Engineering a formal representation of complex temporal Information using the Web Ontology Language

Yusuf Aminat Bolatito [a*], Junaidu Sahalu Balarabe [b], Obiniyi Afolayan Ayodele [b], Aliyu Salisu [b], Kana Armand Florentin Donfack [b] and Oyelade Olaide Nathaniel [b]

[a] *Usmanu Danfodiyo University Sokoto, Sokoto Nigeria*
E-mail: aminat.bola@udusok.edu.ng, aminbolly@gmail.com
[b] *Ahmadu Bello University Zaria, Kaduna Nigeria*

Editors: First Editor, University or Company name, Country; Second Editor, University or Company name, Country

Solicited reviews: First Solicited Reviewer, University or Company name, Country; Second Solicited Reviewer, University or Company name, Country

Open reviews: First Open Reviewer, University or Company name, Country; Second Open Reviewer, University or Company name, Country

Abstract.

Ontologies, a common approach for knowledge representation, encounter challenges in effectively handling time expressions, particularly in languages like OWL. Despite various proposed mechanisms for achieving consistent time representation within a single ontology, OWL-Time constructs have limitations in modelling complex recurring time expressions, crucial for domains such as clinical or historical. Hence, extending the OWL framework to represent recurring temporal information identified in domain-specific challenges is necessary. This paper enriches OWL-Time with the concept of recurring time by identifying five recurring time-related problems namely strongly, nearly, intermittent, non-recurring, and stochastic recurring time, covering the spectrum in classical ontologies. We propose BOLA as a novel ontology, extending OWL-Time's primitive entities, calendar descriptions, and introducing a new concept for recurring time rules. This extension creates 33 core temporal classes, offering a comprehensive representation within OWL-Time constructs. We evaluate the ontology's completeness, expressiveness, and complexity. Experimental results show that BOLA effectively represents a wide range of complex recurring temporal information within OWL-Time constructs, achieving completeness scores of 87 and 88 for class and property coverage hierarchy, expressiveness score of 13 for complex expressions with 49 cardinality restrictions, and a complexity reflection of 266 axioms, including 123 logical axioms.

Keywords: Ontology, OWL-Time, Recurring Time, Primitive Entities

1. Introduction

The historical progression of computer science, particularly in areas like Artificial Intelligence, databases, and distributed systems, has driven a continual quest for modeling concepts and structures conducive to the seamless integration of time [1], [2]. Within the logical theoretical frameworks addressing time models, two primary dimensions have emerged: the representation or reasoning of time units and entities [3], [4], [5], [6]. The realm of time units encompasses fundamental elements such as time points, moments, and intervals, each demanding careful attention to nomenclature and properties. Conversely, time entities involve entities whose truth values evolve, unraveling the temporal order and sequence of changes, including states, properties, events, and processes [6], [7]. The categorization into simple time, as elucidated by [8], distinguishes between the basic theory of time (time units) and the theory of temporal incidence (time entities). As we navigate this temporal terrain, the need arises to explore complex time—those involving multiple instances of an event. Complex entities in the temporal landscape involve time multi-granularities [9], [10], recurring time patterns [11], [12], [13], and imprecise time [14], [15]. Logic has played a pivotal role in capturing the essence of complex recurring time, with notable frameworks by [16], [17], [18] addressing planning and scheduling on recurring times and [11], [12], [13], [19] focusing on user-specified patterns in calendar dates on recurring time in database management system.

Various research contributions offer valuable insights into the intricate nature of recurring time phenomena. For instance, [13] contributes temporal constraints tailored for repeated actions or periods, classifying recurring time into regular/periodic, irregular/aperiodic, and further differentiating between strongly periodic, nearly aperiodic, and intermittent aperiodic. Examples such as patient A's medical appointment recurring on the first Tuesday of each month (strongly periodic), or cancer medication administered once every week (nearly aperiodic), highlight the complexity inherent in such temporal classifications. However, the study focuses on the logical frameworks of the first two categories, strongly-periodic and nearly aperiodic, while intermittently aperiodic occurrence was not addressed. [20] addresses the gap by extending Tuzhilin and Clifford's work, presenting a model to represent intermittent aperiodic repetitions through temporal relational algebra. Examples such as patient B had seven medical appointments scheduled in 2023 (intermittent aperiodic). On the other hand, [21] offers another perspective on recurring aperiodic time, grouping its occurrences into nearly aperiodic, random aperiodic, and stochastic aperiodic. The work explores scenarios where events time gaps i.e., temporal intervals lack regularity (random aperiodic) or occur at regular temporal intervals but with associated probabilities or conditions for starting time or exceptions (stochastic aperiodic). The meticulous classification of aperiodic recurring time extends further in the work of [22], who identified non-recurring aperiodic events time. This was defined as composite events with shared temporal intervals, where each recurrence follows the same pattern within a specified time frame but exhibits different time spans between successive recurrences, as exemplified by a swimming competition held on Saturday from 2 PM to 6 PM and on Sunday from 9 AM to 12 PM.

The focus of [12], [13], [21], [22] on recurring time, where time elapses between instances of events, aligns with user-defined periodicity in calendar time representation. In this paper, we strive to contribute similar knowledge to one of the dynamic ontology languages, specifically the OWL-Time. This offers concepts and relations that comprehensively describe complex recurring time, covering all categories of user-defined periodicity found in classical counterparts. Over the years, dynamic ontologies offered temporal representation and reasoning that has emerged in the field of Semantic Web technologies, thanks to a sound theoretical framework in the classical domain [23], [24]. Dynamic ontologies provide a formal and widely accepted conceptualization for representing time in various domains. They aim to create a human-readable, machine-readable, and interoperable temporal information model, enhancing semantic knowledge representation. Incorporating Semantic Web specifications in areas like clinical narratives facilitates temporal reasoning and supports time-oriented queries [25]. Despite advancements in formalizing complex recurring time at different levels within classical domains, there is a persistent need for effective recurring time modelling mechanisms in the Semantic Web [26]. While dynamic ontologies have successfully addressed simple time representation needs, challenges arise in domains like medicine, where complex recurring time data requires more sophisticated modelling beyond the capabilities of existing dynamic ontologies [27], [28].

Some well-known dynamic ontologies include Web Ontology Language Time (OWL-Time) [29], SWRL Temporal Ontology[30], Reusable Time Ontology [31], TimeLine Ontology[32] and SOWL Ontology[33]. Temporal concepts presented by ontology OWL-Time, provides rich descriptions of temporal intervals, points, durations, and calendar terms. In October 2017, it became a W3C recommendation as part of general domain ontology[34]. OWL-Time, while covering temporal expressions and relations to a remarkable degree, falls short in handling the complexity of

recurring time[26], [28], necessitating an extension to adapt to strongly periodic, nearly aperiodic, non-recurring aperiodic, intermittent aperiodic, and stochastic aperiodic occurrences. For this reason, we propose the BOLA ontology, building upon the OWL-Time ontology to represent complex recurring time information. The key contributions of this paper can be summarized as follows:

1. The proposed BOLA ontology introduces 33 core time classes, providing a comprehensive and robust modelling of recurring time concepts and relations within OWL-Time constructs. These classes cover all categories of user-defined periodicity, addressing strongly-periodic, nearly aperiodic, non-recurring aperiodic, intermittent aperiodic, and stochastic aperiodic occurrences, as found in classical counterparts mentioned in earlier works[12], [13], [21], [22] .
2. We systematically defined recurring time-related classes and their relationships, presenting strongly-periodic, nearly aperiodic, non-recurring aperiodic, intermittent aperiodic, and stochastic aperiodic occurrences illustrations through clear diagrams.
3. In validating our ontology, we empirically assessed its effectiveness with real-world datasets, considering ontology metrics like completeness, complexity, and expressivity.

2. Related Work

The incorporation of temporal representation into dynamic ontologies and the semantic web have been the subject of many discussions. This study examines several dynamic ontologies that fall under the category of general domain ontologies. It also reviews several attempts made in the literature to enrich these ontologies with complex time, particularly recurring time.

1.1. General domain temporal ontologies

The Reusable Time ontology developed by [31] serves as a foundational theory, conceptualizing time as a continuous and linear entity with an infinite timeline denoted by Infinite-Past and Infinite-Future concepts. It introduces "point zero" for partitioning time and accommodates various interval types—open, half-open, closed, convex, or non-convex. While addressing recurring time through the Regular-Non-ConvexTime-Interval subclass, it primarily focuses on granularity for time points, assuming uncertainty in their precise location within an interval. Despite its strengths, Reusable Time falls short in capturing all aspects of time-related concepts, particularly those related to the fuzziness of time, offering a solid foundation for linear time representation but lacking in handling fuzzy time expressions comprehensively.

OWL-Time[29], formerly DAML ontology time[35], utilizes first-order predicate calculus to address topological temporal relations, encompassing intervals, events, dates, and times. In OWL representation, it introduces vocabularies for various temporal elements like instants, intervals, durations, date-time, and Allen's time relations. Centered around the TemporalEntity concept, with subclasses Instant and Interval, OWL-Time excels in precise timestamp-based time representation, focusing on simple time expressions. While effective for definitive points and intervals, it lacks the versatility to handle complex time expressions like recurring time, fuzzy time, and multi-granularity representation.

Performance Simulation Initiative (PSI) Time ontology [36] addresses medium complexity in time, offering diverse perspectives. It models time as open-ended intervals with a linear, discrete, anisotropic, relativist, and absolutist stance on relationships. Key features include point-to-point, point-to-interval, and interval-to-interval relationships, covering durations, recurring time, and interval phases. Despite its comprehensive coverage, it lacks representation for the fuzziness of time.

Clinical Narrative Temporal Relations Ontology (CNTRO) introduced by [25], addresses the semantic gap in clinical temporal modeling. It enriches clinical narratives with vocabularies like Event, Time, Duration, Granularity, Precision, and TemporalRelation Statement. However, CNTRO encounters challenges in achieving simplicity for effective representation of complex temporal relationships despite its valuable contributions.

SWRL-Temporal[30] extends OWL for interval-based information. It introduces class definitions for temporal concepts, like granularity and durations, along with SWRL build-ins for temporal reasoning. However, these build-ins operate on specific dateTime values, limiting the support for qualitatively defined intervals in SWRL-Temporal.

1.2. Deepening domain temporal ontologies

Time-Entry ontology [37] offers a unique solution to temporal representation challenges. It introduces a parallel construct, "time-entry," alongside OWL-Time's entities, featuring TemporalSeq to handle recurring time instances, an area where OWL-Time has limitations. Pan enhances temporal representation by introducing two calendar descriptions for TemporalSeq. While effective for periodic entities, the method has limitations, notably in handling aperiodic concepts like stochastic and non-recurring aperiodic time. Despite this, Pan's approach stands out for its clarity and precision in defining properties and concepts for diverse periodic entities.

SOWL ontology[38] addresses issues observed in previous approaches like TOWL [39], emphasizing qualitative interval support. SOWL utilizes dateTime datatype for date representation, employing SWRL rules for temporal reasoning and defining temporal concepts based on OWL-Time ontology. The model employs 4D fluents, utilizing the TimeSlice class and tsTimeSlice properties to represent dynamic objects.

The ontology time by [40] introduces a novel method in the OWL-Time framework, emphasizing the representation of non-convex intervals through the PeriodicInterval class. This class encapsulates key elements, including the interval's inception, conclusion, subinterval durations, period duration, and subinterval count. Poveda et al. effectively utilize OWL-Time properties like hasBeginning, hasEnd, DateTimeDescription, and DurationDescription to model initiation and culmination. However, the ontology primarily addresses periodic entities, neglecting aperiodic representations, and lacks formal reasoning, impacting precision and clarity in temporal representation.

NCO temporal ontology[22] extends the OWL-Time ontology to address temporal representation challenges in ontologies. Their dual-pronged approach introduces the concept of NCInterval within TemporalEntities and a calendar description framework extending GeneralDateTimeDescription to create GeneralDateTimeLimit. This solution adeptly handles both periodic and aperiodic instances of non-recurring time. However, it may overlook temporal aspects like stochastic recurring time, and the absence of formal reasoning could be enhanced for precision and clarity in temporal representation.

Temporal Event Ontology (TEO) by [26] extended the CNTRO. This was to address the limitations of general-purpose ontologies like OWL-Time when dealing with domain-specific requirements, particularly in healthcare contexts. To bridge this gap, TEO handles complex temporal aspects, including periodic recurring time. Their method involved extending the fundamental entity TimeInterval from [25] ontology by introducing a novel concept known as PeriodicInterval. Similar to CNTRO, the extension enriches OWL with additional features tailored to meet the specific requirements of their respective domain but not directly extending OWL-Time. However, TEO is still insufficient in terms of representing and reasoning on fuzzy time, irregular time, and complex relationships.

Clinical Time Ontology (CTO) by [28] extends TEO to overcome its limitations in handling temporal data in the clinical domain. It specifically addresses challenges related to uncertain and aperiodic time, emphasizing nearly aperiodic and stochastic aperiodic scenarios with qualitative properties. The extension introduces a specialized class, ClinicalTime, to handle possible time discrepancies. While enriching OWL with domain-specific features, it does not directly extend OWL-Time.

