Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

An Enhanced OWL-Time Ontology for Complex Recurring Temporal Patterns

Yusuf Aminat Bolatito[1], Junaidu Sahalu Balarabe[2], Obiniyi Afolayan Ayodele[3], Aliyu Salisu[4], and Kana Armand Florentin Donfack[5]

[1]Department of Information Communication Engineering, Usmanu Danfodiyo University Sokoto, Sokoto

[2, 3, 4, 5]Department of Computer Science, Ahmadu Bello University Zaria, Kaduna Nigeria

aminbolly@gmail.com[1], abuyusra@gmail.com[2], aaobiniyi@gmail [3], aliyusalisu@abu.edu.ng[4], donfackkana@gmail.com[5]

Abstract

The effective representation of temporal information across various domains using semantic web specifications is crucial for enabling temporal reasoning and facilitating time-oriented queries. However, existing temporal ontologies often lack consistency in time representation within a single ontology and are typically tailored to specific domains, limiting their applicability to complex temporal information. Notably, prominent frameworks like OWL-Time frequently overlook the intricacies of complex recurring temporal features. To address this limitation within OWL-Time, this study introduces OWL-BOLA 1.0, an extension specifically engineered to enhance OWL-Time for handling complex recurring temporal patterns. The aim of this study is to augment OWL-Time with specialized concepts and properties using a lightweight approach suited for recurring time, enabling a more comprehensive representation of temporal knowledge and enhancing reasoning capabilities. The objective is to provide robust support for handling recurring temporal patterns within OWL-Time constructs, thus enabling the reasoning of both simple and complex recurring temporal patterns. The methodology utilized in this study involves extending OWL-Time by integrating additional concepts and properties that cater to recurring temporal features. This approach ensures a more explicit representation of temporal knowledge while maintaining the lightweight nature of the ontology. The evaluation of the proposed ontology is carried out by analyzing metrics such as the depth of class hierarchy and inheritance richness. The results demonstrate the efficacy of OWL-BOLA 1.0 in providing explicit representation and handling complex recurring temporal patterns. The ontology exhibits a depth of class hierarchy of 6 and a high inheritance richness of 0.915, highlighting its impressive performance and utility compared to existing ontologies. Although it shows slightly lower semantic richness (0.466) and data property richness (0.237) in conceptualization, OWL-BOLA 1.0 effectively reduces redundancy and maintains clarity. In conclusion, OWL-BOLA 1.0 offers a balanced and effective tool for various application domains by enhancing OWL-Time's capability to handle complex recurring temporal patterns. This study contributes to the field by providing a robust solution for temporal reasoning, making it applicable across diverse domains requiring sophisticated temporal information representation.

Keywords: *Temporal Ontologies, OWL-Time, OWL-BOLA 1.0, Recurring Temporal Patterns, Temporal Reasoning*

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

1. Introduction

The Semantic Web, as envisioned by Berners-Lee et al. (2001), endeavors to enrich the World Wide Web with deeper meaning and machine readability (Wang et al., 2019), fostering knowledge exchange among diverse applications (W3C, 2002). With the web's dynamic nature driven by user-generated content, real-time updates, and evolving applications (Achich et al., 2021; Li et al., 2020), semantic web services encapsulate not just static but also dynamic information (Wang et al., 2012). In domains such as e-commerce, e-government, and e-health, effective temporal tracking is paramount for handling time-sensitive data (Zekri et al., 2014). Consider leisure event scheduling, where organizing events into thematic calendars involves incorporating various data facets, including descriptive, spatial, and temporal information (Faucher et al., 2012). This temporal information spans from simple (e.g., an event at 10 AM on 27/04/2024) to complex recurring patterns (e.g., twice every week). Given the Semantic Web's aim to provide explicit data semantics and facilitate knowledge sharing, it is essential for web pages to semantically encode temporal information. This ensures that software agents can access meta-materials vital for short- or long-term event predictions, aligning with the Semantic Web's overarching goal. Building upon existing web technologies like the Resource Description Framework (RDF), Web Ontology Language (OWL), and SPARQL (Almendros-Jiménez & Becerra-Terón, 2021; Deborah & Frank, 2004), the Semantic Web establishes a framework of machine-readable metadata delineating data relationships. However, OWL, the cornerstone language for semantics, encounters challenges in temporal representation due to its binary relationships and logic-based formalism (Krieger, 2008; O'Connor & Das, 2011), often necessitating reliance on XML Schema for temporal specifications (Henry et al., 2009). Furthermore, OWL's temporal expressivity remains constrained by the absence of dedicated temporal syntax or constructs (Hobbs & Pan, 2004), hindering automated temporal validation, deductive rules, and queries (O'Connor & Das, 2011). While initiatives like OWL-Time have emerged to enhance OWL's temporal capabilities, they often struggle to adequately represent complex recurring time, limiting their applicability in domains such as business management, historical events, and healthcare. Various endeavors have sought to enhance OWL-Time's capacity to express complex recurring times (Robert et al., 2020), but existing solutions typically fall short in capturing diverse forms of recurring time, including intermittent (e.g., three times in 2023), stochastic (e.g., every Friday to Sunday, closed from December 25 to January 3), and non-recurring patterns (e.g., every April 25, between 3 p.m. and 6 p.m., and every April 26, between 9 a.m. and 11 a.m.), as observed in classical ontologies (Bento et al., 2017; Terenziani, 2016; Tuzhilin & Clifford, 1995). Despite partial attempts to address these challenges, none have comprehensively remained within OWL-Time constructs. This paper aims to bridge this gap by effectively representing complex recurring time information within the OWL-Time framework, thus enriching the open domain ontology landscape. This represents a significant contribution to the field of computer science, particularly in semantic web technologies and temporal information representation. The key contributions of this paper are summarized as follows:

1. Introduction to OWL-BOLA 1.0, a novel model that extends the OWL-Time model with a sophisticated ontology for representing and reasoning about complex recurring temporal patterns. This extension enhances two key aspects of the OWL-Time model: its primitive entities and calendar descriptions, with recurring primitive entities and periodic calendar

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

descriptions, respectively. This innovation addresses existing limitations in temporal ontologies by enabling the detailed depiction of intricate recurring patterns. Its versatility is evident in its applicability across various domains, including business, healthcare, and e-commerce, highlighting its broad potential impact.

2. Introduction of the concept of recurring primitive entities and periodic calendar description using First-Order Logic (FOL) language, illustrated through diagrams and formulas. FOL provides a rigorous framework for defining and reasoning about intricate temporal relationships, enhancing the expressiveness of the OWL-BOLA 1.0 model. This approach not only aligns with established theoretical foundations but also ensures that the model is scalable and adaptable.
3. Empirical validation of our ontology's effectiveness using real-world datasets, including assessment of ontology metrics such as complexity and conceptualization richness. The empirical validation of these concepts is facilitated by the clear, formal representation provided by FOL, demonstrating their effectiveness and practical utility in various application. The remainder of the paper is structured into four main sections. Section 2 is dedicated to the background knowledge of OWL-Time constructs. Section 3, we offer a summary of related work, providing context for our contributions. Section 4 outlines our methodology, expressed in FOL, detailing our approach to extending the OWL-Time model. The subsequent section, Section 5, presents and discusses the measurement results obtained from our empirical validation, shedding light on the effectiveness of our proposed ontology. Finally, we conclude the paper in Section 6, offering concluding remarks and recommendations for future research directions.

2. Related Works

One of the earliest studies exploring different categorizations of complex recurring time can be found in classical ontologies (Logananthara & Gombrone, 1995; Terenziani, 2016; Tuzhilin & Clifford, 1995). Tuzhilin and Clifford (1995) introduced temporal constraints for repeated actions or periods, contributing to the database management of user-defined periodicities. Their framework categorized recurring time into regular/periodic and irregular/aperiodic entities, distinguishing between strongly periodic, nearly aperiodic, and intermittent aperiodic occurrences. While their study focused on the first two categories, it overlooked intermittent occurrences. Terenziani (2016) extended the work of Tuzhilin and Clifford (1995) by providing a model for representing intermittent aperiodic repetitions and a temporal relational algebra for querying them. Similarly, Logananthara and Gombrone (1995) presented variations of aperiodic time, grouping them into nearly aperiodic, random aperiodic, and stochastic aperiodic time. In the realm of the Semantic Web, enriched temporal ontologies are discussed in works such as Bento et al. (2017), Hu et al. (2022), Li et al. (2020), Pan (2007), and Poveda-Villalón et al. (2014). Specifically focusing on the extension of the OWL-Time ontology, related studies include Bento et al. (2017), Pan (2007), and Poveda-Villalón et al. (2014). Pan (2007) proposed a method to address temporal representation challenges within ontologies, introducing a two-fold approach that diverged from foundational OWL-Time entities. This approach created a parallel construct, "time-entry," outside OWL-Time's basic entities and introduced TemporalSeq to capture recurring time instances. While addressing

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

recurring time effectively, Pan's method had limitations in representing or reasoning about aperiodic entities. Poveda-Villalón et al. (2014) introduced the PeriodicInterval class within OWL-Time to represent non-convex intervals, effectively reusing existing calendar descriptions. Although efficient for modeling periodic entities, their method lacked formalization and primarily focused on periodic representations. Bento et al. (2017) extended the OWL-Time ontology by introducing the NCInterval primitive concept and developing a calendar description framework. Their approach adeptly handled both periodic and aperiodic instances of non-recurring time within the OWL-Time framework. However, it primarily focused on recurring time instances, potentially overlooking other temporal aspects.

