Imposing Restrictions Over Temporal Properties in OWL: A Rule-Based Approach

Sotiris Batsakis and Euripides G.M. Petrakis

Department of Electronic and Computer Engineering
Technical University of Crete (TUC)
Chania, Greece
e-mail:{batsakis, petrakis}@intelligence.tuc.gr

Abstract. Introducing the dimension of time in ontologies turns binary relations into ternary which cannot be handled by OWL. Approaches such as the N-ary relations or the 4D-fluents approach discussed in this work, offer satisfactory solutions to this problem. However, data and property semantics are not preserved in the resulting representations nor can they be handled by ordinary reasoners such as Pellet. We propose a rule-based solution to this problem using SWRL.

1 Introduction

Welty and Fikes [3] showed how quantitative temporal information (i.e., in the form of temporal intervals whose left and right endpoints are well defined) as well as, the evolution of concepts in time, can be represented effectively in OWL using the so-called "4D-fluents approach". In our previous work [1], we extended this approach with qualitative (in addition to quantitative) temporal expressions, allowing for the representation of temporal intervals with unknown endpoints by means of their relation (e.g., "before", "after") to other time intervals, or alternatively, by translating relations between temporal intervals into equivalent relations between time instants [2]. Property semantics such as cardinality restrictions or property constraints are not handled. Typically, property semantics in OWL are defined over binary relations. These relations are turned into ternary with the introduction of the temporal dimension. Existing solutions to this problem (e.g., the 4D-fluents or the N-ary relations approach) suggest introducing new objects into the temporal representation and also rewriting ternary relations as sets of binary ones defined between the old and the new objects. Accordingly, property relations, to become meaningful, need to be applied on the new objects as well, rather than on the objects on which they were meant to be defined originally. Then, property semantics can no longer be handled by ordinary reasoners such as Pellet.

In this work, we propose a mechanism for handling OWL property semantics over temporal representations in conjunction with the 4D-fluents and the N-ary relations approaches. Property semantics are expressed by a set of SWRL rules defined over temporal relations (rather than by OWL axioms as it is typical in

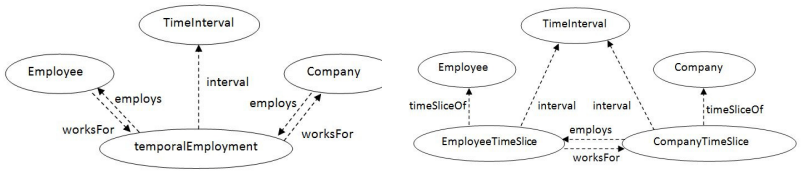
static ontologies, since cardinality restrictions over non simple binary properties lead to undecidability [5]). To the best of our knowledge, this is the only known solution to this problem.

Related work in the field of knowledge representation is discussed in Sec. 2. Restriction checking is discussed in Sec. 3, followed by evaluation in Sec. 4 and conclusions and issues for future work in Sec.5.

2 Background and Related Work

The OWL-Time temporal ontology[1] describes the concepts of time. Apart from language constructs for the representation of time in ontologies, there is still a need for mechanisms for the representation of the evolution of concepts (e.g., events) in time. Representation of the evolution of temporal concepts is achieved using *N-ary relations*[2] or *4D-fluents* [3].

Following the *N-ary relations* approach, the temporal property is represented by two properties, each one related with the new object. Fig.1(a) illustrates the relation *WorksFor(Employee, Company, TimeInterval)* representing the fact that an employee works for a company during a time interval using N-ary relations. This approach requires one additional object for every temporal interval. The *4D-fluents* (perdurantist) approach [3] suggests that concepts in time are



(a) N-ary Relations (b) 4D fluents

Fig. 1. Example of (a) N-ary Relations and (b) 4D-fluents

represented as 4-dimensional objects with the 4th dimension being the time (*timeslices*). Properties having a time dimension are called fluent properties and connect instances of class *TimeSlice*. Fig.1(b) illustrates the 4D-fluents representation for the temporal relation *Works-For(Employee, Company, TimeInterval)* of the same example.