3. Recurring time expressions

OWL-Time adeptly represents simple time expressions like instants and gapless intervals, utilizing constructs such as Instant, Interval, GeneralDateTimeDescription and so on. However, handling complex recurring time requires departing from the simplicity of basic expressions. Developing a specialized semantic framework for recurring time is essential, as highlighted in earlier studies [12], [13], [21], [22]. This initiative represents a crucial advancement towards enhancing representation and reasoning in the semantic web, with a specific focus on optimizing the capabilities of the OWL-Time ontology.

In the upcoming sections, we systematically categorized expressions related to recurring time, based on predefined categories from existing literature and shown in Table 1. This structured framework provides clarity and understanding of recurring time expressions. We analyzed each expression, categorizing its components, and introduced key recurring time concepts to Table 2. The expressions in Table 1 utilize extensions of OWL-Time primitive entities (recurring primitive entities) and periodic rules, along with extensions of OWL-Time calendar descriptions (periodic calendar description), which are presented in syntactic and semantic dimensions in Table 2 to express decomposed recurring time expressions. The term "syntactic" refers to the structural aspects of each recurring time concept root needed for modeling recurring time expression, while "semantic" denotes the entities,

which are subclasses of the syntactic concept explicitly, used to express each decomposed expression within the recurring primitive entities, periodic rule, and periodic calendar description, respectively.

3.1. Categorization of recurring time expressions

Referring to [12], [13], [21], [22] for initial classification of recurring temporal expressions, we refined the categorization based on observations from an anonymized real-world dataset of 503 records [41] and examples from [22]. In summary, we identified and presented five problems, detailed in Table 1.

Table 1: Types of Recurring Time Problems

TYPE	CATEGORY	RECURRING EXPRESSION
1	Strongly Periodic	Every Monday to Saturday, between 10 a.m. and 8 p.m.
2	Nearly Aperiodic	Three times on every week
3	Intermittent Aperiodic	Five times in 2023
4	Non-Recurring Aperiodic	Every April 25, between 15 p.m. and 18 p.m., and every April 26, between 9 a.m. and 11 a.m.
5	Stochastic Aperiodic	Every Friday to Monday Closed on December 20.

We have categorized time descriptions in the dataset based on recurring temporal characteristics, covering both regular (periodic) and irregular (aperiodic) occurrences. Periodic time involves events with regular repetitions, including strongly-periodic time. In contrast, aperiodic time describes events with irregular cycle times, classified into nearly, intermittent, non-recurring, and stochastic aperiodic time.

Nearly aperiodic time features recurrent events at regular intervals, but the intervals are not necessarily equal. Intermittent aperiodic time entails events recurring without a discernible pattern. Non-recurring aperiodic time involves composite event occurrences with consistent temporal distances, but the time spans are regular and distinct. Stochastic aperiodic time follows regular intervals with associated probabilities or conditions governing starting times or exclusion of certain periods.

3.2. Recurring time primitive type

Table 2 outlines recurring time expressions, handled through a synthesis of extended OWL-Time primitive concepts. These expressions fall under the syntax of the recurring primitive TemporalSeq, an extension of OWL-Time Interval entities, representing non-convex intervals for repeating instances. Semantically, when associated with TemporalSeq, these expressions manifest as collections of temporal concepts like PeriodicInstant, PeriodicInterval, PeriodicTimeSpan, or PeriodicException. PeriodicInstant represents recurring entities occurring instantaneously, while PeriodicInterval extends this idea to recurring time periods with duration. PeriodicException captures instances where a recurring event deviates from its regular schedule due to exceptional conditions. PeriodicTimeSpan represents sets of periodic date-time references or reference counts, describing PeriodicInstant, PeriodicInterval, or PeriodicException.

For clarity, each expression in Table 1 is broken down into components and categorized based on the primitive type, detailed in Table 2. This organization simplifies the syntactic and semantic aspects of temporal concepts, showcasing the feasibility of representing non-recurring instances similarly to composite strongly periodic expressions.

3.3. Recurring time periodic rule type

In formulating OWL expressions for entities in Table 1, we introduced relations to guide the understanding of recurrence time expressions. Utilizing periodic rules based on [41], we presented syntactic and semantic dimensions in Table 2. The term "syntactic" refers to the structural aspects, while "semantic" denotes the meaning of entities within the ontology. Recurring time expressions aligned with the PeriodicRule and TemporalSeq class, serving as root elements defining periodicity. A PeriodicRule aggregates elements, each representing a simple periodic phenomenon. Semantically, it can manifest as RecurrenceTime, ContexTime, or ExceptionTime. In common

parlance, ContexTime corresponds to the periodic temporal distance pattern of either PeriodicInterval or PeriodicEntity such as a day of the week (e.g., each Wednesday), week (e.g., every week, each two months), month in a year (e.g., every September, every month), or year (e.g., each year). RecurrenceTime indicates the number of occurrences in a specified span of PeriodicTimeSpan or PeriodicException. ExceptionTime signifies periodic time exclusion pattern within the ContexTime, that is when occurrence does not happen, exemplified in expressions like 'every month except April.'

Table 2: Analysis of recurring time instances into distinct conceptual components.

EXPRESSIONS		PRIMITIVE CONCEPTS		PERIODIC RULE CONCEPTS		PERIODIC CALENDAR DESCRIPTION	
TYPE	COMPOSITION	SYNTACTIC	SEMANTIC	SYNTACTIC	SEMANTIC	SYNTACTIC	SEMANTIC
1	Monday to Saturday	Temporal Seq	PeriodicInterval	Periodic Rule	ContexTime	GeneralDateTime Description	UnitTimeDescription
	10 a.m. to 8 p.m.		PeriodicTimeSpan		Recurrence Time		PeriodicTimeSpan Description
2	Three times	Temporal Seq	PeriodicTimeSpan	Periodic Rule	Recurrence Time	GeneralDateTime Description	PeriodicTimeSpan Description
	Week		PeriodicInstant		ContexTime		UnitTimeDescription
3	Five times	Temporal Seq	PeriodicTimeSpan	Periodic Rule	Recurrence Time	GeneralDateTime Description	PeriodicTimeSpan Description
	2023		Instant		Instant		GeneralDateTime Description
4	April 25	Temporal Seq	PeriodicInstant	Periodic Rule	ContexTime	GeneralDateTime Description	UnitTimeDescription
	15 p.m. to 18 p.m.		PeriodicTimeSpan		Recurrence Time		PeriodicTimeSpan Description
	April 26	Temporal Seq	PeriodicInstant	Periodic Rule	ContexTime	GeneralDateTime Description	UnitTimeDescription
	9 a.m. to 12 a.m.		PeriodicTimeSpan		Recurrence Time		PeriodicTimeSpan Description
5	Friday to Monday	Temporal Seq	PeriodicInterval	Periodic Rule	ContexTime	GeneralDateTime Description	UnitTimeDescription
	December 20		PeriodicException		ContexTime		UnitTimeDescription

Furthermore, we categorized components based on issues addressed in Table 1, organizing syntactic and semantic aspects of temporal rule concepts in Table 2. This simplification aims at periodic rule to imbue meaning into recurring time expressions.

3.4. Recurring time calendar description type

The expressions in Table 1, depicting recurring time scenarios, can be conveyed through syntactic structures and semantic concepts within the extended OWL-Time calendar description. Syntactically, these expressions align

with OWL-Time's GeneralDateTimeDescription, a versatile class accommodating diverse date and time descriptions. Semantically, recurring time expressions are associated with GeneralDateTimeDescription, expanded by two crucial subclasses: UnitTimeDescription and PeriodicTimeSpanDescription. UnitTimeDescription handles periodic calendar descriptions for PeriodicInstant, PeriodicInterval, and PeriodicException, particularly in the ContexTime rule. PeriodicTimeSpanDescription plays a vital role in providing explicit calendar descriptions for features related to PeriodicTimeSpan (in the RecurrenceTime rule) and the time span of PeriodicException (in the RecurrenceTime Rule of ExceptionTime).

UnitTimeDescription and PeriodicTimeSpanDescription explicitly represent instances of recurring time expressions, offering a comprehensive range of date and time descriptions. These instances are categorized based on addressed issues, detailed in Table 1. Table 2 organizes syntactic and semantic aspects of recurring calendar description concepts and relations, correlating them with temporal expression types in Table 1. Through these classes, we aimed to enhance the articulation of recurring time instances in OWL expressions.

4. Proposed ontology

We developed the BOLA ontology for semantic representation and reasoning over the temporal aspects discussed in Section 3. In this section, we elaborated on its module design, which entails the discussion of the recurring primitive entities, periodic rule, and recurring calendar descriptions.

4.1. Overview of recurring primitive entities

The main primitive entities in OWL-Time are Instant and Interval entities, but recurring time instances bound to these entities require extension for non-convex intervals. We introduced TemporalSeq as an extension, with subclasses like PeriodicEntity (PeriodicInstant and PeriodicInterval), PeriodicTimeSpan, and PeriodicException to capture various compositions seen in recurring expressions. This classification accommodates the complexity of recurring time in natural language. Recurrence concepts, being aggregated phenomena, are defined as a collection in our ontology.

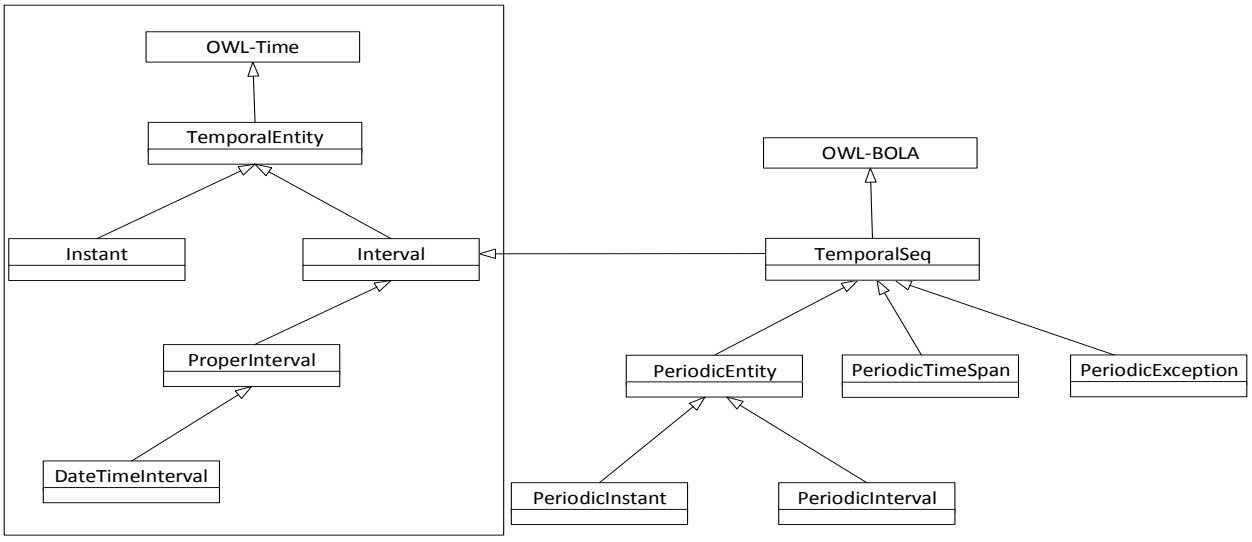

Figure 1: Taxonomy of BOLA Ontology

We derived the first subset of the 33 core classes, as seen in Figure 1, consisting of 11 classes, including TemporalEntity, Instant, Interval, ProperInterval, DateTimeInterval, TemporalSeq, PeriodicEntity, PeriodicInstant, PeriodicInterval, PeriodicTimeSpan, and PeriodicException, among others. Our ontology builds on OWL-Time, allowing some concepts to be reused and considered as part of our core concept. Specifically, we introduced TemporalSeq and connected it to the Interval class through the

hasIntervalTime property. Figure 1 illustrates the taxonomy of BOLA and its relationship with the OWL-Time ontology. The four detailed modules of TemporalSeq are presented in the following sections.

4.1.1. Periodic instant

The PeriodicInstant class, a subclass of PeriodicEntity, is designed to represent sets of recurrent instantaneous temporal entities, including occurrences on specific days ("every April 18"), days of the week ("every Friday"), weeks ("every week"), months in a year ("every June"), individual months ("every month"), or years ("every year"). This class enables the comprehensive depiction of regular instantaneous events with uniform time intervals, accommodating any irregularities within the same recurring event. To represent collections of instantaneous cyclic time, we introduced two object properties: hasPeriod, representing the regular cycle of a periodic instant, and hasNonPeriod, denoting irregular cycles. These properties connect instances of periodic instant with the UnitTimeDescription class, an extension of OWL-Time's GeneralDateTimeDescription. Details on UnitTimeDescription and its supporting class ListDescription will be discussed in section 4.3.

