iCity Transportation Planning Suite of Ontologies

Commented [MK1]: TO DO
- IT-SoS description
- Clean up appendices
- Review entire document

Megan Katsumi
Postdoctoral Fellow
Enterprise Integration Lab
Mechanical and Industrial Engineering
University of Toronto

Mark Fox
Distinguished Professor of Urban Systems Engineering
Professor of Industrial Engineering and Computer Science
Director of the Centre for Social Services Engineering
University of Toronto

Version 1.2

Last Updated: January 30, 2020

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING
Transportation Research Institute

UTTRI

1 Purpose

The purpose of this document is to present the current release of the iCity ontologies. Complementary HTML documentation is automatically generated for each ontology from its OWL file using Widoco[1]; when accessed with a web browser, each iCity ontology IRI dereferences to this documentation.

Commented [MK2]: TPSO

2 Scope

The iCity ontologies define the concepts required to represent the urban system and its behaviour, as informed by work undertaken by the iCity-ORF project teams.

This report includes documentation of the contents of the iCity ontology, along with recommendations for its implementation and maintenance, and examples of its application in the iCity-ORF project. The intended semantics of the ontology's concepts are described in natural language, followed by an overview of the axioms that capture, or in some cases approximate, this semantics. The iCity is made up of sub-ontologies that are axiomatized in OWL 2 {Grau, 2008, OWL 2: The next step for OWL. This report does not go into detail addressing the concepts defined in reused, external ontologies, except where necessary to describe concepts introduced in the iCity ontologies. The reader is referred to the original documentation for these ontologies as required.

3 Role of the Ontology

All of the projects within iCity-ORF are situated in the urban domain, therefore it is not surprising to find many common concepts between them. As such, it stands to reason that some integration between the different applications should be possible. For example, if data is collected about the population, it should be usable by various simulations such as ILUTE [1], but also by the projects developing analysis tools, such as the smart parking application. Unfortunately, there is also ambiguity in how different concepts are used; the same concept may be defined differently in different applications. This provides a challenge not only for integration

[1] https://github.com/dgarijo/Widoco

of the iCity applications, but for shareability and reuse of results: if the knowledge generated by iCity is not defined sufficiently, it will be difficult for any other researchers to understand and leverage it.

The key purpose of the iCity Ontology is to address these challenges of data integration and reuse. The iCity ontology provides a common set of terms with which data can be stored and accessed. The ontology will resolve any ambiguities and disagreements between terms by defining a common set of concepts that completely captures the domain, with agreed-upon definitions. In the case that two applications attribute a different meaning to the same term, the result will be two distinct terms with distinct, precisely defined meanings. In this way we can recognize these differences and clearly identify the relationships between different concepts. The ontology will be used to organize and describe data within the iCity project. It may also be used to support the publishing or sharing data with the research community.

The resulting artifact, often referred to as the *knowledge base* will take the form of a triple-store(s), created by mapping data from the iCity applications to the agreed-upon terminology defined in the iCity ontology. In future work, an alternative architecture may be explored wherein some or all of the data is maintained in its original location, such as a relational database, and accessed via mappings to the ontology. The high-level architecture for the ontology's implementation in the context of the iCity project, is illustrated in Figure 1.

Another purpose of the ontology is to support automated reasoning. Owing to the formal logic that the ontology is encoded in, its axioms are capable of supporting data validation and inference of the information stored in the knowledge base.
The precise and formal nature of the ontology will support the use of services such as inference and data validation. Based on the definitions, we may be able to infer new information that was not originally part of the knowledge base. Data validation is supported as a result of the consistency-checking mechanism. We also hope that identification of relationships may serve to uncover synergies between the projects, by illustrating how data from one project may serve to inform the work of another.

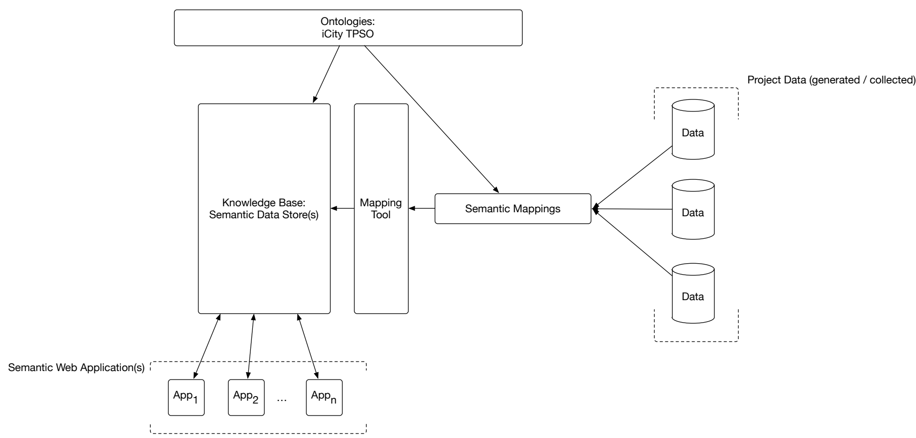


Figure 1: iCity Knowledge Base High-Level Architecture

The sections that follow introduce the ontology required achieve this, in particular, to define the urban system. Beyond this basic architecture, the iCity Ontology may be implemented to support specific applications. Examples of this are discussed in further detail in Section 9.

4 Development Approach

The ontologies presented in this report have been developed based on the guidelines for ontology development set out by both [2] and [3]. This combined approach may be described in terms of the following six activities, the outcomes of which are outlined in subsequent sections of this report.

1. Requirements gather: developing a clear understanding of the domain and required scope of the ontology. This activity is facilitated with the identification of motivating scenarios and eventually made more precise with the specification of Competency Questions. Competency Questions are queries that the ontology should be capable of representing and answering and may be thought of as analogous to functional requirements in Software Engineering.
2. Reuse: where possible, ontologies that were suitable for reuse to (partially) satisfied the requirements were identified. We also considered the reuse of existing vocabularies as appropriate.

3. Ontology Design: the definition of classes, the class hierarchy, and properties was tightly linked and iterative, rather than sequential as described by [2]. These terms were identified with a combined approach: bottom-up design from the required datasets, and top-down from the areas recognized as important to the domain. The reference to "facets" and "slots" by [2] can be interpreted as the task of defining axioms (primarily subsumption and equivalence, but also pertaining to object property characteristics) that relate the classes and properties with one-another.
4. Evaluation: the task of evaluation is an important step that is addressed in several ways in this project:
    a. *Consistency*: The ontologies shall be classified using an automated reasoner to demonstrate consistency of the definitions and absence of unsatisfiable classes.
    b. *Competency:* The ontologies shall be assessed against the Competency Questions specified in the Requirements stage.
5. Application: The ontology shall be applied in a variety of case studies to serve as a concrete demonstration of its viability as a solution for several of the motivating scenarios. A key aspect in all applications will be the definition of instances via the ontology. This will be accomplished with the specification (and materialization) of R2RML mappings from existing data sources to the ontology. These mappings illustrate the adequacy of the ontology to capture data from relevant sources. The processes involved are described in greater detail later in the report.

5 Requirements

In an effort to clarify the domain and scope of an ontology for transportation planning, interviews were conducted and relevant documents were reviewed to reveal required competency areas in two key knowledge categories, outlined in Table 1. The major concepts identified in this effort would eventually form many of the individual ontologies that the iCity TPSO is built from.

Table 1: Key competency requirements.

Knowledge Category	**Competency Areas**

Urban System Characteristics	• Population:
	  ○ People
	  ○ Households

	○ Jobs ○ Schedules ○ Means of travel • Land Use ○ Types of land use ○ Occupied space • Transportation ○ Road networks ○ Transit networks ○ Transportation modes and characteristics of (e.g. access points) ○ Transportation vehicles and characteristics (e.g. capacity, speed, accessible routes/networks)
Urban System Behaviour	• Demographic Update: changes to population (people, household structures) • Labour Market: changes to job situations • Housing Market: changes to housing situations • Auto Ownership: changes to auto ownership • Activity-Based Daily Travel: activity schedules and associated travel • Transportation Emissions & Dispersed Pollution Concentrations • Transportation events: scheduled trips, failures, scheduled maintenance

Delving deeper into the requirements for each research group, several motivating scenarios were identified. For each scenario, relevant data sets and competency questions were identified. The identification of relevant data sets served as a particularly useful source of requirements. Since data collection is a major task in transportation planning, the datasets provided a means of motivating the required scope and level of detail. The resulting ontology could then be assessed in terms of its ability to represent the competency questions as well as the information in the datasets.

Note: Requirements beyond motivating scenarios

The motivating scenarios provided precise, testable requirements and opportunities to apply the ontology in practice. However, scope of the resulting ontology extends considerably beyond the requirements dictated by the motivating scenarios and this should be addressed. This broader scope results from the interviews conducted with subject-matter experts, and the sample data that was provided at this stage. The motivating scenarios that were explored in greater detail were selected based on pragmatic criteria such as data availability, the stage of the various research projects. There exist many additional motivating scenarios, both within and beyond the iCity-ORF project that this ontology is intended to support. This should be explored in future work.

5.1 Motivating Scenario: Land Use and Transportation Simulation

Reviewing the results of large-scale simulations, such as those generated by the ILUTE [1] and TASHA [4] models, can be challenging. The ontology can be used to capture the output and support question-answering to explore the results. Maintaining the data that serves as input to these simulations also poses a challenge for researchers. The ontology may be used to capture and relate historical data to improve access for researchers.

- CQ1-1: What trips originated from/ended in a given zone?
- CQ1-2: What is the occupation breakdown of the travelers whose trips originated/ended in a given zone?
- CQ1-3: What were the purposes of the trips that originated/ended in a given zone?
- CQ1-4: In a particular time period, how many trips originated/ended in a given zone?
- CQ1-5: What were the transportation mode(s) taken by trips that originated/ended in a given zone?
- CQ1-6: Who are the members of a particular household?
- CQ1-7: What trips were performed, by which members of a particular household?
- CQ1-8: What were the purposes of the trips performed by members of a particular household?

CQ1-9: What is the age, sex, and occupation of the traveler who performed a particular trip?

CQ1-10: What land use classification is associated with a particular parcel?

5.2 Motivating Scenario: Transit Research

Transit research activities will often involve collecting, integrating, data from various sources. For example, researchers may need to combine data from various parts of the transit system to assess how some failure event, for example on a streetcar line, may impact nearby bus routes. Even assessing data about a single transit route may require the integration of various datasets, such as data describing the route itself, data describing the actual behavior of vehicles on the route, data on the vehicle's characteristics, and perhaps contextual information such as ridership. The ontology may be employed to facilitate the integration of transit, thereby supporting easier access to information of interest.

In the iCity-ORF project, two areas of transit research were strategies to prevent streetcar bunching, and transit resilience strategies (so-called "bus bridging" where buses are re-routed to serve as shuttle buses in order to delays on the subway lines). As an initial step toward supporting these research areas, we elected to focus on supporting the transit resilience strategy project. This required support for queries to support the detection buses that had been re-routed as shuttle buses. This information could then be used to further analyze the bus bridging strategy and assess its impact on the network.

CQ2-1: What date and time has a subway incident occurred?

CQ2-2: What are the locations of vehicles on a particular route after the occurrence of a subway incident?

CQ2-3: Are any buses located more than a certain distance from their assigned route at a given point in time?

5.3 Motivating Scenario: Smart Parking Applications

Through a tripartite research agreement on transportation and smart cities, a forward-looking motivating scenario for the Chinese University of Hong Kong (CUHK) was identified. Researchers at the CUHK have been investigating the potential for smart parking applications, especially in the context of electric vehicles. Providing parking information to drivers, whether real-time or static, is useful in helping them to locate a suitable parking spot. The question of suitability is complicated for drivers of electric vehicles, as they may require a parking location with access to a particular type of charger. Researchers at the CUHK were investigating the potential for such smart parking applications, and identified an opportunity to use ontologies to facilitate the access and integration of the required data. Based on the envisioned use cases and the currently available data, the following set of competency questions was identified:

CQ3-1 What is the address of the parking lot P?

CQ3-2 What is the capacity of parking lot P?

CQ3-3 Is it accessible by disabled people, and if so how many parking spots are for disabled vehicles?

CQ3-4 Is there a height limit for vehicles for a parking lot P?

CQ3-5 What are the geographic coordinates for parking lot P?

CQ3-6 What building is a particular parking lot located in?

CQ3-7 Is a particular parking lot open to the public at a given time?

CQ3-8 How much does it cost to park in a particular parking lot?

CQ3-9 What types of payment are accepted at a particular parking lot?

CQ3-10 How many parking spots are designated for electric vehicles in a particular parking lot?

CQ3-11 What types of electric vehicle chargers are available in a particular parking lot?

Opportunities for additional competency questions were identified as possibilities for future work, as more data sources become available.

5.4 Motivating Scenario: ATIS via IT-SoS

The tremendous amount and diversity of data generated by ITS (Intelligent Transportation Systems) has become an important source for its services and applications. Travelling from one place to another often involves different information from different ITS services. Unfortunately, the multiplicity of ITS and their complexity has produced a body of heterogeneous data that cannot easily be integrated. Data from different sources must be analyzed, classified and re-organized into a homogenous format to make it universally applicable.

To address this challenge, an architecture has been designed to support scalable and extensible ITS applications using a semantic representation and integration. Many institutions and companies have developed ICT solutions to close the gap and manage data integration and representation by using well-known industrial protocols likes GTFS. Nevertheless, these solutions lack a formal semantics; there is no common standard across systems to manage and exchange data and information.

