Defining Activity Specifications in OWL

Megan Katsumi and Mark Fox

University of Toronto katsumi@mie.utoronto.ca
msf@mie.utoronto.ca

Abstract. As the amount of information on the web increases, so does the importance of ontologies available to capture it. This work is concerned with supporting a correct and meaningful representation of activities on the Semantic Web, with the potential to support tasks such as activity recognition and reasoning about causation. This requires an ontology capable of more than simply documenting and annotating individual activity occurrences; definitions of activity specifications are needed. Current representations of activities in OWL do not meet the basic requirements for activity specifications. Detailed definitions of an activity's preconditions and effects are lacking, in particular with respect to a consideration of change over time. This paper leverages existing work to fill this void with an ontology design pattern for activity specifications in OWL.

1 Introduction

The need to capture the semantics of activities is a common requirement in various domains. In our case, this requirement arose in the development of an ontology for urban systems. The ontology is part of a larger project of urban informatics, iCity [12], and was required to capture both the static and dynamic aspects of an urban system. The project requires the representation of various activities that occur in the urban environment: from population changes (such as deaths, births, and marriages), to everyday travel activities (such as driving, taking transit, and parking). Beyond representing individual occurrences, specifications of these activities are required.

Over time there may be any number of occurrences of an activity and an activity specification provides a useful way to describe these occurrences in general. It is a means of improving the understanding of the domain, as well as assessing the accuracy or completeness of some information (e.g. a model or other implementation from the various research efforts in iCity). Certainly, there may be many and varied concepts involved in activity specifications for different applications. There may be requirements to capture various types of resources, resource consumption, activity participation, and so on. This work focuses solely on the requirements for a fundamental representation of activity preconditions and effects. Specifically, what state(s) must hold in order for an activity to occur, and what state(s) are caused to be true when an activity occurs. A definition of preconditions and effects in an activity specification enriches the knowledge

representation and creates the potential to reason about cause and effect. These preconditions and effects should be defined in other ontology(s), thereby capturing a meaningful connection, and supporting a deeper reasoning about the activities and the rest of the domain. This enables questions such as: *What class of objects could satisfy the precondition for some activity? What activity could have caused a change in some object? Will the precondition for this activity still be satisfied if something in the domain changes?*

Due to the prevalence of the Web Ontology Language (OWL) and the implications this has for shareability through the Semantic Web, the iCity project has committed to the use of OWL for the development of the ontology to represent its knowledge base. One objective of the project is to facilitate the distribution of its various research efforts with other groups. OWL is a desirable implementation language in this case because it provides a widely accepted, formal language to support this. Further, due to its role as the de facto standard language for the Semantic Web, vast amounts of semantically-annotated data are available in OWL. This makes it a compelling representation language choice in order to easily access and integrate this information with project data. While it may be necessary to develop extensions in other languages to support more complex reasoning tasks, the iCity project requires that data be captured and encoded in OWL.

Unfortunately, existing OWL ontologies are lacking with respect to the basic requirements for an activity specification. This work presents the Activity Specification[1] content ontology design pattern [4] to fill this gap in Semantic Web resources. In the section that follows, we review existing approaches to defining preconditions and effects in OWL and explain why they are not suitable solutions. Then, informed by earlier work on representations of fluents and activities, we propose an ontology design pattern capable of satisfying this basic set of requirements for activity specifications. It is applied with minimal reliance on the underlying activity ontology and is described such that this novel approach may easily be applied to other activity or event ontologies to obtain a similar solution.

2 Related Work

There are many OWL ontologies that in some way address the concept of activities. Event ontologies such as *The Event Ontology* [18, 14] focus only on occurrences, and so do not completely capture an activity specification. While *F-model* [17] does include the concepts of cause and effect, they are only defined at the occurrence level and specified as other events rather than states. Other OWL ontologies that do include the concept of activities and their preconditions and effects, such as the activity ontology by Riboni and Bettini[15] and the activity pattern presented by Abdalla and colleagues [1], provide no detail on the semantics of preconditions and effects, nor do they offer any guidance for how the states themselves should be formalized.