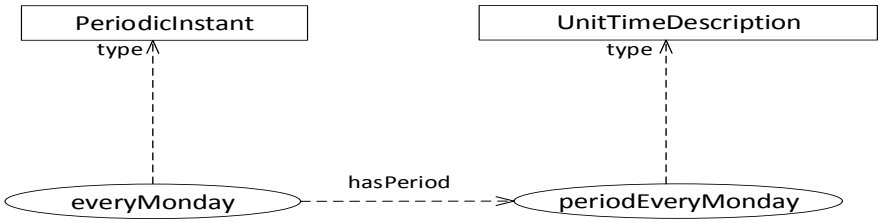

Figure 2: Mapping of the Individual to Periodic Instant

Illustrating with Example 1, “the orthopedic clinic holds every Monday,” Figure 2 depicts only its recurring temporal pattern. An instance named "everyMonday" is created, falling under the type PeriodicInstant. This instance is linked to another instance, "periodEveryMonday," through the hasPeriod object property, defined as belonging to the type UnitTimeDescription. This connection solidifies the recurring nature of the temporal pattern, with “periodEveryMonday” acting as a descriptor for the unit of time associated with each recurrence.

4.1.2. Periodic interval

The PeriodicInterval, a subclass of PeriodicEntity, extends the concept of PeriodicInstant to define recurring time spans with specified durations within a non-convex interval (TemporalSeq). PeriodicIntervals, like intervals “every April 18 to April 20” or “every Monday to Saturday”, require the identification of periodic instants at the beginning and end roles for accurate reasoning about start and end points. This is ensured by associating the PeriodicInstant instances through the object properties hasLowerLimit and hasUpperLimit, belonging to the same class, PeriodicInterval. These properties enable precise specifications and facilitate the use of associated properties within UnitTimeDescription and ListDescription.

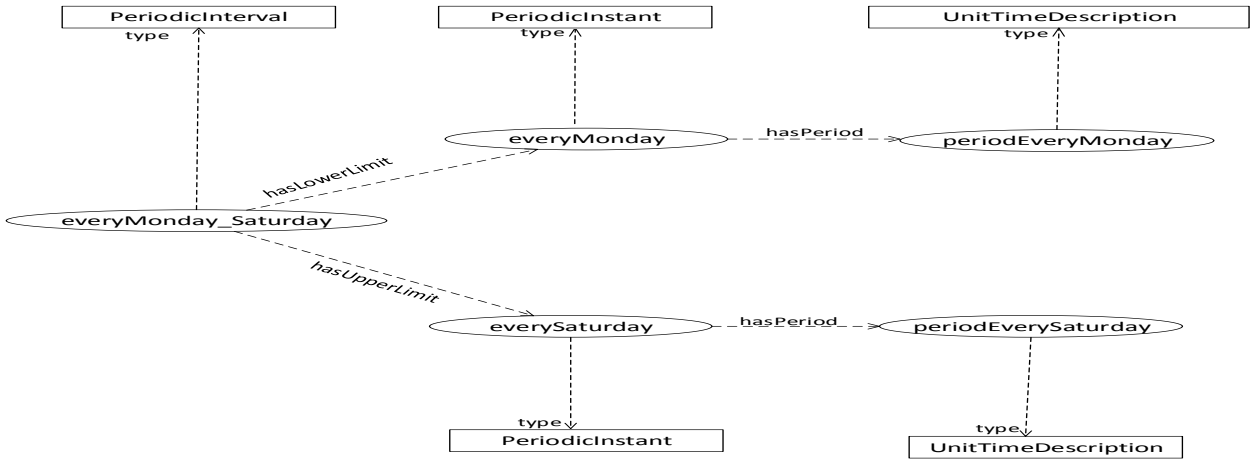

Figure 3: Mapping of the individual of PeriodicInterval entity to instances of PeriodicInstant and UnitTimeDescription

Illustrating with Example 2: "medication and check-up events in clinical settings every Monday to Saturday," showcased in Figure 3, the process of representing this recurring temporal pattern using PeriodicInterval is elucidated. We create an instance named "everyMonday_Saturday," a type of PeriodicInterval, intricately connected to individual instances "everyMonday" and "everySaturday" through the object properties hasLowerLimit and hasUpperLimit. Both "everyMonday" and "everySaturday" are instances of PeriodicInstant, establishing a representation for each day within the recurring interval. To emphasize the recurring nature, each instance is linked to "periodEveryMonday" and "periodEverySaturday" through the object property hasPeriod, forming a periodic interval time description in Figure 3. Further discussions on UnitTimeDescription, implicit at this stage, will be explored in future sections.

4.1.3. Periodic time span

Instances of PeriodicInstant and PeriodicInterval can be enriched with supplementary information for a more comprehensive depiction of periodicity. This additional information is captured by the class PeriodicTimeSpan, which includes details like periodic date-time references (e.g., "between 10 a.m. and 8 p.m." in "Every Monday to Saturday, between 10 a.m. and 8 p.m.") and reference counts (e.g., "three times" in "Three times every week," "first Wednesday" in "first Wednesday of every month"). PeriodicTimeSpan introduces complexity to the semantic breakdown of recurring time. Importantly, PeriodicTimeSpan is distinct from PeriodicEntity and is linked through the hasPeriodSpan property. However, it does not directly define additional details. Instead, the object property hasSpan is introduced to associate these details with the periodic calendar description. The explicit representation of these details is discussed in Section 4.3 which is focused on f PeriodicTimeSpanDescription.

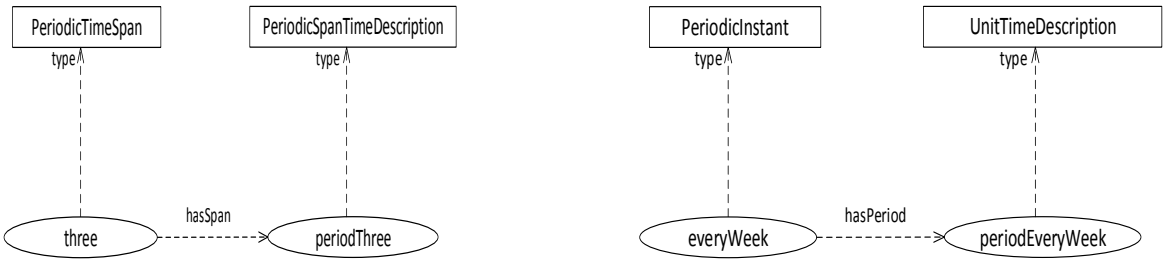

Figure 4: The separate mapping of the individual of PeriodicTimeSpan and the individual of PeriodicInstant

Consider Type 2 in Example 3: "Medication review three times every week." This involves a periodic instant (“everyWeek”) and a periodic time span (“three”), as shown in Figure 4. An instance named "everyWeek," falling under the type PeriodicInstant is linked to another instance, "periodEveryWeek," through the hasPeriod object property, defined as the type UnitTimeDescription. This connection solidifies the recurring nature of the temporal pattern, with "periodEveryWeek" serving as a descriptor for the unit of time associated with each recurrence. Additionally, an instance named "three," falling under the type PeriodicTimeSpan, is linked to another instance, "periodThree," through the hasSpan object property, defined as the type PeriodicTimeSpanDescription. This connection also solidifies the recurring nature of the temporal pattern, with "periodThree" acting as a descriptor for the unit of time associated with each recurrence. This example underscores the importance of future discussions on UnitTimeDescription and PeriodicTimeSpanDescription, as their representation is implicit at this stage.

4.1.4. Periodic exception

In expanding OWL-Time entities to encompass regular and irregular patterns, the introduction of the PeriodicException concept is a crucial addition. This concept precisely captures exceptions within regularly recurring patterns, addressing irregularities in stochastic aperiodic times. PeriodicException is conceptualized as a class that houses temporal instances where a recurring event deviates from its regular schedule due to exceptional conditions. For instance, in Type 5 instances, such as excluding specific days or times from the regular schedule, like "The museum opens every Friday to Monday, closed on December 20 for an important event."

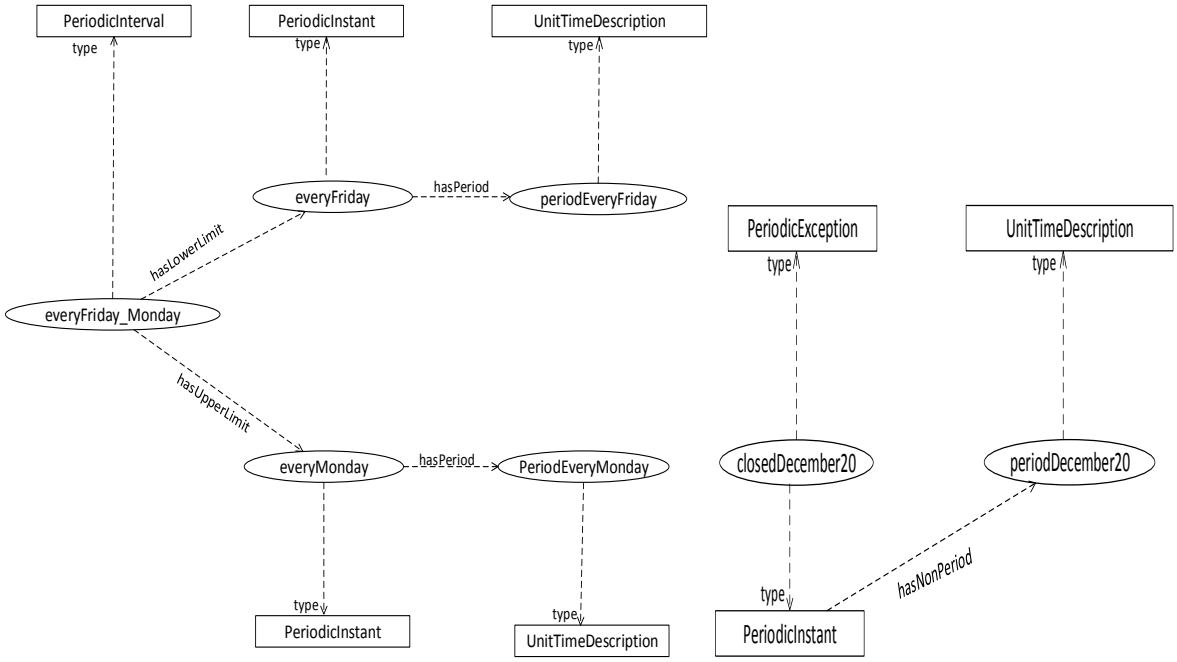

Figure 5: The separate mapping of the individual of regular periodicity and the individual of irregular periodicity (PeriodicException)

Unlike other recurring entities, PeriodicException does not have a unique property; instead, it reuses the properties of PeriodicRule, as explained in Section 4.2. This approach allows PeriodicException to invoke a PeriodicInstant and utilizes the hasNonPeriod property of PeriodicInstant. The illustrations of Example 3 are showcased in Figure 5 for Type 3 recurring time, constituting a periodic interval (everyFriday_Monday) and a periodic exception (closedDecember20). The process of representing this recurring temporal pattern using PeriodicInterval

is elucidated. We create an instance named "everyFriday_Monday," a type of PeriodicInterval, intricately connected to individual instances "everyFriday" and "everyMonday" through the object properties hasLowerLimit and hasUpperLimit. Both "everyFriday" and "everyMonday" are instances of PeriodicInstant, establishing a representation for each day within the recurring interval. To emphasize the recurring nature, each instance is linked to "periodEveryFriday" and "periodEveryMonday" through the object property hasPeriod, forming a periodic interval time description in Figure 3. Additionally, an instance named "closedDecember20" is created, falling under the type PeriodicException and PeriodicInstant. This instance is linked to another instance, "periodDecember20," through the hasNonPeriod object property, defined as the type UnitTimeDescription.

4.2. Overview of periodic rules

In this section, we introduced the PeriodicRule class to the OWL-Time ontology, a pivotal element in exploring recurring time patterns. Inspired by the iCalendar system, our class and its subclasses were customized to enrich our understanding of recurring time phenomena. Derived from [41] classes, our adaptation, represented in OWL-Time format, enhances the ontology's accessibility in complex cases.

