http://ns.inria.fr/pwkso/>
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:MyRelation rdfs:subClassOf pwkso:KripkeRelation .

:param1 rdfs:subPropertyOf pwkso:param ;
    rdfs:domain :MyRelation .
## ...
:paramN rdfs:subPropertyOf pwkso:param ;
```

```
    rdfs:domain :MyRelation .

:index1 rdfs:subPropertyOf pwkso:index ;
    rdfs:domain :MyRelation .
## ...
:indexM rdfs:subPropertyOf pwkso:index ;
    rdfs:domain :MyRelation .

:a_possible_world a :PossibleWorld ;
    pwkso:hasRelation [
        a :MyRelation ;
        pwkso:param1 :arg1 ;
        ## ...
        pwkso:paramN :argN ;
        pwkso:index1 :value1 ;
        ## ...
        pwkso:indexM :valueM .
        ] .
```

6.3. Complete examples

In this section, we illustrate the chosen pattern on the binary, ternary, and unary accessibility relations from the example Kripke models of Sections 3.1 and 5.1.

Representing an indistinguishable relation (epistemic case). The PWKSO generic usage pattern offers a simple way for adding information about binary Kripke relations. For instance, if we want to specify the agent associated with the relation, we can simply use the index property `pwkso:index` as in:

```
:IndividualKnowledge a owl:Class ;
    rdfs:label "Individual Knowledge"@en ;
    rdfs:comment "The equivalence Kripke relation of epistemic logic"@en ;
    rdfs:subClassOf pwkso:BinaryKripkeRelation ,
     [ a owl:Restriction ;
       owl:cardinality 1 ;
       owl:onProperty pwkso:index ] ,
     [ a owl:Restriction ;
       owl:allValuesFrom :CognitiveAgent ;
       owl:onProperty pwkso:index ] .

:CognitiveAgent a owl:Class ;
    rdfs:label "Cognitive Agent"@en ;
    rdfs:comment "The class of cognitive agents"@en .
```

The epistemic multi-modal Kripke frame $(\mathcal{W}, \{\mathcal{K}_{\text{Alice}}, \mathcal{K}_{\text{Bob}}\})$ of Section 5.1 can then be represented as follows:

```
<w1> pwkso:hasRelation [ a :IndividualKnowledge ;
                         pwkso:param <w2> ;
                         pwkso:index <alice> ] ,
                       [ a :IndividualKnowledge ;
                         pwkso:param <w1> ;
                         pwkso:index <alice> ] .

<w2> pwkso:hasRelation [ a :IndividualKnowledge ;
                         pwkso:param <w2> ;
                         pwkso:index <alice> ] ,
                       [ a :IndividualKnowledge ;
                         pwkso:param <w1> ;
```

```
                          pwkso:index <alice> ] .

<w1> pwkso:hasRelation [ a :IndividualKnowledge ;
                         pwkso:param <w1> ;
                         pwkso:index <bob> ] .
```

Representing a ternary accessibility relation (deontic case). The dyadic deontic operator, e.g., $O(\psi, \phi)$ expresses that "It ought to be that ϕ is verified when ψ". The corresponding Kripke relation is a ternary relation that maps a possible world to a couple of possible worlds, where the first represents the context that should be satisfied, and the second represents what should ideally be verified. We model the relation with class `:DyadicObligation` which represents an obligation. It is defined by a context capturing when the obligation is applicable (parameter property `:fromContext`) and by what the obligation intends to enforce (parameter property `:itOughtToBe`):

```
:DyadicObligation a owl:Class ;
    rdfs:label "Dyadic Obligation"@en ;
    rdfs:comment "The dyadic obligation Kripke relation"@en ;
    rdfs:subClassOf pwkso:KripkeRelation ,
     [ a owl:Restriction ;
       owl:cardinality 1 ;
       owl:onProperty :fromContext  ] ,
     [ a owl:Restriction ;
       owl:cardinality 1 ;
       owl:onProperty :itOughtToBe ] .

:fromContext a owl:ObjectProperty ;
    rdfs:label "from context"@en ;
    rdfs:comment "links a dyadic obligation to the context world"@en ;
    rdfs:subPropertyOf pwkso:param ;
    rdfs:domain :DyadicObligation ;
    rdfs:range pwkso:PossibleWorld .