2.1 Temporal ontology

According to Welty and Fikes [3], to add a time dimension to an ontology, classes *TimeSlice* and *TimeInterval* with properties *timeSliceOf* and *timeInterval* are introduced. Properties having a temporal dimension are called *fluent properties* and connect instances of class *TimeSlice* (as in Fig.1(b)). In [1], the 4D-fluents

[1] http://www.w3.org/TR/owl-time
[2] http://www.w3.org/TR/swbp-n-aryRelations

representation was enhanced with qualitative temporal relations (i.e., relations holding between time intervals whose starting and ending points are not specified). A temporal relation can be one of the 13 pairwise disjoint Allen's relations.

Definitions for temporal entities (e.g., instants and intervals) are provided by incorporating OWL-Time into the ontology. A representation based on temporal instants (rather than intervals) is also feasible [2]. Each interval (which is an individual of the *ProperInterval* class of OWL-Time) is related with two instants (individuals of the *Instant* class) that specify it's starting and ending points using the *hasBegining* and *hasEnd* object properties respectively. Notice that, only one of the three relations ($before$, $after$, or $equals$) may hold between any two temporal instants with the obvious interpretation. These relations can be asserted even when exact dates of the time instants are unknown. Relations between intervals are expressed as time instant relations between their starting and ending points.

2.2 Temporal Reasoning

Reasoning is realized by introducing a set of SWRL[3] rules operating on temporal relations. Reasoners that support DL-safe rules such as Pellet[4] can be used for inference and consistency checking over temporal relations. Table 1 represents the result of the composition of two temporal relations pairs (relations $before$, $after$ and $equals$, are denoted by symbols "<",">","=" respectively).

Table 1. Composition Table for point-based temporal relations.

Relations	$<$	$=$	$>$
$<$	$<$	$<$	$<,=,>$
$=$	$<$	$=$	$>$
$>$	$<,=,>$	$>$	$>$

For example, if relation R_1 holds between $instant1$ and $instant2$ and relation R_2 holds between $instant2$ and $instant3$, then the entry of the Table 1 corresponding to line R_1 and column R_2 denotes the possible relation(s) holding between $instant1$ and $instant3$. Compositions of relations R_1, R_2 yielding a unique relation R_3 as a result are expressed in SWRL using rules of the form:

$$R_1(x, y) \wedge R_2(y, z) \rightarrow R_3(x, z)$$

The following is an example of such a temporal composition rule:

$$before(x, y) \wedge equals(y, z) \rightarrow before(x, z)$$

A series of compositions of relations may imply relations which are inconsistent with existing ones (for example the above rule will yield a contradiction if *after(x,z)* has been asserted into the ontology for specific values of x,y,z). Consistency checking is achieved by ensuring path consistency [7] by applying formulas of the the form:

[3] http://www.w3.org/Submission/SWRL/
[4] http://clarkparsia.com/pellet/

$$\forall x, y, k\ R_s(x, y) \leftarrow R_i(x, y) \cap (R_j(x, k) \circ R_k(k, y))$$

representing intersection of compositions of relations with existing relations (symbol $\cap$ denotes intersection, symbol $\circ$ denotes composition and R_i, R_j, R_k, R_s denote temporal relations). A set of rules defining the result of intersecting relations holding between two instances must also be defined in order to implement path consistency. These rules are of the form:

$$R_1(x, y) \wedge R_2(x, y) \rightarrow R_3(x, y)$$

where R_3 can be the empty relation. For example, the intersection of relations $before$ and $after$ yields the empty relation, and an inconsistency is detected:

$$before(x, y) \wedge after(x, y) \rightarrow \bot$$

Path consistency is implemented by defining compositions of relations using SWRL rules and declaring the three basic relations as disjoint. It is sound and complete when applied on the three basic relations [7].