3. Methodology

Traditionally, OWL-Time (Hobbs & Pan, 2004; Hobbs et al., 2017) only allows for the representation of simple time using Instant and Interval primitive entities. These entities are initially introduced at the onset of temporal expression, with subsequent connections of individual instances generated from the primitive entities to their respective calendar descriptions, enabling explicit representation of time. For illustration purposes, consider **Example 1**, where the representation of a competition taking place on Tuesday, October 8th, 2023, at 10 AM is encoded. This representation is visually depicted in Figure 1.

```
1. <competition> rdf:type Instant;
2.       inDateTime <beginning>;
3. <beginning> rdf:type GeneralDateTimeDescription;
4.       unitType “unitDay”;
5.       dayOfWeek “Tuesday”;
6.       day “10”;
7.       month “08”;
8.       year “2023”;
```

Figure 1. Defining Occurrence of Instant Primitive

Figure 1 visually represents the scheduling of a competition event occurring on Tuesday, the 10th of August 2023, at 10 AM. The central node denotes the competition, classified as an "Instant" in time. The competition's start time, labeled "beginning," is described in detail, including the day of the week, day, month, and year. These properties provide a structured representation of the event's temporal information in OWL format, enabling machine-readable interpretation within the Semantic Web.

In this section, we undertake an expansion of the fundamental OWL-Time primitive entities, aimed at enriching the representation of temporal information. Our approach involves expanding OWL-Time primitive Interval entity by introducing a non-convex interval named TemporalSeq, a novel concept designed to accommodate complex recurring time patterns. We also represent patterns within the TemporalSeq through the creation of specialized subclasses, including PeriodicInstant, PeriodicInterval, PeriodicTimeSpan, and PeriodicException. These subclasses are tailored to represent the nuanced properties associated with TemporalSeq, facilitating dedicated representations for diverse recurring temporal aspects (periodic and aperiodic recurring time) as covered in the related studies. Furthermore, we extend the OWL-Time calendar description to

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

incorporate periodic calendar descriptions. This extension builds upon the existing GeneralDateTimeDescription (Hobbs et al., 2017) by employing a lightweight technique. The result is the creation of simplified classes that prioritize ease of use and minimal complexity within GeneralDateTimeDescription. Notably, this extension is supported by two significant subclasses: UnitTimeDescription and PeriodicTimeSpanDescription.

The substantial contributions outlined in enriching OWL-Time is illustrated in Figure 1.1. The coloured pink section indicates the recurring primitive entities and the blue colured indicates the periodic calendar description contributions to the existing OWL-Time concepts.

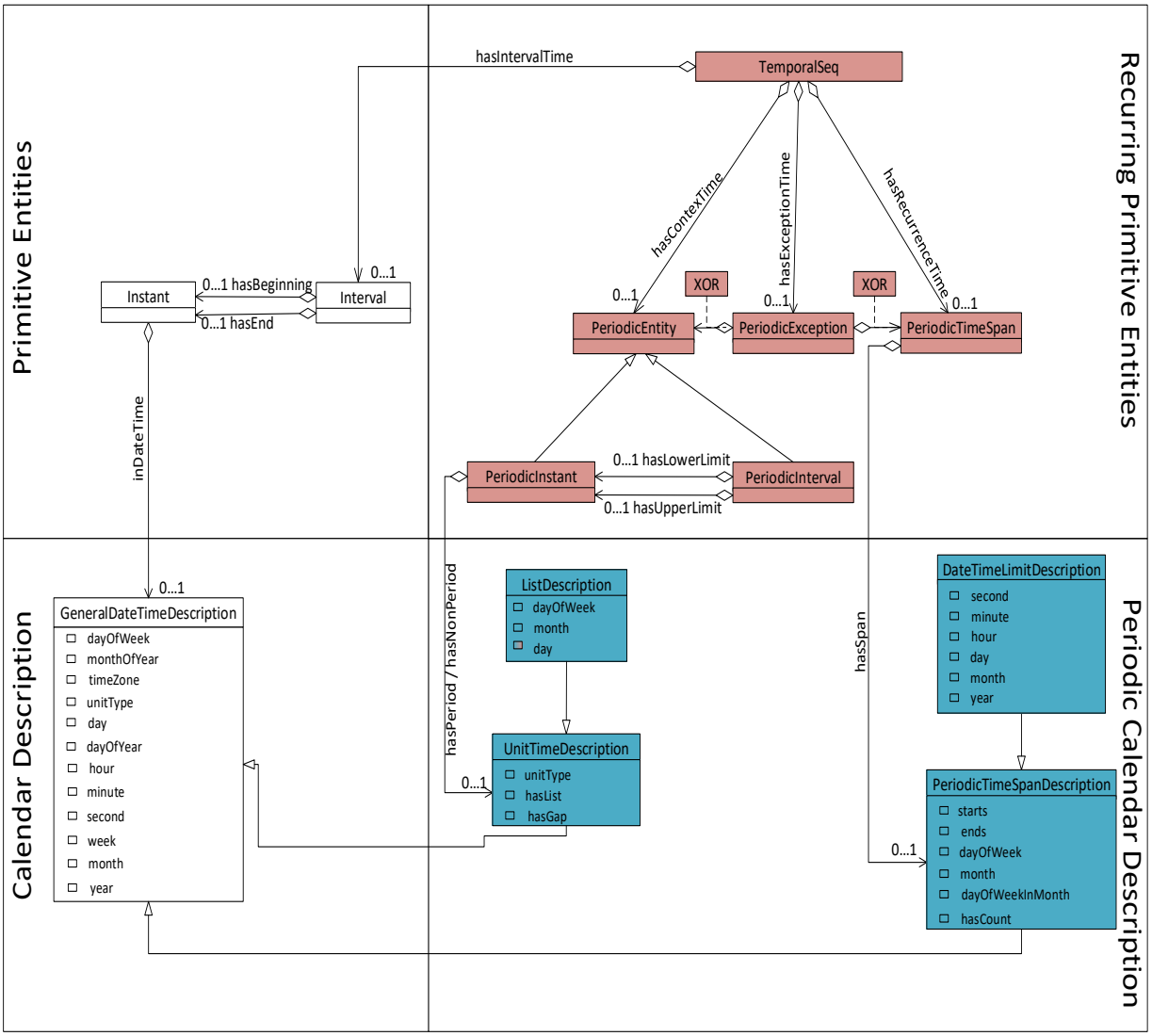

Figure 2. An overview of the proposed ontology (OWL-BOLA 1.0)

The figure 2 illustrates two essential relationships: the aggregation association and the subclass relationship. In the aggregation association, the arrows depict the cardinality, representing the

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

connection between the domain and the range. This cardinality varies, indicated by [0 to 1], shows multiple elements in the range. Conversely, the subclass relationship delineates the generalization hierarchy's scope, elucidating how subclasses inherit attributes and behaviors from their super classes. As shown in figure, we enriched the expressive capabilities of OWL-Time by introducing recurring periodic entities and periodic description, summarized in section 3.1 and 3.2.

3.1 Recurring Primitive Entities

The OWL-Time framework is built upon two foundational entities: Instant and Interval, both encompassed within the TemporalEntity framework. Instant refers to a singular, point-like unit of time, while Interval represents a period of time with a distinct duration. Notably, the OWL-Time specification emphasizes ProperInterval, a subclass characterizing Intervals with two distinct Instant endpoints. However, a notable gap exists in representing non-convex Intervals within the OWL-Time framework. To address this limitation, we extend the Interval class by introducing TemporalSeq and its associated subclasses. This innovative extension enables seamless definition of non-convex Intervals within the context of calendar datetime. The visual representation of this hierarchical structure, is presented Figure 2 under the category of recurring periodic entities and each element are discussed as follows.