:itOughtToBe a owl:ObjectProperty ;
    rdfs:label "it ought to be"@en ;
    rdfs:comment "links a dyadic obligation to the ideal world"@en ;
    rdfs:subPropertyOf pwkso:param ;
    rdfs:domain :DyadicObligation ;
    rdfs:range pwkso:PossibleWorld  .
```

The dyadic deontic Kripke structure $(\mathcal{W}, \mathcal{T})$ from Example 3.2 can then be represented as follows:

```
<w1> pwkso:hasRelation [ a :DyadicObligation ;
                         :fromContext <w1> ;
                         :itOughtToBe <w1> ] ,
                       [ a :DyadicObligation ;
                         :fromContext <w2> ;
                         :itOughtToBe <w2> ].
<w2> pwkso:hasRelation [ a :DyadicObligation ;
                         :fromContext <w1> ;
                         :itOughtToBe <w1> ] ,
                       [ a :DyadicObligation ;
                         :fromContext <w2> ;
                         :itOughtToBe <w2> ] .
```

As a second example of a dyadic deontic Kripke structure $(\mathcal{W}', \mathcal{T}')$, let us assume that in a possible world $w_{\text{Norm}} \in \mathcal{W}'$, there is a norm $n :=$"Alice must answer to Bob when she receives a request from Bob". The context world $w_{\text{Context}} \in \mathcal{W}'$ and the ideal world $w_{\text{Ideal}} \in \mathcal{W}'$ may be represented as follows:

```
## world wContext
<p> { <alice> ex:receivedARequestFrom <bob> . }
<p> pwkso:verifiedIn <wContext> .

## world wIdeal
<q> { <alice> ex:answeredTo <bob> . }
<q> pwkso:verifiedIn <wIdeal> .
```

Then, to express that the norm n holds in w_{Norm}, we represent $(w_{\text{Context}}, w_{\text{Ideal}}) \in \mathcal{T}(w_{\text{Norm}})$ as follows:

```
<wNorm> pwkso:hasRelation [ a :DyadicObligation ;
                            :fromContext <wContext> ;
                            :itOughtToBe <wIdeal> ].
```

This norm is applicable if the following holds:

```
<p> pwkso:verifiedIn <wNorm> .
```

And the norm will be fulfilled if in addition:

```
<q> pwkso:verifiedIn <wNorm> .
```

Representing a unary relation for nullary modal operators. The action formalism DL-MA presented in Section 3.1 describes actions that have been done in a possible world using nullary modal operators. Their associated Kripke relations is unary. For example, one can represent that the Kripke relation $\triangle_1$ ("One proclaims oneself President of France") is true in world w as follows:

```
:ProclaimsPresidentOfFrance a owl:Class ;
    rdfs:subClassOf pwkso:UnaryKripkeRelation .

<w> pwkso:hasRelation [ a :ProclaimsPresidentOfFrance ] .
```

7. Discussion and possible extensions

This section aims to discuss different important open questions related to possible hyperworlds. We first discuss the function f between a set of triples and a formula in $\mathcal{L}$. Then, we give examples of how to query, instantiate, and reason with possible hyperworlds using SPARQL request. Next, we notice that the framework enables the definition of nested possible worlds and nested kripke structure in a possible world. Thus, we discuss the meaning of defining possible worlds contained in a possible world and the link with propositional dynamic logics. Finally, we discuss how existing modal logic formalisms could be leveraged for example to merge knowledge graphs, represent the knowledge or beliefs of agents, represent the effects of actions or norms, or develop model checking applications.

7.1. Discussing the function f

Our intuition is that in the context of an RDF dataset D, some IRI $i \in \mathcal{I}$ or sets of triples $g \in \mathcal{G}$ could be translated to a formula $\phi \in \mathcal{L}$ using a function $f \subseteq \mathcal{I} \cup \mathcal{G} \times \mathcal{L}$. We do not want to prescribe one unique way to define the function f, and, in this section, we only open a discussion of some of its desired characteristics. As a reminder, the normal modal logic language $\mathcal{L}$ is generated by the following grammar given in Backus-Naur Form:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi \wedge \psi \mid \nabla(\phi_1, \ldots, \phi_N)$$

We adopt the usual notations $\mathcal{T}$ and $\mathcal{G} = 2^{\mathcal{T}}$ for the set of RDF triples and the set of RDF graphs, respectively. An RDF dataset is noted $D = (g, \{n, g_n\}_{n \in I})$ where $g \in \mathcal{G}$ is the default graph, I is a set of IRIs, and for all $n \in I$, $g_n \in \mathcal{G}$ is the graph named n. At least, f must be such that: $\forall n \in I, f(n) = f(g_n)$.

Representation of $\bot$ (or $\top$). One could explicitly represent $\bot$ (or $\top$) using a dedicated vocabulary. For example:

```
:top :verifiedIn <w> .
:bottom :verifiedIn <w> .
```

We see immediately that some reasoner should treat such situations as violating the definition of Kripke models. In fact, by definition $\bot$ is verified in no world. One could also choose to define f such that no set of triples translates to $\bot$. That is, $\forall g \in \mathcal{G}, f(g) \neq \bot$. Thus, using `:bottom` as the name of a graph would immediately violate this rule.

Representation of propositional atoms. Our simplification in Section 4.1 consists in considering that f maps some statements to propositional atoms. Although this seems like a reasonable assumption, not all statements should be mapped to propositional atoms. For example, triple `{<t1> :verifiedIn <w> .}` describes that a formula $f($`<t1>`$)$ is valid in a possible world, and should not be mapped to a propositional atom itself. Doing so would result in a strange loop, where the representations of the semantics and of the language are mixed. To deactivate such cases, such triples could be left out of the domain of f. One could also define some sort of punning where this triple is considered to describe the semantics in some context, and to represent a propositional atom in other contexts.

We can imagine at least two other ways to represent propositional atoms. First, propositional atoms could be resources explicitly typed with a class such as `:Atom`. For example propositional atom p could be defined as $f($`<p>`$) = f($`{<p> a :Atom}`$) = p$. Second, propositional atoms could require more than one triple in their representation. As an example, the code below is an alternative representation of propositional atom t_1:

```
[] a :Presidency ;
   :agent <x> ;
   :country <France> .
```

Representation of negation. Assuming the closed-world assumption, or using property `:notVerifiedIn`, are only indirect ways to represent negation in our formalism. There are different alternatives to represent negation explicitly.

For instance, one can define a specific property, such as `ex:isNotPresidentOf`, and interpret `{<x> ex:isNotPresidentOf <France>}` as the negation of `{<x> ex:isPresidentOf <France>}`. This approach allows for more explicit and fine-grained representation of negated statements, but it assumes some shared understanding on the fact that `ex:isNotPresidentOf` is the negation of `ex:isPresidentOf`. A dedicated vocabulary could be proposed to explicit this, such as `{ex:isNotPresidentOf :isNegationOf ex:isPresidentOf}`.

The OWL negative object property assertions provide an alternative way of modeling that `<x>` is not connected by the property `ex:isPresidentOf` to `<France>`. This would be written as follows:

```
[] a owl:NegativePropertyAssertion ;
   owl:sourceIndividual <x> ;
   owl:assertionProperty ex:isPresidentOf ;
   owl:targetIndividual <France> .
```

Another alternative uses as a starting point that $\neg\phi$ is a construct where formula ϕ itself may be the translation of an IRI n, potentially being the name of a graph g_n. The default graph in the dataset below could represent $\neg t_2$:

```
<t2> { <x> ex:isPresidentOf <England> . }
[] a :Negation ;
   :isNegationOf <t2> .
```

Representation of conjunction (and disjunction). As a simple extension of our example in Section 4.1, consider the following representation where named graph `<t1_and_t2>` contains the two triples that represent t_1 and t_2:

```
## world w3
<t1_and_t2> {
    <x> ex:isPresidentOf <France> .
    <x> ex:isPresidentOf <England> .
}
<t1_and_t2> :notVerifiedIn <w1> .
```

It seems natural that function f maps `<t1_and_t2>` to $t_1 \wedge t_2$, and that the above dataset represents $\mathcal{M}, w_1 \not\models t_1 \wedge t_2$. More generally, for any two sets of triples g_1 and g_2, the function f may be defined such that $f(g_1 \oplus g_2) = f(g_1) \wedge f(g_2)$, where $g_1 \oplus g_2$ is the merge[2] of g_1 and g_2.

Another alternative is to consider that $\phi_1 \wedge \phi_2$ is a construct where formula ϕ_1 and ϕ_2 themselves may be the translation of IRIs i_1 and i_2, potentially being the names of graphs g_1 and g_2. Then, the default graph in the dataset below could represent $t_1 \wedge t_2$:

```
## world w1
<t1> { <x> ex:isPresidentOf <France> . }
<t2> { <x> ex:isPresidentOf <England> . }
[] a :Conjunction ;
   :member <t1> , <t2> .
```

Some similar representation could be defined for disjunction using class `:Disjunction`, offering some syntactic sugar.

Translation of the empty graph. Consider the following RDF representation involving an empty graph $g_0 = \{\} \in \mathcal{G}$, whose corresponding formula $f(g_0)$ is valid in world w, i.e., $\mathcal{M}, w \models f(g_0)$:

```
<g0> { }
<g0> :verifiedIn <w> .
```

If we assume that $\forall g_1, g_2 \in \mathcal{G}, f(g_1 \oplus g_2) = f(g_1) \wedge f(g_2)$, then the empty graph represents an empty conjunction, and should therefore always be translated to $\top$ by postulate. $f(g_0) = \top$.

Representation of the modal operator. Depending on the arity of the modal operator, there may be different ways to represent formula $\nabla(\phi_1, \ldots, \phi_N)$ in a natural manner. We will only present one option, that uses as a starting point the idea that for every k, ϕ_k is a translation of an IRI i_k, potentially being the name of a graph g_k. The default graph in the dataset below could represent $O(t_1, \neg t_2)$:

```
<t1>     { <x> ex:isPresidentOf <France> . }
<t2>     { <x> ex:isPresidentOf <England> . }
<not_t2> { [] a :Negation ; :isNegationOf <t2> . }

[] a :OughtTo ;
```

[2]Merging RDF graphs takes care of the fact that blank nodes are considered local to a named graph, and that their identifier may need to be changed in the merge.