ITS tools require integration of many heterogeneous data sources. Adaptability is challenging for traditional ITS frameworks due to the overhead to integrate new and changing data sources. The IT-SoS architecture, originally proposed by [5], is intended to leverage the ontology to support data integration. In general, the range of queries required to support the IT-SoS architecture will vary greatly as a function of the ITS application(s) to be supported. In the iCity-ORF project (1.2), the IT-SoS architecture was demonstrated by way of the Advanced Traveler Information System (ATIS). To support this implementation, the iCity ontology must be capable of capturing data and formulating competency questions regarding the traffic status data on various road segments in the transportation network.

CQ4-1: What are the averages of the TTI_Max values that have been observed over some period of time?

CQ4-2: What are the averages of the TTI_Max values that have been observed at some location?

CQ4-3: What are the averages of the TTI_Max values that have been observed at some location, over some period of time?

In the questions above TTI_Max refers to the Maximum Transportation Travel Index; TTI_Max is a measurement used to indicate traffic conditions by way of a comparison of the observed rate of travel to the maximum throughput speed on a road segment. The questions were specified with respect to the average value because at the time of this work the ATIS application was restricted to work with loop detector readings that had been aggregated over road segments and ten-minute intervals in time.

5.5 Motivating Scenario: ArcGIS Query Support

ESRI Canada provides geospatial information system (GIS) solutions used for transportation research, urban planning, and a variety of other applications. These tools provide users with a wealth of data and powerful tools for visualization and analysis. Despite this, query formulation and revision can be challenging, in particular for less experienced users. These difficulties may be addressed with use of an ontology to formalize the terms of interest and provide a single interface with which complex queries may be formulated. Streamlined access to the geospatial data in ArcGIS will support all users, and may be particularly valuable for use cases for ongoing work on NextGen-911 services.

The same ontology developed to provide query support might serve as a specification of recent standardization efforts by the Canadian Transportation Infostructure Initiative (CTII). The CTII is currently working to develop a Community Map of Canada – a complete and accurate base map of Canada that is created by integrating data from various municipalities and other regions. Central to this initiative is the GeoFoundation Exchange (GFX), an effort to collect, unify, and publish base map data. Beyond this, there may be opportunities to employ the ontology for automated reasoning services such as classification or validation.

An initial set of CQs was identified to explore the use of the ontology for query support. These CQs are example queries requiring the combination of multiple GFX datasets to retrieve the required information. They are derived from a prototype application that requires contextual information about a particular route.

CQ5-1: What neighbourhood(s) does a particular route go through?

CQ5-2: What types of land use does a particular route go through?

CQ5-3: What types of land cover does a particular route go through?

CQ5-4: What points of interest does a particular route pass by?

CQ5-5: What types of road does a particular route travel on?

CQ5-6: What (if any) parts of a route travel on a road segment that is above grade?

CQ5-7: What (if any) parts of a route travel on a road segment that is below grade?

6 Urban System Characteristics and Behaviour

In the urban system, we recognize the following key concepts that must be defined:

- Person
- Organization
- Household
- Building
- Parking
- Vehicle
- Transportation Networks

- Public Transit
- Land Use
- Travel

The semantics of each of these concepts will be defined by a generic ontology. These generic ontologies will then be used in the iCity ontology to define the urban system and its behaviour; its population, land use, transportation infrastructure, and the travel that occurs within it. This representation may then be extended to capture the individual iCity applications so that they may be integrated with one another and sufficiently well-defined so as to be shareable and reproducible with the research community. Foundational Ontologies will be also required in order to define the core concepts that apply across the transportation domain. These will be introduced first, followed by the presentation of each generic ontology in more detail.

6.1 Foundational Ontologies

In addition to the concepts that are specific to an urban system, there exist foundational concepts that are required to fully define the domain. In particular, the foundational ontology captures the concepts of time, space, change, activities, and resources; each concept is defined its own sub-ontology.

6.1.1 Location Ontology

http://ontology.eil.utoronto.ca/icity/SpatialLoc.owl

To effectively capture the location of some object, several concepts must be introduced. First, a distinction must be made between the object and its location. Objects have some location, that is the region in space – a so-called spatial feature – that they occupy. The ontology must not only support a representation of these concepts, but a representation of relationships between spatial features. In particular, topological relationships are important as they allow for the identification of how one area in space is situated relative to another. For example, is one area contained in another? Are two areas disconnected?

Finally, to precisely describe the location of an object in space, some notion of geometry is required. This is important to represent the quantitative aspects of the feature, which may be represented as a point or perhaps some other area such as a polygon or a line.

6.1.1.1 The Ontology

The Location ontology reuses and extends the GeoSPARQL [6] standard to specify the concepts of interest. GeoSPARQL specifies the required vocabulary of spatial relations. It is particularly attractive as it has been published as a standard by the OGC; in addition, its defined relations are implemented, to various extents, as functions for querying spatial data by some knowledge base tools.

The ontology represents the location of objects using two key classes: Feature and Geometry, as shown in Table 2. A Feature is a spatial object, as opposed to a Geometry which is a more abstract object that may be used to describe the shape of some spatial object(s). The key properties, shown in Table 3, are largely made up of topological relations between Feature objects. In addition, the ontology specifies the property hasLocation to capture the relationship between non-spatial objects (e.g., train station) and the spatial locations they occupy. Similarly, the associatedLocation is introduced to capture the association of some non-spatial object to a particular location. For example, a train station may occupy a fairly large spatial location but be associated with a particular point.

In order to capture the quantitative geospatial information, spatial features may be associated with geometry objects, via the hasGeometry property. These geometries may then be encoded with coordinate information through the specification of WKT (well-known text) values with the data property asWKT. The default reference system for the coordinate values is assumed to be WGS84. While the GeoSPARQL specification allows for the identification of alternate reference systems, captured as IRIs and concatenated with the coordinates, it should be noted that current support is not widespread or standardized, therefore automated translation between these systems should not be assumed.

Table 2: Key classes in the Location Ontology

Object	Property	Value
geo:Feature	rdf:subClassOf	geo:SpatialObject
geo:Geometry	rdf:subClassOf	geo:SpatialObject

Table 3: Key properties in the Location Ontology

Property	Characteristic	Value (if applicable)
geo:sfEquals	Domain and Range	geo:SpatialObject
geo:sfDisjoint	Domain and Range	geo:SpatialObject
geo:sfIntersects	Domain and Range	geo:SpatialObject
geo:sfTouches	Domain and Range	geo:SpatialObject
geo:sfWithin	Domain and Range	geo:SpatialObject
geo:sfContains	Domain and Range	geo:SpatialObject
geo:sfOverlaps	Domain and Range	geo:SpatialObject
geo:sfCrosses	Domain and Range	geo:SpatialObject
geo:hasGeometry	Domain	geo:Feature
geo:hasGeometry	Range	geo:Geometry
hasLocation	Range	geo:Feature
associatedLocation	Range	geo:Feature
geo:asWKT	Range	geo:wktLiteral
as_nDLatLon	Domain	geo:Geometry

	Range	http://franz.com/ns/allegrograph/5.0/geo/nd#_lat_la_-9.+1_+9.+1_+1.-4_+1.-1_lon_lo_-1.8+2_+1.8+2_+1.-4[2]

6.1.1.2 An Example

For example, consider the location of a vehicle. A vehicle may be located at a person's home or work. Similarly, a transit vehicle may be located at some station, maintenance yard, or at some point on a particular transit route. The Spatial Feature where the vehicle is located may be represented by some geometry (e.g. a point), and may have relationships of interest with other spatial features. For example, the location of the vehicle may be contained in some other spatial feature (corresponding to a traffic zone, for example). The resulting representation is illustrated in Figure 2.

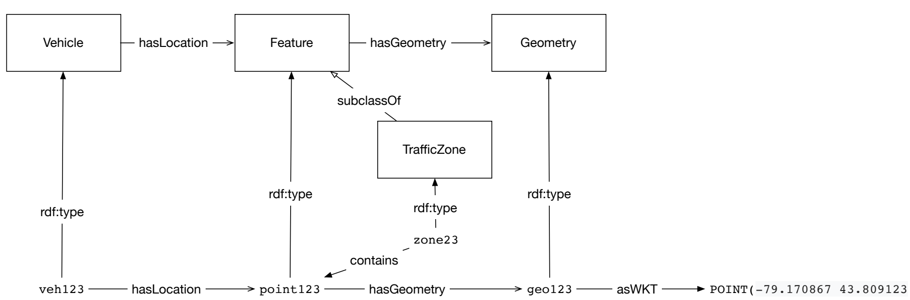


Figure 2: An example representation of a location information for a vehicle.

6.1.1.3 Future Work

The GeoSPARQL standard supports the identification of alternate coordinate reference systems, captured as IRIs and concatenated with the coordinates. However, support for translation between these systems is limited. Future work should address this in greater detail.

---

[2] AllegroGraph-generated nD datatype for lat-lon location data

Implicit in the description of a spatial feature and its geometry are its dimensions. In theory, properties such as area, height, and length may be inferred from the geometry of some spatial feature. However, in practice these properties are captured separately. Future work should attempt to formalize the relationship between these properties. If this relationship cannot be precisely captured in OWL, it should at least be represented in the OWL ontology, defined in natural language, and ideally captured in some more expressive extension.

Reused Ontologies:

1. GeoSPARQL: http://www.opengis.net/ont/geosparql#

6.1.2 Time Ontology

http://ontology.eil.utoronto.ca/icity/Time.owl

The concept of time is so pervasive that its definition is often taken for granted. In order to define an ontology for time, the objects of interest must be identified. What are the *things* that will be described? In general, three approaches to a representation of time have been identified: point-based, interval-based, and mixed. In a point-based representation, the objects of interest are timepoints. The passing of time is described as an ordering over time points, and periods of time may be represented as a series of timepoints. In an interval-based representation the objects of interest are time intervals, whereas the mixed representation includes both timepoints and time intervals. Key to all of these representations is that there is an ordering that holds over these time objects. We must be able to describe whether a time object is before another, and in the case of time intervals we must be able to describe other relationships such as whether one interval is contained in or overlaps with another.

6.1.2.1 The Ontology

Time is a concept that is fundamental, not only to transportation planning, but many other domains. For this reason, it is not surprising that a well-established ontology of time already exists, published as a W3C standard [7] and originally presented in work by [8] . This representation is reused directly, however rather than import the ontology directly into the transportation planning ontology, the time ontology is imported by a transportation-specific time ontology. This is done for two reasons: (1) It allows for the application of an organizational structure to the terms defined in the ontology; all classes are defined as subclasses of a TimeOntologyThing, similarly all object properties are subproperties of a TimeOntologyProperty, and likewise with data properties. These classes are superficial, but allow us to precisely organize the terms. This provides an added level of clarity in cases with large ontologies where multiple ontologies are imported. (2) In addition, it provides the flexibility for possible extensions to the time ontology in the iCity TPSO, while maintaining a clear relationship to the Time Ontology that is the W3C standard. In other words, any additions or changes may be made by defining new concepts in the transportation-specific time ontology, and *relating them* (e.g. via the subclass relation) to concepts in the W3C's Time Ontology standard. These new concepts will be clearly identifiable their IRI.

The Time Ontology adopts a mixed representation of time, including both time instant and time interval classes. Definitions of the key classes and properties in the Time ontology are depicted in Table 4.

Table 4: Key classes in the Time Ontology

Object	Property	Value
time:TemporalEntity	EquivalentClass	time:Instant and time:Interval
	time:before	only time:TemporalEntity
	time:after	only time:TemporalEntity
	time:hasBeginning	only time:Instant
	time:hasEnding	only time:Instant
	time:hasDuration	only time:Duration
time:Instant	subClassOf	time:TemporalEntity
	time:inside	only time:Interval
	time:inTimePosition	max 1 time:TimePosition
	time:inXSDDateTimeStamp	max 1 xsd:DateTimeStamp
time:Interval	subClassOf	time:TemporalEntity

	time:meets	only time:Interval
	time:overlaps	only time:Interval
	time:starts	only time:Interval
	time:finishes	only time:Interval
	time:during	only time:Interval
	time:equals	only time:Interval
time:DateTimeDescription	time:day	max 1 rdfs:Literal
	time:dayOfWeek	max 1 owl:Thing
	time:dayOfYear	max 1 rdfs:Literal
	time:hour	max 1 rdfs:Literal
	time:minute	max 1 rdfs:Literal
	time:month	max 1 rdfs:Literal
	time:second	max 1 rdfs:Literal

6.1.2.2 An Example

Returning to the example of representation of a vehicle. Should we wish to represent an instant in time at which the vehicle exists, relative to some earlier time before the vehicle exists, this would involve the introduction of two Instant objects that could be related via the before property. Should the data be available, the instants could be further described with the date-time stamp using the inXSDDateTime data property, or using the inDateTime property to relate the instants to a DateTimeDescription object.

Alternatively, we might know the interval but the not the precise instant. If specific data were known regarding the date and time of these interval, say that it began at 09:22 EST on June 19, 2019 and ended at 11:33 EST on July 12, 2019, this could be specified using the inXSDDateTime data property. In this case, the instant might simply be described as being in the interval using the inside property. This example representation is depicted in Figure 3: Example use of the Time Ontology.