3. 1.1. Temporal Sequence

In the domain of OWL-Time Interval, a variety of Interval types are encountered, ranging from proper and positive/negative half-infinite to finite Intervals. However, a notable gap exists in the representation of non-convex Intervals, distinguished by their characteristic gaps. To address this limitation, we introduce TemporalSeq, an extension of the Interval class aimed at accommodating the representation of non-convex Intervals within calendar datetime contexts. TemporalSeq encompasses finite, positive half-infinite, negative half-infinite, and infinite variations. Drawing inspiration from its predecessor Interval, TemporalSeq inherits critical properties such as hasBeginning and hasEnd, utilized to denote Interval boundaries, with their invocation facilitated through the object property hasIntervalTime. This property serves as a linkage between individual TemporalSeq instances and OWL-Time Intervals, as depicted in Axiom 1. Additionally, three additional object properties attributed to TemporalSeq—hasContexTime, hasRecurrenceTime, and hasExceptionTime—are introduced as shown in Axiom 2 to 4. These properties have ranges in PeriodicEntity, PeriodicTimeSpan, and PeriodicException, respectively, and will be elaborated upon in the forthcoming sections, from 3.1.2 to 3.1.4.

Axiom 1: Defining hasIntervalTime Property Relationship between TemporalSeq and Interval.

$$\forall\, x, y(hasIntervalTime(x, y) \rightarrow (TemporalSeq(x) \wedge Interval(y)))$$

Axiom 2: Defining hasContexTime Property Relationship between TemporalSeq and PeriodicEntity.

$$\forall\, x, y(hasContexTime(x, y) \rightarrow (\, TemporalSeq(x) \wedge PeriodicEntity(y)))$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Axiom 3: Defining hasRecurrenceTime Property Relationship between TemporalSeq and PeriodicTimeSpan.

$$\forall x, y(hasRecurrenceTime(x, y) \rightarrow (TemporalSeq(x) \wedge PeriodicTimeSpan(y)))$$

Axiom 4: Defining hasExceptionTime Property Relationship between TemporalSeq and PeriodicException.

$$\forall x, y(hasExceptionTime(x, y) \rightarrow (TemporalSeq(x) \wedge PeriodicException(y)))$$

Definition 1: Finite TemporalSeq with Intervals and Instants.
If there is a TemporalSeq x, and it is of type finite TemporalSeq, then there is an Interval y, an Instant u, and an Instant v that satisfy the conditions: x is linked to y via hasIntervalTime, y is an Interval, y is linked to u via hasBeginning, and y is linked to v via hasEnd.

$$\forall x\ (TemporalSeq(x) \wedge FiniteTemporalSeq(x) \rightarrow \exists y, u, v\ (hasIntervalTime(x, y) \wedge Interval(y) \wedge hasBeginning(y, u) \wedge hasEnd(y, v)))$$

Example 2: The 2023 second-semester lectures occurred from September 2, 2023 to December 24, 2023.

```
1. <secondSemesterLectures> rdf:type FiniteTemporalSeq;
2.       hasIntervalTime <september022023ToDecember242023>;
3. <september022023ToDecember242023> rdf:type Interval;
4.       hasBeginning <september0220233>;
5.       hasEnd <december242023>;
```

Figure 3. Defining Occurrence of FiniteTemporalSeq

Figure 3 illustrates Example 2, depicting the Definition 1 of the occurrence of second-semester lectures from September 2, 2023, to December 24, 2023 which is a FiniteTemporalSeq. The RDF triples represent this interval as a finite sequence of time intervals, denoted by <secondSemesterLectures>, with a start date of September 2, 2023, and an end date of December 24, 2023.

Definition 2: Defining Positive Half-Infinite TemporalSeq with Intervals and Instants.
If there is a TemporalSeq x, and it is of type positive half-infinite TemporalSeq, then there are Interval y and Instant u such that x is linked to y with hasIntervalTime, y is an Interval, and y is linked to u with hasBeginning.

$$\forall x\ (TemporalSeq(x) \wedge PositiveHalfInfiniteTemporalSeq(x) \rightarrow \exists y, u\ (hasIntervalTime(x, y) \wedge Interval(y) \wedge hasBeginning(y, u)))$$

Example 3: The recurring yoga class starts from January 01, 2023.

```
1. <yogaClass> rdf:type PositiveHalfInfiniteTemporalSeq;
2.       hasIntervalTime <fromJanuary012023>;
3. <fromJanuary012023> rdf:type Interval;
4.       hasBeginning <january012023>;
```

Figure 4. Defining Occurrence of PositiveHalf InfiniteTemporalSeq

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Figure 4 illustrates Example 3, depicting the Definition 2 where the recurring yoga class initiates from January 01, 2023. It illustrates a positive half-infinite temporal sequence denoted by <yogaClass>, indicating an ongoing series of events starting from January 01, 2023. The RDF triples define this sequence with a starting point on January 01, 2023.

Definition 3: Defining Negative Half-Infinite TemporalSeq with Intervals and Instants.

If there is a TemporalSeq x, and it is of type negative half-infinite TemporalSeq, then there are Interval y and Instant v such that x is linked to y with hasIntervalTime, y is an Interval, and y is linked to v with hasEnd.

$$\forall\, x\, (TemporalSeq(x) \wedge NegativeHalfInfiniteTemporalSeq(x) \rightarrow \exists\, y,u\, (hasIntervalTime(x,y)\ \wedge\ Interval(y)\ \wedge\ hasEnd(y,v)))$$

Example 4: The weekly webinar series will be held until December 31, 2024.

```
1. <webinarSeries> rdf:type NegativeHalfInfiniteTemporalSeq;
2.       hasIntervalTime <untilDecember312024>;
3. <untilDecember312024> rdf:type Interval;
4.       hasEnd <december312024>;
```

Figure 5: Defining Occurrence of NegativeHalfInfiniteTemporalSeq

Figure 5 illustrates Example 4, depicting definition 3 a negative half-infinite temporal sequence denoted by <WebinarSeries>. This sequence represents a recurring event, specifically a weekly webinar series that will continue until December 31, 2024. The RDF triples define this sequence with an endpoint on December 31, 2024.

3.1.2. Periodic Entity

Here, we delve into the exploration of patterns within TemporalSeq, which are manifested through collections of instants and intervals. These collections encompass PeriodicInstant (e.g., every Monday, every two Thursdays) and PeriodicInterval (e.g., every Monday to Friday, every January to March), representing collections of instants and intervals respectively, and are encapsulated as subclasses of PeriodicEntity. PeriodicEntity instance is attached to TemporalSeq through the hasContexTime property in Axiom 2. We provided the formal representation of the PeriodicEntity concept in Axiom 5. In Axiom 6, we introduce two object properties, hasPeriod and hasNonPeriod, for PeriodicInstant, establishing connections between PeriodicInstant and the periodic calendar description discussed in 3.2. Similarly, Axiom 7 establishes the relationship between the beginning and end of a PeriodicInterval, we employ the object properties hasLowerLimit to define the starting point and hasUpperLimit to specify the endpoint. These properties not only enable the specification of PeriodicInstants as the starting or ending points of a periodic Interval but also facilitate the utilization of UnitTimeDescription as outlined in Axiom 6.