```
    :firstArgument <t1> ;
    :secondArgument <not_t2> .
```

and the dataset below could represent that $\mathcal{M}, w_1 \models O(t_1, \neg t_2)$

```
<t1>     { <x> ex:isPresidentOf <France> . }
<t2>     { <x> ex:isPresidentOf <England> . }
<not_t2> { [] a :Negation ; :isNegationOf <t2> . }
<ot1nott2> {
    [] a :OughtTo ;
       :firstArgument <t1> ;
       :secondArgument <not_t2> .
}

<ot1nott2> :verifiedIn <w1> .
```

Conclusion of the discussion about f. In this section, we only scratched the surface of the representation of formulas from the normal modal logic language. We consider that this task goes beyond the scope of this article and requires some sort of consensus in the Modal Logic and the Semantic Web communities. Therefore, the identifiers introduced in this section (`:Negation`, `:Conjunction`, etc.) just served as examples, and are not part of the PWKSO ontology. It is important to note that representing formulas in RDF can be context-specific and dependent on the particular requirements of the application or domain. The choice of a representation method should be guided by the desired semantics, compatibility with existing tools and systems, and the specific reasoning capabilities needed for the given use case. Additionally, one could employ some query language (e.g., SPARQL), rule languages, and inference mechanisms to support some reasoning about consequences with these syntactic representation. Finally, one may wonder how one may interpret OWL constructs and axioms mixed with such representations. Thus, this section just opened the door for more research.

7.2. Querying, instantiating, and reasoning with Kripke models

A potential application and extension of the proposed framework is the ability to query and instantiate Kripke models using SPARQL. This section briefly discusses querying, instantiating, and reasoning with Kripke models expressed using PWKSO.

Querying Kripke models. SPARQL query can be used to extract information from RDF data and SPARQL update can be used to instantiate Kripke models. For example, the SPARQL query below extracts all the pairs of indistinguishable worlds according to Alice, and optionally the associated named graphs verified in these worlds.

```
SELECT ?w1 ?w2 ?g WHERE {
    ?w1 pwkso:hasRelation [
        a :IndividualKnowledge ;
        pwkso:param ?w2 ;
        pwkso:index <alice> ] }
    OPTIONAL { ?g pwkso:verifiedIn ?w2  }
}
```

Round-tripping with existing binary representation Although we promote one pattern for representing accessibility relations using (N+1)-ary relations, it is possible to round-trip with an alternative legacy representation. The query below translates from the PWKSO n-ary pattern to the binary pattern introduced in the last example of Section 5.1, where a binary indistinguishability relation is represented using a property, and statements are annotated using named graphs.

```
INSERT {
        GRAPH ?g { ?w1 :indistinguishableFrom ?w2 }
        ?g :associatedToAgent <alice> }
WHERE {
        ?w1 pwkso:hasRelation [
             a :IndividualKnowledge ;
             pwsko:param ?w2 .
             pwkso:index <alice> ] .  }
```

The query below is the reciprocal, ensuring the possibility to roundtrip between the two representations.

```
INSERT { ?w1 pwkso:hasRelation [
             a :IndividualKnowledge ;
             pwsko:param ?w2 .
             pwsko:index <alice> ] . }
WHERE {
         GRAPH ?g { ?w1 :indistinguishableFrom ?w2 }
         ?g :associatedToAgent <alice>  }
```

By instantiating such Kripke structure using SPARQL queries, an agent both insure interoperability and use legacy tools based on binary representations to reason about the relationships between different possible worlds and use them to make inferences and draw conclusions.

Reasoning with PWKSO and SPARQL SPARQL queries may also be used to model the standard frame conditions, in particular for binary accessibility relations. For example the query below models symmetry (necessary for axiom **B**):[3]