3 Restriction Checking over Temporal Properties

Checking for restrictions holding on time dependent (fluent) properties in the 4D-fluents representation requires particular attention. If a fluent property holds between two objects, then, these objects are only indirectly associated through one or more artificial objects (e.g., *TimeSlice* object in *4D-fluents*). A fluent property is declared between the artificial object and an actual object (as in Figure 1(a)) or between two artificial objects (as in Figure 1(b)). Checking for property restrictions requires adjusting the domain and range of this property from the artificial to the actual objects (e.g., to *Company* and *Employee* objects in Figure 1(a)). For example, for the *worksfor* property in Figure 1(b), the domain of the property is no longer class *Employee* but *timeslice of Employee*. Accordingly, its range is *timeslice of Company*.

Similar adjustments must be made in the case of N-ary relations but in this case, combining transitivity of properties while retaining domain and range restrictions becomes problematic. For example, the $worksfor$ relation in Fig. 1(a) must be provided with two alternative domains and ranges. Other restrictions on properties such as symmetry, asymmetry, reflexiveness, irreflexiveness and transitivity can be applied directly on the temporal property retaining the intended semantics.

Universal restrictions (e.g., "all Employees work for a company") also require adjusting domains and ranges (i.e., all timeslices of employees $workfor$ timeslices of companies). Existential restrictions are adjusted as well (if for example each employee must work for some company, then timeslices of employees must work for some timeslices of companies). Notice, that an existential restriction corresponds to an *at least one* qualified cardinality restriction in OWL and the way it is handled is discussed in the rest of this section.

Adjusting cardinality restrictions, functional and inverse functional properties is somewhat more complicated. Functional properties are a special case of

cardinality restrictions (i.e., if a property is functional, then each object must be connected to *at most* one subject which is different for each object). Cardinality restrictions are amenable to two different interpretations depending on the specific application and the intended semantics:

Cardinality restrictions may be interpreted either as restricting the total number of individuals of a class (e.g., *Company*) related with each individual of another class (e.g., *Employee*) through a fluent property at all times or as restricting the number of individuals for each specific temporal interval over which the fluent property holds true. Each interpretation calls for different ways of handling which are discussed in the following.

The first interpretation is handled simply by counting on the number of individuals of a class related to this property and is implemented in SWRL. SWRL rules are applied because OWL cardinality restrictions cannot handle fluent properties connecting objects through intermediate objects. Specifically, imposing cardinality restrictions as OWL axioms in the chain of properties involving the intermediate objects (which are non-simple properties) leads to undecidability [6].

The following rule expresses the restriction that each employee can work for at most n companies. If $n+1$ company individuals are found to connect with an employee individual, then the restriction is violated (the $Alldifferent$ keyword is an abbreviation for a series of axioms imposing that the n+1 individuals $z_1, z_2 ... z_{n+1}$ are all different). By imposing a *max* cardinality restriction of 0 over property *error* at the definition of class *Employee*, the violation of the cardinality restriction is detected by standard reasoners such as Pellet using the rule:

$$(At-most-rule1) Employee(x) \wedge (tsTimesliceOf(x_1, x)$$

$$\wedge ... \wedge tsTimesliceOf(x_{n+1}, x) \wedge worksfor(x_1, y_1) \wedge worksfor(x_{n+1}, y_{n+1})$$

$$\wedge tsTimesliceOf(y_1, z_1) ... \wedge tsTimesliceOf(y_{n+1}, z_{n+1})$$

$$\wedge Alldifferent(z_1, z_2, ..., z_{n+1}) \wedge Company(z_1) ... \rightarrow error(x, z_1)$$

An *at-least* restriction is expressed similarly: an *at-most* $n-1$ rule is applied (changing the asserted property to $satisfies(x, n)$) combined with an *at-least-one* cardinality restriction on the $satisfies$ property for class *Employee*. All rules impose also a restriction on the type of objects involved (e.g., they require that only company objects are involved by checking only for objects connected with timeslices of companies). Dropping such a check leads to an unqualified numeric restriction on the property. Notice that the *Open World Assumption* of OWL will cause a reasoner (e.g., Pellet) not to detect an inconsistency of an $at-least$ restriction as future assertions might cause invalidity of this inconsistency detection. Instead, the user can retrieve individuals that are not yet proven to satisfy the restriction using the following SPARQL query:

The second interpretation imposes restrictions on the number of individuals associated with an individual of a specific class through a fluent property, for every temporal interval that the property holds true. Checking for such restrictions

```
select distinct ?x
where {
?x rdf:type ex1:Employee.
OPTIONAL{
?x ex1:satisfies ?y.}
FILTER(!bound(?y))}
```

requires applying reasoning rules over the relations between the temporal intervals associated with the fluent property as described in Sec. 2.2. The next step is to detect overlapping and non-overlapping intervals. After the Allen's relations holding between pairs of intervals have been inferred, Allen's properties *during, contains, starts, startedby, finishes, finishedby, overlaps, overlapedby, equals* are defined as subproperties of property *overlapping*, thus detecting overlapping and non-overlapping intervals. Moreover, properties *before, after, meets, metby* are defined as subproperties of property *non-overlapping*.

Expressing an *at most* restriction for every time interval is based on the following observation: the restriction is violated *if and only if* $n+1$ distinct individuals are connected with a given individual (through their timeslices) with the relation at hand, and their corresponding intervals are all pairwise overlapping. If $n+1$ intervals are pairwise overlapping, then, there exists an interval such that $n+1$ intervals share a common sub-interval, and this can be proven by induction on n. The existence of such an interval implies that for this interval the *at least* restriction is violated. The corresponding rule (used in combination with a cardinality restriction on property error for inconsistency detection by reasoners) is expressed as (the *pairwiseoverlapping* is an abbreviation for a set of *overlapping* relations between all pairs of intervals at hand):

$$(At-most-rule2)Employee(x) \wedge (tsTimesliceOf(x_1,x))$$

$$\wedge ... \wedge tsTimesliceOf(x_{n+1},x) \wedge hasinterval(x_1,w_1)...$$

$$\wedge hasinterval(x_{n+1},w_{n+1}) \wedge worksfor(x_1,y_1) \wedge ... \wedge worksfor(x_{n+1},y_{n+1})$$

$$\wedge tsTimesliceOf(y_1,z_1)... \wedge tsTimesliceOf(y_{n+1},z_{n+1}) \wedge$$

$$Alldifferent(z_1,...,z_{n+1}) \wedge pairwiseoverlapping(w_1,...w_{n+1})$$

$$\wedge Company(z_1)... \rightarrow error(x,z_1)$$

As in the case of the first interpretation, the *at-least* restriction cannot be imposed using only SWRL rules due to the Open World Assumption of OWL. Nevertheless, SWRL rules combined with an SPARQL query that detects individuals that are not satisfying the restriction yet, (in a way analogous to the first interpretation) can be applied.

All rules (with both interpretations) involve a time consuming selection of all possible subsets of individuals and intervals. Therefore, expressing restrictions using SWRL may become a tedious task and also detecting inconsistencies can be time consuming. Specifically, rules for imposing a cardinality of *at-least* or *at most* n involves the selection of all combinations of n among k timeslices, or reified relations, (where k is the number of temporal individuals in the ontology), thus it is not scalable for large values of n. In the case of reification or N-ary

relations, cardinality constraints are expressed accordingly, using appropriate adjustments on classes and on properties of objects involved.