[1] http://ontologydesignpatterns.org/wiki/Submissions:ActivitySpecification

Although it is not discussed in the documentation, DUL+dns [3] does support a means of defining preconditions and effects as states (Situations). However, this ontology requires the use of reification for the N-ary relations approach to capture changes in states. The reader is directed to [13] for more detail on this approach in OWL. There are known issues that arise with reification (discussed in [19, 11], for example). In particular, it imposes limitations on reasoning, complicates reuse, and requires the introduction of additional terms. These issues are especially problematic for the iCity project and likely many other applications that aim to reuse existing foundational and domain ontologies (and vocabularies) wherever possible.

3 The Solution: A Pattern for Activity Specification

The proposed solution adopts a view of causality similar to the Event Calculus [10], employing the concept of manifestations to describe the states (fluents). The representation of activity specifications is based on the activity clusters introduced by Fox, Sathi, and colleagues [2, 16]. A precursor to the TOVE [6] and PSL [5] activity ontologies, an activity cluster provides a basic structure for representing activity specifications. Illustrated in Figure 1, it consists of an activity connected to an enabling and caused state, each of which may be a state tree that defines complex states via decomposition into conjunctions and disjunctions of states. It is important to clarify that in this work, an activity is

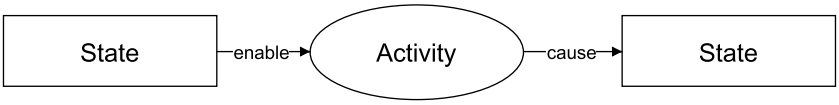

Fig. 1. A generic activity cluster, as represented in earlier work by Fox, Sathi, and colleagues [2, 16].

interpreted as a class of occurrences. There are arguments for and against this approach, as compared to a more traditional AI approach where activities are separate entities related to occurrences via an *occurrence of* relation, however this debate is outside of the scope of this work[2]. Before presenting the pattern for activity specifications, the concept of a manifestation is introduced to provide the required background. Then, a basic activity ontology is introduced and we describe the intuition of our solution: interpret manifestations as states, and define preconditions and effects using these states to achieve the required definition of an activity specification. Finally, we present the pattern as a whole.

[2] It is worthwhile to note that a similar solution can be applied for this alternate representation as well.

3.1 Manifestations

The representation of fluents in OWL via a 4-dimensionalist (4D) view was originally proposed by Welty, Fikes and Makarios [19]. Rather than relations being fluent and varying between entities as a function of time, in this approach relations simply hold (or do not hold) between temporal parts of the entities. This 4D view was later reinterpreted by Krieger [11], offering several practical advantages over the original approach. In particular, it eliminated the need for re-writing atemporal domain ontologies for reuse in a 4D representation. This reinterpretation has been implemented for the iCity project; the Ontology of Change[3] introduces two key classes to represent a concept that is subject to change: Manifestation and TimeVaryingEntity. The TimeVaryingEntity class corresponds to the invariant part of the concept. It is defined by properties that do not change over time. As per Krieger [11], TimeVaryingEntities are viewed as perdurants (things that occur over time, i.e. processes). A TimeVaryingEntity has Manifestations that demonstrate its changing properties over time, and these Manifestations exist at some Instant (or possibly Interval) in time. In addition to the manifestationOf relationship between a TimeVaryingEntity and a Manifestation, the ontology introduces the sameTimeVaryingEntity property for a Manifestation, to capture other Manifestations of the same TimeVaryingEntity.

The Activity Specification design pattern requires that the domain of interest is defined to capture change via this approach. To implement this representation in a particular domain requires that the changing concept (e.g. the Vehicle class) is defined as a subclass of Manifestation, and its invariant (perdurant) part is introduced as a subclass of TimeVaryingEntity. This process turns out to be relatively straightforward and has been submitted as an Ontology Design Pattern[4] as described in related work on a logical design pattern for change [9].

3.2 Activity and Time Ontologies

In this pattern an intentionally weak commitment is made in the definition of an Activity in order to focus on the representation of preconditions and effects and ensure its relevance to a wider audience. Should the pattern be used as-is, it is likely that some desired semantics may be found lacking. In particular, this simplistic representation does not account for composition of, or ordering over activities. The resulting ontology can easily be extended with a stronger definition of an Activity. Likewise, the pattern can easily be applied to other activity or event ontologies. It assumes only that an Activity is something that occurs over some point in time:

$\mathsf{Activity} \sqsubseteq \exists_{=1}\ \mathsf{occursAt}.\mathsf{Interval}$

It is also useful to include beginOf and endOf properties for an Activity to describe when it starts and ends. Assuming the use of a time ontology that defines a couple of basic relations regarding the start and end of an Interval,