```
INSERT {
    ?w2 pwkso:hasRelation ?bn2 .
    ?bn2 a ?class ;
         ?param ?w1 ;
         ?index ?value .
}
WHERE {
    ?w1 pwkso:hasRelation ?bn .
    ?bn a ?class ;
        ?param ?w2 .
    ?param rdfs:subPropertyOf* pwkso:param .
  OPTIONAL{
     ?bn ?index ?value .
     ?index rdfs:subPropertyOf* pwsko:index . }
  BIND(ex:bnode(?bn) as ?bn2)
}
```

7.3. The case of nested possible worlds and nested Kripke structures: possible worlds and Kripke structures contained in a possible world

In the context of semantic frameworks like Description Logics and modal logics, the notion of one possible world being contained within another refers to the relationship between different interpretations or models. In such frameworks, a possible world represents a specific interpretation of a knowledge base

[3]This query assumes that custom function `ex:bnode(_:a)` generates a blank node distinct from `_:a`, but shared across all the solution mappings of the query. See also https://github.com/w3c/sparql-dev/issues/36

or a set of statements. Each possible world provides an assignment of truth values to atomic statements or propositions within the knowledge base.

When we say that one possible world is contained within another, it means that it represents a possible world contained and possible w.r.t. the former. It can be useful to represent "possible" Kripke structure to capture, for example, possible effects of actions in Dynamic Epistemic Logics (DEL) [35] or Propositional Dynamic Logics (PDL) [4]. Consider the following representation:

```
<t2> { <x> ex:isPresidentOf <England> . } ## Proposition t2: <x> is president of England
<p> {  <t2> :verifiedIn <w2> . }
<p> :verifiedIn <w1> .
```

This represents two possible worlds w_1 and w_2. Named graph <p> indicates that proposition t_2 is verified in the world w_2. Since we consider that the named graph <p> is verified in w_1, by transitivity it means that the proposition t_2 is also verified in w_1. We conclude that we have described a situation where possible world are nested, and w_2 is contained in w_1.

Representing a Kripke structure within a possible world can be particularly useful when dealing with dynamic effects, as seen in DEL [35] or PDL [4]. These logics are concerned with capturing changes and updates to knowledge, beliefs, or states over time. By verifying a Kripke structure in a possible world, we can assess the truth and validity of modal formulas within that specific world, considering the dynamic aspects of the system. This allows us to reason about how knowledge, beliefs, or states evolve and interact within different possible worlds.

The representation of a Kripke structure within a possible world enables the modeling of dynamic effects, such as the updates to knowledge or the evolution of states, by using the accessibility relations between worlds. These relations define the transitions or changes that are possible between states, providing a framework for reasoning about the dynamics of the system.

In DEL for example, the verification of a Kripke structure in a possible world allows us to reason about the knowledge updates or belief revisions that occur in different situations. We can analyze how the knowledge or beliefs of agents change over time and in response to various events or actions.

Similarly, in PDL, the representation of a Kripke structure within a possible world facilitates the modeling and analysis of dynamic effects in systems that involve state transitions, actions, and updates.

By representing Kripke structures within possible worlds, PWKSO offers a powerful framework to capture and reason about dynamic effects in formal systems, providing insights into the evolution of knowledge, beliefs, or states over time.

7.4. A large panel of applications and domain specific extensions

PWKSO and extensions can be directly applied for different use cases from the families of motivating scenarios mentioned in Section 1. In addition, this section details a number of potential applications that leverage existing modal logic formalisms.

Capturing agents' knowledge and beliefs. Firstly, although we propose a general framework to handle (N+1)-ary Kripke relations, we leave abstract the representation of these Kripke relations. Therefore, a primary perspective is to define an epistemic frame not only to represent knowledge but also to capture the beliefs of agents using a doxastic frame [36]. As demonstrated in this article, this RDF representation has potential applications in merging knowledge graphs. Merging knowledge graphs is a common task in knowledge representation and semantic web applications. With the proposed framework, merging different knowledge graphs becomes more flexible and efficient. Rather than relying on heuristics to decide which triples should be deleted to maintain consistency, the framework allows for the representation of these triples in different possible worlds, which can then be merged without conflicts. Moreover, the use of possible worlds can facilitate the identification and resolution of inconsistencies in merged knowledge graphs, which can be challenging with traditional merging approaches. By pinpointing the source of inconsistencies across different possible worlds, users can address them in a more targeted manner.

Capturing agents' norms and actions. Secondly, another application of the proposed framework is to reason about norms and actions since Kripke frames are also used to represent the effects of actions [37], or norms [38]. Within the domain of formalizing actions and their effects, various formalisms exist, such as STRIPS [39], or different modal logics as, e.g., STIT [40], BIAT [41], PDL [4], etc. Thus, this framework can be used to propose a way to represent actions formalisms. For instance, in STRIPS [39], actions are depicted through a pre-condition and some effects. The pre-condition outlines the necessary conditions for the action to be applicable, while the effect delineates how the world undergoes changes upon the execution of the action. Thus, with PWKSO and the RDF representation of possible worlds and Kripke relation, we can represent possible effects by the fact some possible world can be reached after some actions have been performed. This facilitates, e.g., the sharing of actions that agents can perform in their environments, and their possible effects. Furthermore, by representing the possible effects of actions, we can represent and deduce what actions are forbidden and allowed w.r.t. their effects [42]. Thus, this framework would also help us to represent the normative status of actions in different contexts. For example, given a norm "it is forbidden to drive above a certain speed limit" and a set of actions that include driving at different speed in different possible worlds, it becomes possible to reason about which actions violate the norm. This could have applications in legal reasoning, ethical decision-making, and policy development. However detecting if a norm has been respected or violated implies to check the current state of the system and this is done with model-checking.