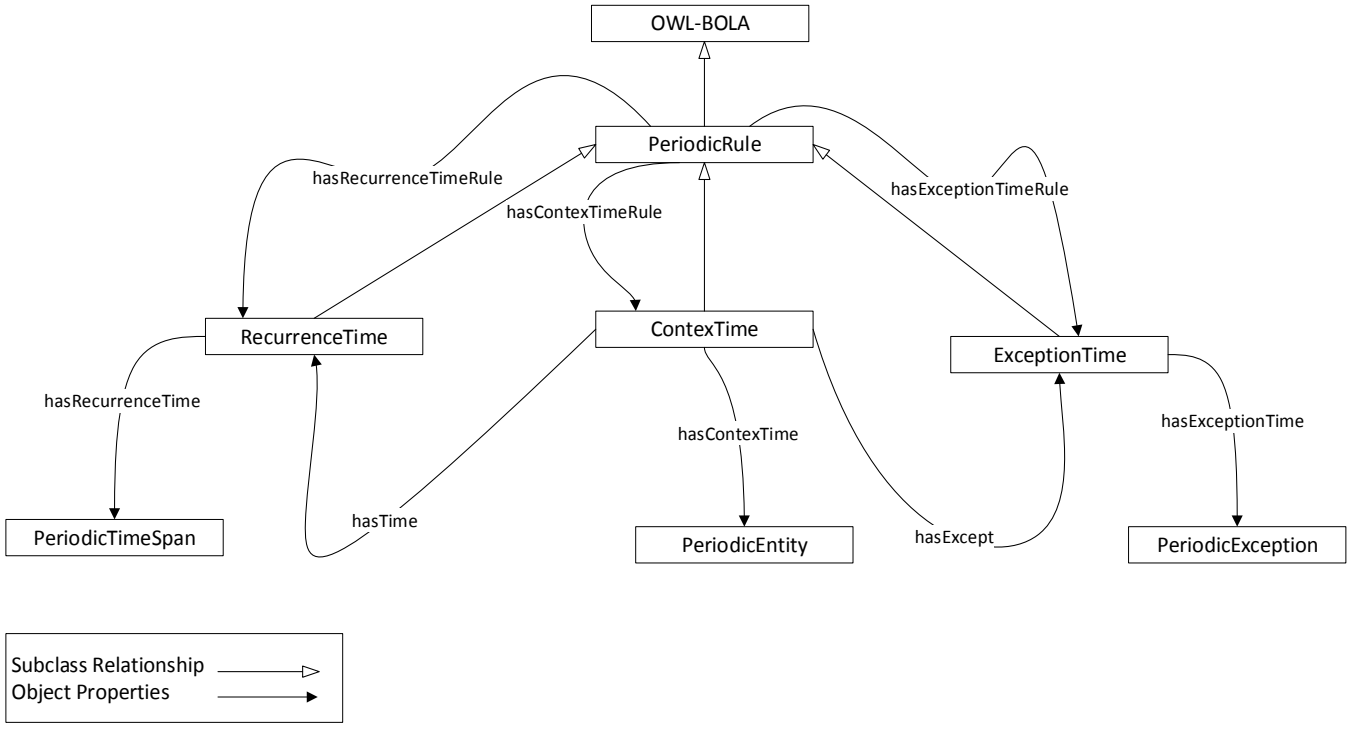

Figure 6: Taxonomy of Periodic Rule

In Figure 6, PeriodicRule class consolidates properties within the defined TemporalSeq interval, effectively amalgamating frequencies of subintervals within a non-convex interval. Our extended version prioritizes reusability across subclasses, featuring ContexTime, RecurrenceTime, and ExceptionTime. We derived six classes for rule concepts: PeriodicRule, FrequencyWithDurationReference, FrequencyWithCalendarReference, ContexTime, RecurrenceTime, and ExceptionTime, building on Faucher et al.'s foundation. While commonalities exist, such as FrequencyWithDurationReference and FrequencyWithCalendarReference being unions of ContexTime, our ontology introduces RecurrenceTime and ExceptionTime, absent in their ontologies but supported by the iCalendar system's concepts of RDATE and EXDATE. Figure 6 illustrates the taxonomy of the periodic rule, with details on the three important modules of PeriodicRule presented in the following sections.

4.2.1. Context time

The ContexTime class aligns with the periodic temporal distance patterns observed in the PeriodicEntity. It encompasses patterns in specific calendar units, such as “each Wednesday”, “every week”, or “every September”, and extends to any calendar unit supported by a numeric value, defining the gap, like “every two weeks” or “every three months”. To consolidate the pattern of PeriodicEntity within the PeriodicRule, we employed the hasContexTimeRule property. This property is explicitly outlined with the domain set as PeriodicRule and the range as ContexTime. The ContexTime class is then linked to the PeriodicEntity through the object property hasContexTime.

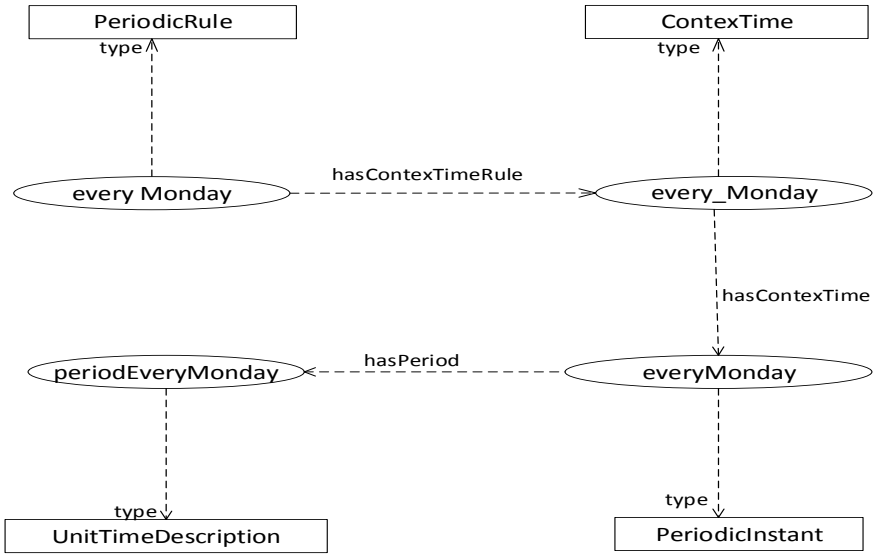

Figure 7 : Association of Individuals to Context Time Rule and Primitive Entities

The practical of ContexTime is exemplified in Figure 7, using the scenario outlined in Example 1, to illustrate the relationship among PeriodicRule, ContexTime, and PeriodicInstant. An instance, "every Monday," is instantiated as a type of PeriodicRule. This instance is connected to another named "every_Monday" through the hasContexTimeRule object property, specified as the type ContexTime. Subsequently, "every_Monday" is associated with another instance, "everyMonday," through the hasContexTime object property, defined as the type PeriodicInstant. Further, "everyMonday" is linked to an additional instance, "periodEveryMonday," through the hasPeriod object property, specified as the type UnitTimeDescription. This intricate connection solidifies the recurring nature of the temporal pattern, with "periodEveryMonday" serving as a descriptor for the unit of time associated with each recurrence.

4.2.2. Recurrence time

RecurrenceTime corresponds to the periodic time span pattern of the PeriodicTimeSpan. This rule defines the pattern of time span within a ContexTime, such as in “three times every 3rd month”. We aggregate the pattern of PeriodicTimeSpan within the PeriodicRule, utilizing the property hasRecurrenceTimeRule. The hasRecurrenceTimeRule is defined with the domain set as PeriodicRule, and the range as RecurrenceTime. The class

RecurrenceTime is then associated with the object property hasRecurrenceTime, linking the RecurrenceTime class to the PeriodicTimeSpan.

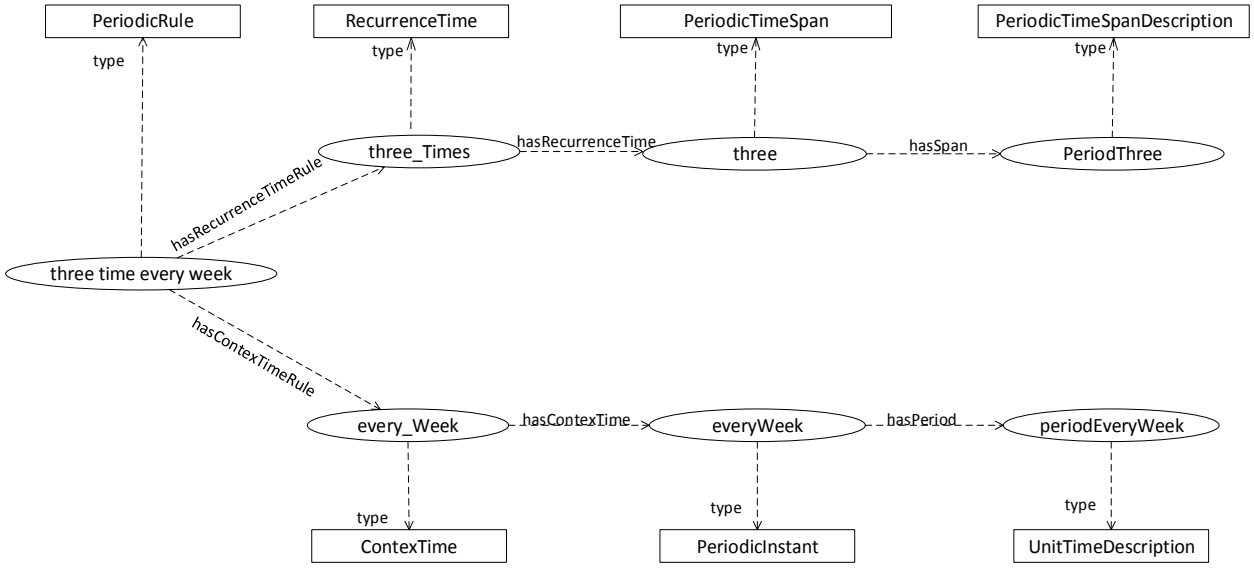

Figure 8: Association of Instances with RecurrenceTime Rule and Primitive Entities

The practical of the RecurrenceTime class is exemplified in Figures 8, utilizing the scenario detailed in Example 2. The visual representation showcased the root element as PeriodicRule with three interconnected properties, depicted by ContexTime triggering PeriodicInstant and RecurrenceTime activating the PeriodicTimeSpan class. An instance, "three times every week," is instantiated as a type of PeriodicRule. This instance is linked to another named "three_Times" through the hasRecurrenceTimeRule object property, specified as the type RecurrenceTime. Subsequently, "three_Times" is associated with another instance, "three," through the hasRecurrenceTime object property, defined as the type PeriodicTimeSpan. Further, "three" is connected to an additional instance, "periodThree," through the hasSpan object property, specified as the type PeriodicTimeSpanDescription. The remaining part of "three times every week" is linked to another named "every_Week" through the hasContexTimeRule object property, specified as the type ContexTime. Subsequently, "every_Week" is associated with another instance, "everyWeek," through the hasContexTime object property, defined as the type PeriodicInstant. Further, "everyWeek" is connected to an additional instance, "periodEveryWeek," through the hasPeriod object property, specified as the type UnitTimeDescription.

4.2.3. Exception time

ExceptionTime defines the exclusion pattern within ContexTime, aligning with PeriodicException's exclusion patterns. For instance, in the example “every month except April”, ContexTime is “every month”, and ExceptionTime is “except April”. In this context, ContexTime is linked to PeriodicEntity, while ExceptionTime is linked to PeriodicException. The integration of the exclusion pattern of PeriodicException into PeriodicRule is facilitated by the hasExceptionTimeRule property, connecting ContexTime and ExceptionTime. This connection, represented by the hasExceptionTime object property, captures the relationship between the regularity defined by ContexTime and the exceptions specified by ExceptionTime.

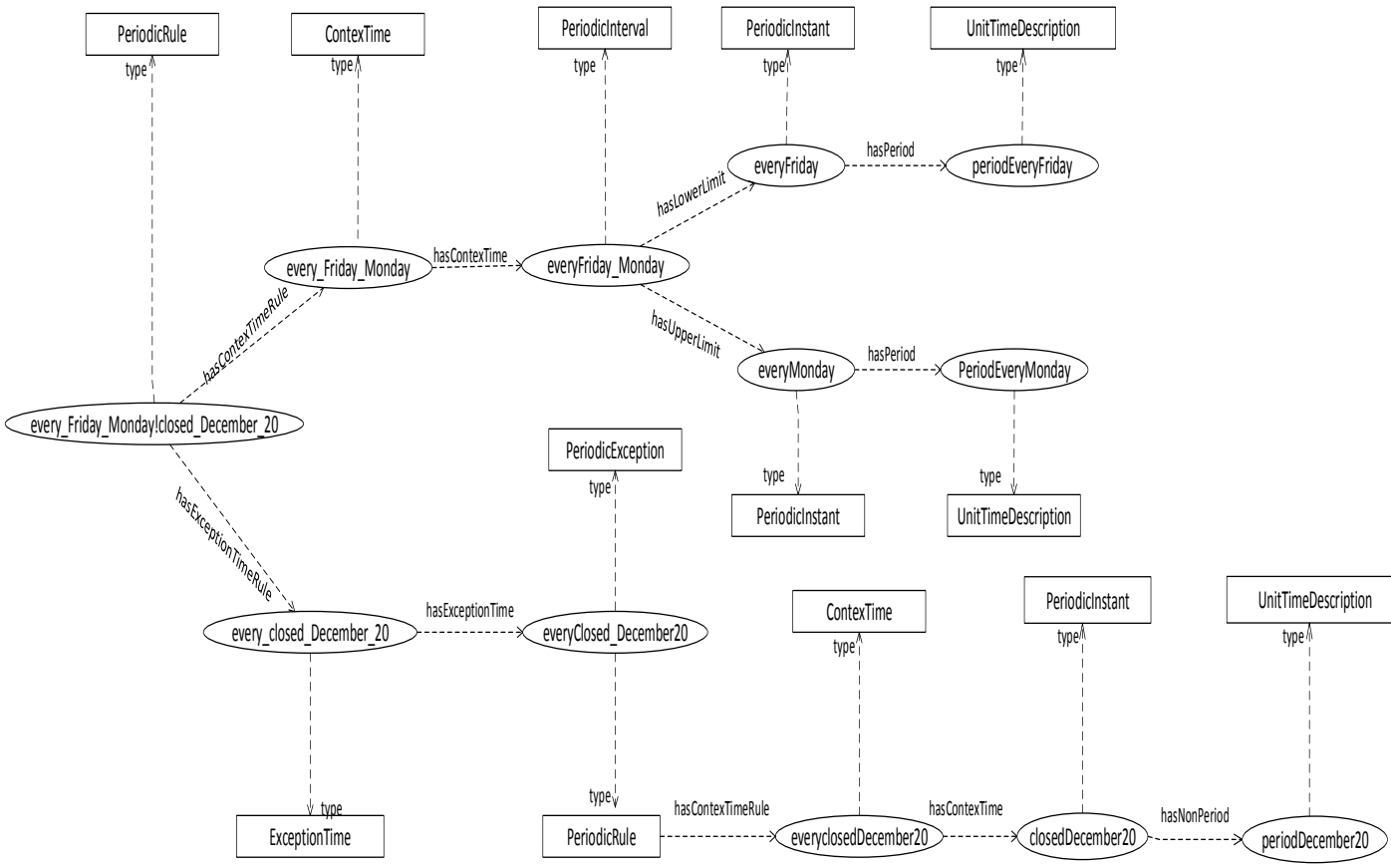

Figure 9 : Association of Instances with Exception Time Rule and Primitive Entities