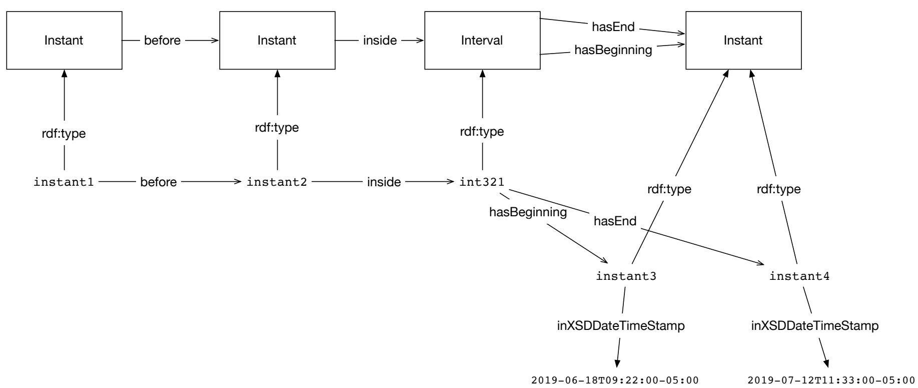


Figure 3: Example use of the Time Ontology

Reused Ontologies:

- time: W3C Time Ontology[3] originally presented by [8]

6.1.3 Change Ontology

http://ontology.eil.utoronto.ca/icity/Change.owl

Many of the concepts identified in the urban system ontologies are subject to change. For example, a Vehicle will have one location at one time, and another location at a later time; it may have only one passenger at one time, and four passengers at a later time. Similarly, many attributes of Persons, Households, and even Transportation Networks are subject to change.

Change over time plays a role in many domains, and is by no means a new research topic. In fact, several approaches for capturing change in OWL have been proposed [9],[10]. Despite these solutions, we have found that Semantic Web practitioners currently lack clear and precise methods for how to apply these approaches to capture change at a domain level, whether reusing

[3] https://www.w3.org/TR/owl-time/

an ontology that does not account for change over time or developing an ontology from scratch. The Change Ontology serves as a clear guide to support a consistent approach to formalizing how things change over time throughout the iCity TPSO.

6.1.3.1 The Ontology

An approach to representing changing properties, or *fluents*, that leverages the 4-dimensionalist perspective was proposed by [9]. We adopt a similar approach, based on the design pattern presented in [11], requiring the representation of objects that are subject to change with two classes: invariant and variant parts of the concept; we refer to these as TimeVaryingConcept and Manifestation classes, respectively. By distinguishing between these class types and recognizing the properties that are (and aren't) subject to change, the ontology supports the capture of both the static and dynamic aspects of a particular entity.

A class that is subject to change is defined as a type of TimeVaryingConcept (e.g. Vehicle may be a subclass of TimeVaryingConcept). The TimeVaryingConcept itself is invariant and defined by properties that do not change over time. As per [10], we represent TimeVaryingConcepts as perdurants (things that occur over time, i.e. processes). This is done to enable the required representation however we do not adopt the ontological commitment of these objects as processes: a TimeVaryingConcept is distinct from a process or event. A TimeVaryingConcept has Manifestations that demonstrate their changing (variant) properties over time. Different types (subclasses) of TimeVaryingConcept may be defined based on the Manifestations that are part of them. The key classes and properties of the ontology are outlined in

Property	Characteristic	Value (if applicable)
hasManifestation	inverseOf	manifestationOf
	Inverse Functional	-
manifestationOf	Functional	-
existsAt	Ranges	time:TemporalEntity

Property	Characteristic	Value (if applicable)
hasManifestation	inverseOf	manifestationOf
	Inverse Functional	-

manifestationOf	Functional	-
existsAt	Ranges	time:TemporalEntity

6.1.3.2 An Example

A key question to answer in the representation of changing objects is what properties may be subject to change, as opposed to other properties which have values that are part of the object's identity. The vehicle identifier (VIN) is a unique identifier for a vehicle that is assigned by the manufacturer and remains constant throughout a vehicle's lifetime. Therefore, the VIN should be a property of the TimeVaryingConcept (a class typically denoted with "PD"[4], for example VehiclePD) object. On the other hand, a vehicle's location may change over time. Therefore, the location should a property of the Manifestation object (a member of the Vehicle class). Note that the Change Ontology has implications not only on how instance data is represented, but also on how domain-specific classes are defined. This example representation is depicted in Figure 4. The individual "veh1t1" represents a manifestation of the individual vehicle "veh1"; in other words, veh1t1 captures a snapshot of veh1 in time. While veh1 has a single VIN for its entire existence, its location will change over time. Therefore, it is related to a series of individual manifestations (veh1t1 and others) that capture changing properties, such as location. When the location changes, this will be represented by another individual manifestation of veh1. Not captured in the diagram is the fact that each manifestation exists during some point or interval in time and thus may be related to a different temporal entity.

Commented [MK3]: Revise to "-S" for static

[4] Note: in order to avoid confusion that may result from the use of the "-Process" suffix (e.g. VehicleProcess,OrganizationProcess), we opt instead to use the suffix "PD", i.e. short for "Perdurant".

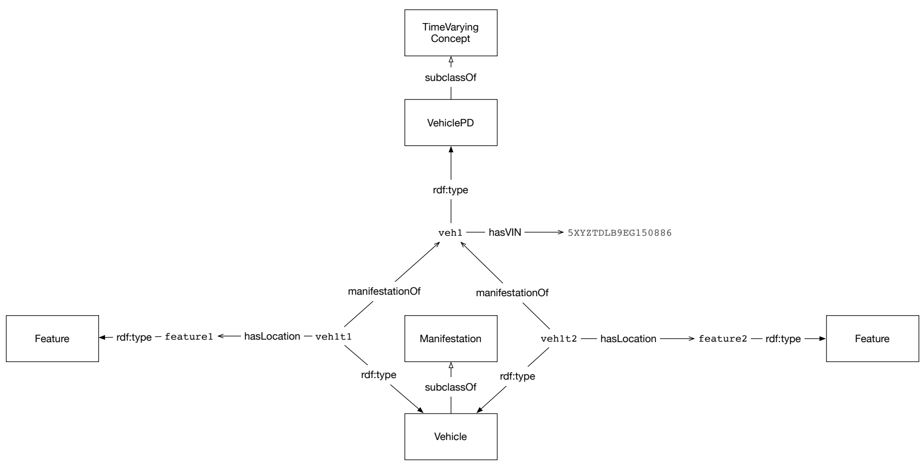


Figure 4: Example use of the Change Ontology

Reused Ontologies:

- iCity-Time

6.1.3.3 Future Work

Future work should clarify the distinction between the adoption of the 4-dimensionalist approach to capture change and the adoption of the 4-dimensionalist position. There are many implications in defining a class as a Perdurant (Occurant) or an Endurant (Continuant). Future work should consider alignment of the iCity Ontology to an Upper Ontology [12] such as DOLCE [13] or BFO [14] in order to make these commitments explicit.

6.1.4 Activity Ontology

http://ontology.eil.utoronto.ca/icity/Activity.owl

The concept of activities arises in several cases in the domain of transportation planning. Of particular importance are the trip activities that contribute to the demand on a transportation system, and the routine activities that motivate these trips. Trips are defined more precisely in an extension of the activity ontology, however both types of activities share the same foundational semantics. In the most general sense, activities may be thought of things that happen; events that occur (scheduled or not) or actions that are performed by some agent. It is not only the activity but the time of its occurrence and any things that are participants in some way that are involved in the description of an activity. Finally, central to understanding an activity, and thus central to its definition, is the effect it has or should have on the world.

6.1.4.1 The Ontology

There are many OWL ontologies that in some way address the concept of activities, however most are lacking with respect to the basic representation requirements. The Activity Ontology adopts the Activity Specification design pattern that was presented by [15] as a solution to address these limitations. The proposed solution adopts a view of causality similar to the Event Calculus [16], employing the concept of manifestations to describe the states (fluents) that hold before and after an activity. The representation of activity specifications is based on the activity clusters introduced by Fox, Sathi, and colleagues [17, 18].

A precursor to the TOVE [19] and PSL [20] activity ontologies, an activity cluster provides a basic structure for representing activity specifications. Illustrated in Figure 5, 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.

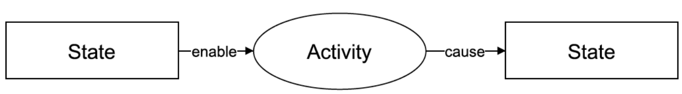


Figure 5: A generic activity cluster

It is important to clarify that in this approach an activity is interpreted as a class of occurrences, in contrast other approaches where activities are separate entities that are related to occurrences via an *occurrence of* relation. This decision was motivated by several pragmatic factors: in many cases it is sufficient to capture information regarding individual activities (i.e. occurrences or events). These activities may be categorized via different subclasses of “Activity”, but there is no need to associate them with a single activity type entity, unless we wish to characterize the activity type itself. The capability for this more complex formalization is supported, should it be required, by the Recurring Event ontology (presented below). Dividing these representations into two separate ontologies allows users of this representation the discretion to only include what they need. In addition, much of the semantics that relate activity types and occurrences – as defined in PSL for example – is not expressible in OWL. There would be little value in forcing such an ontology in OWL, which would only superficially capture the intended semantics. Instead, the Activity Ontology works within the limitations of OWL to capture the concepts of activities, their composition, preconditions and effects, and ordering. The key terms are described below:

An Activity describes something that occurs in the domain. It may have precondition and/or effect states, and may be further decomposed into subactivities. An Activity may be enabled by or cause some States. An enabling/causing state is a generalization of a precondition/effect; an Activity is enabled by or causes some State if it has a subactivity with a precondition or effect (respectively) of that State. An Activity occurs at some point in time or over some interval, and space, and may have some participants. Finally, though it is not possible to fully define the semantics in OWL, some notion of an ordering on activity occurrences must be captured in some cases. To address this, the properties: "occursBefore" and "occursDirectlyBefore" are introduced in the Activity ontology.

While we cannot fully define this semantics of an ordering over occurrences in OWL, we can leverage the start and end times of an activity to describe the occursBefore property using object property chaining:

- An activity occursBefore another if its endOf instant is before the beginOf instant of the other activity: endOf o before o inverse (beginOf) -> occursBefore. The occursBefore relation is also defined as transitive.
- An activity occursDirectlyBefore another if it occursAt an interval that meets the interval of the other activity; this can be captured similarly with object property chaining: occursAt o intervalMeets o inverse(occursAt) -> occursDirectlyBefore.

A state refers to a subclass of manifestations, as defined in the Change Ontology. It may be an immediate precondition or effect of some Activity, or more generally it may enable or be caused by some Activity (in which case, it might be a direct precondition or effect of some subactivity of the activity). A state may be complex and refer to some combination of classes of manifestations.

- A State may be either non-terminal or terminal. A terminal state has no child states, and therefore refers directly to a class of manifestations, whereas a non-terminal state has child states, which may define some classes of manifestations, or further define some other complex state types. A state type cannot be both non-terminal and terminal.
- A terminal state has cannot be decomposed, in other words it has no substates. It corresponds to a particular class of manifestations. A terminal state is achieved at some time if and only if there exists a manifestation within its defined classification, that exists at that time.
- A non-terminal state may be conjunctive or disjunctive. Naturally, a conjunctive state is defined by the conjunction of its child state, whereas a disjunctive state is defined by the disjunction of its child states. A state cannot be both conjunctive and disjunctive.

  Conjunctive and disjunctive states, which *do* have substates, are achieved at some time if their decomposition of state is achieved.

  Note that in this representation the decomposition of (*decomp_of*) property is not a transitive relation, it only refers to the direct children of a non-terminal state. A more general relation that *is* transitive is the substate relation.

The key classes that formalize these concepts are summarized in Table 5 and illustrated in Figure 6.

Table 5: Key classes in the Activity Ontology

Object	Property	Value
Activity	hasSubactivity	only Activity
	hasPrecondition	only State
	enabledBy	only State
	hasEffect	only State
	causes	only State
	occursAt	some time:Interval
	beginOf	some time:Instant

	endOf	some time:Instant
	spatial_loc:hasLocation	only spatial_loc:SpatialFeature
	hasParticipant	only change:Manifestation
	occursBefore	only Activity
	occursDirectlyBefore	only Activity
State	preconditionOf	only Activity
	enables	only Activity
	effectOf	only Activity
	causedBy	only Activity
	achievedAt	only time:TemporalEntity
TerminalState	subClassOf	State
	disjointWith	NonTerminalState
	subClassOf	change:Manifestation and (preconditionOf some Activity or effectOf some Activity)
	hasDecomp	exactly 0 StateType
NonTerminalState	subClassOf	State
	disjointWtih	TerminalState
	hasDecomp	only State and min 2 State
	hasSubstate	only State
ConjunctiveState	subClassOf	NonTerminalState
	disjointWith	DisjunctiveState
DisjunctiveState	subClassOf	NonTerminalState
	disjointWith	ConjunctiveState

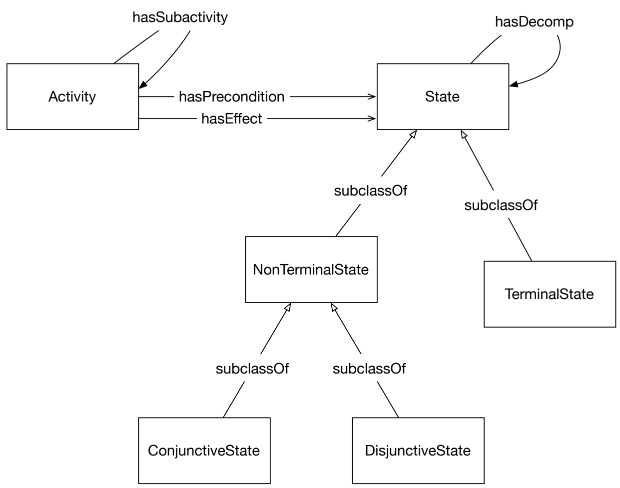


Figure 6: Relationship between key classes in the Activity Ontology

6.1.4.2 An Example

As an example, consider the activity of driving to work. This activity occurs before the activity of working; axioms at the class-level could be added to state that all instances (occurrences) of the DriveToWork activity occur before some instances (occurrences) of the Work activity, though such statements may be too strong in some cases. There are also certain preconditions and effects of the activity that might be important to represent. For example, an effect of the DriveToWork activity is that both the driver and the car are at work. This could be represented as a complex, Conjunctive State. This state may then be decomposed into more precise sub-states that capture the intended semantics using concepts from other parts of the iCity TPSO. This example formalization of the DriveToWork activity is illustrated in Figure 7. Note that the activity DriveToWork might also be decomposed into subactivities (e.g. parts of the trip) as required. When the resulting Activity and State subclasses are instantiated, additional details regarding a particular occurrence of an activity may be added. For example, the location of the person and vehicle may be specified thus providing additional detail on the state before the particular activity occurrence. This is depicted in Figure 8.