Model checking software agents. A third example of applications is the model-checking of software agents. Model checking is a formal verification technique used to automatically verify whether a given system model satisfies a desired property or specification. In model checking, the system behavior is typically represented as a formal model, such as a finite-state automaton or a transition system. Properties of interest, expressed in a formal specification language, are then checked against this model. If a property violation is found, the model checker provides a counter-example, which helps in diagnosing the cause of the failure. Model checking has applications in various domains, including hardware and software verification, concurrent and distributed systems, protocol verification, and cybersecurity. It is particularly useful for detecting design errors, software bugs, and security vulnerabilities early in the development process, or if a norm has not been respected. By leveraging PWKSO to represent Kripke frames, we can represent systems such as automata, enabling model-checkers to query the system's current states using SPARQL. PRISM [43] is a famous example of model-checker which allows the verification of properties expressed in probabilistic temporal logics such as Probabilistic Computation Tree Logic (PCTL), making it particularly suitable for systems with uncertain or probabilistic behavior. Integrating our RDF framework with such model-checking tools opens up new avenues for verifying automatically properties of probabilistic systems described with PWKSO, enhancing the scalability and efficiency of the verification process.

8. Conclusion

In this article, we represented normal Kripke models using RDF. This approach represents possible worlds and (N+1)-ary Kripke relations using named graphs and the W3C N-ary pattern.

First, we compared the different approaches for asserting that triples are verified or not verified in possible worlds: RDF reification, singleton properties, named graphs, RDF-star, and N3. We showed that all of the approaches allow us to effectively capture the concept of possible worlds and valuation function under the closed-world or the open-world assumption. We adopted the named graph approach to represent possible hyperworlds as it is standardized, largely adopted, and it may be useful to represent modal logic formulas, going beyond simple propositional atoms.

Then we compared the different approaches for representing (N+1)-ary accessibility relations between worlds, and motivated the adoption of the pattern "Introducing a new class for a relation" introduced in the W3C Working Group Note [23]. We selected its variant for modeling relation instances with a distinguished participant, where different instances of the same relation with the same arguments are equivalent and are recommended to be identified using blank nodes.

The resulting Possible Worlds and Kripke Structures Ontology (PWKSO) compiles our design rationale as a small OWL 2 DL ontology with 6 classes and 5 object properties, published under license CC-BY 4.0 under the namespace http://ns.inri.fr/pwkso/ using the recommended content negotiation rules. We proposed a simple usage pattern and illustrated it with different examples: binary indistinguishability relations in the context of epistemic multi-modal logic, a ternary relation for the dyadic deontic logic, and unary relations for the dynamic logic of mental attitudes and joint actions (DL-MA).

We thus demonstrated that our framework can represent all classical aspects of modalities as, e.g., knowledge of an agent, beliefs, possibility, necessity, effects of actions, or norms. It can also represent N-ary modalities such as for dyadic deontic operators, mental states or agency operators.

Our approach assumed a function f from IRIs and sets of triples to modal logic formulas. As a first step, in this article we only considered that f maps some statements to propositional atoms, but we opened the discussion on its desired characteristics. Therefore, this version of PWKSO does not include specific constructs to express modal logic formulas, and opens the door for more research in this direction and more generally for extending the top ontology provided in PWKSO with specific ontologies to cover specific languages and logics.

Other possible extensions of this work include further investigations regarding querying, instantiating, and reasoning with possible hyperworlds, and studying of how possible worlds or Kripke structures can be nested as a way to model Dynamic Epistemic Logics or Propositional Dynamic Logics.

Different direct applications of PWKSO and possible domain extensions have been envisioned and discussed, including 1. contributing to a standard exchange format for modal logic applications, including reasoners; 2. facilitating the Open Science initiative in the modal logic research community; 3. supporting the representation and management of uncertainty or other modalities; 4. facilitating interoperability between heterogeneous systems; and 5. leveraging existing modal logic formalisms for example to merge knowledge graphs, represent the knowledge or beliefs of agents, represent the effects of actions or norms, or develop model checking applications.

Acknowledgment

We thank the 3IA Côte d'Azur ANR-19-P3IA-0002 and the HyperAgents project ANR-19-CE23-0030 for their support.

References

[1] G.H. Von Wright, An essay in modal logic (1951). doi:10.2307/2216596.
[2] P. Blackburn, M. De Rijke and Y. Venema, *Modal Logic: Graph. Darst*, Cambridge University Press, New York, 2002.
[3] R. Fagin, J.Y. Halpern, Y. Moses and M. Vardi, *Reasoning About Knowledge*, The MIT Press, 2004. ISBN 9780262256094. doi:10.7551/mitpress/5803.001.0001.
[4] D. Harel, D. Kozen and J. Tiuryn, *Dynamic Logic*, in: *Handbook of Philosophical Logic*, Springer Netherlands, 2001, pp. 99–217. ISBN 9789401704564. doi:10.1007/978-94-017-0456-4_2.
[5] J. Horty, *Agency and Deontic Logic*, Oxford University Press, 2001. doi:10.1093/0195134613.001.0001.
[6] T. Dalmonte, A. Mazzullo, A. Ozaki and N. Troquard, *Non-Normal Modal Description Logics*, in: *Logics in Artificial Intelligence*, S. Gaggl, M.V. Martinez and M. Ortiz, eds, Springer Nature Switzerland, 2023, pp. 306–321. ISSN 1611-3349. ISBN 9783031436192. doi:10.1007/978-3-031-43619-2_22.
[7] F. Baader and S. Ghilardi, Unification in modal and description logics, *Logic Journal of IGPL* **19**(6) (2010), 705–730. doi:10.1093/jigpal/jzq008.
[8] P. Woltert and M. Zakharyaschev, Modal description logics: modalizing roles, *Fundamenta Informaticae* **39**(4) (1999), 411–438. doi:10.3233/fi-1999-39405.
[9] F. Baader and A. Laux, Terminological Logics with Modal Operators (1995), 808–814.
[10] H. Prakken and M. Sergot, *Dyadic Deontic Logic and Contrary-to-Duty Obligations*, in: *Defeasible Deontic Logic*, Springer Netherlands, 1997, pp. 223–262. ISBN 9789401588515. doi:10.1007/978-94-015-8851-5_10.
[11] E. Lorini, *On the Logical Foundations of Moral Agency*, in: *Deontic Logic in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 108–122. ISSN 1611-3349. ISBN 9783642315701. doi:10.1007/978-3-642-31570-1_8.