The utilization of the ExceptionTime class is portrayed in Figure 9 for Example 3. The visual representation highlights PeriodicRule as the central element, featuring three interconnected properties. Specifically, ContexTime activates PeriodicInterval, RecurrenceTime triggers the PeriodicTimeSpan class, and ExceptionTime introduces the representation of PeriodicException. An instance named "every_Friday_Saturday!Closed_December_20" is instantiated as a type of PeriodicRule. This instance is linked to another named "every_Friday_Saturday" through the hasContexTimeRule object property, specified as the type ContexTime. Subsequently, "every_Friday_Saturday" is associated with another instance, "everyFriday_Saturday," through the hasContexTime object property, defined as the type PeriodicInterval connected to individual instances "everyFriday" and "everyMonday" through the object properties hasLowerLimit and hasUpperLimit. Both "everyFriday" and "everyMonday" are instances of PeriodicInstant, establishing a representation for each day within the recurring interval. To emphasize the recurring nature, each instance is linked to "periodEveryFriday" and "periodEveryMonday" through the object property hasPeriod. The remaining part of "every_Friday_Saturday!Closed_December_20" is linked to another named "every_Closed_December_20" through the hasExceptionTimeRule object property, specified as the type ExceptionTime. Subsequently, "every_Closed_December_20" is associated with another instance, "everyClosed_December20," through the hasExceptionTime object property, defined as the type PeriodicRule and PeriodicException. Further, "everyClosed_December20" is connected to an additional instance, "everyClosedDecember20," through the hasContexTimeRule object property, specified as the type ContexTime. "everyClosedDecember20" is associated with another instance, "closedDecember20," through the hasContexTime object property specified as the type PeriodicInstant. This instance is linked to another instance, "periodDecember20," through the hasNonPeriod object property, defined as the type UnitTimeDescription.

4.3. Overview of recurring calendar descriptions

In this section, we outlined a systematic approach to map basic periodic entities, as discussed in Section 4.1, to an expanded OWL-Time calendar description known as the periodic calendar description for explicit representation of diverse recurring time date and time descriptions.

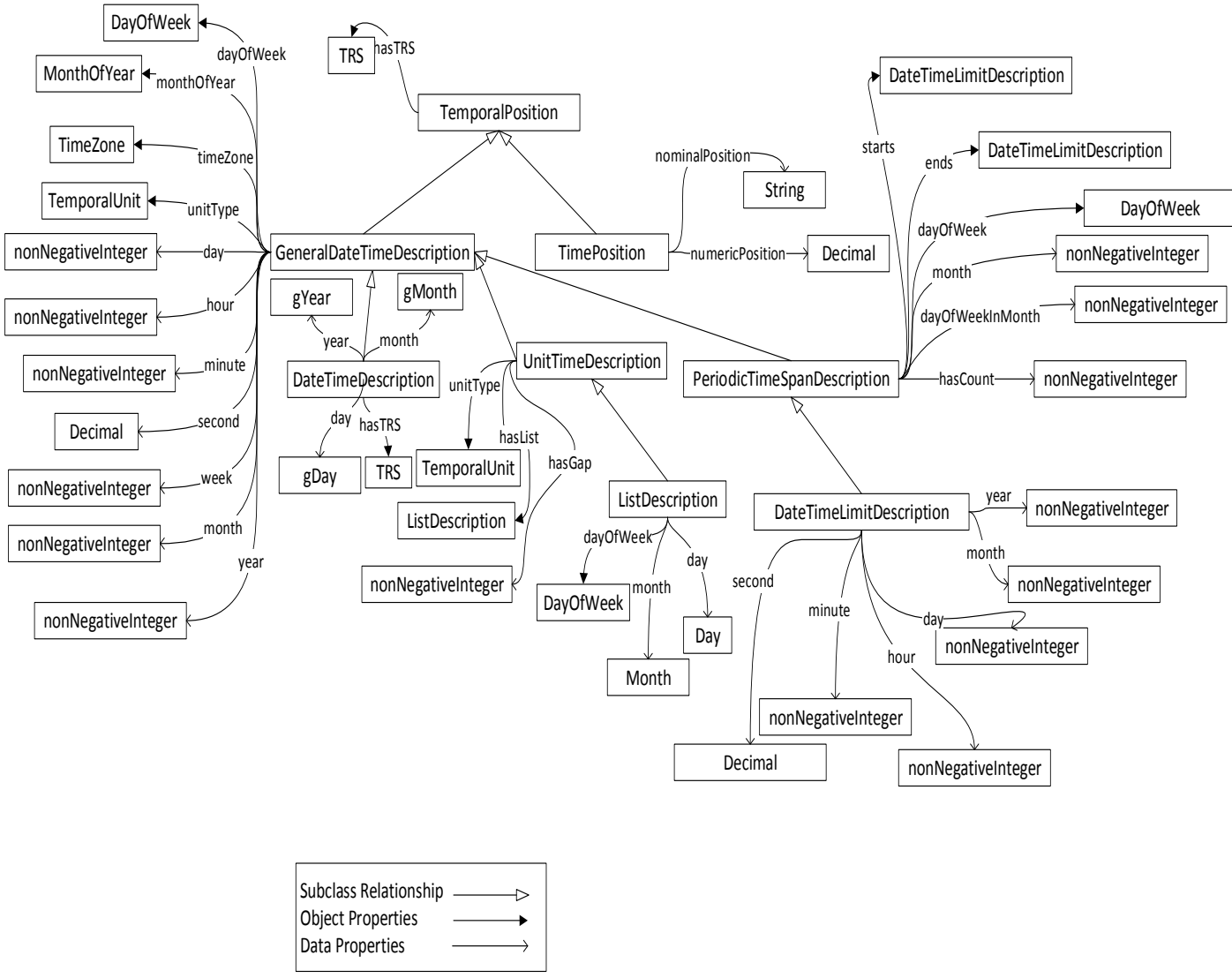

Figure 10: Taxonomy of Periodic Calendar Description

To enhance the expressive and reasoning capabilities of OWL-Time, we have introduced four subclasses—UnitTimeDescription, ListDescription, PeriodicTimeSpanDescription, and DateTimeLimitDescription—into the GeneralDateTimeDescription, as depicted in Figure 10. These subclasses will be detailed in the upcoming discussion. We illustrated the application of UnitTimeDescription, ListDescription, PeriodicTimeSpanDescription, and DateTimeLimitDescription through practical examples outlined in Table 1 through Figure 11 to 16. While some of these scenarios were implicitly covered in our earlier discussions (Sections 4.1 and 4.2), we now extend their representation by incorporating the properties of the periodic calendar description as needed.

4.3.1. Unit time description

The UnitTimeDescription class manages periodic calendar descriptions related to PeriodicInstant, PeriodicInterval, and PeriodicException within the ContexTime rule. As an extension of OWL-Time GeneralDateTimeDescription, it incorporates specific properties—unitType, hasGap, and hasList—alongside a supporting class named ListDescription. Utilizing the object property unitType, it represents periodicity frequencies (e.g., "every year," "every month") linked to TemporalUnit. The hasGap data property complements unitType, enabling the representation of periodic entities with numeric values (e.g., "every three months"). The object property hasList connects UnitTimeDescription to ListDescription, facilitating the definition of specific days or months. Implicitly, the hasList property in UnitTimeDescription supports expressions like "every Wednesday" or "every August.

4.3.2. List description

The ListDescription, a subclass of UnitTimeDescription in our ontology, enhances the representation by introducing the hasList property associated with specific calendar values. This augmentation provides explicit details, extending the capability discussed in Section 4.3.1, to represent days of the week and months explicitly. This enhancement is applied to the ContexTime rule, mirroring its application in handling periodic calendar descriptions for PeriodicInstant, PeriodicInterval, and PeriodicException. Notable features of ListDescription include the dayOfWeek property, inherited from OWL-Time, representing specific recurring days of the week. This property is linked to the inferred class DayOfWeek. Additionally, ListDescription incorporates data properties: day, representing specific days in a calendar month, and month, representing specific recurring months. These properties enable the explicit representation of statements like “every Wednesday,” “every 19th”or“every August”.

4.3.3. Periodic time span description

The PeriodicTimeSpanDescription class is crucial for explicit calendar descriptions related to PeriodicTimeSpan and the time span of PeriodicException. It includes various components such as defined starting and ending points (linked to the DateTimeLimitDescription subclass), specific days of the week using the dayOfWeek property, association with parts of the month through the month property, incorporation of specific days within a month (e.g., “first Wednesdays”), and reference counts indicating occurrences (e.g., “Twice every month”). These components are expressed through object properties like starts, ends, dayOfWeek, month, and dayOfWeekInMonth, along with the data property hasCount. Together, they provide a comprehensive framework for representing diverse recurring temporal expressions. With UnitTimeDescription, ListDescription, and PeriodicTimeSpanDescription discussed, we can now fully express Type 2 and Type 3 expressions from Table 1, as depicted in Figures 11 and 12.

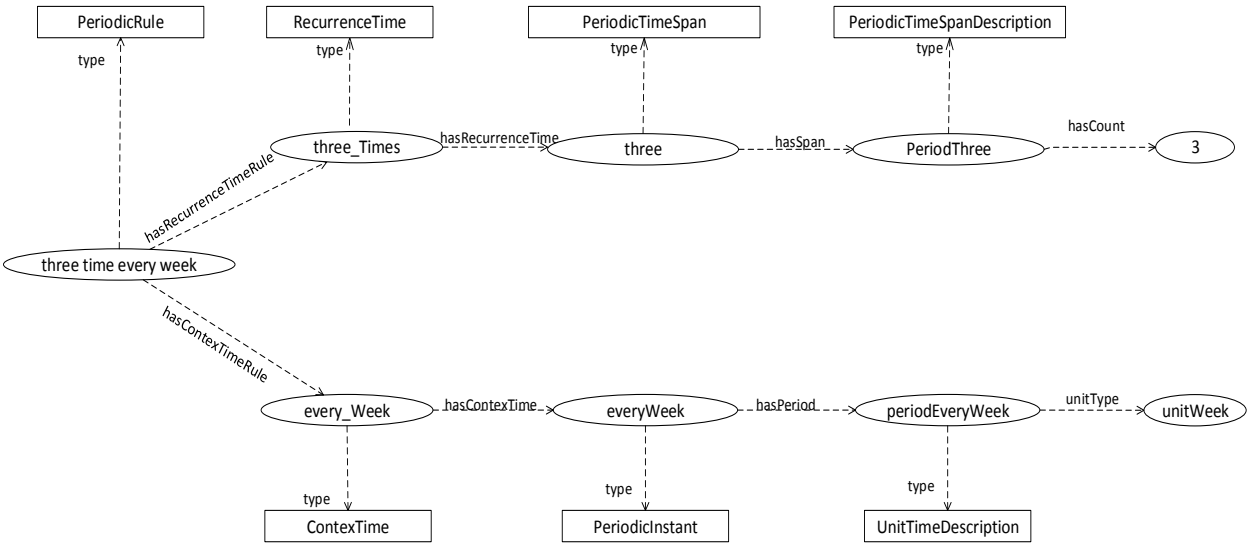

Figure 11: Representation of "Three times on every week" – Type 2

Figure 11 builds upon the insights provided in the discussion of Figure 8. In this context, we successfully capture the data type property of "periodThree" by utilizing the hasCount property within PeriodicTimeSpanDescription, setting it to the value of "3". Additionally, the representation of "periodicEveryWeek" is achieved through the application of the unitType object property within UnitTimeDescription, designated as "unitWeek."