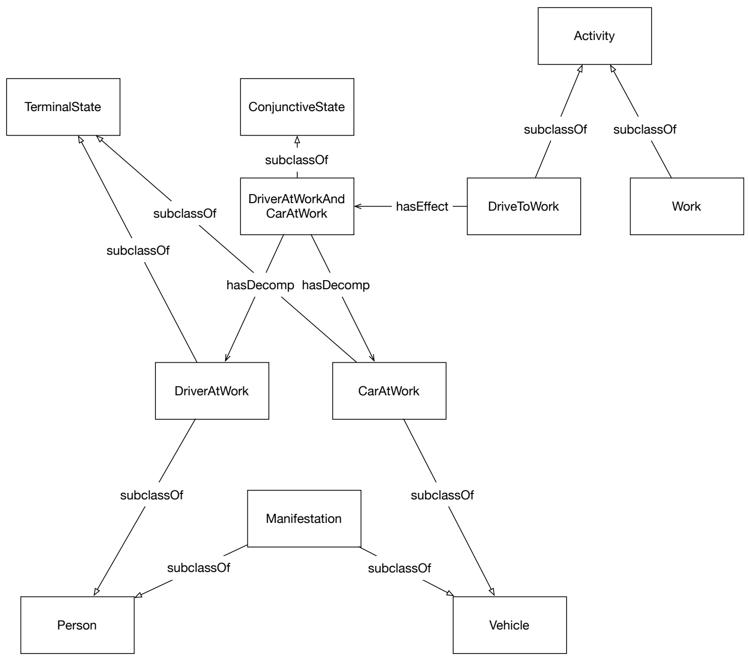


Figure 7: Example formalization of the DriveToWorkActivity

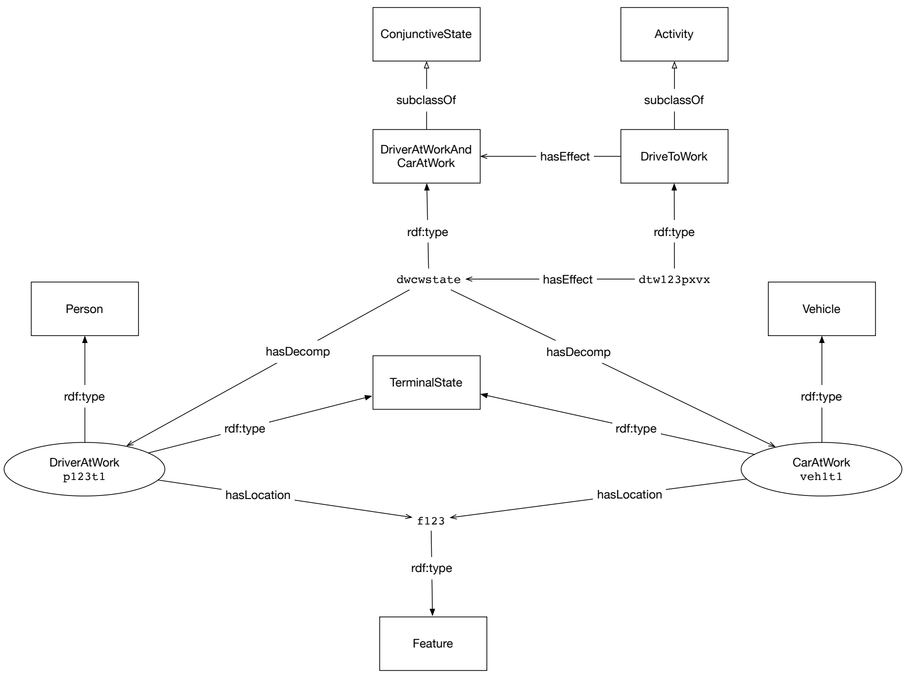


Figure 8: Example use of the Activity Ontology

6.1.4.3 Future Work

As noted, this represented is influenced by earlier work on Activities in the TOVE ontologies. However, this ontology does not directly reuse the more recent OWL version of the TOVE Activity ontology released by the Enterprise Integration Laboratory[5]. Future work should address this by attempting to either revise and converge these ontologies or to formalize the relationship between the two.

Reused Ontologies:

- Change Ontology

[5] http://ontology.eil.utoronto.ca/tove/activity.owl

- Location Ontology

6.1.5 Recurring Event ontology

http://ontology.eil.utoronto.ca/icity/RecurringEvent

A specification of recurring events, in particular those that are defined according to calendar dates (e.g. every Monday, every March), is required to capture information regarding hours of operation, road restrictions, restrictions on parking policies, and so on. A recurring event is a means of describing scenarios where some activity is scheduled to recur at some regular interval. It is important to note that recurring events such as scheduled transit trips and operating hours represent planned or usual occurrences. For example, while a business may be open at some recurring intervals, it's possible that given some exceptional circumstances (e.g power failure) they may not be open during the predefined days and times.

6.1.5.1 The Ontology

The design of this ontology was inspired by previous work on an ontology for city services[6] for the Global City Indicator (GCI) Ontology [21], however due to incompatibilities in the scope and semantics of the GCI ontology we do not directly reuse it in the iCity TPSO. The GCI Ontology defines recurring events specifically as "Service" events, whereas the transportation requires a more general notion of recurring events. The GCI Ontology employs the concept of a time interval to capture when some event recurs, however we observe that this is misleading as recurring events will occur at *multiple* intervals in time. In the iCity TPSO, we opt for a more precise representation that identifies the individual occurrences (that occur at a particular time interval) of some recurring event.

The Recurring Event Ontology adopts the following representation of recurring events: daily, weekly, and monthly recurring events (and their related properties) are defined, however the ontology may be extended with similar definitions of other type of recurring events, as required. This approach is based on the GCI Ontology work and adapted to provide a more suitable and complete representation of recurring events for the transportation domain.

[6] http://ontology.eil.utoronto.ca/city-services/city-services.owl#

An instance of a recurring event corresponds to a class of activities (e.g., all of the occurrences of a Tuesday, all of the occurrences of the weekly waste pickup). The intuition is that the occurrences of a recurring event are all the same type of activity. What defines a recurring event is a combination of the activity type (e.g. a transit trip from point A to point B or the provision of a service) and the frequency at which it recurs.

The ontology captures the associated activity type with the *hasOccurrence* property that relates recurring events to activities. Classes of recurring events may be captured by identifying their associated classes of Activities, while individual recurring events may be associated with one or more instances of an activity.

The Recurring Event ontology reuses the Activity ontology, as the concept of an activity is central to the notion of a recurring event: the activities are the recurrences. It is important to note that while the concept of Activity defined in the Activity ontology and is necessary for the definition of a RecurringEvent, it is *not* the case that the concept of RecurringEvents is required for the definition of an Activity. This allows the iCity TPSO to maintain a simpler representation of events in cases where the notion of recurrence not be required.

Recurring events are also identified based on the regular interval at which they occur; this is captured using some combination of the hasTime, dayOfWeek, hasMonth, and dayOfMonth properties. Using these properties, the ontology supports definitions of specializations of the RecurringEvent class. In particular, subclasses for daily, weekly, monthly, and yearly recurring events are defined; other classes of recurring events may be defined similarly, as required.

- A DailyRecurringEvent occurs every day. It has a maximum of one associated time – the start time. Typically, a daily recurring event will occur at the same time every day, however there may be no commitment to a recurring start time for the event, in which case no start time is specified. A DailyEvent does not necessarily have a recurring end time (this would require a constant duration), therefore this is not part of the definition (although it is possible to specify).
- A WeeklyRecurringEvent recurs regularly on the same day of the week, as specified by the schema:dayOfWeek property.
- A MonthlyRecurringEvent recurs regularly on the same day of each month, as specified by the dayOfMonth data property. Note that there is often ambiguity regarding the

semantics of a monthly recurring event: in this formalization, a MonthlyRecurringEvent is any event that recurs regularly on the same *day* of each month; other interpretations sometimes consider events that recur on the same day of week, or first or last day, in which case the day of month will vary. Such a representation is not included in this ontology, but could be captured in an extension.

- A YearlyRecurringEvent recurs regularly on the same day of the same month, as specified by the hasMonth and dayOfMonth properties. As with MonthlyRecurringEvent, there may be ambiguity regarding the semantics of a yearly recurring event, however this formalization captures only the notion of an event that recurs on the same day of the same month (e.g. a birthday).

Exceptions to recurring events may also be defined. For example, a business may normally operate on Monday-Friday, except for public holidays. Exceptions may also be defined on *specific* dates (e.g. June 23, 2018), for example due to construction. If applicable, exceptions may be defined for recurring events with the recursExcept property. Conversely, so-called exceptions may involve an additional, unusual occurrences. This is captured with the recursAddition property.

As with an Activity, a RecurringEvent may be decomposed/decomposed into simpler/more complex RecurringEvents to support varying levels of granularity. This decomposition may be specified with the hasSubRecurringEvent property. The key classes in the Recurring Event Ontology are summarized in Table 6 and illustrated in Figure 9.

Table 6: Key classes in the Recurring Event Ontology

Object	Property	Value
RecurringEvent	hasOccurrence	only activity:Activity
	spatial_loc:associatedLocation	only spatial_loc:Feature
	hasSubRecurringEvent	only rec:RecurringEvent
	startTime	only xsd:time
	endTime	only xsd:time

	schema:dayOfWeek	only DayOfWeek
	endDayOfWeek	only DayOfWeek
	hasMonth	only Month
	endMonth	only Month
	dayOfMonth	only rdfs:Literal
	endDayOfMonth	only rdfs:Literal
	beginsRecurring	only time:TemporalEntity
	endsRecurring	only time:TemporalEntity
	recursExcept	only time:TemporalEntity or DayOfWeek
	recursAddition	only time:TemporalEntity or DayOfWeek
DailyRecurringEvent	subclassOf	RecurringEvent
	startTime	max 1 xsd:time
WeeklyRecurringEvent	subclassOf	RecurringEvent
	schema:dayOfWeek	exactly 1 DayOfWeek
MonthlyRecurringEvent	subclassOf	RecurringEvent
	dayOfMonth	exactly 1 rdfs:Literal
YearlyRecurringEvent	subclassOf	RecurringEvent
	hasMonth	exactly 1 Month
	dayOfMonth	exactly 1 rdfs:Literal

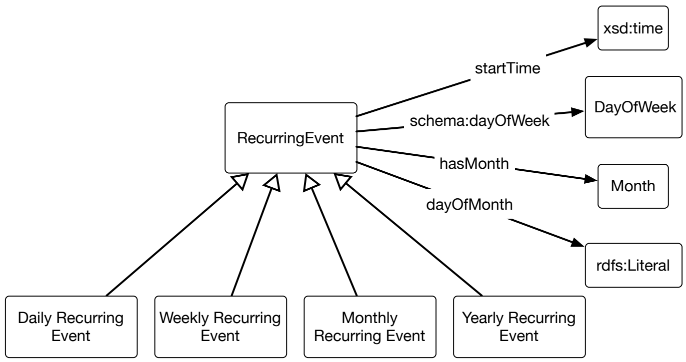


Figure 9: Basic structure of the Recurring Event Ontology

6.1.5.2 An Example

As an example, consider the representation of scheduled transit trips. The Activity Ontology may be used to define classes of Transit Trip activities, and these classes may be instantiated with instances that correspond to individual occurrences of these trips, however in order to capture the schedule – i.e. that some trip occurs every day at 08:00am – the notion of a recurring event is required. A class of recurring events that captures scheduled bus trips may be defined as having only BusTrip activities as occurrences. Instances of the ScheduledBusTrip class may include recurring events with different start times, perhaps corresponding to different routes or different routes on the same trip. An individual scheduled bus trip with a start time of 08:00am corresponds to multiple occurrences. As daily recurring event, we can expect there will be a corresponding occurrence of the bus trip activity every day, thus an individual recurring event (an instance of a scheduled bus trip) will correspond to multiple instances of a particular activity type (a bus trip). The Recurring Event object provides information on the way in which the activity recurs (e.g. daily at 08:00am). This example is illustrated in Figure 10.