[3] Available at: https://w3id.org/icity/iCity-Change

[4] http://ontologydesignpatterns.org/wiki/Submissions:Change_of_Time_Varying_Entities

these properties are easily definable through object property chaining with the occursAt property as follows:
occursAt o hasBeginning $\rightarrow$ beginOf
occursAt o hasEnd $\rightarrow$ endOf
Our implementation employs the OWL-Time ontology [7], however this is not a requirement of the solution. The pattern is relatively agnostic as to what time ontology is used; it requires only some notion of time points and intervals such that we can refer to when an activity is occurring, as well as when it begins and ends.

3.3 Manifestations as States in an Activity Specification

Observe that a Manifestation corresponds to (part of) a state of the world. Thus, similar in intuition to the Event Calculus approach to representing causality with fluents [10], manifestations may be interpreted as states that describe the preconditions and effects of activities. It is interesting to note that according to this view, both activities and their preconditions and effects are perdurants. However, in this pattern there is a distinction between Activities and States because Activities, (as opposed to TimeVaryingEntities) do not have manifestations[5].

Consider the activity, DriveToWork. A precondition of this activity might be that there is a vehicle with some gas. Rather than define the precondition relationship between individuals, the DriveToWork Activity can be defined by describing the class of manifestations that satisfies the precondition of a Vehicle with gas[6]:
$\mathsf{DriveToWork} \sqsubseteq \exists\ \mathsf{hasPrecondition}.(\mathsf{Vehicle} \sqcap \forall\ \mathsf{hasGas}.(\mathsf{Gas} \sqcap \mathsf{hasVolume} > 0)$
Admittedly, this is a very naive representation. More accurately, the volume of gas would required to be not only greater than zero, but some minimum value, (dependent on the distance to work). This has been simplified for presentation purposes. Complex states and the relationships between them will be addressed subsequently.

Effects of activities can be defined similarly, as classes of Manifestations. The corresponding ontology representation for this example is shown in Figure 2. This approach captures the semantics of the entity(s) that is a precondition or effect of the activity (in this case, the definition of Vehicle, which might be imported from some other ontology), while also accounting for its temporal, variable nature. In other words, the same vehicle that satisfies that precondition at some time t_1, may no longer satisfy the precondition at a later time, t_2.

Complex States The activity of driving to work may require not only a car that has gas, but also that there is a person available to drive it. An activity

[5] While this is not explicitly precluded in the axioms, we do not find it useful or necessary to require it.

[6] Note that the examples presented here assume the existence of the relevant domain ontology(s), using the representation of change described previously; in this case one that defines Vehicles, including properties and concepts such as its volume of gas.

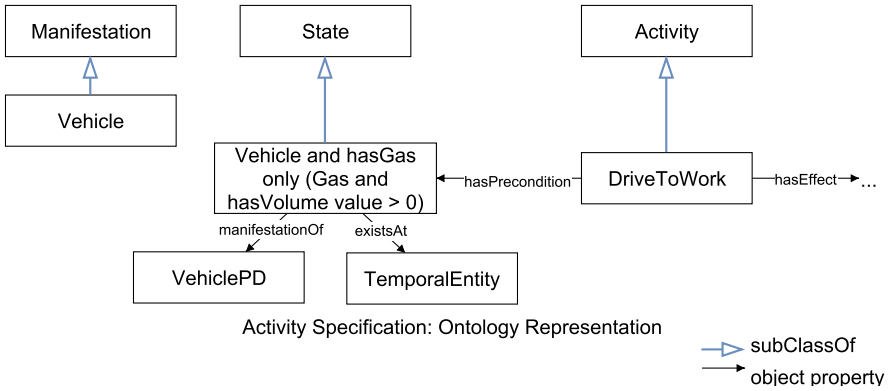

Fig. 2. A simple example illustrating the use of the Manifestation class to represent a State in an activity specification.

specification must be able to capture such instances where the precondition or effect is a *complex* state, involving multiple manifestations. To address this, the state tree approach used in the specification of activity clusters [16] is adopted. Two subclasses of State are introduced: a non-terminal and a terminal state. A TerminalState has no "children" (i.e. is not a complex state), and therefore refers directly to a class of manifestations. In fact, TerminalState is the subclass of Manifestations that are a precondition or effect of some Activity.