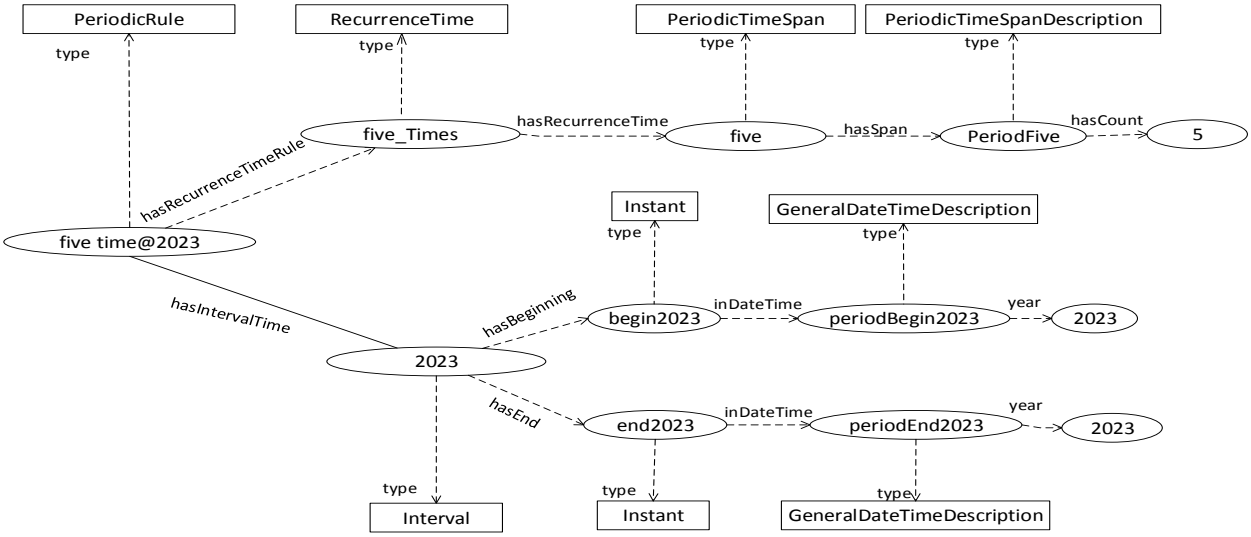

Figure 12 : Representation of "Five times in 2023"- Type 3

In Figures 12, we delve into the scenario outlined in Type 3, where we introduce an instance labeled "five times @ 2023," classified as a type of PeriodicRule. This instance first establishes a connection with another instance named "five_Times" through the hasRecurrenceTimeRule object property, specified as the type RecurrenceTime. Subsequently, "five_Times" is further linked to another instance, "five," using the hasRecurrenceTime object property, designated as the type PeriodicTimeSpan. To provide a detailed representation, "five" is intricately connected to an additional instance named "periodFive" through the hasSpan object property, specified as the type PeriodicTimeSpanDescription and efficiently represented utilizing the hasCount property. The second segment of "five times @ 2023" is associated with an instance named "2023," instantiated as a type of Interval. In continuation,

"2023" establishes connections to individual instances "begin2023" and "end2023" through the object properties hasBeginning and hasEnd, where both "begin2023" and "end2023" are instances of Instant. To emphasize the temporal interval, each instance is further linked to "periodBegin2023" and "periodEnd2023" through the object property inDateTime. These instances can be distinctly represented using the year property within GeneralDateTimeDescription specifying the beginning as 2023 and end as 2023.

4.3.4. Date time limit description

Instances of PeriodicTimeSpan defining the time span (e.g., “9:30 AM and 11:30 AM” in “every Wednesday between 9:30 AM and 11:30 AM”) are implicitly linked to the starts and/or ends object properties in PeriodicTimeSpanDescription. For explicitly representation of instances, we introduced the concept of DateTimeLimitDescription. This class handles fine time span granularity, allowing explicit representation of start or end times using six datatype properties (second, minute, hour, day, month, and year). These properties connect individuals of the starts and ends object properties to their data values in strings or XML dateTime format. This facilitates detailed representation, including instances like “09:30 AM”, “10 PM”, “21 hours”, “90 minutes”, “55 seconds”, or specific dates like “25th December”. Conversion to hours, minutes, or seconds may be needed for “AM” or “PM.” With UnitTimeDescription, ListDescription, PeriodicTimeSpanDescription, and DateTimeLimitDescription covered, we can now comprehensively express Type 1, Type 4, and Type 5 expressions from Table 1, as shown in Figures 13 to 15.

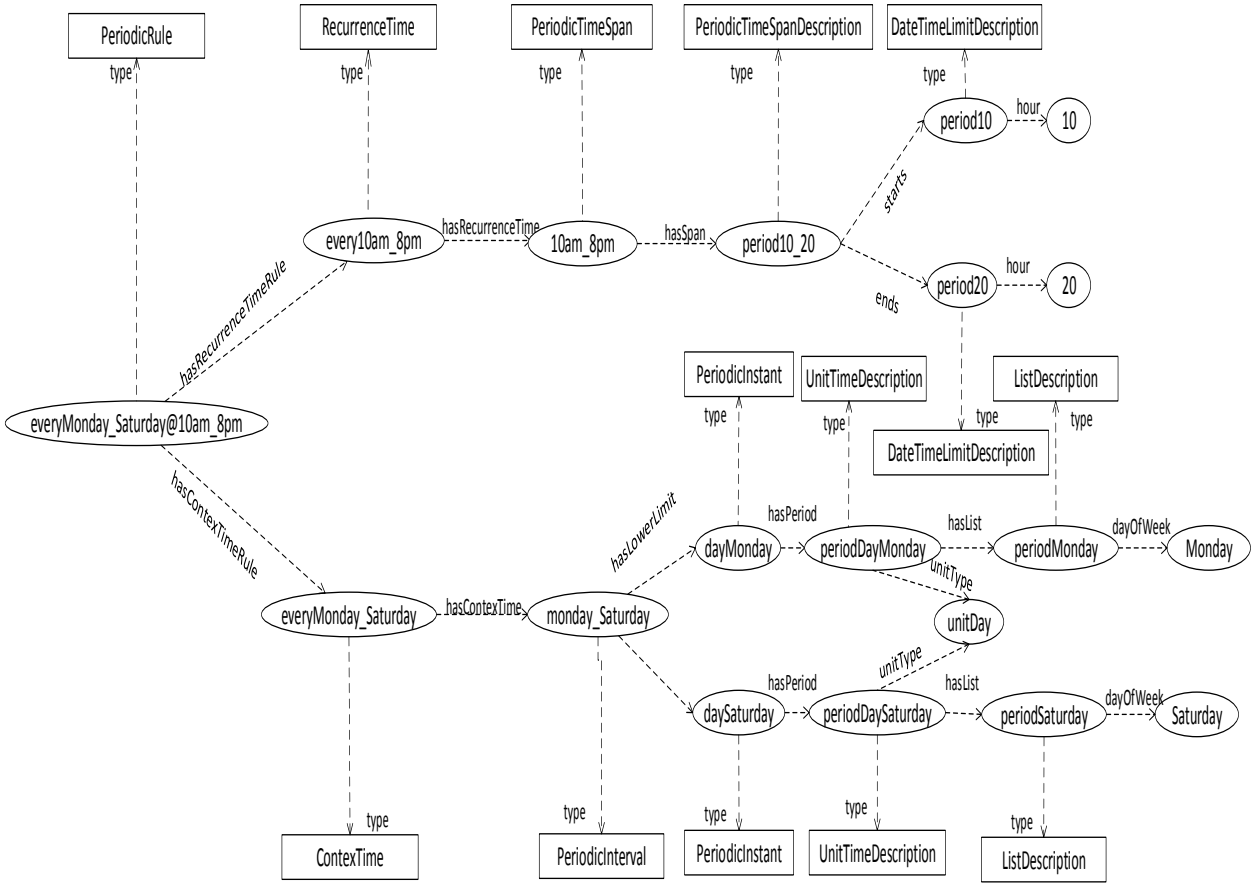

Figure 13 : Representation of “Every Monday to Saturday, between 10 a.m. and 8 p.m.”- Type 1

In Figures 13, we illustrate Type 1 with an instance named "everyMonday_Saturday@10am_8pm," a type of PeriodicRule, linked first to "everyMonday_Saturday" through hasContexTimeRule, specified as ContexTime. Subsequently, "everyMonday_Saturday" utilizes hasContexTime to connect with "monday_Saturday," a PeriodicInterval, further establishing associations with "dayMonday" and "daySaturday" through the object properties hasLowerLimit and hasUpperLimit. Both "dayMonday" and "daySaturday" are instances of PeriodicInstant. To emphasize recurrence, each links to "periodDayMonday" and "periodDaySaturday," types of UnitTimeDescription, through hasPeriod also specified as "unitDay" via the unitType property. Each of "periodDayMonday" and "periodDaySaturday" were further linked to "periodMonday" and "periodSaturday," through the hasList property of UnitTimeDescription, enabling a connection to ListDescription where we utilize the dayOfWeek property to specify them as “Monday” and “Saturday” respectively. The second part of "everyMonday_Saturday@10am_8pm" links to "every10am_8pm" through hasRecurrenceTimeRule, specified as RecurrenceTime. Subsequently, "every10am_8pm" utilizes hasRecurrenceTime to connect with "10am_8pm," a PeriodicTimeSpan, connected to "period10_20" through hasSpan, specified as PeriodicTimeSpanDescription, linked to "period10" and "period20" through starts and ends. Both "period10" and "period20" are instances of DateTimeLimitDescription, specifying the hour property as “10” and “20.”

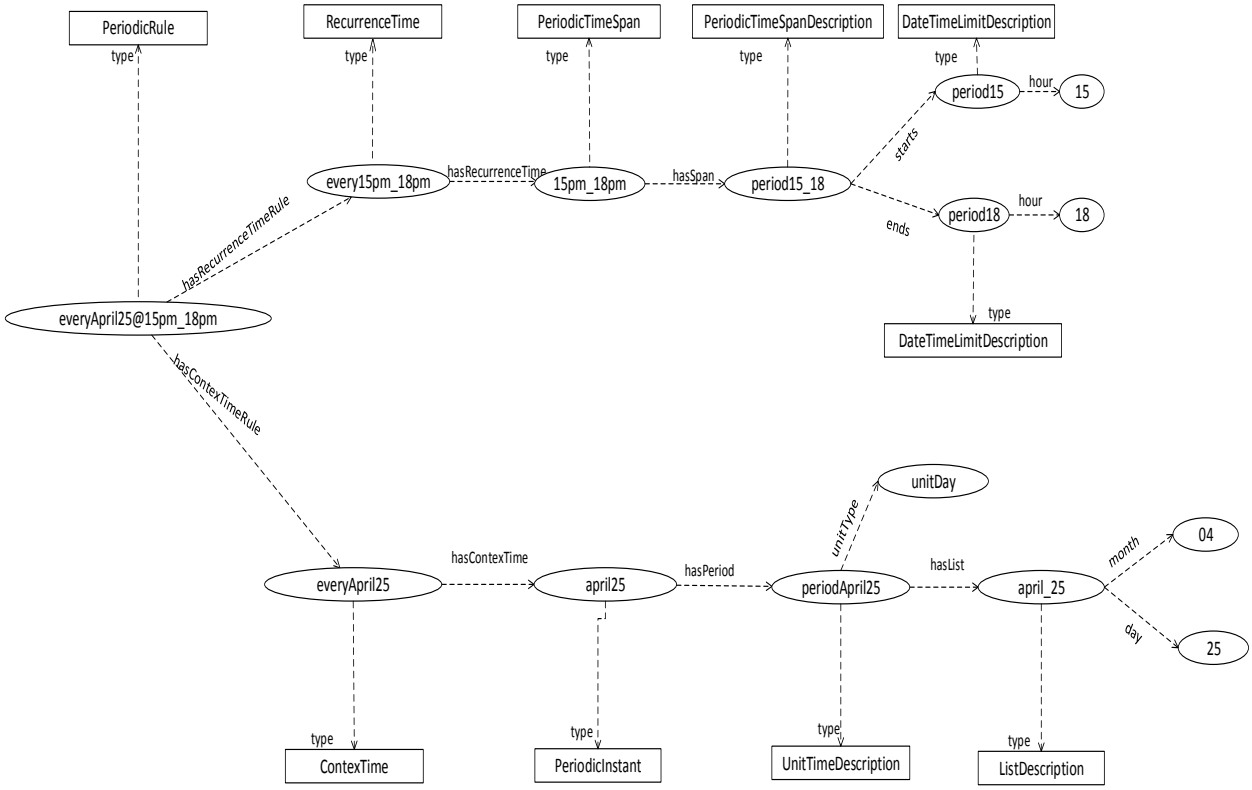

Figure 14: Representation of “every April 25, between 15 p.m. and 18 p.m.” -Type 4

Type 4 of the recurring time expression unfolds in Figures 14 and 15. In Figure 14, we encounter an instance titled "everyApril25@15pm_18pm," classified as a PeriodicRule, linked to "everyApril25" through the hasContexTimeRule, designated as ContexTime. Following this, "everyApril25" utilizes the hasContexTime property to establish a connection with "april25," identified as a PeriodicInstant, further solidifying an association with "periodApril25" through the object property hasPeriod. Notably, "periodApril25" falls under the category of UnitTimeDescription, specified as "unitDay" via the unitType property. Additionally, "periodApril25" is intricately linked to “april_25” through the hasList property of UnitTimeDescription, enabling a connection to ListDescription

where we utilize the day property to specify it as “25” and the month property to specify it as “04.” The remaining segment of "everyApril25@15pm_18pm" links to "every15pm_18pm" through the hasRecurrenceTimeRule, identified as RecurrenceTime. Subsequently, "every15pm_18pm" utilizes hasRecurrenceTime to connect with "15pm_18pm," recognized as a PeriodicTimeSpan, intricately linked to "period15_18" through the hasSpan property, specified as PeriodicTimeSpanDescription. This connection extends to "period15" and "period18" through the starts and ends properties. Both "period15" and "period18" are instances of DateTimeLimitDescription, specifying the hour property as “15” and “18.”