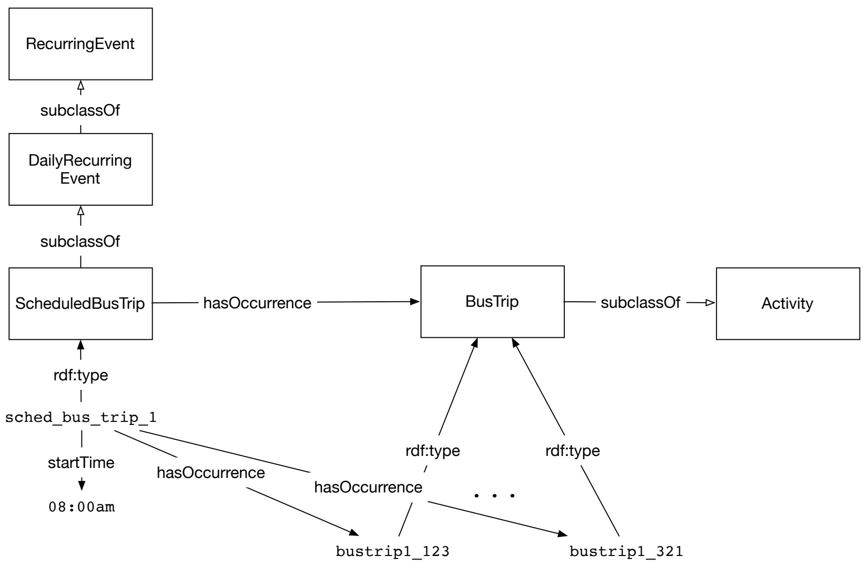


Figure 10: Example use of the RecurringEvent Ontology

6.1.5.3 Future work

- Address the relationship between a Recurring (Service) Event and an Event (Activity) in more detail. Based on the properties of a recurring event, additional constraints on its occurrences (related activities) may be inferred.
- Additional temporal constraints may be specified to describe the relationship between a Recurring Event and its sub-Recurring Events: the sub-Recurring Events may only recur during the times at which the Recurring Event recurs.
- An ordering relationship over sub-Recurring Events may be useful in future implementations, however this is not currently captured or required.

Reused Ontologies

- Activity Ontology

6.1.6 Resource Ontology

http://ontology.eil.utoronto.ca/icity/Resource.owl

Resources are an important aspect of activities; they often capture important preconditions and effects of activities. In the context of transportation planning, resources such as vehicles, income, and transit passes will impact travel behaviour. The representation of resources is also important for tasks related to asset management; for example, transit vehicles and their scheduled maintenance and failure rates are important factors for predicting the performance of the transit system.

6.1.6.1 The Ontology

The Resource Ontology provides a generic representation of resources that contain core properties generic across all transportation uses. We take the view presented in the TOVE model [22] that "*...being a resource is not an innate property of an object but a property that is derived from the role the object plays with respect to an activity*". In this sense, Resources are a class of Manifestations; a Resource is a manifestation of some *other* perdurant class in the ontology when it plays the role of Resource for some Activity. For example, an instance of a Vehicle, (a manifestation of some VehiclePD) may also be an instance of a Resource, whereas some other instance of a Vehicle, (some later manifestation of the same VehiclePD) may not be a Resource, or it may be a *different* type of Resource. For example, when the Vehicle is used for transportation, it is one type of resource, but when it is being used for scrap metal, it is a different type. This definition of a resource is dependent on its participation in an activity, thus the Resource ontology reuses the Activity ontology.

A Resource may have some Location, amount or availability, according to the definition of the Manifestation or TimeVaryingEntity. In addition, it may have some associated location and may have some owner. As with the precondition and effect properties defined in the Activity Ontology, the decomposition of an activity must be considered: there are atomic-level relationships of consumption and use, but also more general relationships based on inheritance through composition. For example, if Activity A has subactivity B, then a resource used by Activity B is also used by Activity A.

For additional detail, a Resource maybe classified according to more specific resource types. A Resource may *either* be a Divisible Resource or a Non-Divisible Resource, but not both. As the names indicate, a Divisible Resource may be divided for use or consumption between multiple activities at any point in time, whereas a Non-Divisible Resource may only be used for a single activity at once – even if it isn't fully utilized. Continuing our example, a Vehicle used for transportation is non-divisible but if used for scrap then it is divisible. The key classes in the ontology are summarized in Table 7.

Various other types (subclasses) of Resource may be defined as required. A Resource Type may be used by or consumed by some Activity; the specification of the Resource Type defines the quantity of a particular resource that will be used or consumed by a particular activity. If some resource type is used by an activity, then when the activity occurs, there must be some resource of that type that is (partially) not available. If a resource type is consumed by an activity, then the resource and the entity it is a manifestation of (partially) cease to exist by the end of the occurrence.

Table 7: Key classes in the Resource Ontology

Object	Property	Value
Resource	subClassOf	change:Manifestation
	change:existsAt	exactly 1 TemporalEntity
	spatial_loc:hasLocation	only spatial_loc:SpatialFeature
	hasCapacity	only om:CapacitySize
	capacityInUse	only om:CapacitySize
	activity:participatesIn	min 1 activity:Activity
	usedInOccurrence	only activity:Activity
	consumedInOccurrence	only activity:Activity
DivisibleResource	subClassOf	Resource
	disjointWith	NonDivisibleResource
	hasAvailableCapacity	only om:CapacitySize
NonDivisibleResource	subClassOf	Resource
	disjointWith	DivisibleResource

	usedBy	exactly 1 activity:Activity

6.1.6.2 An Example

Consider the representation of a vehicle as an example. A Vehicle might be used as a non-divisible resource for transportation, and then later as a divisible resource for some metal recycling process. While these examples might refer to the same car over the span of its lifetime, each one in fact refers to a different manifestation of the car, and hence a different resource. The resources differ in their divisibility because each one is defined with respect to a different activity (e.g. travel, versus metal recycling). A divisible resource may be used by or consumed by more than one activity, whereas a non-divisible resource may only be used by one activity (i.e. the object may only be used by one activity at a time). This example is illustrated in Figure 11

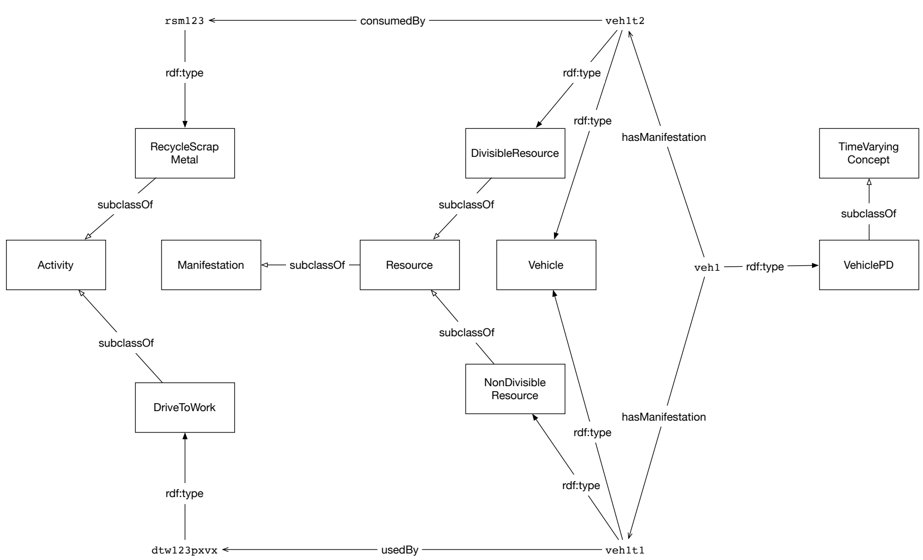


Figure 11: Example use of the Resource Ontology

6.1.6.3 Future Work

The current representation is very general and simplistic. Future versions will likely need to expand on the types of resources as well as their relationship between activities.

Reused Ontologies:

- iCity Activity Ontology

6.1.7 Parthood Ontology

http://ontology.eil.utoronto.ca/icity/Mereology.owl

Notions of parthood are ubiquitous in and beyond the transportation domain. While sometimes conflated, there are clear distinctions which can be made between different types of parthood and similar relations. The parthood ontology goes beyond classical mereology and is intended to capture any generic part-whole relations of interest; thus far, the ontology focuses on the part-of, contained-in, and component-of relations, attempting to make the distinctions between them explicit. These distinctions may be best explained with the use of examples. An item may be *contained in* my car, but that does not make it a *component of* my car. For example, we may wish to describe passengers or cargo being *contained in* a vehicle, but this relation must be distinguished from the parts and components that make up a vehicle. Similarly, the front of my car is intuitively a part of my car, but not a component of my car. While we may define components of a vehicle, different zone systems (wards, postal codes) are not components, but proper parts of larger areas.

6.1.7.1 The Ontology

The Parthood Ontology introduces the following different relations as object properties: proper-part-of, component-of, and contained-in. A more detailed analysis reveals clear, ontological distinctions between each of these relations that may formalized clearly with a set of first-order logic axioms. This analysis, presented in [23] also identifies the expressive limitations of OWL, which prevent a complete representation of this semantics, and discussed the various possible approximations. It is important to consider what should be captured, and what distinctions should be made in the introduction of properties, in contrast with what is actually expressible in the logic. Since we cannot completely capture the required semantics in OWL, some trade-off(s) is required for any partial specification, (e.g. OWL only allows the specification of transitivity for simple object properties).

The difficulty with such an approximation is that the resulting theory defines a semantics for something else entirely. Inherently, some semantics are omitted, which may not be required for one application but may be important for another. For example, if transitivity is a key aspect of some required reasoning, then perhaps a parthood relation would be defined as transitive, and some omissions would be made with respect to the formalization other restrictions (e.g. cardinality) that should be applied to the parthood relation. Certainly, the use of approximations will be required in some cases, for example in order to support some desired reasoning problems. However, precisely which axiomatization is most suitable will vary between different usage scenarios. The Parthood Ontology therefore omits a detailed, partial axiomatization in favour of an under-axiomatized specification of the key relations, in order to avoid prescribing one trade-off over another. This leaves the commitment open-ended and variable to suit individual applications' needs. The key properties are summarized in Table 8.

This ontology defines the general properties such that the commonality between domain-specific part-of relations may be captured, and more detailed semantics may be defined in extensions of the properties. This creates a means of indicating the intended semantics of a relation by identifying the *type* of parthood that it is intended to capture, while allowing for the specification of different partial approximations of the semantics (and possibly also specializations of this semantics), as required. For example, a notion of parthood arises in the description of a building and the units it is divided into. In this case, this relationship may be identified as a sort of hasComponent relation; a new property 'hasBuildingUnit' may be identified then as a subPropertyOf hasComponent. We are free to assess, for the 'hasBuildingUnit' relation, which approximations of the component-of relation are the most suitable. The approximation chosen for one type of parthood relation does not constrain the choice of approximation for another.

Table 8: Key properties in the Parthood Ontology

Property	Characteristic	Value (if applicable)
properPartOf	inverseOf	hasProperPart
hasProperPart	inverseOf	properPartOf
componentOf	subPropertyOf	properPartOf
	inverseOf	hasComponent
hasComponent	subPropertyOf	hasProperPart

	inverseOf	componentOf
immediateComponentOf	subPropertyOf	componentOf
containedIn	inverseOf	contains
contains	inverseOf	containedIn
immediatelyContainedIn	subPropertyOf	containedIn

6.1.7.2 An Example

For example, consider the representation of various parts in a vehicle. This is a component-of type property, therefore a property ‘hasVehicleComponent’ as a sub-property of the ‘hasComponent’ property. Like hasComponent, hasVehicleComponent should be both transitive and irreflexive. However, owing to the restriction on non-simple object properties in OWL, it is not possible to capture both characteristics. In the context of a vehicle’s component decomposition, it is likely the case that transitivity of the property may be more important than its irreflexivity. Therefore, the subproperty hasVehicleComponent may be approximated as being transitive while maintaining the under-axiomatized definition of hasComponent. On the other hand, it may be the case that for component relation for Buildings and BuildingUnits the anti-symmetry of the property is the most important aspect to capture. The hasBuildingUnit property may be approximated as anti-symmetric rather than transitive, thus allowing for different trade-offs to be made to capture the component-hood relationship in different domains. This example is illustrated in Figure 12.

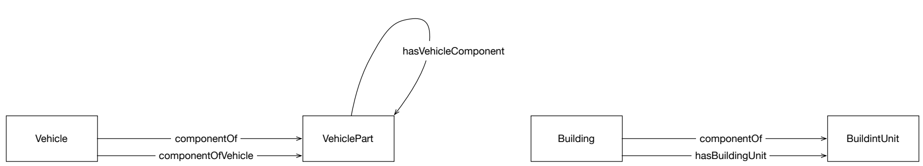


Figure 12: Example use of the Parthood Ontology

6.1.7.3 Future work

In addition to the aforementioned work by [23], various approaches to the partial capture of these mereological relationships in OWL have been proposed that may be used to extend the ontology presented here, such as in the W3C's Best Practices [24], and also in upper ontologies such as [14]. Future revisions may benefit from considering the relationship between these ontologies and other OWL formalisms.

This ontology takes a simple approach to supporting a taxonomy of part-whole relations, however recent work presents strong arguments for the adoption of mereological pluralism [25]. This work is currently restricted to physical objects, and so does not address the part-whole relationship between abstract objects (e.g. household membership), nevertheless future work should examine the reuse of these physical part-whole relations as a means to make the iCity ontology more precise. One limitation on such an effort may be the expressive restrictions of the OWL language. The proposed ontology for mereological pluralism is axiomatized in first-order logic, and we speculate that many of the distinctions made between the types of part-whole relations may not be expressible in OWL. On the other hand, regardless of whether or not the distinctions may be logically formalized, it may be sufficient to distinguish them superficially, in the language of the ontology.

Finally, there is a close connection between the Parthood Ontology and the Spatial Ontology that has not yet been explored: the relationship between the spatial properties and the part-whole properties should be formalized with some theory of mereotopology.

6.1.8 Units of Measure Ontology

uom: http://ontology.eil.utoronto.ca/icity/OM.owl

Units of measure are an important concept due to the observational nature of data collection for transportation planning. In particular, it is important to capture the relationship between some quantity and the unit of measure it is described with. This allows for a representation in which the same individual quantity may be associated with several values, according to different units of measurement.