Axiom 5: Two Mutually Exclusive Options of PeriodicEntity.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

If there is a PeriodicEntity x, then it is an ordered temporal recurring entity which is either a PeriodicInstant or a PeriodicInterval

$$\forall x \left(PeriodicEntity(x) \rightarrow \left(PeriodicInstant(x) \vee PeriodicInterval(x)\right)\right)$$

Axiom 6: Defining properties hasPeriod and hasNonPeriod for PeriodicInstants.
Every instance of PeriodicInstant x must have the property hasPeriod linking to an instance of UnitTimeDescription y and may have the property hasNonPeriod linking separately a UnitTimeDescription z.

$$\forall x\ (PeriodicInstant(x) \rightarrow (\exists y\ (hasPeriod(x,y) \wedge UnitTimeDescription(y)) \vee (\exists z\ (hasNonPeriod(x,z) \wedge UnitTimeDescription(z)))))$$

Example 5: The meeting will be held every Monday from September 2, 2023, to December 24, 2023.

```
1. <meeting> rdf:type FiniteTemporalSeq;
2.       hasIntervalTime <september022023ToDecember242023>;
3.       hasContexTime <everyMonday>;
4. <september022023ToDecember242023> rdf:type Interval;
5.       hasBeginning <september0220233>;
6.       hasEnd <december242023>;
7. <everyMonday> rdf: type PeriodicInstant;
8.       hasPeriod <periodMonday>;
9. <periodMonday> rdf: type UnitTimeDescription;
```

Figure 6. TemporalSeq with features FiniteTemporalSeq and PeriodicInstant

Figure 6 illustrates Example 5 within the OWL-BOLA 1.0 constructs. It encodes a representation of a repeating event in FiniteTemporalSeq, which has two properties: hasIntervalTime (september022023ToDecember242023) for its interval and hasContextTime (everyMonday) for its PeriodicInstant, signifying its recurrent occurrence. The properties hasBeginning and hasEnd from OWL-Time are reused to represent the interval start (september022023) and end (december242023). To encode "everyMonday," the concept of PeriodicInstant is used to denote its repetitive nature. This representation utilizes the property hasPeriod, linking it to a UnitTimeDescription specifying the recurring period, identified in this case as PeriodMonday, as covered in Axiom 6.

Axiom 7: Defining properties hasLowerLimit and hasUpperLimit for PeriodicIntervals.
Every PeriodicInterval x has a hasLowerLimit property that links it to a PeriodicInstant and a hasUpperLimit property that links it to a PeriodicInstant

$$\forall x\ (PeriodicInterval(x) \rightarrow (\exists y\ (hasLowerLimit(x,y) \wedge PeriodicInstant(y)) \wedge \exists z\ (hasUpperLimit(x,z) \wedge PeriodicInstant(z))))$$

Example 6: The Champions League starts on January 01, 2023, and occurs every Monday to Friday.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

```
1. <championsLeague> rdf:type PositiveHalfInfiniteTemporalSeq;
2.       hasIntervalTime <startsJanuary012023>;
3.       hasContexTime <everyMonday_Friday>;
4. <startsJanuary012023> rdf:type Interval;
5.       hasBeginning <january012023>;
6. <everyMonday_Friday> rdf: type PeriodicInterval;
7.       hasLowerLimit <everyMonday>;
8.       hasUpperLimit<everyFriday>;
9. <everyMonday> rdf: type PeriodicInstant;
10.      hasPeriod <periodMonday>;
11. <periodMonday> rdf: type UnitTimeDescription;
12. <everyFriday> rdf :type PeriodicInstant;
13.      hasPeriod <periodFriday>;
14. <periodFriday> rdf: type UnitTimeDescription;
```

Figure 7. TemporalSeq with features PositiveHalfInfiniteTemporalSeq and PeriodicInterval

Figure 7 illustrates Example 6 within the OWL-BOLA 1.0 constructs. It encodes a representation of a repeating event in PositiveHalfInfiniteTemporalSeq, which has two properties: hasIntervalTime (startsJanuary012023) for its interval and hasContextTime (everyMonday_Friday) for its PeriodicInterval, signifying its recurrent occurrence throughout the week. The property hasBeginning from OWL-Time is reused to represent the interval start (startsJanuary012023). The property hasLowerLimit links to a PeriodicInstant denoting the starting point, "every Monday," while the property hasUpperLimit links to another PeriodicInstant representing the endpoint, "every Friday," of the PeriodicInterval "everyMonday_Friday." These PeriodicInstants are associated with UnitTimeDescriptions specifying the recurring periods, PeriodMonday and PeriodFriday, respectively. This encoding adheres to the specifications outlined in Definition 2, Axiom 6, and Axiom 7.

3.1.3 Periodic Time Span

The complexity of a PeriodicEntity can be compounded by the presence of another recurring time, which may include periodic date-time references or frequency counts. To enhance comprehension, we introduce the PeriodicTimeSpan class, designed to accommodate such additional information. For instance, expressions like "on the first Wednesday of every month" feature a PeriodicTimeSpan denoted by "first Wednesday" and a PeriodicInstant represented by "every month." Similarly, expressions like "every Monday to Wednesday, between 09:30 a.m. and 10:00 p.m." involve a PeriodicInterval for the weekdays and a PeriodicTimeSpan for the time range. The attachment of PeriodicTimeSpan instances to TemporalSeq is facilitated through the hasRecurrenceTime property in Axiom 3. Also, PeriodicTimeSpan class is distinct from PeriodicEntity, and is linked through the hasPeriodSpan property outlined in Axiom 8. However, instead of directly defining additional details, we introduce the hasSpan property to implicitly associate them with the periodic calendar description of section 3.2. This property serves as a placeholder for specifying the finest granularity of PeriodicTimeSpan, with explicit representation deferred to the PeriodicTimeSpanDescription in section 3.2. The relationship between hasSpan and the PeriodicTimeSpan and PeriodicTimeSpanDescription classes is elucidated in Axiom 9.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Axiom 8: Defining hasPeriodSpan Relationship between PeriodicEntity and PeriodicTimeSpan. For all instances x and y, if there exists a relationship hasPeriodSpan between x and y, then x must be an instance of PeriodicEntity, and y must be an instance of PeriodicTimeSpan.

$$\forall\, x, y(hasPeriodSpan(x, y) \rightarrow (PeriodicEntity(x) \wedge PeriodicTimeSpan(y)))$$

Axiom 9: Defining Relationship between PeriodicTimeSpan and PeriodicTimeSpanDescription. Every instance of PeriodicTimeSpan x must have the property hasSpan linking to an instance of PeriodicTimeSpanDescription.

$$\forall\, x \left(PeriodicTimeSpan(x) \rightarrow \exists\, y\big(hasSpan(x, y) \wedge PeriodicTimeSpanDescription(y)\big)\right)$$

Example 7: The weekly webinar series will be held until December 31, 2024, every Monday at 21:00 hours.

```
1. <webinarSeries> rdf:type NegativeHalfInfiniteTemporalSeq;
2.       hasIntervalTime <untilDecember312024>;
3.       hasContexTime <everyMonday>;
4.       hasRecurrenceTime <every21Hour>;
5. <untilDecember312024> rdf:type Interval;
6.       hasEnd <december312024>;
7. <everyMonday> rdf: type PeriodicInstant;
8.       hasPeriod <periodMonday>;
9. <periodMonday> rdf: type UnitTimeDescription;
10. <every21Hour> rdf: type PeriodicTimeSpan;
11.      hasSpan <period21Hour>;
12. <period21Hour> rdf: type PeriodicTimeSpanDescription;
```

Figure 8. TemporalSeq with features NegativeHalfInfiniteTemporalSeq, PeriodicInstant and PeriodicTimeSpan

Figure 8 illustrates Example 7 within the OWL-BOLA 1.0 constructs. It encodes a representation of a repeating event in a NegativeHalfInfiniteTemporalSeq that has three properties: hasIntervalTime (untilDecember312024) for its interval, hasContextTime (everyMonday) for its PeriodicInstant, and hasRecurrenceTime (every21Hour). The property hasEnd from OWL-Time is reused to represent the interval ending on "untilDecember312024." To encode its "everyMonday," we evoke the concept of PeriodicInstant to denote its repetitive nature. This representation utilizes the property hasPeriod, linking it to a UnitTimeDescription specifying the recurring period, identified in this case as PeriodMonday, as covered in Axiom 6. Additionally, the time of occurrence is represented as a PeriodicTimeSpan denoted by the instance every21Hour. This PeriodicTimeSpan is associated with a PeriodicTimeSpanDescription named period21Hour, which defines the specific time interval of '21 hours.' This encoding adheres to the specifications outlined in Axiom 6 and Axiom 9.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

3.1.4 Periodic Exception

The concept of regularity has been adequately addressed through the PeriodicEntity class. However, recurring expressions may also exhibit irregular patterns, which in this case in our ontology is captured using the PeriodicException. A PeriodicException refers to a specific instance of time that deviates from the regular recurring pattern. It represents a recurring time during which the recurrence is temporarily disrupted or exceptional. Examples of PeriodicException instances include excluding certain days or times from a regular schedule (e.g., every day at 5 p.m. except on Sundays at 2 p.m.). This type of exclusion is regular and is bound within the TemporalSeq Interval. The attachment of PeriodicException instances to TemporalSeq is facilitated through the hasExceptionTime property in Axiom 4. To represent and reason with the PeriodicException concept, we do not create its unique property, unlike other recurring entities such as PeriodicEntity and PeriodicTimeSpan. Instead, we opt to reuse TemporalSeq properties like hasContexTime and hasRecurrenceTime. This approach allows PeriodicException to leverage existing properties attributed to TemporalSeq through inheritance features. For instance, using the hasContexTime property, PeriodicException can involve a PeriodicInstant and utilize the hasNonPeriod property of PeriodicInstant. Similarly, with the hasRecurrenceTime property, PeriodicException can invoke PeriodicTimeSpan.