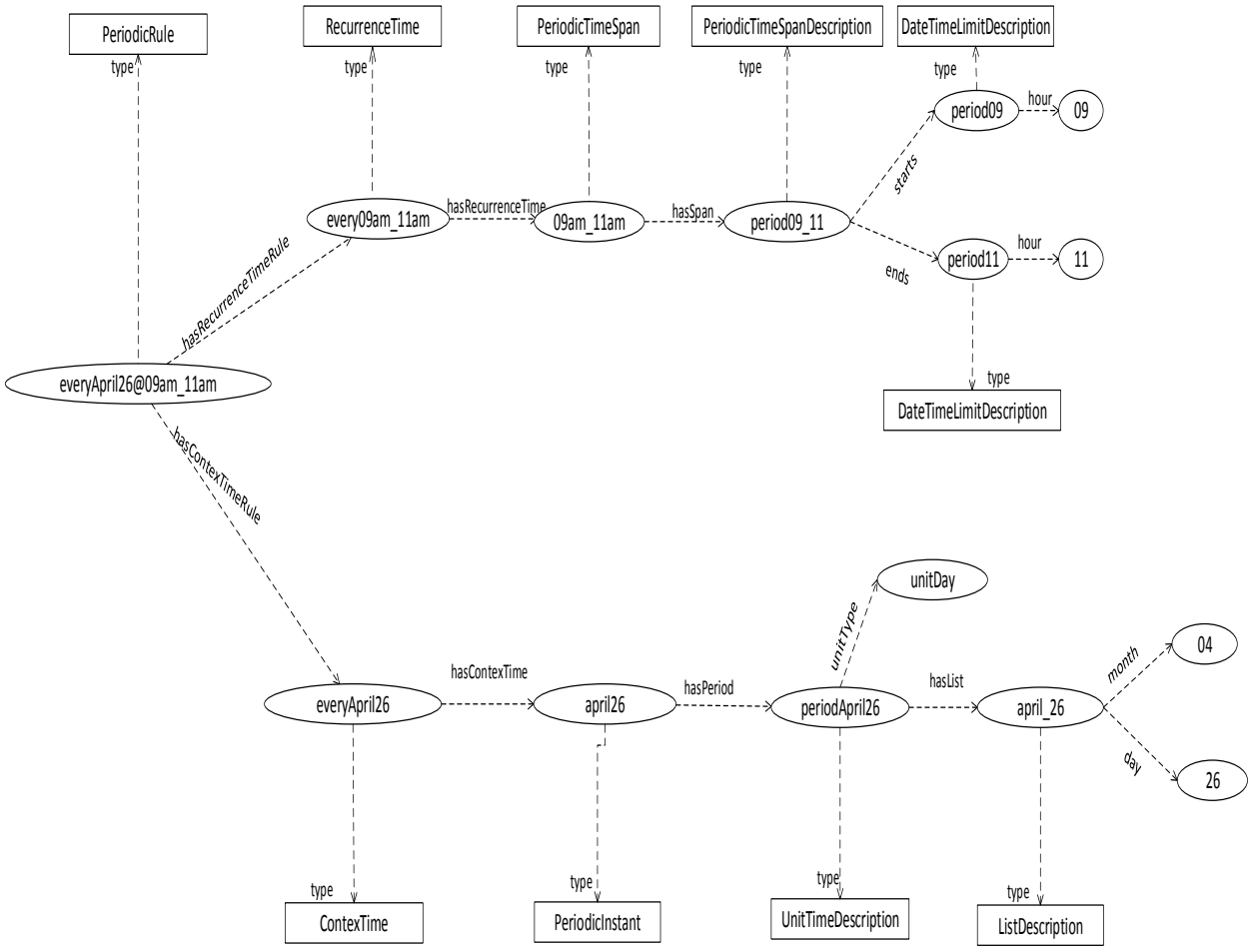

Figure 15 : Representation of “every April 26, between 9 a.m. and 11 a.m.”-Type 4

In Figure 15, we come across an instance denoted as "everyApril26@09am_11am," categorized as a PeriodicRule, linked to "everyApril26" through the hasContexTimeRule, identified as ContexTime. Subsequently, "everyApril26" employs the hasContexTime property to establish a link with "april25," recognized as a PeriodicInstant, further reinforcing the association with "periodApril26" through the object property hasPeriod. Importantly, "periodApril25" falls into the realm of UnitTimeDescription, specified as "unitDay" via the unitType property. Furthermore, "periodApril26" intricately connects to "april_26" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription where we use the day property to specify it as “26” and the month property to specify it as “04.” The remaining part of "everyApril26@09am_11am" links to "every09am_11am" through the hasRecurrenceTimeRule, recognized as RecurrenceTime. Following this, "every09am_11am" employs hasRecurrenceTime to link with "09am_11am," acknowledged as a PeriodicTimeSpan, intricately associated with "period09_11" through the hasSpan property, specified as PeriodicTimeSpanDescription. This connection extends to "period09" and "period11" through the starts and ends properties. Both "period09" and "period11" are instances of DateTimeLimitDescription, specifying the hour property as “09” and “11.”

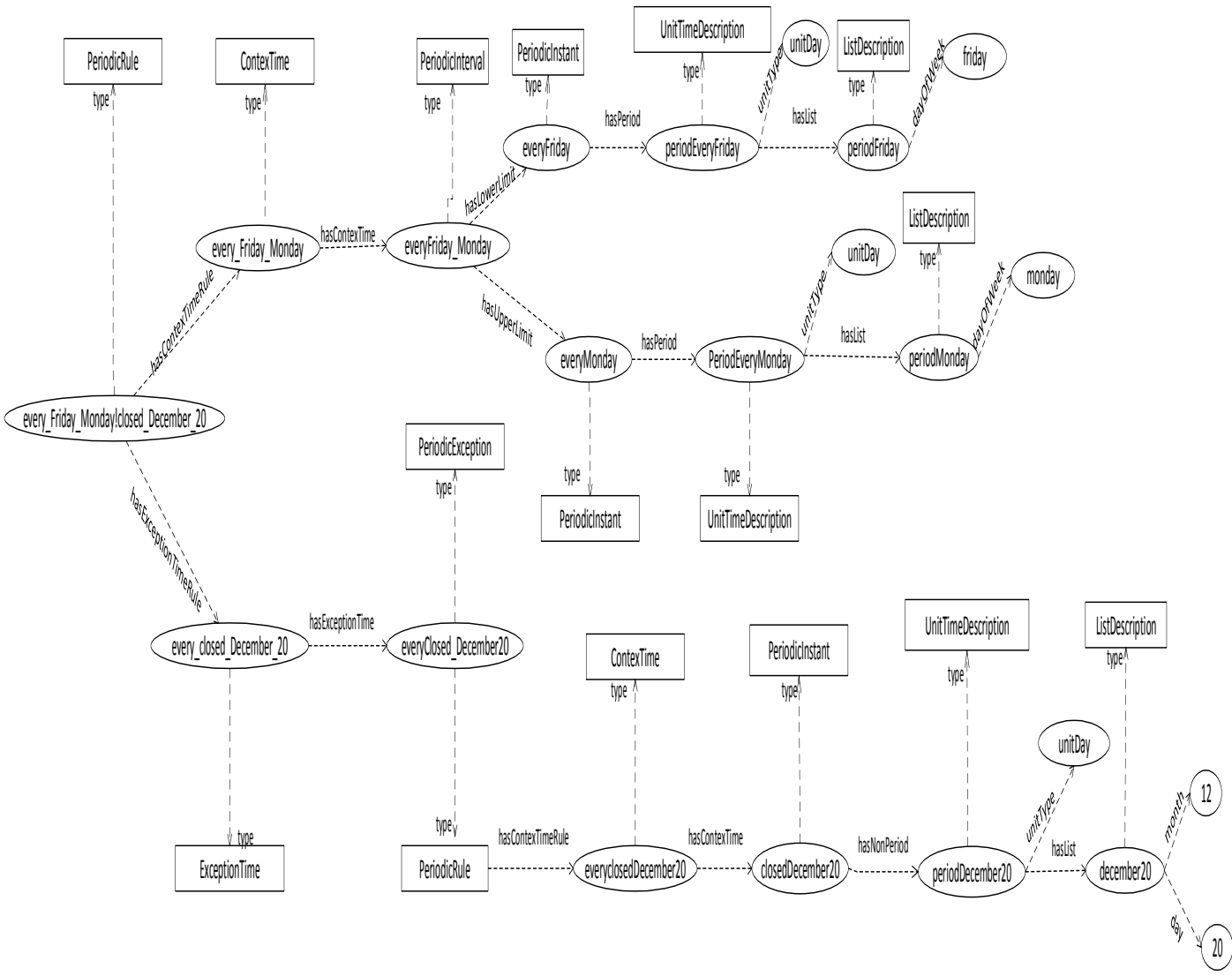

Figure 16 : Representation of "Every Friday to Monday closed on December 20"- Type 5

Figure 16 expands on the insights discussed in Figure 9. In this context, we extended instances such as "periodEveryFriday" and "periodEveryMonday," categorizing them under UnitTimeDescription and specifying "unitDay" via the unitType property. These instances were intricately linked to "periodFriday" and "periodMonday" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription. For "periodFriday," we utilized the dayOfWeek property to specify it as "Friday," and for "periodMonday," we used the dayOfWeek property to specify it as "Monday."

Additionally, the representation of the irregularity in "periodicDecember20" is achieved similarly. Here, "periodDecember20" is categorized under "UnitTimeDescription" and specified as "unitDay" via the unitType property. Moreover, "periodDecember20" intricately connects to "december20" through the hasList property of UnitTimeDescription, establishing a connection to ListDescription where the day property specifies it as "20", and the month property specifies it as "12."

5. Result and Discussion.

In this section, we provide a comprehensive overview of the evaluation and insightful discussions on the ontology results. The development and experimentation of the ontology were conducted using protégé editor version 4.3.0, build 304, on a host system configured with an Intel (R) Core i5-5200U CPU at 2.20 GHz, 2.20 GHz, and 8 Gbytes of RAM, running on Windows 8.1 OS. Our examination delves into the richness of the BOLA ontology, analyzing its expressivity, completeness, and complexity within the Protégé editor. Additionally, we evaluate the coverage of the domain and assess the quality of information accessible from the associated concepts and data types. This evaluation includes a comparative analysis with other temporal ontologies, exploring their respective ontologies for a comprehensive understanding.

5.1. Evaluation of the ontology

In order to verify the ability of the BOLA ontology to express time information in the recurring dataset, we selected a set of case texts an anonymized realistic dataset of 503 records [41] and examples from [22]. We selected 5 statements covering all aspects of the recurring time problems, which contained 118 individuals using our ontology. A number of standard ontology metrics were considered for the evaluation of the ontology designed in this study. Each metric presents its value, as shown in Table 3. For instance, it is clear that the ontology is rich in axiom and logical axiom, which are statements about concepts and relations. Other notable metrics are the class coverage, property coverage, class hierarchical, cardinality restriction, depth of class hierarchy etc. A description of these metrics and their count values are summarized in Table 3. This clarification of the atomic metrics became necessary to allow for their use in the computation of the composite ontology metrics described later in the section.

Table 3: A listing of metrics and their respective counts in the BOLA ontology

	Metrics	**Number of counts**
Completeness	Class coverage	87
	Property coverage	88
	Class hierarchy	100
Expressivity	Class hierarchy	100
	Property hierarchy	100
	Cardinality restrictions	49
	Qualified cardinality restrictions	0
	Complex class expressions	13
Complexity	Number of Classes	38
	Number of Object Properties	59
	Number of Individuals	118
	Depth of Class Hierarchy	6
	Number of Axioms	266
	Number of Logical Axioms	123
	Number of Role Chains	10
	Number of Equivalent Classes	7
	Number of Disjoint Classes	5
	Number of Inferred Subclass Relationships	6

Considering the list in Table 3, the following definitions hold:

i. **Class coverage** measures the percentage of relevant classes or concepts in the domain that are represented in the ontology.
ii. **Property coverage** measures the percentage of relevant properties or relationships in the domain that are represented in the ontology.
iii. **Class hierarchy** measures the percentage of essential superclass-subclass relationships that are explicitly defined.
iv. **Properties hierarchy** measures percentage of essential property characteristics
v. **Cardinality restrictions** measures the number of classes with cardinality restrictions on properties.
vi. **Qualified cardinality restrictions** measures the number of specifying cardinality restrictions based on conditions (qualified cardinality restrictions) are more expressive.
vii. **Complex class** expressions number of classes with set operations (union, intersection, complement) and existential or universal quantification.
viii. **Number of Classes** measures the total number of classes defined in the ontology.
ix. **Number of Object Properties** measures total number of object properties defined in the ontology.
x. **Number of Individuals** measures the total number of instances in the ontology.
xi. **Depth of class hierarchy** measures the maximum length of the inheritance chain in the class hierarchy.
xii. **Number of axioms** measures the total number of axioms in the ontology, including subclass relationships, property assertions, and constraints.
xiii. **Number of logical axioms** measures the number of related logical relationships and constraints, excluding annotation and metadata.
xiv. **Number of role chains** measures the total number of sequences of object properties defined in the ontology.
xv. **Number of equivalent classes** measures the number of equivalent class axioms that define classes with the same meaning.
xvi. **Number of disjoint classes** measures the number of disjointness axioms that state classes have no common instances.
xvii. **Number of inferred subclass** relationships measures the size of the inferred subclass hierarchy resulting from automated reasoning.