On the other hand, a NonTerminal state has child states (defined with the hasDecomp property), which may themselves be Terminal or NonTerminal. A NonTerminal state may be conjunctive or disjunctive. Naturally, a ConjunctiveState is defined by the conjunction of its child states, whereas a DisjunctiveState is defined by the disjunction of its child states. The resulting state tree may be multiple levels deep, comprised of a combination of manifestations that form the Terminal States (i.e., the leaves of the state tree).

3.4 The Representation

The resulting ontology design pattern, illustrated in Figure 3, defines the following key classes for an activity specification: Activity, State, TerminalState, NonTerminalState, ConjunctiveState, and DisjunctiveState. It provides the infrastructure necessary to more accurately define the preconditions and effects of an activity, and may be implemented with alternate theories of activity and time. For an example of a complete implementation of the pattern, the reader is referred to the iCity Activity Ontology[7]. The representation of an activity specification focuses on capturing preconditions and effects at the general, activity level. In order to define the *semantics* of these conditions we must consider the implications for individual occurrences and states – in particular, the relationship between when an activity occurs and when a state is true. The status of a

[7] Available at: https://w3id.org/icity/iCity-Activity

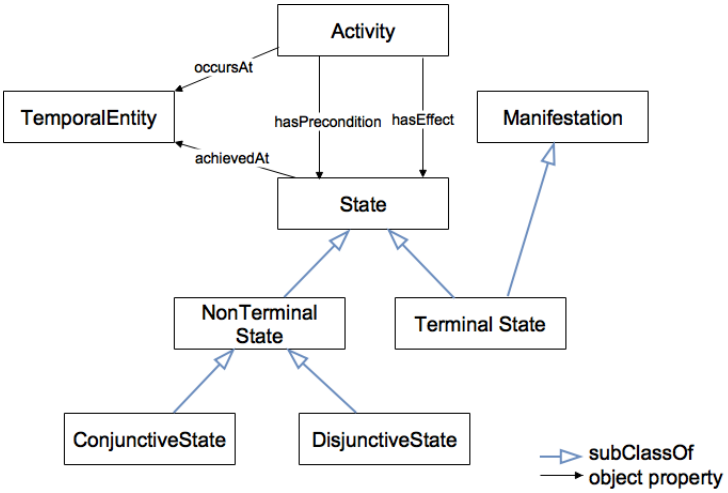

Fig. 3. Depiction of the Activity Specification design pattern.

state is further defined in the ontology via the object property achievedAt. The precondition of an activity must be true (achievedAt) when the activity begins:
inverse(hasPrecondition) o beginOf → achievedAt
Similarly, effect must be true (achievedAt) when the activity ends:
inverse(hasEffect) o endOf → achievedAt

Both Manifestations and States have a temporal projection. The ontology should also describe the relationship between the status of a state and the existence of its related manifestation(s). This requires the introduction of specialized sub-properties of achievedAt for each type of State.

Terminal states are the simplest sort of State. A specialization of the achievedAt property for terminal states, terminalAchievedAt, can be defined as a subproperty of existsAt, (the property that defines the temporal extent of a Manifestation).
terminalAchievedAt subPropertyOf existsAt

Conjunctive and disjunctive state do have substates. A ConjunctiveState is achieved only when *all* of its decomposed states are achieved. A DisjunctiveState is achieved at some time if *any* of its decomposed states is achieved. This intended semantics for conjunctive and disjunctive states is illustrated in Figure 4, where t_3 and t_4 depict the intervals at which the non-terminal state s is achieved, as a conjunctive or disjunctive state, respectively. Ideally the conjunctiveAchievedAt and disjunctiveAchievedAt properties would be defined as follows:

$$\begin{aligned}(\forall s,t)conjunctiveAchievedAt(s,t) &\iff (\forall s_1,t_1)hasDecomp(s,s_1)\\ &\quad \wedge achievedAt(s_1,t_1) \wedge during(t,t_1)\\ (\forall s,t)disjunctiveAchievedAt(s,t) &\iff (\exists s_1,t_1)hasDecomp(s,s_1)\\ &\quad \wedge achievedAt(s_1,t_1) \wedge during(t,t_1)\end{aligned}$$