Besides representation of cardinality restrictions, value restrictions and adjustments of domains and ranges, the following object property semantics are redefined as follows:

- *Functional*: It is handled as an at most 1 unqualified cardinality restriction.
- *Inverse Functional*: The inverse property is handled as an at most one unqualified cardinality restriction.
- *Symmetric*: The fluent property is *symmetric* too, thus the symmetry axioms apply on the interval that the involved timeslices exist (i.e., the temporal property is declared symmetric).
- *Asymmetric*: This is handled as a cardinality restriction, where the same property cannot hold for interchanged subjects and objects for timeslices that share an overlapping interval.
- *Equivalent*: The fluent properties are equivalent too.
- *Reflexive*: The fluent property is reflexive too; when a timeslice has the property for an interval, it is also the subject of the property for this interval.
- *Irreflexive*: This is handled as a cardinality restriction; two timeslices of an object cannot be related with the property in question if their intervals overlap.
- *Subproperty*: subproperty axioms apply for the fluent properties with the intended semantics.
- *Transitive*: Fluent properties are declared transitive since related timeslices must have equal intervals (by the definition of the 4D-fluent model) and for these intervals transitivity is applied.

Datatype properties have fewer characteristics (i.e., *subproperty, equivalence disjointness, functional*) and they are handled as is the case of object properties. In the case of N-ary relations, the above adjustments must take into account the different objects involved (i.e., *Events* instead of *timeslices*).

4 Evaluation

The resulting OWL ontology is decidable, since it complies with the corresponding OWL 2 specifications[5] . Introducing the set of temporal qualitative rules of Sec. 2.2 retains decidability since rules are DL-safe rules[6] and they apply only on named individuals of the ontology Abox using Pellet (which support DL-safe rules). Furthermore, computing the rules has polynomial time complexity since tractable subsets of Allen's temporal relations are used. Examples of tractable sets include [7]. As shown in [4], by restricting the supported relations set to a tractable subset of Allen's interval algebra, path consistency has $O(n^5)$ worst

[5] http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/
[6] http://www.w3.org/TR/rif-rdf-owl

time complexity (with n being the number of intervals) and it is sound and complete. Notice that extending the model to the full set of relations would result into an intractable reasoning procedure.

By applying the *closure method* [4] over Allen's relations, the minimal tractable sets containing the basic relations consist of 29 relations. For this set, the required number of OWL axioms and SWRL rules is 983. An implementation with temporal instants does not require additional relations. Then, the number of OWL axioms and SWRL rules required for reasoning is limited to 20 [2], implying that the representation based on temporal instants must be preferred over the one based on temporal intervals. However, restriction checking mechanism work with either representation. The restriction checking rules are dependent on the specific cardinality restriction imposed (i.e., the complexity of an *at most r* restriction is a parameter of r). Specifically, restriction checking has $O(n^r)$ time complexity.

5 Conclusions and Future Work

We introduce a rule-based approach for handling data and property semantics in temporal ontologies in OWL. Addressing performance and scalability issues for large scale applications are important issues for future research.

Acknowledgement

This research leading to these results has received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement No 296170 (Project PortDial).

References

1. S. Batsakis and E.G.M. Petrakis. "SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0". *5th International Symposium on Rules: Research Based and Industry Focused (RuleML' 2011)*, pages 242–249, 2011.
2. S. Batsakis, K. Stravoskoufos, and E.G.M. Petrakis. "Temporal Reasoning for Supporting Temporal Queries in OWL 2.0". *15th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES' 2011)*, pages 558–567, 2011.
3. C. Welty and R. Fikes. "A Reusable Ontology for Fluents in OWL". *Frontiers in Artificial Intelligence and Applications*, 150:226–236, 2006.
4. J. Renz, and B. Nebel. "Qualitative Spatial Reasoning using Constraint Calculi". In Handbook of Spatial Logics, Springer, Netherlands , pp. 161-215, 2007.
5. I. Horrocks, O. Kutz and U. Sattler "The Even More Irresistible SROIQ" In: Proc. KR 2006, Lake District, UK, 2006.
6. I. Horrocks, U. Sattler, and S. Tobies. "Practical Reasoning for Expressive Description Logics". In *Logic for Programming and Automated Reasoning*, pages 161–180. Springer, 1999.
7. P. van Beek, and R. Cohen "Exact and approximate reasoning about temporal relations." Computational intelligence, Vol 6(3), pp. 132–147, 1990.