5.2 Evaluating the depth of temporal ontologies

We considered some temporal ontologies like TEO, CTO, Time-Entry, and NCO, which we compared with the BOLA ontology on their ability to represent the five problems presented in Section 3. Table 4 shows the result, and we used mark "√" to indicate those ontologies capable of getting problem's by querying.

Table 4: Comparison of different temporal ontologies on recurring time representations.

Type / Ontology	1	2	3	4	5
TEO	√				
CTO	√	√		√	
Time-Entry	√	√			
NCO	√			√	
BOLA	√	√	√	√	√

In addressing different types of time problems, each ontology has distinct strengths and limitations: For Type-1 problems, characterized by strongly periodic time, any ontology can effectively represent regular cyclic occurrences. In contrast, for Type-2 problems involving nearly aperiodic time, NCO primarily focuses on Type-4 (non-recurring aperiodic) scenarios, lacking the ability to represent other forms of aperiodic time. Type-3 problems, characterized as intermittent, are approached by Time-Entry and CTO, but these ontologies exhibit limitations in handling the intermittent counterpart. For instance, they lack the capability to represent instances like "three times in 2023" within a single ontology, requiring separate representations for the instant 2023 and the three-time reference object. Moving to Type-5 problems involving stochastic time, CTO introduces an intriguing feature in IrregularCollection, enabling a stage-by-stage representation of time and potentially enabling Type-4 representation. Their stochastic time representation is binary-dependent such as in "taking medicine A is not earlier than the end of treatment B." Their algorithm "not" is designed for representing the negation of temporal relationships, such as not before, not after, and not overlap. However, BOLA ontology surpasses this by representing stochastic problems as aggregated concepts and not as the negation. BOLA extends OWL-Time comprehensively, covering various periodic and aperiodic time expressions, including those not explicitly mentioned, thanks to its rich recurring calendar description.

In summary, BOLA ontology stands out as superior among time ontologies, excelling in representing recurring time comprehensively, including both periodic and aperiodic instances.

6. Conclusion and Future Work

In this paper, we addressed the challenges associated with recurring time-related problems across diverse domains, identifying five key problem types: strongly periodic, nearly aperiodic, intermittent aperiodic, non-recurring aperiodic, and stochastic. To tackle these issues, we introduced the BOLA ontology, a comprehensive solution featuring a taxonomy of intricate temporal concepts. Our ontology extends OWL-Time primitive entities and calendar descriptions, providing clear concept rules for recurring time components. The taxonomy encompasses periodic rules, recurring primitive entities, and recurring calendar descriptions, offering a structured framework for expressing various components in recurring time expressions. Experimental results demonstrate the ontology's completeness, with a 100% hierarchy for both classes and properties, expressive capabilities with a complexity expression of 13 and 49 cardinality restrictions, and complexity score with a substantial number of 266 axioms, including 123 logical axioms. In summary, the BOLA ontology stands as a robust solution capable of representing a wide range of complex recurring temporal information.

While our ontology presents a significant advancement, there are avenues for improvement and extension. First, the current work does not address reasoning over fuzzy time, representing a potential area for future research. Additionally, we proposed building practical applications for recurring time utilizing a knowledge graph based on the BOLA ontology. This application-focused approach aims to validate and enhance the ontology's utility in real-world scenarios.

REFERNCES

[1] V. Ermolayev, S. Batsakis, N. Keberle, O. Tatarintseva, and G. Antoniou, “ONTOLOGIES OF TIME: REVIEW AND TRENDS,” p. 59, 2014.

[2] J. M. Górriz *et al.*, “Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications,” *Neurocomputing*, vol. 410, pp. 237–270, Oct. 2020, doi: 10.1016/j.neucom.2020.05.078.

[3] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983, doi: 10.1145/182.358434.

[4] D. Mcdermott, “A TemporalLogicfor Reasoning About Processesand Plans,” *Cognitive science*, no. 6, pp. 101–155, 1982.

[5] Y. Shoham and N. Goyal, “Temporal Reasoning in Artificial Intelligence,” in *Exploring Artificial Intelligence*, Elsevier, 1988, pp. 419–438. doi: 10.1016/B978-0-934613-67-5.50015-0.

[6] L. Vila, “A Survey on Temporal Reasoning in Artificial Intelligence,” p. 25, 1994.

[7] A. H. Van de Ven and M. S. Poole, “Alternative Approaches for Studying Organizational Change,” *Organization Studies*, vol. 26, no. 9, pp. 1377–1404, Sep. 2005, doi: 10.1177/0170840605056907.

[8] L. Vila, “Revisiting time and temporal incidence,” *Scientic research council of spain*, pp. 1–36, 1996.

[9] C. Bettini, X. S. Wang, S. Jajodia, and J.-L. Lin, “Discovering frequent event patterns with multiple granularities in time sequences,” *IEEE Trans. Knowl. Data Eng.*, vol. 10, no. 2, pp. 222–237, Apr. 1998, doi: 10.1109/69.683754.

[10] Li, X. S. Wang, and S. Jajodia, “Discovering Temporal Patterns in Multiple Granularities,” in *Temporal, Spatial, and Spatio-Temporal Data Mining*, vol. 2007, J. F. Roddick and K. Hornsby, Eds., in Lecture Notes in Computer Science, vol. 2007. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 5–19. doi: 10.1007/3-540-45244-3_2.

[11] P. Terenziani, “Integrated Temporal Reasoning with Periodic Events,” *Computational Intell*, vol. 16, no. 2, pp. 210–256, May 2000, doi: 10.1111/0824-7935.00112.

[12] P. Terenziani, “Irregular Indeterminate Repeated Facts in Temporal Relational Databases,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1075–1079, Apr. 2016, doi: 10.1109/TKDE.2015.2509976.

[13] A. Tuzhilin and Clifford, “On periodicity in temporal databases,” *Information system*, vol. 20, no. 8, pp. 619–639, 1995.

[14] H. J. Ohlbach, “Relations between fuzzy time intervals,” in *Proceedings. 11th International Symposium on Temporal Representation and Reasoning, 2004. TIME 2004.*, Tatihou, Normandie, France: IEEE, 2004, pp. 44–51. doi: 10.1109/TIME.2004.1314418.

[15] S. Ribaric, B. Dalbelo Basic, and N. Pavesic, “A model for fuzzy temporal knowledge representation and reasoning,” in *FUZZ-IEEE’99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, Seoul, South Korea: IEEE, 1999, pp. 216–221 vol.1. doi: 10.1109/FUZZY.1999.793237.

[16] B. O. Akinkunmi, “The problem of coincidence in a theory of temporal multiple recurrence,” *Journal of Applied Logic*, vol. 15, pp. 46–68, May 2016, doi: 10.1016/j.jal.2015.12.001.

[17] B. O. Akinkunmi, “A Partitioning Algorithm for Detecting Eventuality Coincidence in Temporal Double recurrence,” p. 33, 2017.

[18] J. A. G. M. Koomen, “Reasoning about recurrence,” *Int. J. Intell. Syst.*, vol. 6, no. 5, pp. 461–496, Aug. 1991, doi: 10.1002/int.4550060503.

[19] P. Terenziani, “Toward a Unifying Ontology Dealing with Both User–Defined Periodicity and Temporal Constraints About Repeated Events,” *Computational Intelligence*, vol. 18, no. 3, pp. 336–385, Aug. 2002, doi: 10.1111/1467-8640.00194.

[20] P. Terenziani, “Irregular Indeterminate Repeated Facts in Temporal Relational Databases,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 1075–1079, Apr. 2016, doi: 10.1109/TKDE.2015.2509976.

[21] R. Logananthara and S. Gombrone, “Representation of, and reasoning with, near-periodic recurrent events,” *9th IJCAI Workshop on Spatial and Temporal Reasoning.*, pp. 1–6, 1995.

[22] A. M. Bento, D. C. Viegas, and N. Correia, “Temporal Reasoning with Non-convex Intervals,” in *Knowledge Engineering and Semantic Web*, vol. 786, P. Różewski and C. Lange, Eds., in Communications in Computer and Information Science, vol. 786. , Cham: Springer International Publishing, 2017, pp. 127–142. doi: 10.1007/978-3-319-69548-8_10.

[23] J. Chicaiza and P. Valdiviezo-Diaz, “A Comprehensive Survey of Knowledge Graph-Based Recommender Systems: Technologies, Development, and Contributions,” *Information*, vol. 12, no. 6, p. 232, May 2021, doi: 10.3390/info12060232.

[24] A. Rhayem, M. B. A. Mhiri, and F. Gargouri, “Semantic Web Technologies for the Internet of Things: Systematic Literature Review,” *Internet of Things*, vol. 11, p. 100206, Sep. 2020, doi: 10.1016/j.iot.2020.100206.

[25] C. Tao, W.-Q. Wei, M. H. R. Solbrig, G. Savova, and C. G. Chute, “CNTRO: A Semantic Web Ontology for Temporal Relation Inferencing in Clinical Narratives,” *Journal of the American Medical Informatics Association*, vol. 7, p. 27, 2010.

[26] F. Li *et al.*, “Time event ontology (TEO): to support semantic representation and reasoning of complex temporal relations of clinical events,” *Journal of the American Medical Informatics Association*, vol. 27, no. 7, pp. 1046–1056, 2020, doi: 10.1093/jamia/ocaa058.

[27] J. Gu, D. Wang, D. Hu, F. Gao, and F. Xu, “Temporal Extraction of Complex Medicine by Combining Probabilistic Soft Logic and Textual Feature Feedback,” *Applied Sciences*, vol. 13, no. 5, p. 3348, 2023, doi: 10.3390/app13053348.

[28] D. Hu, M. Wang, F. Gao, F. Xu, and J. Gu, “Knowledge Representation and Reasoning for Complex Time Expression in Clinical Text,” *Data Intelligence*, vol. 4, no. 3, pp. 573–598, Jul. 2022, doi: 10.1162/dint_a_00152.

[29] J. R. Hobbs and F. Pan, “An ontology of time for the semantic web,” *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 1, pp. 66–85, Mar. 2004, doi: 10.1145/1017068.1017073.

[30] M. J. O’Connor and A. K. Das, “A Method for Representing and Querying Temporal Information in OWL,” in *Biomedical Engineering Systems and Technologies*, vol. 127, A. Fred, J. Filipe, and H. Gamboa, Eds., in Communications in Computer and Information Science, vol. 127. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 97–110. doi: 10.1007/978-3-642-18472-7_8.

[31] R. Fikes and Q. Zhou, “A Reusable Time Ontology,” *Proceeding of the AAAI Workshop on Ontologies for the Semantic Web*, pp. 1–6, 2002.

[32] Y. Raimond and A. Samer, “The event ontology,” pp. 1–6, 2007.

[33] S. Batsakis, E. G. M. Petrakis, I. Tachmazidis, and G. Antoniou, “Temporal representation and reasoning in OWL 2,” *SW*, vol. 8, no. 6, pp. 981–1000, Aug. 2017, doi: 10.3233/SW-160248.

[34] F. Ghorbel, F. Hamdi, E. Métais, N. Ellouze, and F. Gargouri, “Ontology-based representation and reasoning about precise and imprecise temporal data: A fuzzy-based view,” *Data & Knowledge Engineering*, vol. 124, p. 101719, Nov. 2019, doi: 10.1016/j.datak.2019.101719.

[35] J. R. Hobbs, G. Ferguson, and J. F. Allen, “A daml ontology of time.,” 2002, [Online]. Available: http://www.cs.rochester.edu/

[36] V. Ermolayev, N. Keberle, and W. E. matzke, “An Upper-Level Ontological Model for Engineering Design Performance Domain,” *27th Int’l Conf. on Conceptual Modeling*, no. LNCS 5231, pp. 98–113, 2008.

[37] F. Pan, “An Ontology of Temporal Concepts for the Semantic Web and Natural Language,” p. 66, 2007.

[38] S. Batsakis and E. G. M. Petrakis, “SOWL: spatio-temporal representation, reasoning and querying over the semantic web,” in *Proceedings of the 6th International Conference on Semantic Systems - I-SEMANTICS ’10*, Graz, Austria: ACM Press, 2011, p. 1. doi: 10.1145/1839707.1839726.

[39] F. Frasincar, V. Milea, and U. Kaymak, “tOWL: Integrating Time in OWL,” in *Semantic Web Information Management*, R. de Virgilio, F. Giunchiglia, and L. Tanca, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 225–246. doi: 10.1007/978-3-642-04329-1_11.

[40] V. M. Poveda, F. M. C. Suárez, and P. A. Gómez, “A Pattern for Periodic Intervals,” *Semantic Web Journal*, pp. 1–10, 2014.
[41] C. Faucher, J.-Y. Lafaye, and F. Bertrand, “Modelling composite periodic Events,” p. 15, 2012.