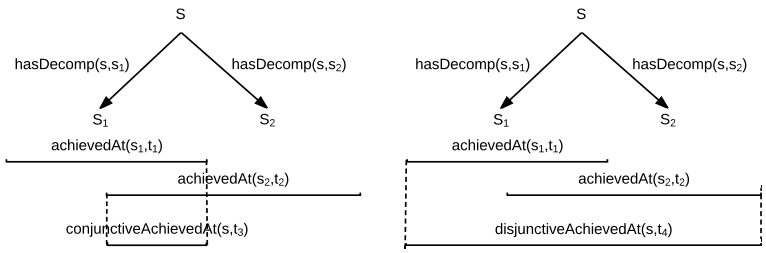

Fig. 4. A simple example of conjunctive and disjunctive states and their associated achievedAt properties

However, these axioms are beyond the expressive capabilities of OWL. Instead, this semantics is approximated for conjunctive states with the use of object property chaining:
inverse(hasDecomp) o conjunctiveAchievedAt o during $\rightarrow$ achievedAt
A similar axiom for disjunctiveAchievedAt is conceivable, though not expressible in OWL as it results in a cyclic dependency with the achievedAt property. Collectively, the pattern for activity specifications is shown in Figure 5.

Relationships Between States In addition to the composition of states and their status, there may exist relationships between the states in an activity specification. A simple example of this can be seen by considering the additional precondition of a driver for the DriveToWork activity. Adding this precondition results in a complex (conjunctive) state; the resulting specification is illustrated as an activity cluster, and using the Activity Specification design pattern in Figures 6,7 (respectively). Based on the design pattern, it is defined as follows:
DriveToWork $\sqsubseteq \forall$ hasPrecondition.DTWPre
DTWPre $\sqsubseteq$ ConjunctiveState
DTWPre $\sqsubseteq \exists$ hasDecomp.Person
DTWPre $\sqsubseteq \exists$ hasDecomp.(Vehicle $\sqcap \forall$ hasGas.(Gas $\sqcap$ hasVolume< 0))
These axioms provide a correct representation, however they omit a stronger relationship that is likely present between the Person and Vehicle states: they should share the same location before the activity can occur. Similarly, there may be relationships between the precondition and effect states of an activity. One effect of the DriveToWork activity might be a state where a Vehicle is at some particular location (work). Intuitively, this is the same vehicle as that which satisfied the precondition, however the same vehicle at different points in time corresponds to a *different* Vehicle that is a Manifestation of the same TimeVaryingEntity. This common relationship between preconditions and effects can be expressed using the sameTimeVaryingEntity property described earlier.

Within OWL these relationships may be expressed at the individual level. Additionally, some relationships are expressible when more precise classes of activities are defined. For example, the activity *Alice drives to work* allows for the