[12] G. Bonnet, C. Leturc, E. Lorini and G. Sartor, *Influencing Choices by Changing Beliefs: A Logical Theory of Influence, Persuasion, and Deception*, in: *Deceptive AI*, Springer International Publishing, 2021, pp. 124–141. ISSN 1865-0937. ISBN 9783030917791. doi:10.1007/978-3-030-91779-1_9.
[13] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider and D. Nardi, *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, 2007. ISBN 9780521150118. doi:10.1017/cbo9780511711787.
[14] K. Schild, A correspondence theory for terminological logics: preliminary report, in: *Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'91, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991, pp. 466–471–. ISBN 1558601600.
[15] R. Rosati, On the Semantics of Epistemic Description Logics., *Description Logics* **96** (1996), 05.
[16] F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt and A. Schaerf, An epistemic operator for description logics, *Artificial Intelligence* **100**(1–2) (1998), 225–274. doi:10.1016/s0004-3702(98)00009-5.
[17] A. Artale and E. Franconi, A temporal description logic for reasoning about actions and plans, *Journal of Artificial Intelligence Research* **9** (1998), 463–506. doi:10.1613/jair.516.
[18] A. Artale and E. Franconi, A survey of temporal extensions of description logics, *Annals of Mathematics and Artificial Intelligence* **30** (2000), 171–210.
[19] C. Areces and B. ten Cate, *14 Hybrid logics*, in: *Studies in Logic and Practical Reasoning*, Elsevier, 2007, pp. 821–868. ISSN 1570-2464. doi:10.1016/s1570-2464(07)80017-6.
[20] S. Muñoz, J. Pérez and C. Gutierrez, *Minimal Deductive Systems for RDF*, in: *The Semantic Web: Research and Applications*, Springer Berlin Heidelberg, 2007, pp. 53–67. ISSN 1611-3349. ISBN 9783540726678. doi:10.1007/978-3-540-72667-8_6.
[21] F. Orlandi, D. Graux and D. O'Sullivan, Benchmarking RDF metadata representations: Reification, singleton property and RDF, in: *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, IEEE, 2021, pp. 233–240. doi:10.1109/icsc50631.2021.00049.
[22] V. Nguyen, O. Bodenreider and A. Sheth, Don't like RDF reification? Making statements about statements using singleton property, in: *Proceedings of the 23rd international conference on World wide web*, 2014, pp. 759–770. doi:10.1145/2566486.2567973.
[23] N. Noy and A. Rector, Defining N-ary Relations on the Semantic Web, W3C Working Group Note, W3C, 2006-04. https://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/.
[24] V. Nguyen, H.Y. Yip, H. Thakkar, Q. Li, E. Bolton and O. Bodenreider, Singleton Property Graph: Adding A Semantic Web Abstraction Layer to Graph Databases., in: *BlockSW/CKG@ ISWC*, 2019, pp. 1–13.
[25] J.J. Carroll, C. Bizer, P. Hayes and P. Stickler, Named graphs, *Journal of Web Semantics* **3**(4) (2005), 247–267. doi:10.1016/j.websem.2005.09.001.
[26] R. Crouch and A.-L. Kalouli, Named graphs for semantic representation, in: *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, 2018, pp. 113–118. doi:10.18653/v1/s18-2013.
[27] A. Zimmermann, RDF 1.1: On Semantics of RDF Datasets, W3C Working Group Note, W3C, 2014-02. https://www.w3.org/TR/rdf11-datasets/.
[28] O. Hartig, Foundations of RDF* and SPARQL*:(An alternative approach to statement-level metadata in RDF), in: *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017.*, Vol. 1912, Juan Reutter, Divesh Srivastava, 2017.
[29] O. Hartig and P.-A. Champin, Metadata for rdf statements: the rdf-star approach, in: *Lotico*, 2021.
[30] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf and J. Hendler, N3Logic: A logical framework for the World Wide Web, *Theory and Practice of Logic Programming* **8**(3) (2008), 249–269. doi:10.1017/s1471068407003213.
[31] T. Berners-Lee, D. Connolly, E. Prud'hommeaux and Y. Scharf, Experience with N3 rules, in: *W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA*, W3C, 2005. http://www.w3.org/2004/12/rules-ws/paper/94.
[32] J. Frey, K. Müller, S. Hellmann, E. Rahm and M.-E. Vidal, Evaluation of metadata representations in RDF stores, *Semantic Web* **10**(2) (2019), 205–229. doi:10.3233/sw-180307.
[33] S. Sen, M.C. Malta, D. Katoriya, B. Dutta and A. Dutta, Labeled k-partite Graph for Statement Annotation in the Web of Data, in: *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, IEEE, 2020, pp. 63–71. doi:10.1109/wiiat50758.2020.00014.
[34] S. Sen, D. Katoriya, A. Dutta and B. Dutta, RDFM: An alternative approach for representing, storing, and maintaining meta-knowledge in web of data, *Expert Systems with Applications* **179** (2021), 115043. doi:10.1016/j.eswa.2021.115043.
[35] H. van Ditmarsch, W. van der Hoek and B. Kooi, *Dynamic Epistemic Logic*, Springer Netherlands, 2008. ISBN 9781402058394. doi:10.1007/978-1-4020-5839-4.
[36] J.-J.C. Meyer, Modal Epistemic and Doxastic Logic, in: *Handbook of Philosophical Logic*, Springer Netherlands, 2003, pp. 1–38. ISBN 9789401745246. doi:10.1007/978-94-017-4524-6_1.
[37] A. Herzig, E. Lorini and N. Troquard, *Action Theories*, in: *Introduction to Formal Philosophy*, Springer, Cham, 2018, pp. 591–607. ISSN 2569-8753. ISBN 9783319774343. doi:10.1007/978-3-319-77434-3_33.
[38] G.H. Von Wright, *On the Logic of Norms and Actions*, in: *New Studies in Deontic Logic*, Springer Netherlands, 1981, pp. 3–35. ISBN 9789400984844. doi:10.1007/978-94-009-8484-4_1.

[39] V. Lifschitz, On the semantics of STRIPS, in: *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Elsevier, 1987, pp. 1–9. doi:10.1016/b978-0-934613-30-9.50004-4.

[40] J. Broersen, *A Complete STIT Logic for Knowledge and Action, and Some of Its Applications*, in: *Declarative Agent Languages and Technologies VI*, Springer Berlin Heidelberg, 2009, pp. 47–59. ISSN 1611-3349. ISBN 9783540939207. doi:10.1007/978-3-540-93920-7_4.

[41] C. Leturc and G. Bonnet, A Deliberate BIAT Logic for Modeling Manipulations, in: *19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, 2020, pp. 699–707. ISBN 9781450375184.

[42] G. Boella, L. van der Torre and H. Verhagen, Introduction to normative multiagent systems, *Computational & Mathematical Organization Theory* **12**(2–3) (2006), 71–79. doi:10.1007/s10588-006-9537-7.

[43] M. Kwiatkowska, G. Norman and D. Parker, *PRISM 4.0: Verification of Probabilistic Real-Time Systems*, in: *Computer Aided Verification*, Springer Berlin Heidelberg, 2011, pp. 585–591. ISSN 1611-3349. ISBN 9783642221101. doi:10.1007/978-3-642-22110-1_47.