Example 8: The yoga class starts from January 01, 2023 every Monday at 10 hour except on holiday Monday at 14 hour.

```
1. <yogaClass> rdf:type PositiveHalfInfiniteTemporalSeq;
2.       hasIntervalTime <startsJanuary012023>;
3.       hasContexTime <everyMonday>;
4.       hasRecurrenceTime <every10Hour>;
5.       hasExceptionTime <everyHolidayMonday14Hour>;
6. <startsJanuary012023> rdf:type Interval;
7.       hasBeginning <january012023>;
8. <everyMonday> rdf: type PeriodicInstant;
9.       hasPeriod <periodMonday>;
10. <periodMonday> rdf: type UnitTimeDescription;
11. <every10Hour> rdf: type PeriodicTimeSpan;
12.      hasSpan <period10Hour>;
13. <period10Hour> rdf: type PeriodicTimeSpanDescription;
14. <everyHolidayMonday14Hour> rdf: type PeriodicException;
15.      hasContexTime <everyHolidayMonday>;
16.      hasRecurrenceTime <every14Hour>;
17. <everyHolidayMonday> rdf: type PeriodicInstant;
18.      hasNonPeriod <periodMonday>;
19. <periodMonday> rdf: type UnitTimeDescription;
20. <every14Hour> rdf: type PeriodicTimeSpan;
21.      hasSpan <period14Hour>;
22. <period14Hour> rdf: type PeriodicTimeSpanDescription;
```

Figure 9. TemporalSeq with features PositiveHalfInfiniteTemporalSeq, PeriodicInstant, PeriodicTimeSpan and PeriodicException

Figure 9 illustrates Example 8 is encoded using OWL-BOLA 1.0 constructs. The yoga class schedule is defined as a PositiveHalfInfiniteTemporalSeq, indicating its ongoing nature. The property hasIntervalTime links it to an Interval instance named "startsJanuary012023," which represents the timeframe of the yoga class schedule. The recurring pattern of the yoga class schedule is specified using two PeriodicInstants: "everyMonday" and "everyHolidayMonday," representing regular Mondays and holiday Mondays, respectively. These PeriodicInstants are

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

associated with the UnitTimeDescription "periodMonday," denoting the recurring period of Mondays. Additionally, the time of occurrence for regular yoga classes, "10 hours," is represented as a PeriodicTimeSpan named "every10Hour," which is linked to a PeriodicTimeSpanDescription, specifying the time interval of 10 hours. For holiday Mondays, occurring at 14 hours, the deviation from the regular schedule is captured using a PeriodicException instance named "everyHolidayMonday14Hour." This PeriodicException is linked to PeriodicInstants representing holiday Mondays and a PeriodicTimeSpan representing the time of 14 hours. These representations align with the specifications outlined in Axiom 1 to 4 and Definition 4 of our ontology.

3.2 Periodic Calendar Description

This section introduces an extended OWL-Time calendar description known as the periodic calendar description, designed for the explicit representation of repeating time values categorized into their respective classes or concepts. This extension addresses the limitations of OWL-Time in representing simple time notations, enabling the incorporation of periodic milestones. The aim is to establish a comprehensive framework for vocabulary specifications related to periodic calendar descriptions, facilitating the detailed representation of recurring entities. To enhance the expressive and reasoning power of OWL-Time's GeneralDateTimeDescription, four classes have been added: UnitTimeDescription, ListDescription, PeriodicTimeSpanDescription, and DateTimeLimitDescription. The visual representation of this hierarchical structure, is presented Figure 2 under the category of periodic calendar description and each element are discussed as follows.

3.2.1 Unit Time Description

In OWL-BOLA 1.0, the UnitTimeDescription concept was introduced for handling periodic calendar descriptions related to PeriodicInstant, PeriodicInterval, and PeriodicException. UnitTimeDescription extends GeneralDateTimeDescription by introducing properties and values specific to the periodicity of section 3.1 classes. UnitTimeDescription includes three primary properties: unitType, hasGap, and hasList. The unitType property of Axiom 10 represents periodicity frequency (e.g., "every year," "every month") and uses TemporalUnit of Axiom 11 (e.g., unitDay, unitMonth). The hasGap property of Axiom 12 supports unitType by enabling additional numeric values (e.g., "every three months" with hasGap: 3). The hasList property of Axiom 13 relates periodic entities to ListDescription, a subclass of UnitTimeDescription, defining specific days (e.g., "every Wednesday") or months (e.g., "every August").

Axiom 10: Defining Relationship between Time Description Types and TemporalUnit.
For any instances x and y, if there is a relationship unitType between x and y, then x must be of type UnitTimeDescription or GeneralDateTimeDescription, and y must be of type TemporalUnit.

$$\forall x, y\ (unitType(x, y) \rightarrow ((UnitTimeDescription(x) \vee GeneralDateTimeDescription(x)) \wedge TemporalUnit(y)))$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Axiom 11: Valid Units for TemporalUnit.
Any instance of TemporalUnit must be one of the valid units (unitYear, unitMonth, unitWeek, unitDay, unitHour, unitMinute, unitSecond).

$$\forall x \left(TemporalUnit(x) \rightarrow (x = unitYear \vee x = unitMonth \vee x = unitWeek \vee x = unitDay \vee x = unitHour \vee x = unitMinute \vee x = unitSecond)\right)$$

Axiom 12: The Valid Range of Description Types for Periodic Gap Property.
For any instances x and y, if there is a relationship hasGap between x and y, then x must be of type UnitTimeDescription, and y must be of type nonNegativeInteger.

$$\forall x, y\ (hasGap(x, y) \rightarrow (UnitTimeDescription(x) \wedge nonNegativeInteger(y)))$$

Axiom 13: Defining Relationship between UnitTimeDescription and ListDescription.
For any instances x and y, if there is a relationship hasList between x and y, then x must be of type UnitTimeDescription, and y must be of type ListDescription.

$$\forall x, y\ (hasList(x, y) \rightarrow (UnitTimeDescription(x) \wedge ListDescription(y)))$$

3.2.2 List Description

The ListDescription class, a subclass of UnitTimeDescription within our ontology, enhances UnitTimeDescription by associating the hasList property with specific calendar values. It manages periodic calendar descriptions related to PeriodicInstant, PeriodicInterval, and PeriodicException, providing explicit details not covered in Section 3.2.1. Among the key properties of ListDescription is dayOfWeek, an object property inherited from OWL-Time and described in Axiom 14 and Axiom 15, representing all possible days in a week. For instance, "every Wednesday" is specified using unitType: unitDay and hasList, linked to ListDescription with dayOfWeek: Wednesday. Additionally, ListDescription includes a data property called day (Axiom 16), representing days within each calendar month as non-negative integers from 1 to 31. For example, "every 19th" is specified using unitType: unitDay and hasList, linked to ListDescription with day: 19. The month property (Axiom 17) describes specific recurring months, using non-negative integers from 1 to 12. For example, "every August" is specified using unitType: unitMonth and hasList, linked to ListDescription with month: 08.

Axiom 14: Defining Relationship between DaysOfWeek and Description Types.
For any instances x and y, if there is a relationship dayOfWeek between x and y, then x must be of type ListDescription or GeneralDateTimeDescription, and y must be of type DayOfWeek.

$$\forall x, y \left(dayOfWeek(x, y) \rightarrow \left(\left(ListDescription(x) \vee GeneralDateTimeDescription(x)\right) \wedge DayOfWeek(y)\right)\right)$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Axiom 15: Valid Units for DayOfWeek.

Any instance of DayOfWeek must be one of the valid units (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday).

$$\forall x \left(DayOfWeek(x) \rightarrow (x = Monday \vee x = Tuesday \vee x = Wednesday \vee x = Thursday \vee x = Friday \vee x = Saturday \vee x = Sunday)\right)$$

Axiom 16: The Valid Range of Description Types for day Property.

For any instances x and y, if there is a relationship day between x and y, then x must be of type ListDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer from value 1 to 31.

$$\forall x, y \left(day(x, y) \rightarrow \left(\left(ListDescription(x) \vee GeneralDateTimeDescription(x)\right) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 31\right)\right)$$

Axiom 17: The Valid Range of Description Types for Months Property.