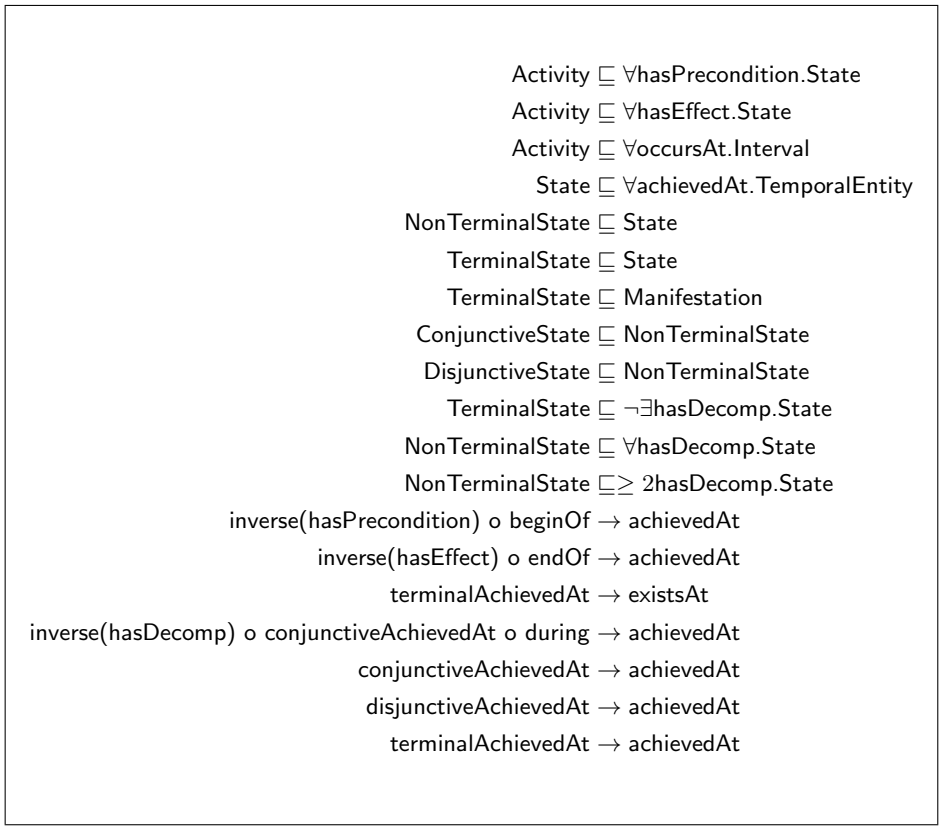

Fig. 5. The Activity Specification Content Design Pattern

definition of a complex precondition with a vehicle at Alice's home, and Alice at her home; the shared location is defined with respect to the activity. A more general axiomatization of these inter-state relationships is beyond the representation abilities of OWL, thus while the representation is capable of capturing activity specifications, there is limit with respect to the type of reasoning that may be supported.

It is not surprising that enforcing a more advanced semantics is not possible in OWL, nor should it be a source of concern. In our experience, it is common practice to define the necessary concepts in a "neat" way on the Semantic Web, while reasoning is implemented in a "scruffy" way with external applications supplementing the required semantics. Should more advanced reasoning be required, it is possible to define these relationships between states outside of the OWL representation. For example, in SWRL [8]:
DriveToWork(?a), hasPrecondition(?a,?s_1), hasDecomp(?s_1,?v), Vehicle(?v), hasEffect(?a,?s_2), Vehicle(?s_2) → sameTimeVaryingEntity(?s_2,?v)

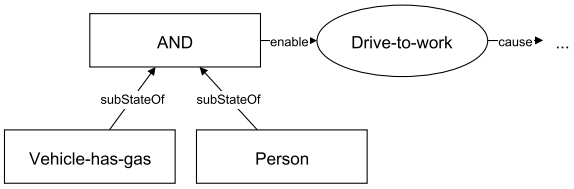

Fig. 6. An example representation of a complex precondition as an activity cluster.

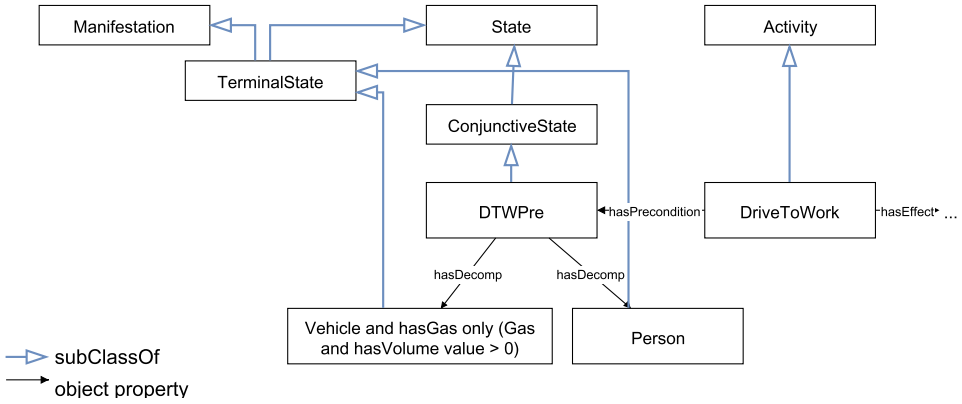

Fig. 7. An example representation of a complex precondition captured with the Activity Specification pattern.

Likewise, stronger definitions of the precondition, effect, and achievedAt properties are possible if we consider extensions beyond OWL. That being said, limitations in OWL expressivity should not be a deterrent from developing a representation capable of approximating the semantics and capturing the relevant data. The Activity Specification pattern provides a meaningful connection between the activity and the associated domain(s), in a way that captures causality and change over time. An OWL representation is critical to enable Semantic Web support for sharing, integrating, and linking the data. Should the OWL representation be unable to support all of the key reasoning tasks, this can be addressed at alternate levels of implementation (scruffy, or otherwise).