6.1.8.1 The Ontology

The Ontology of Units of Measure provides a structured vocabulary to describe, among other things, the different values (measures) that we associate to given quantities. This allows us to provide greater detail regarding specific measurements that are defined in the ontology. Rather than simply have a simple data property to describe the length of some road segment as "10 m", with the units of measure ontology we are able to describe the nature of the quantity (i.e. length), its value as a Measure (10 m), and also describe the unit that the measure's numerical value is given in (e.g. meters). The core formalism is based on the Units of Measure ontology defined by [26]. The Units of Measure ontology is not directly reused as it is quite large and includes many concepts that are out of scope for city data measures. The relationship with the quantities, and units of measure defined as classes and individuals in [26] may be formalized in the future if required. Existing concepts may be added from the original ontology or this ontology may be extended as to capture new units of measure as required.

Quantities, units, and/or measures that are defined using domain-specific concepts (e.g. vehicles, lanes) are defined by reusing and extending the units of measure ontology in the relevant ontologies, such that the necessary concepts may be captured and the foundational ontology is not complicated with domain-specific concepts. The key classes used in the definition of quantities and measures are summarized in Table 9.

Table 9: Key classes in the Units of Measure Ontology

Object	Property	Value
Quantity	hasValue	only Measure
Measure	hasUnit	only Unit
Speed_unit	subClassOf	Unit
Amount_of_money_unit	subClassOf	Unit
...	subClassOf	Unit
MonetaryValue	subClassOf	Measure
	hasRelativeYear	exactly 1 xsd:gYear
	hasUnit	only Amount_of_money_unit
ValueOfMoney	subClassOf	Quantity

	subClassOf	AmountOfMoney
	hasValue	only MonetaryValue

6.1.8.2 An Example

For example, consider the representation of the speed of a Vehicle, and a particular point in time. The Vehicle's speedometer may indicate a speed of 62 mph, whereas the speed observed by some radar gun or loop detector may record a speed of 100 km/h. Both values represent the same quantity but use different units of measure. Using the Units of Measure Ontology, the two distinct values and their units of measure may be captured and associated with a single instance of the vehicle's speed, as illustrated in Figure 13.

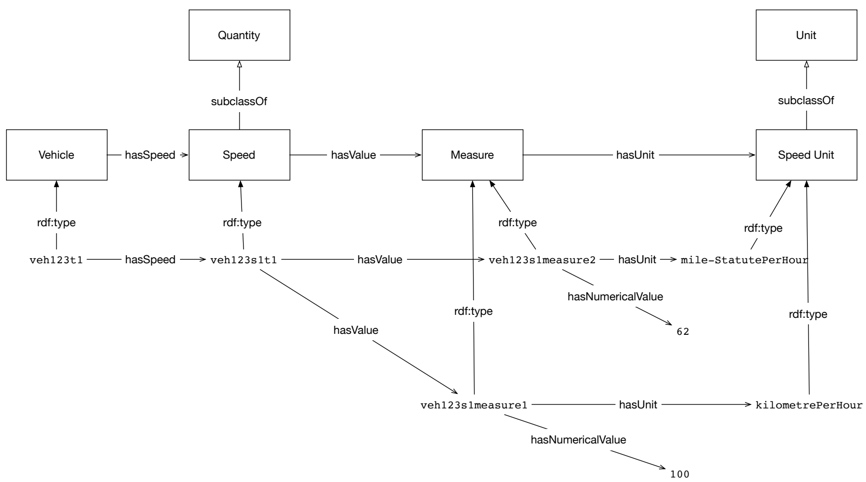


Figure 13: Example use of the Units of Measure Ontology.

A note on populations and cardinality:

In order to represent populations, we reuse the following classes from the GCI-Foundation ontology: gci:PopulationSize, gci:PopulationSizeMeasure, and gci:CardinalityUnit. Refer to the working paper on the GCI Ontology for more details on this approach. The meaning of population is general here, while it may define a population of residents within some zone, it may also be used to describe the population of vehicles occupying some stretch of the road network.

The quantity of interest (population size being measured/described) is defined as gci:Population_Size, a subclass of Quantity. Population_Size has some unit of measure (a cardinality unit), and has_value some Population_Measure (with an associated numeric value). The elements associated with a population quantity are captured through the defined_by property that relates a Population to some class of objects. For example, consider the measurement of the number of cars on some road segment, we could specify: Population_Size and cardinalityOf only (Population and definedBy only (Vehicle)). The defining population might be even more precisely captured for a given Road Segment, X, as depicted in Figure 14: definedBy only (Vehicle and onSegment value X). These specializations are defined, as required, within the relevant module; for example, a vehicle population would be defined in a module that contains the required concepts of vehicles and road segments. The units of measure ontology captures only the core concepts of Population Size, Population Measure, Cardinality Unit, and Population, as depicted in Figure 14. Capacity and its associated quantity and measure are defined similar to population.

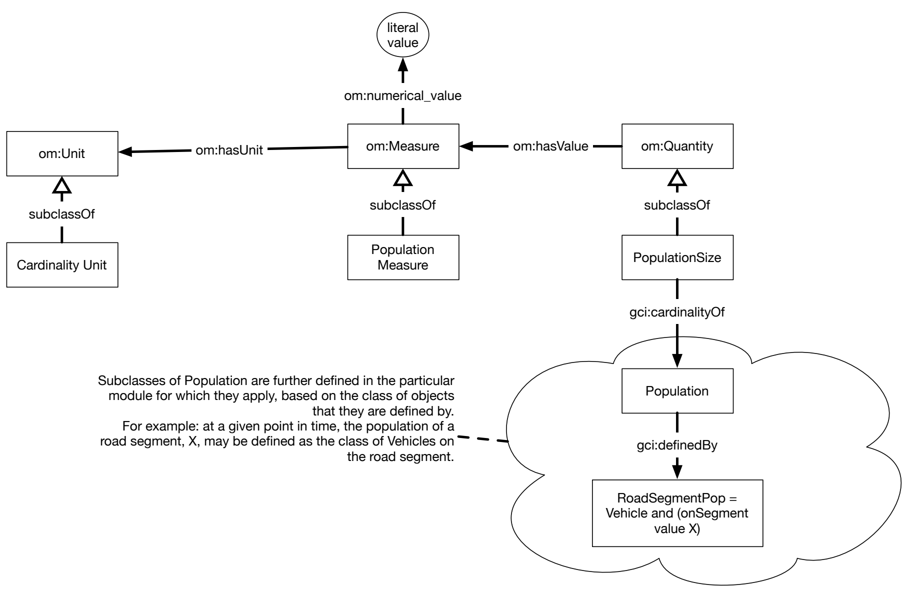


Figure 14: Specialization of populations.

Object	Property	Value
Quantity	hasValue	only Measure
Measure	hasUnit	only Unit
Length_unit	subClassOf	Unit
Mass_unit	subClassOf	Unit
Area_unit	subClassOf	Unit
Acceleration_unit	subClassOf	Unit
Volume_unit	subClassOf	Unit
Speed_unit	subClassOf	Unit
Amount_of_money_unit	subClassOf	Unit
Geo_Position_unit	subClassOf	Unit
gci:Cardinality_unit	subClassOf	Unit
UnitDivision	subClassOf	Unit
Cardinality_unit_per_time	subClassOf	UnitDivision
	hasNumerator	only gci:Cardinality_unit
	hasDenominator	only TimeUnit
...	subClassOf	Unit_of_measure
MonetaryValue	subClassOf	Measure
	hasRelativeYear	exactly 1 xsd:gYear
	hasUnit	only Amount_of_money_unit
gci:Population_measure	subClassOf	Measure
	subClassOf	CardinalityMeasure
CardinalityMeasure	subClassOf	Measure
	hasUnit	only gci:Cardinality_unit
ValueOfMoney	subClassOf	Quantity
	subClassOf	AmountOfMoney
	hasValue	only MonetaryValue
Length	subClassOf	Quantity

	hasValue	only (Measure and hasUnit only Length_unit)
gci:PopulationSize	subClassOf	Quantity
	hasValue	only gci:Population_measure
	gci:cardinalityOf	exactly 1 gci:Population
CapacitySize	subClassOf	Quantity
	hasValue	only gci:Cardinality_measure
	gci:cardinalityOf	exactly 1 Capacity
CapacityRate	subclassOf	Quantity
	hasValue	only (hasUnit only CardinalityUnitPerTime)
	gci:cardinality_of	exactly 1 Capacity
Mass	subClassOf	Quantity
	hasValue	only (hasUnit only Mass_unit)
Area	subClassOf	Quantity
	hasValue	only (hasUnit only Area_unit)
Volume	subClassOf	Quantity
	hasValue	only (hasUnit only Volume_unit)
Acceleration	subClassOf	Quantity
	hasValue	only (hasUnit only Acceleration_unit)
Speed	subClassOf	Quantity
	hasValue	only (hasUnit only Speed_unit)

Property	Characteristic	Value (if applicable)
hasBaseUnit	domain	System_of_units
hasBaseUnit	range	Unit
hasDenominator	domain	UnitDivision
hasDenominator	range	Unit
hasNumerator	domain	UnitDivision
hasNumerator	domain	Unit
hasAggregateFunction	domain	Quantity
	range	function
aggregateOf	domain	Quantity
aggregateOver	domain	Quantity

6.1.8.3 Future work

Future extensions should consider whether it is more accurate to describe the position coordinates as quantities that are measured in degrees *that are relative to a geodetic datum* (e.g. NAD83), as it is important that we are able to distinguish between different position systems. In particular, WGS84 and NAD83, which were originally nearly equal are now considerably different (depending on the area) due to changes that have occurred to the earth since 1984. Note that http://data.ign.fr/def/ignf/20150505.en.htm may be a relevant ontology.

Reused Ontologies:

- Global City Indicators Foundation Ontology[7]

6.1.9 Observations Ontology

http://ontology.eil.utoronto.ca/icity/Observations

[7] http://ontology.eil.utoronto.ca/GCI/Foundation/GCI-Foundation-v2.owl#

In the iCity TPSO, the Observations ontology is included with the Foundational Ontologies due to the importance of data collection for transportation planning activities. Data collection efforts take various forms – whether through surveys, the use of sensors, or manual observation. With growing access to the Internet of Things, data from available sensors will continue to expand, likely to include observations about persons, vehicles, and so on. It is important to not only capture the data that is gathered, but the source of the observations.

6.1.9.1 The Ontology

The Observations ontology reuses the SSN (Semantic Sensor Network) ontology[8], a W3C recommendation that has been widely adopted to represent sensors and their observations. It is this widespread use which has motivated the adoption of the SSN ontology to capture sensors and their observations in the domain of transportation planning. The SSN Ontology defines a Sensor as a device that makes some observation, and may be triggered by some stimulus. An Observation has some feature of interest – the thing whose property is being detected by the sensor. An observation observes some ObservableProperty. A phenomenon time (i.e. the time at which the property was demonstrated) and result time may be associated with a particular observation.

The Observations Ontology generalizes concepts from the SSN Ontology and expands the representation to include observations collected without the use of a sensor. To achieve this, the concept of an Observer is introduced; an Observer is a generalization of a Sensor and could also include concepts such as Persons or Surveys. The key concepts are summarized in Table 10.

The SSN ontology does not make any commitments as to whether instances of ssn:Property should be generic (e.g. ex:temperature) or specific to the feature of interest (e.g. ex:mybodytemperature); current documentation suggests that this is a choice for the modeler. On the other hand, the iCity TPSO prescribes a definition of instances of ssn:Property at a generic level; this enables the querying of sensors that observe some property (e.g. vehicle presence) regardless of the location. This is useful as there may be different kinds of sensors that observe the same properties (e.g. loop detectors vs Bluetooth sensors) and while they might not share the

[8] http://www.w3.org/ns/ssn/

exact feature of interest, they may be in close enough proximity to be related and so a property indicating their similarity is desirable.

Table 10: Key classes in the Observations Ontology

Object	Property	Value
Observation	observedBy	only Observer
Observer	inverse(observedBy)	only Observation
sosa:Sensor	subclassOf	Observer
	sosa:madeObservation	only sosa:Observation
	sosa:observes	only sosa:ObservableProperty
	ssn:detects	only ssn:Stimulus
sosa:Observation	subclassOf	Observation
	sosa:madeBySensor	exactly 1 sosa:Sensor
	sosa:hasFeatureOfInterest	exactly 1 owl:Thing and only sosa:FeatureOfInterest
	sosa:hasResult	exactly 1 owl:Thing and only sosa:Result
	sosa:observedProperty	exactly 1 owl:Thing and only sosa:ObservableProperty
	sosa:phenomenonTime	exactly 1 owl:Thing
	sosa:resultTime	exactly 1 rdfs:Literal
	ssn:wasOriginatedBy	exactly 1 owl:Thing and only ssn:Stimulus
sosa:ObservableProperty	subClassOf	ssn:Property
	inverse ('is proxy for')	only ssn:Stimulus

	inverse ('observed property')	only sosa:Observation
	sosa:'is observed by'	only sosa:Sensor
sosa:FeatureOfInterest	ssn:'has property'	min 1 owl:Thing and ssn:Property
sosa:Result	sosa:'is result of'	min 1 owl:Thing

6.1.9.2 An Example

As an example, consider the representation of a loop detector and its observations on the road network. The Observations ontology may be extended to capture the class of Loop Detector sensors. For a particular Loop Detector, we may specify that it makes some observation at a particular time, and that the result of this observation is some Vehicle Volume on the RoadSegment of interest (i.e. the segment being observed). The same observation may be associated with multiple results. In the case of the loop detector this might include not only vehicle volume, but also average vehicle speed. This example is illustrated in Figure 15. Note that the Units of Measure ontology also plays a role in capturing the observed values.