For any instances x and y, if there is a relationship month between x and y, then x must be of type ListDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer from value 1 to 12.

$$\forall x, y \, (month(x, y) \rightarrow ((ListDescription(x) \vee GeneralDateTimeDescription(x)) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 12))$$

3.2.3 Periodic Time Span Description

The PeriodicTimeSpanDescription class, a subclass of OWL-Time GeneralDateTimeDescription, provides explicit calendar descriptions for features associated with PeriodicTimeSpan and time span of PeriodicException, allowing detailed periodic date and time references such as "Wednesday of each month," "every Wednesday between 9 a.m. and 11 p.m.," and "every August on Wednesday at 21:00." This class includes components like defined start and end points using starts and ends properties, days of the week using the dayOfWeek property, months using the month property, and specific days of the week within a month using the dayOfWeekInMonth property. It also describes occurrences with a reference count using the hasCount property. For instance, "Three times every August" uses hasCount: 3 and month: 08. The formal specification of PeriodicTimeSpanDescription and its properties is detailed in Axioms 18 to 23.

Axiom 18: Defining starts Relationship between PeriodicTimeSpan and DateTimeLimitDescription.

For any x and y, if there is a starts property linking x to y, then x must be an instance of PeriodicTimeSpanDescription, and y must be an instance of DateTimeLimitDescription.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

$$\forall x, y\ (starts(x, y) \rightarrow (PeriodicTimeSpanDescription(x) \wedge DateTimeLimitDescription(y)))$$

Axiom 19: Defining ends Relationship between PeriodicTimeSpan and DateTimeLimitDescription.
For any x and y, if there is a ends property linking x to y, then x must be an instance of PeriodicTimeSpanDescription, and y must be an instance of DateTimeLimitDescription.

$$\forall x, y\ \big(ends(x, y) \rightarrow PeriodicTimeSpanDescription(x) \wedge DateTimeLimitDescription(y)\big)$$

Axiom 20: Defining dayOfWeek Relationship for Description Types and DayOfWeek.
For any instances x and y, if there is a relationship dayOfWeek between x and y, then x must be of type ListDescription, PeriodicTimeSpanDescription or GeneralDateTimeDescription, and y must be of type DayOfWeek.

$$\forall x, y\ \Big(dayOfWeek(x, y) \rightarrow \Big(\big(ListDescription(x) \vee PeriodicTimeSpanDescription(x) \vee GeneralDateTimeDescription(x)\big) \wedge DayOfWeek(y)\Big)\Big)$$

Axiom 21: The Valid Range for month Property on Description Types.
For any instances x and y, if there is a relationship month between x and y, then x must be of type ListDescription, PeriodicTimeSpanDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer from value 1 to 12.

$$\forall x, y\ \Big(month(x, y) \rightarrow \Big(\big(ListDescription(x) \vee PeriodicTimeSpanDescription(x) \vee GeneralDateTimeDescription(x)\big) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 12\Big)\Big)$$

Axiom 22: The Valid Range for dayOfWeekInMonth Property on PeriodicTimeSpanDescription.
For any instances x and y, if there is a relationship dayOfWeekInMonth between x and y, then x must be of type PeriodicTimeSpanDescription and y must be of type nonNegative integer from value 1 to 5.

$$\forall x, y\ (dayOfWeekInMonth(x, y) \rightarrow (PeriodicTimeSpanDescription(x) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 5))$$

Axiom 23: The Valid Range for hasCount Property on PeriodicTimeSpanDescription.
For any instances x and y, if there is a relationship hasCount between x and y, then x must be of type PeriodicTimeSpanDescription and y must be of type nonNegative integer.

$$\forall x, y\ (hasCount(x, y) \rightarrow (PeriodicTimeSpanDescription(x) \wedge nonNegativeInteger(y)))$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

3.2.4 Date Time Limit Description

Instances of PeriodicTimeSpan within PeriodicTimeSpanDescription detail the start and/or end of a time span. For example, "every Wednesday between 9:30 AM and 11:30 AM" is defined as a PeriodicTimeSpan from "9:30 AM to 11:30 AM." This association is managed by the starts and/or ends object properties of PeriodicTimeSpanDescription, as outlined in Section 3.2.3. To provide explicit representation, we introduced DateTimeLimitDescription. This subclass allows for precise granularity of time spans, representing starts or ends using six datatype properties: second, minute, hour, day, month, and year. These properties link individuals of the starts and ends object properties to their data values in strings or the XML dateTime format, enabling representation of instances like "09:30 AM," "10 PM," "21 hours," "90 minutes," "55 seconds," or dates such as "25th December." The formal specification presents the complete representation in first-order logic for the DateTimeLimitDescription class and its properties, as defined in Axioms 24 to 29.

Axiom 24: Defining Valid Range for second Property on Description Types.
For any instances x and y, if there is a relationship second between x and y, then x must be of type DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type decimal value.

$$\forall x, y \left(second(x, y) \rightarrow \left(\left(DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x) \right) \wedge decimal(y) \right) \right)$$

Axiom 25: Defining Valid Range for minute Property on Description Types.
For any instances x and y, if there is a relationship minute between x and y, then x must be of type DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer.

$$\forall x, y \left(minute(x, y) \rightarrow \left(\left(DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x) \right) \wedge nonNegativeInteger(y) \right) \right)$$

Axiom 26: Defining Valid Range for hour Property on Description Types.
For any instances x and y, if there is a relationship hour between x and y, then x must be of type DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer.

$$\forall x, y \left(hour(x, y) \rightarrow \left(\left(DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x) \right) \wedge nonNegativeInteger(y) \right) \right)$$

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Axiom 27: Defining Valid Range for day Property on Description Types in DateTimeLimitDescription.
For any instances x and y, if there is a relationship day between x and y, then x must be of type ListDescription, DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer from value 1 to 31.

$$\forall x, y\ (day(x, y) \rightarrow ((ListDescription(x) \vee DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x)) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 31))$$

Axiom 28: Defining Valid Range for month Property on Description Types in DateTimeLimitDescription.
For any instances x and y, if there is a relationship month between x and y, then x must be of type ListDescription, DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer from value 1 to 12.

$$\forall x, y\ \Big(month(x, y) \rightarrow \Big(\big(ListDescription(x) \vee DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x)\big) \wedge nonNegativeInteger(y) \wedge 1 \leq y \leq 12\Big)\Big)$$

Axiom 29: Defining Valid Range for year Property on Description Types in DateTimeLimitDescription.
For any instances x and y, if there is a relationship year between x and y, then x must be of type DateTimeLimitDescription or GeneralDateTimeDescription, and y must be of type nonNegative integer.

$$\forall x, y\ (year(x, y) \rightarrow ((DateTimeLimitDescription(x) \vee GeneralDateTimeDescription(x)) \wedge nonNegativeInteger(y)))$$

The concept of a periodic calendar can be illustrated by referring to examples presented in Examples 7 and 8. To further enrich the discussion of Example 7 (Figure 8), we will OWL-Time GeneralDateTimeDescription and OWL-BOLA 1.0 periodic calendar description (UnitTimeDescriotion, ListTimeDescription, PeriodicTimeSpanDescription and DateTimeLimitDescription) incorporated into Figure 10.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

```
1. <webinarSeries> rdf:type NegativeHalfInfiniteTemporalSeq;
2.       hasIntervalTime <untilDecember312024>;
3.       hasContexTime <everyMonday>;
4.       hasRecurrenceTime <every21Hour>;
5. <untilDecember312024> rdf:type Interval;
6.       hasEnd <december312024>;
7. <december312024> rdf: type GeneralDateTimeDescription;
8.        unitType “unitDay”;
9.       day “31”;
10.      month “12”;
11.      year “2024”;
12. <everyMonday> rdf: type PeriodicInstant;
13.      hasPeriod <periodMonday>;
14. <periodMonday> rdf: type UnitTimeDescription;
15.       unitType “unitDay”;
16.      hasList <periodListMonay>;
17. <periodListMonday> rdf: type ListDescription;
18.       dayOfWeek “Monday”;
19. <every21Hour> rdf: type PeriodicTimeSpan;
20.      hasSpan <period21Hour>;
21. <period21Hour> rdf: type PeriodicTimeSpanDescription;
22.      starts <21Hour>;
23. <21Hour> rdf: type DateLimitTimeDescription;
24.       hour “21”;
```

Figure 10. Detail representation of Example 7

Figure 10 expands on the insights discussed in Figure 8. In this context, we extended the instance of "december312024," classifying it under GeneralDateTimeDescription and detailing its attributes: "unitDay" via the unitType property, "31" via the day property, "12" via the month property, and "2024" via the year property. Additionally, we expanded the instance "periodMonday," which was of type UnitTimeDescription, representing its "unitDay" via the unitType property. This instance is intricately linked to "periodListMonday" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription. For "periodListMonday," we utilized the dayOfWeek property to specify it as "Monday." Subsequently, we expanded "period21Hour," which is of type PeriodicTimeSpanDescription. This connection extends to "21Hour" through the starts property. The "21Hour" is an instance of DateTimeLimitDescription, specifying the hour property as "21."