4 Conclusion

This work presents a novel approach to capturing the semantics of preconditions and effects in OWL based on existing solutions for the representation of fluents. The proposed ontology design pattern adopts an approach that supports an ease of reuse of existing domain ontologies that is not possible with reification. Further, it is specified with a weak semantics of Activities that may be easily reused and strengthened as required (e.g. for complex activities), or integrated with existing Activity ontologies. This work provides a foundation for the adoption of a meaningful representation of activity specifications on the Semantic Web. Future work should look toward capturing other aspects such as resource consumption and participation. The notion of relative strength or priority between alternative preconditions and effects would also be an interesting extension.

While not the focus of this pattern, an additional consequence of this approach is the ability to easily develop more complex representations by extending the representations of the Activity and State objects. For example, spatial information is a particularly common requirement: spatial knowledge about a precondition or effect may be defined by capturing the location of the Manifestation. Other possible extensions are concepts of participation, activity composition, and specializations of the precondition and effect properties.

As noted, this work is motivated by a project on urban informatics [12]. We gratefully acknowledge support provided by the Ontario Ministry of Research and Innovation through the ORF-RE program.

References

1. Abdalla, A., Hu, Y., Carral, D., Li, N., Janowicz, K.: An ontology design pattern for activity reasoning. In: Proceedings of the 5th International Conference on Ontology and Semantic Web Patterns-Volume 1302. pp. 78–81. CEUR-WS. org (2014)
2. Fox, M.S.: Constraint-Directed Search: A Case Study of Job-Shop Scheduling. Ph.D. thesis, Computer Science Department Carnegie-Mellon University (1983)
3. Gangemi, A.: Dolce+dns ultralite, http://lov.okfn.org/dataset/lov/vocabs/dul
4. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on ontologies, pp. 221–243. Springer (2009)

5. Grüninger, M.: Using the psl ontology. In: Handbook on Ontologies, pp. 423–443. Springer (2009)
6. Gruninger, M., Fox, M.S.: An activity ontology for enterprise modelling. Submitted to AAAI-94, Dept. of Industrial Engineering, University of Toronto 321 (1994)
7. Hobbs, J.R., Pan, F.: An ontology of time for the semantic web. ACM Transactions on Asian Language Information Processing (TALIP) 3(1), 66–85 (2004)
8. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al.: Swrl: A semantic web rule language combining owl and ruleml. W3C Member submission 21, 79 (2004)
9. Katsumi, M., Fox, M.: A guide to representing change over time in owl. In: Submitted to: WOP 2017 8th Workshop on Ontology Design and Patterns
10. Kowalski, R., Sergot, M.: A logic-based calculus of events. New generation computing 4(1), 67–95 (1986)
11. Krieger, H.U.: Where temporal description logics fail: Representing temporally-changing relationships. In: Annual Conference on Artificial Intelligence. pp. 249–257. Springer (2008)
12. Miller, E.J.: icity: Urban informatics for sustainable metropolitan growth; a proposal funded by the ontario research fund, research excellence, round 7. Tech. rep., University of Toronto Transportation Research Institute (2014)
13. Natasha Noy, Alan Rector, P.H.C.W.: Defining n-ary relations on the semantic web (2006), https://www.w3.org/TR/swbp-n-aryRelations/
14. Raimond, Y., Abdallah, S.: The event ontology. Tech. rep., Citeseer (2007), http://motools.sourceforge.net/event
15. Riboni, D., Bettini, C.: Owl 2 modeling and reasoning with complex human activities. Pervasive and Mobile Computing 7(3), 379–395 (2011)
16. Sathi, A., Fox, M.S., Greenberg, M.: Representation of activity knowledge for project management. IEEE Transactions on pattern analysis and machine intelligence (5), 531–552 (1985)
17. Scherp, A., Franz, T., Saathoff, C., Staab, S.: F–a model of events based on the foundational ontology dolce+ dns ultralight. In: Proceedings of the fifth international conference on Knowledge capture. pp. 137–144. ACM (2009)
18. Van Hage, W.R., Malaisé, V., Segers, R., Hollink, L., Schreiber, G.: Design and use of the simple event model (sem). Web Semantics: Science, Services and Agents on the World Wide Web 9(2), 128–136 (2011)
19. Welty, C., Fikes, R., Makarios, S.: A reusable ontology for fluents in owl. In: Formal Ontology in Information Systems (FOIS). vol. 150, pp. 226–236 (2006)