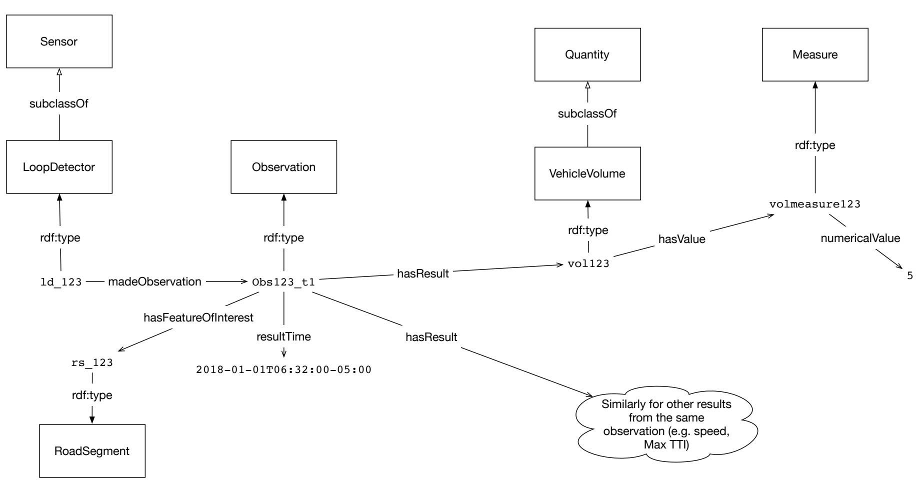


Figure 15: Example use of the Observations Ontology.

6.1.9.3 Future work

Add logic to relate the values of observable property, observation, feature of interest, and result: the observable property indicates how (by what property) the result relates to the feature of interest; e.g. the location of the loop detector indicates the identity of the feature of interest of its observations.

Reused Ontologies:

- W3C SSN Ontology

6.2 Contact Ontology

http://ontology.eil.utoronto.ca/icity/Contact

Namespace: contact

Contact information is relevant for a range of concepts in the transportation domain. For example, a building may have some associated address, similarly a person or an organization may have some contact address (or phone number, email, etc). Note that a person's contact address may differ from their place of residence. The iContact ontology is reused to provide the core concepts necessary to define this type of information. The Contact ontology uses concepts from the spatial location ontology in order to associate an address with a location. It also introduces a more specific definition of hours of operation as a specialization of the RecurringEvent class.

Object	Property	Value
contact:Address	hasStreetNumber	exactly 1 xsd:nonNegativeInteger
	hasStreet	only xsd:string
	hasCity	exactly 1 schema:city
	…	
	spatialloc:hasLocation	exactly 1 geo:Feature
	subClassOf	iContact:Address
contact:HoursOfOperation	subClassOf	icontact:HoursOfOperation
	subClassOf	rec:RecurringEvent

Reused Ontologies:

- iContact: http://ontology.eil.utoronto.ca/icontact.owl
- iCity Spatial Location: *http://ontology.eil.utoronto.ca/icity/SpatialLoc/*

Future Work:

In future extensions it may be useful to consider the addition of properties such as the time zone (time:TimeZone) associated with an address, as well as the primary language of correspondence. The iContact ontology also introduces an object property: has Geo Coordinates. Future work should consider how the relationship between the coordinates of an address and the location it occupies can be formalized. Are the address coordinates always contained within the location in space, or are there some exceptions?

6.3 Person Ontology

http://ontology.eil.utoronto.ca/icity/Person

Namespace: person

- Person: A Person may have a **unique identifier**.

  A Person has a **date of birth**, and may have a **date of death**.

  A Person has a **mother** and **father**, and may have a **spouse** and/or **child**(ren). Note that we define the parent relation as the legal relation as opposed to biological. This property may be specialized and restricted, for example hasBiologicalMother: exactly 1 Person.

  A Person may **have** some **Job** and associated **Income**.

  A Person has an **address** of residence and may have other contact information such as **E-mail**, **phone number**, etcetera.

  A Person has some age and exactly 1 sex, and sex may be one of only male or female. The definition of sex is distinct from that of a person’s gender: “Sex refers to sex assigned at birth. Sex is typically assigned based on a person's reproductive system and other physical characteristics.”[9] Future extensions may incorporate a representation of gender, should it be required.

  A person has some Age may or may not be a licensed driver.

Object	Property	Value

[9] http://www23.statcan.gc.ca/imdb/p3Var.pl?Function=DEC&Id=24101

PersonPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Person and change:hasManifestation only Person
	change:existsAt	exactly 1 time:Interval
	hasPersonID	only PersonId
	schema:birthDate	exactly 1 time:Instant
	hasSex	exactly 1 Sex
	schema:deathDate	max 1 time:Instant
Person	equivalentClass	change:manifestationOf some PersonPD and change:manifestationOf only PersonPD
	subclassOf	change:Manifestation
	change:existsAt	exactly 1 time:TemporalEntity
	hasAge	exactly 1 om:duration
	isLicensedDriver	exactly 1 xsd:boolean
	schema:parent	only Person
	schema:spouse	only Person
	schema:children	only Person
	hasIncome	only MonetaryValue
	schema:address	some schema:PostalAddress
	hasSkill	only Skill
	hasQualification	only Qualification
Sex	equivalentClass	{person:male, person:female}

Reused Ontologies:

- schema.org[10] (A vocabulary as opposed to an ontology)
- Change ontology
- Units of measure ontology
- Time ontology

[10] http://schema.org/

Future work:

- Attributes such as isLicensedDriver are currently captured as (Boolean) data properties. Future extensions may capture these attributes as object properties, should a more detailed representation be required (e.g. the introduction of a DriversLicense class, with attributes such as its category, expiration date, province of issue, etc). This possibility for future extension applies to many of the defined data properties in the icity ontologies in general.

6.4 Household Ontology

http://ontology.eil.utoronto.ca/icity/Household.owl

Namespace: household

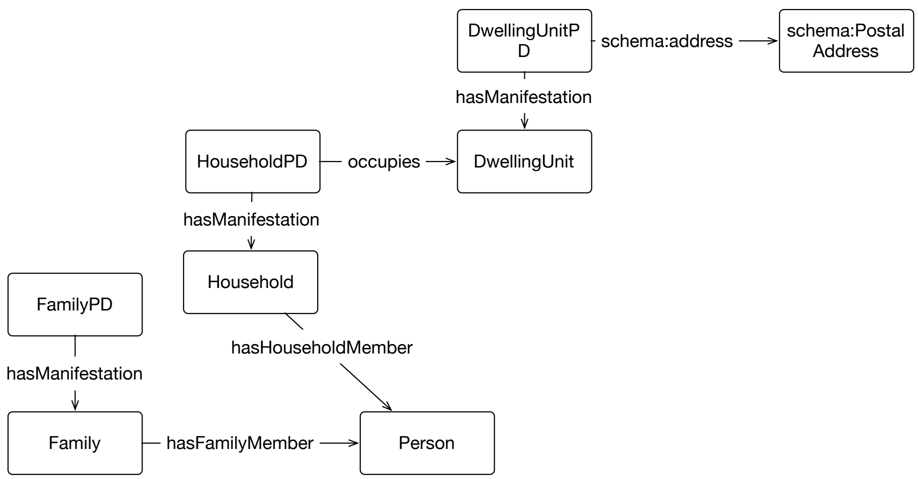
In order to define a Household, we require the following classes and properties:

- Family: The notion of Family simply makes the commitment that it is a group of people who are connected via the has-spouse or has-child properties. From these, we can derive grandparents, aunts, uncles, etcetera.
  One question to consider is to what degree the general/extended Family concept makes sense or is useful. After a few generations the concept of a family will become quite large and confusing, with Persons belonging to many different Families. It may be more useful to consider a relatedTo property between Persons, or only defining restricted subclasses of Family; for example, different types of Family (e.g. Immediate, Extended) may be defined.
- Household: A Household **occupies** a particular Dwelling, according to some **tenure** type. It is defined by this location, so that if the members move (even collectively), the new residence constitutes a new Household.
  Note that a Household, and likely many other classes may have different definitions in different contexts/applications. To address this we may be required to introduce specializations of the class (e.g. ILUTE_Household, TTS_Household) in future extensions.

- Dwelling Unit: A Dwelling Unit is **occupied** by a Household.

  A Dwelling Unit has a **market value**.

  A Dwelling Unit has some Location.



Object	Property	Value
FamilyPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Family and change:hasManifestation only Family
	change:existsAt	exactly 1 time:Interval
Family	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some FamilyPD and change:manifestationOf only FamilyPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasFamilyMember	min 2 person:Person
HouseholdPD	subclassOf	change:timeVaryingConcept
	equivalentClass	change:hasManifestation some Household and change:hasManifestation only Household
	change:existsAt	exactly 1 time:Interval
	occupies	exactly 1 DwellingUnit

Household	subclassOf	change:Manifestation
	subClassOf	gci:Household
	equivalentClass	change:manifestationOf some HouseholdPD and change:manifestationOf only HouseholdPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasHouseholdMember	only person:Person and some person:Person
DwellingUnitPD	subclassOf	change:TimeVaryingConcept
	subclassOf	building:BuildingUnitPD
	equivalentClass	change:hasManifestation some DwellingUnit and change:hasManifestation only DwellingUnit
	change:existsAt	exactly 1 time:Interval
	schema:address	only schema:PostalAddress
	spatial_loc:hasLocation	only spatial_loc:SpatialFeature
DwellingUnit	subclassOf	change:Manifestation
	subclassOf	building:Building and building:BuildingUnit
	equivalentClass	change:manifestationOf some DwellingUnitPD and change:manifestationOf only DwellingUnitPD
	change:existsAt	exactly 1 time:TemporalEntity
	occupiedBy	exactly 1 Household
	hasValue	only monetary:MonetaryValue
	tenureType	only Tenure

Property	Characteristic	Value (if applicable)
occupiedBy	inverseOf	Occupies
hasFamilyMember	subPropertyOf	mer:hasComponent

hasHouseholdMember	subPropertyOf	mer:hasComponent

Future Work:

- Extend the definitions of the classes beyond OWL to capture the different notions of family membership and the types (subclasses) of Family that result.
  - hasParent subpropertyOf isRelated
  - hasChild subpropertyOf isRelated
  - hasParent o hasChild subpropertyOf isRelated
  - hasChild o hasParent subPropertyOf isRelated
  - hasParent o (hasParent)- subPropertyOf isRelated
  - …

Reused Ontologies:

- schema.org
- gci: GCI-Shelter Ontology[11]
- mer:Mereology Ontology

6.5 Organization Ontology

http://ontology.eil.utoronto.ca/icity/Organization.owl

Namespace: org

- Organization: A company or other sort of group of individuals in the urban system with some goal(s).

  An Organization may **own** Property, including different types of Buildings.

  An Organization may have an address.

  An Organization has at least 2 members.

  An Organization has some Goal(s); this represents some state or complex states, and allows for the representation of various groups' responsibilities.

  An Organization may be divided into Divisions.

[11] http://ontology.eil.utoronto.ca/GCI/Shelters/GCI-Shelters.html

- Organization Agent: Members of an organization.
  Organization Agents have goals, authority, and may be members of some team.
  An Organization Agent plays a Role within the Organization.
- Role: A Role has a single (possibly complex) Goal.
  A Role has some authority, requires some skill, and may also have some associated processes.
- Firm: A Firm is a type of organization.
  A Firm has an address and an industry type, and some Employees.
  A Firm may have a Business Establishment(s).
- Business Establishment: A Business establishment is a physical location where a Firm conducts business.
  A Business Establishment has a Location and may have an address.
- Employee: A Firm **has** some **Employees**, whom it **employs for** some Occupation.
  An Employee is a type of Organization Agent.
  An Employee may be employed at a particular Business Establishment.
  An Employee may be responsible for one or more Roles within the Organization.
  An Employee is **employed by** some Organization, unless the Person is self-employed.
  An Employee **has a Wage/Salary** and **may work at** some **Location** (this may be the location of the Firm, an alternate Location, or a Location that is subject to change).
  An employee has some employment status. An employment status may be categorized as one of: full-time regular, part-time regular, full-time-work-at-home, part-time-work-at-home
- Student: A Student is a kind of Organization Agent (and Person) who is enrolled in some EducationalInstitution
- Occupation: An occupation describes the type of work performed by some employee.
  Different classes of occupations may be defined, such as: General Office / Clerical, Manufacturing / Construction / Trades, Professional / Management / Technical, Retail Sales and Service

Object	Property	Value
OrganizationPD	subclassOf	change:TimeVaryingConcept

	equivalentClass	change:hasManifestation some Organization and change:hasManifestation only Organization
	change:existsAt	exactly 1 time:Interval
Organization	subclassOf	change:Manifestation
	subclassOf	tove:Organization
	equivalentClass	change:manifestationOf some OrganizationPD and change:manifestationOf only OrganizationPD
	change:existsAt	exactly 1 time:TemporalEntity
	schema:address	only schema:PostalAddress
	tove:has_goal	only tove:Goal
	tove:consists_of	only tove:Division
	spatialloc:assiociatedLocation	only geosparql:Feature
tove:Role	tove:has_goal	only tove:Goal
	tove:has_process	only (tove:Process or activity:Activity)
	tove:has_authority	only tove:Authority
	tove:requires_skill	only tove:Skill
	tove:has_resource	only resource:ResourceType
tove:Goal	subClassOf	StateType
FirmPD	subclassOf	tove:Organization
	hasFirmId	only FirmId
	equivalentClass	change:hasManifestation some Firm and change:hasManifestation only Firm
	change:existsAt	exactly 1 time:Interval