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

```
<yogaClass> rdf:type PositiveHalfInfiniteTemporalSeq;
      hasIntervalTime <startsJanuary012023>;
      hasContexTime <everyMonday>;
      hasRecurrenceTime <every10Hour>;
      hasExceptionTime <everyHolidayMonday14Hour>;
<startsJanuary012023> rdf:type Interval;
      hasBeginning <january012023>;
<january012023> rdf: type GeneralDateTimeDescription;
      unitType “unitDay”;
      day “01”;
      month “01”;
      year “2023”;
<everyMonday> rdf: type PeriodicInstant;
      hasPeriod <periodMonday>;
<periodMonday> rdf: type UnitTimeDescription;
       unitType “unitDay”;
      hasList <periodListMonay>;
<periodListMonday> rdf: type ListDescription;
       dayOfWeek “Monday”;
<every10Hour> rdf: type PeriodicTimeSpan;
      hasSpan <period10Hour>;
<period10Hour> rdf: type PeriodicTimeSpanDescription;
      starts <10Hour>;
<10Hour> rdf: type DateLimitTimeDescription;
       hour “10”;
<everyHolidayMonday14Hour> rdf: type PeriodicException;
      hasContexTime <everyHolidayMonday>;
      hasRecurrenceTime <every14Hour>;
<everyHolidayMonday> rdf: type PeriodicInstant;
      hasNonPeriod <periodMonday>;
<periodMonday> rdf: type UnitTimeDescription;
      unitType “unitDay”;
      hasList <periodListMonay>;
<periodListMonday> rdf: type ListDescription;
       dayOfWeek “Monday”;
<every14Hour> rdf: type PeriodicTimeSpan;
      hasSpan <period14Hour>;
<period14Hour> rdf: type PeriodicTimeSpanDescription;
      starts <14Hour>;
<14Hour> rdf: type DateLimitTimeDescription;
       hour “14”;
```

Figure 11. Detail representation of Example 8

The enrichment of Example 8 (Figure 9) is also demonstrated with the OWL-Time GeneralDateTimeDescription and OWL-BOLA 1.0 periodic calendar description (UnitTimeDescription, ListTimeDescription, PeriodicTimeSpanDescription, and DateTimeLimitDescription) incorporated into Figure 11.

Figure 11 expands on the insights discussed in Figure 9. In this context, we extended the instance of "january012023," classifying it under GeneralDateTimeDescription and detailing its attributes: "unitDay" via the unitType property, "01" via the day property, "01" via the month property, and "2023" via the year property. Additionally, we expanded the instance of the regular occurrence "periodMonday," which was of type UnitTimeDescription, representing its "unitDay" via the unitType property. This instance is intricately linked to "periodListMonday" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription. For "periodListMonday," we utilized the dayOfWeek property to specify it as "Monday."

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013
Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Subsequently, we expanded "period10Hour," which is of type PeriodicTimeSpanDescription. This connection extends to "10Hour" through the starts property. The "10Hour" is an instance of DateTimeLimitDescription, specifying the hour property as "10." Additionally, the representation of irregularities in "periodMonday" and "period14Hour" are achieved similarly. Here, "periodMonday" is categorized under UnitTimeDescription and specified as "unitDay" via the unitType property. Moreover, "periodMonday" intricately connects to "periodListMonday" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription where the dayOfWeek property specifies it as "Monday." Similarly, for "period14Hour," which is of type PeriodicTimeSpanDescription, the connection extends to "14Hour" through the starts property. The "14Hour" is an instance of DateTimeLimitDescription, specifying the hour property as "14."

4. Result and Discussion

This study presents a detailed evaluation and discussion of the ontology results. The ontology was developed and tested using Protégé editor version 5.6.3, on a system equipped with an Intel (R) Core i5-5200U CPU at 2.20 GHz and 8 GB of RAM, running Windows 8.1 OS. Our analysis examines the OWL-BOLA 1.0 ontology, focusing on its complexity and conceptualization richness within the Protégé editor. The quality of information was assessed based on the associated concepts and data types related to both simple and complex recurring time. This evaluation includes a comparative analysis with the base ontology and other recurring temporal ontologies, providing a comprehensive understanding.

4.1. Comparison of the system's ontology with others

The ontology developed in this research significantly enhances the performance of the OWL-Time framework, which is primarily focused on simple time representation. To evaluate the effectiveness and significance of the knowledge representation formalism employed in this study, we decided to compare the efficiency of the proposed ontology (OWL-BOLA 1.0) with other related ontologies used in studies on complex recurring time. This comparison is based on the following metrics:

i. Complexity Richness: This means how complicated or detailed the ontology's organization and parts are. The more complex an ontology is, the more intricate and comprehensive its representation of domain knowledge tends to be. Number of classes, Number of Object Properties, Number of Individuals, Depth of Class Hierarchy, Number of Axioms, Number of Logical Axioms, Number of Role Chains, Number of Equivalent Classes, Number of Disjoint Classes, Number of Inferred Subclass Relationships
ii. Conceptualization Richness: This focus on the depth and quality of the concepts represented in the ontology. They assess how well the ontology captures and organizes the domain knowledge. The metric aim to assess the richness of the ontology in terms of its semantic content, use of data properties, and inheritance structure.

Based on these metrics, the performance measurements in the following subsections are presented. Table 1 shows the derivation and description of the metrics used in computation of results in Table 2.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Table 1. A summary of the metrics and their respective counts in the OWL-BOLA 1.0 ontology

Metrics	Counts	Descriptions
Number of Classes	118	measures the total number of classes defined in the ontology
Number of Object Properties	55	measures total number of object properties defined in the ontology.
Number of Individuals	145	measures the total number of instances in the ontology
Depth of class hierarchy	6	measures the maximum length of the inheritance chain in the class hierarchy.
Number of axioms	540	measures the total number of axioms in the ontology, including subclass relationships, property assertions, and constraints.
Number of logical axioms	308	measures the number of related logical relationships and constraints, excluding annotation and metadata.
Number of role chains	10	Number of role chains measures the total number of sequences of object properties defined in the ontology.
Number of equivalent classes	6	measures the number of equivalent class axioms that define classes with the same meaning.
Number of disjoint classes	4	measures the number of disjointness axioms that state classes have no common instances.
Semantic	55/118	normalized measure that reflects not only the number of object properties but also how they are distributed across classes
Data Property	28/118	normalized measure that reflects not only the number of data properties but also how they are distributed across classes
Inheritance	108/118	measures the extent to which inheritance (i.e., subclass relationships) is utilized within an ontology

The results of Tables 2 show the richness of the proposed ontology alongside with related works. The performance of the OWL-BOLA 1.0 ontology, as earlier described in Table 1, was compared with a similar study extending the OWL-Time ontology. The results revealed that the proposed ontology outperformed the comparison to certain extent, as presented in Table 2, displaying a comprehensive evaluation of richness in complexity and conceptualization.

Kasu Journal of Computer Science Vol. 1 No. 2 [June, 2024], pp. 340-365
https://doi.org//10.47514/kjcs/2024.1.2.0013

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Table 2. A comparison of related ontologies used in temporal studies with the proposed ontology

Ontology Metrics												
	Complexity									Conceptualization		
Studies	Number of Class	Number of Object Properties	Number of Individual	Depth of Class Hierarchy	Number of Axioms	Number of Logical Axioms	Number of Role Chains	Number of Equivalent Classes	Number of disjoint Classes	Semantic	Data Property	Inheritance
OWL-Time	55	33	27	4	208	188	7	4	1	0.6	0.456	0.909
(Bento et al., 2017)	89	51	95	2	386	258	8	5	1	0/573	0.315	0.933
Proposed Ontology (OWL-BOLA 1.0)	118	55	145	6	540	308	10	6	4	0.466	0.237	0.915

The evaluation of the proposed ontology (OWL-BOLA 1.0) reveals significant insights into its richness in both complexity and conceptualization. The ontology demonstrates a high degree of comprehensiveness and inclusiveness, ensuring that essential components such as classes, properties, and hierarchical structures are well-defined. The depth of the class hierarchy, with a value of 6, surpasses that of both Bento et al. (2017) and OWL-Time, highlighting the richness and depth of knowledge representation within the domain. In terms of conceptualization richness, the proposed ontology balances semantic richness, data property richness, and inheritance richness effectively. While its semantic richness (0.466) and data property richness (0.237) are slightly lower compared to other ontologies, these values indicate a streamlined and clear representation of relationships and data properties, reducing redundancy. The high inheritance richness score (0.915) demonstrates robust utilization of hierarchical structures, ensuring a flexible framework for knowledge representation. Overall, the proposed ontology (OWL-BOLA 1.0) shows a balanced and effective approach to capturing domain knowledge, maintaining simplicity and clarity. This balance supports its usability and effectiveness in representing complex recurring time, ensuring essential concepts and relationships are clearly defined and accessible.

5. Conclusion

In this paper, we identified the challenges associated with the OWL-Time ontology in representing complex recurring time-related problems across diverse domains. To tackle these issues, we introduced the OWL-BOLA 1.0 ontology, providing first-order logic formalization with axioms and definitions. Our comprehensive solution features a taxonomy of intricate temporal concepts. The ontology extends OWL-Time primitive entities and calendar descriptions for recurring time components. The taxonomy encompasses recurring primitive entities and periodic calendar descriptions, offering a structured framework for expressing various components in recurring time expressions. Experimental results of the proposed ontology (OWL-BOLA 1.0) demonstrate its richness in both complexity and conceptualization. With a depth of class hierarchy of 6, it surpasses Bento et al. (2017) and OWL-Time, highlighting its comprehensive knowledge representation.

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Despite slightly lower semantic (0.466) and data property (0.237) richness compared to other ontologies, OWL-BOLA 1.0 effectively reduces redundancy and maintains clarity. Its high inheritance richness (0.915) showcases robust use of hierarchical structures. Overall, OWL-BOLA 1.0 balances complexity and simplicity, making it a usable and effective tool for representing complex recurring time, with clearly defined and accessible concepts and relationships. While our ontology presents a significant advancement, there are avenues for improvement and extension. Efforts can be directed towards enhancing semantic richness by incorporating additional object properties and refining existing ones, thereby capturing more intricate relationships within the domain. Further refinement of data properties can strike a balance between simplicity and comprehensiveness, enhancing clarity and usability. Extending the ontology to cover more aspects of the domain, including additional classes, properties, and axioms relevant to complex recurring time events, can enrich its knowledge base. Conducting extensive validation and testing with real-world data and collaborating with domain experts can ensure accuracy and applicability. Additionally, exploring the integration of OWL-BOLA 1.0 with other related ontologies can enhance interoperability and broaden its application. Addressing these areas can further improve the ontology, providing a more comprehensive and effective framework for representing complex recurring time within the domain.

References

Achich, N., Ghorbel, F., Hamdi, F., Metais, E., & Gargouri, F. (2021). *Dealing with Uncertain and Imprecise Time Intervals in OWL2: A Possibility Theory-Based Approach* (S. Cherfi, A. Perini, & S. Nurcan, Eds.; Vol. 415). Springer International Publishing. https://doi.org/10.1007/978-3-030-75018-3

Almendros-Jiménez, J. M., & Becerra-Terón, A. (2021). Discovery and diagnosis of wrong SPARQL queries with ontology and constraint reasoning. *Expert Systems with Applications*, *165*, 113772. https://doi.org/10.1016/j.eswa.2020.113772

Bento, A. M., Viegas, D. C., & Correia, N. (2017). Temporal Reasoning with Non-convex Intervals. In P. Różewski & C. Lange (Eds.), *Knowledge Engineering and Semantic Web* (Vol. 786, pp. 127–142). Springer International Publishing. https://doi.org/10.1007/978-3-319-69548-8_10

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). *Semantic Web* (pp. 34–43). Scientic American.

Deborah, L., & Frank, V. H. (2004). *OWL Web Ontology Language Overview*. w3.org/TR/2004/REC-owl-features-20040210/

Faucher, C., Lafaye, J.-Y., & Bertrand, F. (2012). *Modelling composite periodic Events*. 15.

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Henry, S. T., Noah, M., David, B., Murray, M., Shudi, G., & Sperberg-McQueen, C. M. (2009). *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. https://www.w3.org/TR/xmlschema11-1/

Hobbs, J. R., & Pan, F. (2004). An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing*, *3*(1), 66–85. https://doi.org/10.1145/1017068.1017073

Hobbs, J. R., Pan, F., Cox, S., & Little, C. (Eds.). (2017). *Time Ontology in OWL*. W3C working draft, W3C. https://www.w3.org/TR/owl-time/

Hu, D., Wang, M., Gao, F., Xu, F., & Gu, J. (2022). Knowledge Representation and Reasoning for Complex Time Expression in Clinical Text. Data Intelligence, 4(3), 573–598. https://doi.org/10.1162/dint_a_00152

Krieger, H.-U. (2008). Where Temporal Description Logics Fail: Representing Temporally-Changing Relationships. In A. R. Dengel, K. Berns, T. M. Breuel, F. Bomarius, & T. R. Roth-Berghofer (Eds.), *KI 2008: Advances in Artificial Intelligence* (Vol. 5243, pp. 249–257). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-85845-4_31

Li, F., Du, J., He, Y., Song, H.-Y., Madkour, M., Rao, G., Xiang, Y., Luo, Y., Chen, H. W., Liu, S., Wang, L., Liu, H., Xu, H., & Tao, C. (2020). Time event ontology (TEO): To support semantic representation and reasoning of complex temporal relations of clinical events. *Journal of the American Medical Informatics Association*, *27*(7), 1046–1056. https://doi.org/10.1093/jamia/ocaa058

Logananthara, R., & Gombrone, S. (1995). Representation of, and reasoning with, near-periodic recurrent events. *9th IJCAI Workshop on Spatial and Temporal Reasoning.*, 1–6.

O'Connor, M. J., & Das, A. K. (2011). A Method for Representing and Querying Temporal Information in OWL. In A. Fred, J. Filipe, & H. Gamboa (Eds.), *Biomedical Engineering Systems and Technologies* (Vol. 127, pp. 97–110). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-18472-7_8

Pan, F. (2007). *An Ontology of Temporal Concepts for the Semantic Web and Natural Language*. 66.

Poveda, V. M., Suárez, F. M. C., & Gómez, P. A. (2014). *A Pattern for Periodic Intervals*. 1–10.

Robert, A., Genest, D., & Loiseau, S. (2020). Temporal Cognitive Maps. In ICAART (2) (pp. 58-68).

Terenziani, P. (2016). Irregular Indeterminate Repeated Facts in Temporal Relational Databases. *IEEE Transactions on Knowledge and Data Engineering*, *28*(4), 1075–1079. https://doi.org/10.1109/TKDE.2015.2509976

Online ISSN: 1597-2216
Print ISSN: 1597-2178
www.kjcs.edu.ng

Tuzhilin, A., & Clifford. (1995). On periodicity in temporal databases. *Information System*, *20*(8), 619–639.

W3C. (2002). *Semantic Web*. https://www.w3.org/standards/semanticweb/

Wang, Chang, L., Zhu, C., & Dong, R. (2012). Reasoning about Semantic Web Services with an Approach Based on Temporal Description Logic. In Z. Shi, D. Leake, & S. Vadera (Eds.), *Intelligent Information Processing VI* (Vol. 385, pp. 286–294). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-32891-6_36

Wang, H.-T., Department of Computer Science, The Graduate Center, The City University of New York, USA, Tansel, A. U., Department of Computer Science, The Graduate Center, The City University of New York, USA, Tansel, A. U., Paul H. Chook Department of Information Systems and Statistics, Baruch College, The City University of New York, USA, Tansel, A. U., & School of Engineering, Thammasat University Thailand. (2019). Temporal Extensions to RDF. *Journal of Web Engineering*, *18*(1), 125–168. https://doi.org/10.13052/jwe1540-9589.18134

Zekri, A., Brahmia, Z., Grandi, F., & Bouaziz, R. (2014). *τOWL: A Framework for Managing Temporal Semantic Web Documents*.