Firm	subclassOf	tove:Organization
	equivalentClass	change:manifestationOf some FirmPD and change:manifestationOf only FirmPD
	change:existsAt	exactly 1 time:TemporalEntity
	schema:address	exactly 1 schema:PostalAddress
	hasIndustryType	only IndustryType
	hasEstablishment	only BusinessEstablishment
BusinessEstablishmentPD	subclassOf	change:TimeVaryingConcept
	change:existsAt	exactly 1 time:Interval
	hasBusinessId	only BusinessId
	equivalentClass	change:hasManifestation some BusinessEstablishment and change:hasManifestation only BusinessEstablishment
BusinessEstablishment	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some BusinessEstablishmentPD and change:manifestationOf only BusinessEstablishmentPD
	change:existsAt	exactly 1 time:TemporalEntity
	spatial_loc:hasLocation	exactly 1 spatial_loc:SpatialFeature
	schema:address	only schema:PostalAddress
tove:OrganizationAgent	tove:member_of	only tove:Division
	tove:plays	only tove:Role
	tove:has_goal	only tove:Goal
	tove:has_authority	only tove:Authority

Employee	subclassOf	tove:OrganizationAgent
	employedAs	some Occupation
	hasPay	some Wage or Salary
	worksAt	some spatial_loc:SpatialFeature
	hasEmploymentStatus	only EmploymentStatus
FullTimeEmployee	subClassOf	Employee
FullTimeHomeEmployee	subClassOf	FullTimeEmployee
FullTimeRegEmployee	(subClassOf	FullTimeEmployee) and (not FullTimeHomeEmployee)
PartTimeEmployee	subClassOf	Employee
PartTimeHomeEmployee	subClassOf	PartTimeEmployee
PartTimeTimeRegEmployee	(subClassOf	PartTimeEmployee) and (not PartTimeHomeEmployee)
Wage	hourlyPay	exactly 1 monetary:MonetaryValue
	overtimePay	only monetary:MonetaryValue
Salary	hasAnnualPay	exactly 1 monetary:MonetaryValue
tove:Activity	equivalentClass	activity:Activity
tove:Resource	equivalentClass	resource:Resource
EmploymentStatus	equivalentClass	{fulltime_regular, parttime_regular, fulltime_home, parttime_home}
GeneralOffice	subClassOf	Occupation
Trades	subClassOf	Occupation
Professional	subClassOf	Occupation
Sales	subClassOf	Occupation
EducationalInstitution	subClassOf	Organization

Student	subClassOf	OrganizationAgent
	enrolledIn	min 1 EducationalInstitution
FullTimeStudent	subClassOf	Student
PartTimeStudent	subClassOf	Student

- **hasOrgMember subPropertyOf tove:hasMember**
- **org:Organization hasOrgMember min 2 tove:OrganizationAgent**

Reused Ontologies:

- tove: The TOVE Organization ontology[12], as originally presented by [27] with modifications to account for the difference in our representation of states, where a Goal is a subclass of StateType, and where Activities are enabled/caused by state types.
  This modification also results in the removal of the StateEmpowerment class. Note that it is possible to introduce a similar concept if required, however this would likely take the form of a property that relates an organization agent to some state-types (where the states they are empowered to take an object to, and the object itself, are described by the state type).
- icity-foundation: iCity-Foundation Ontology
- schema.org (vocabulary)

Future Work:

- Define part-time / full-time employees and students in more detail (e.g. with respect to their work locations).
- Define part-time /full-time students according to some enrollment criteria

6.6 Building Ontology

http://ontology.eil.utoronto.ca/icity/Building.owl

Namespace: building

- Building: A Building is a structure with some location in the urban system. Many properties of a Building may change over time, (even the exact location of the Building in may change due to construction), but its Address cannot.

[12] http://ontology.eil.utoronto.ca/tove/organization.html

There are different types (**subclasses**) of buildings, such as House, Apartment Building, Office Building, and so on.

A Building or BuildingUnit may contain some Building Facility(s), e.g. kitchen, bath, or air conditioning. Note that this is distinct from the notion of including amenities that are not a physical part of the Building (Unit), but which may be part of the Tenure.

A Building has a market **value**.

A Building **has** some **Location**.

A Building has some height, some footprint area, and some floor area. The floor area is often greater than the footprint area as it accounts for the area of each floor of the building. However, floor area excludes unoccupied areas such as basements. These properties are considered variant as it is possible for a building to undergo construction to increase its dimensions.

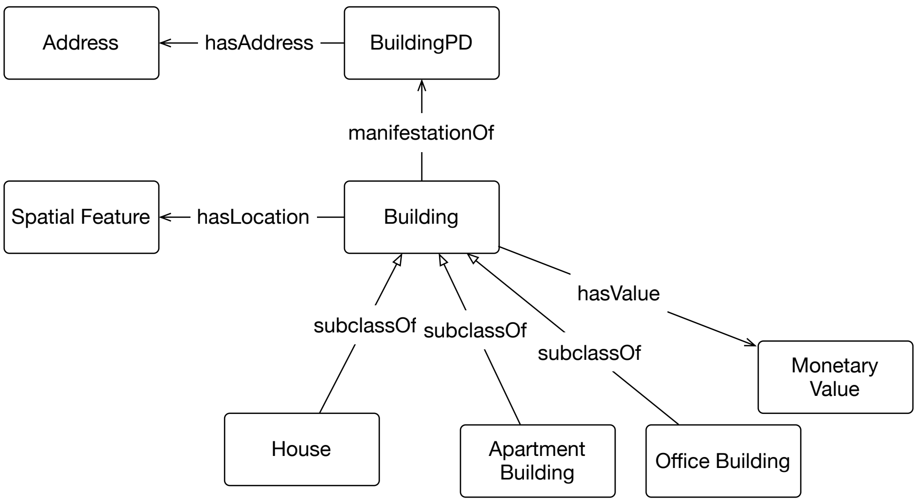
A Building contains one or many units.

- BuildingFacility: A Building Facility refers to services/features that are included in the Building/Building unit by nature of its physical design (e.g. HVAC, kitchen, bathroom, etc)
- BuildingUnit: A BuildingUnit has a size (square footage, number of rooms)

  A Building or BuildingUnit may contain some Facility(s), e.g. kitchen, bath, or air conditioning. Note that this is distinct from the notion of including amenities that are not a physical part of the Building (Unit), but which may be part of the Tenure.

  A BuildingUnit has an address.

  A BuildingUnit has a value, and may have some rental fee.



Object	Property	Value
BuildingPD	subClassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Building and change:hasManifestation only Building
	contact:hasAddress	only contact:Address
	change:existsAt	exactly 1 Interval
Building	equivalentClass	change:manifestationOf some BuildingPD and change:manifestationOf only BuildingPD
	subClassOf	change:Manifestation
	change:existsAt	exactly 1 TemporalEntity
	spatial_loc:hasLocation	exactly 1 spatial_loc:SpatialFeature
	monetary:hasValue	only monetary:MonetaryValue
	hasBuildingFacility	only BuildingFacility
	hasBuildingUnit	only BuildingUnit
House	subclassOf	Building
ApartmentBuilding	subclassOf	Building
OfficeBuilding	subclassOf	Building

IndustrialBuilding	subclassOf	Building
BuildingUnitPD	subclassOf	change:TimeVaryingConcept
	change:existsAt	exactly 1 Interval
	equivalentClass	change:hasManifestation some BuildingUnit and change:hasManifestation only BuildingUnit
	unitInBuilding	exactly 1 Building
	Contact:hasAddress	exactly 1 contact:Address
BuildingUnit	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some BuildingUnitPD and change:manifestationOf only BuildingUnitPD
	change:existsAt	exactly 1 TemporalEntity
	monetary:hasValue	only monetary:MonetaryValue
	hasRent	only monetary:MonetaryValue
	hasUnitSize	only om:area
	hasRooms	only xsd:int
	~~hasFacility~~ hasBuildingFacility	only Facility

Property	Characteristic	Value (if applicable)
hasBuildingFacility	subPropertyOf	mer:hasComponent
hasBuildingUnit	inverseOf	unitInBuilding
	subPropertyOf	mer:hasComponent
	subPropertyOf	mer:contains
unitInBuilding	inverseOf	hasBuildingUnit
	subPropertyOf	mer:componentOf
	subPropertyOf	mer:containedIn

Reused Ontologies:

- Change
- Units of measure
- Mereology
- Spatial location

Future work:

- Consider adding an BuildingAmenity class to capture common spaces or features may be included / excluded for occupants by virtue of some rental agreement

6.7 Vehicle Ontology

http://ontology.eil.utoronto.ca/icity/Vehicle.owl

Namespace: icity-vehicle

- Vehicle: A Vehicle provides a means of transportation within the urban system.
  A Vehicle is **associated with some Mode** of transportation.
  A Vehicle has a Vintage.
  A Vehicle has a Manufacturer (make).
  There are different types (**subclasses**) of vehicles: Motorcycle, Sedan, Truck, Bus, Commercial Cargo Vehicle, … These types may be identified and defined in different, complementary ways. The VehicleType class allows for the specifications of various types of vehicles, which may or may not also be captured as subclasses of the Vehicle class. Should a vehicle type also be a subclass, then the subclass should be defined such that it is equivalent to the class of all individuals that have the vehicle type as a property *hasVehicleType value <vehicle type>*.
  A Vehicle **has a capacity** of passengers
  A Vehicle **has a capacity** of cargo
  A Vehicle **has a Speed** at some point in time
  A Vehicle **has a location** at some point in time.

Object	Property	Value
VehiclePD	equivalentClass	change:hasManifestation some Vehicle and

		change:hasManifestation only Vehicle
	subclassOf	change:TimeVaryingConcept
	change:existsAt	exactly 1 time:Interval
	hasVehicleType	only VehicleType
	schema:productionDate	only time:DateTimeDescription
	schema:brand	only schema:Brand
	schema:vehicleSeatingCapacity	exactly 1 xsd:int
	schema:cargoVolume	only om:volume
	hasCargoCapacityLoad	only om:Quantity
	schema:driveWheelConfiguration	schema:DriveWheelConfigurationValue
	schema:fuelConsumption	schema:QuantitativeValue
	schema:fuelEfficiency	schema:QuantitativeValue
	schema:fuelType	schema:QualitativeValue
	schema:mileageFromOdometer	schema:QuantitativeValue
	schema:numberOfDoors	only xsd:int
	schema:numberOfAxels	only xsd:int
Vehicle	equivalentClass	change:manifestationOf some VehiclePD and change:manifestationOf only VehiclePD
	subclassOf	change:Manifestation
	change:existsAt	exactly 1 time:TemporalEntity
	schema:purchaseDate	only time:DateTimeDescription
	hasSpeed	only om:speed
	spatial_loc:hasLocation	only spatial_loc:SpatialFeature
	accommodatesWheelchair	max 1 xsd:Boolean

	accommodatesBicycle	max 1 xsd:Boolean
schema:QualitativeValue	subClassOf	om:quantity

Ontologies Reused:

- Schema.org (vocabulary)
- iCity-Foundation

6.8 Transportation System Ontology

http://ontology.eil.utoronto.ca/icity/TransportationSystem.owl

Namespace:transport

While most existing work attempts to describe the network based on its physical constructs, we model the network flow and the physical infrastructure separately. The motivation for this is that the constraints on transportation flow are something that is *applied to* the physical infrastructure. These constraints are distinct from the physical characteristics and so should be defined separately. Although some constraints may be related, such as flow constraints imposed by the size of the lane that an arc accesses, this is a specific relationship that should be captured rather than conflating the concepts. For example, there is nothing to stop a vehicle from going the wrong way on a road, except for the flow of traffic that is imposed on the system (and these constraints may change with time). This results in the identification of two key concepts: the Transportation Network (a directed graph), and the Transportation Complex (a physical feature where transportation occurs).

We relate the Network and the Infrastructure by relating an Arc to a Transportation Complex (or other Road Segment) with the "accesses" property. In this way, we may define an Arc accessing various Transportation Complexes at different Levels of Detail (LOD).
Both Nodes and Arcs may have implicit locations based on the infrastructure they access, however unlike the infrastructure classes, Nodes and Arcs are *not* Spatial Things. A Node may have a control (e.g. a signal) with a physical presence somewhere else (traffic lights apply to one side of the intersection, but are actually located on the other side of the intersection); by

separating the physical infrastructure and the network flow we are able to accurately represent this.

The OTN (Ontology of Transportation Networks[13]) ontology, as presented by [28], also defines terms such as nodes, arcs, and road/rail elements. The lack of maintenance and activity on the OTN poses a potential issue, and the lack of modularity in its structure makes it difficult to use. Therefore, although its scope is similar, we have elected not to reuse it in the design of this ontology.

- Network: A collection of Nodes and Arcs that enables transportation. A Network may have some cost associated to its access.
- Link: A directed connection in the Network that enables transportation via some Mode(s) from one Node to another.
  A link contains one or more Arcs that represent individual flows of traffic (e.g. traffic lanes, bicycle lanes).
  A link begins and ends at a source and sink Node.
  A link has some (straight-line) length description, in km.
  A link is associated with, or considered to be *in*, a municipality and a planning district.
  A link supports one or more Mode(s) of access.
- Arc: A directed connection in the Network that enables transportation via a particular Mode(s) from one Node to another.
  An Arc begins and ends at the source and sink of the Link it is contained in.
  An Arc has access to some Spatial Thing (such as a road), which may change over time.
  An Arc may impose access restrictions (for example, based on the size of vehicle), which are subject to change.
  An Arc may have some cost associated to its travel.
  An Arc supports one or more Modes of access.
  An Arc may have some posted and/or free flow speed. It may also be described with a volume delay function (VDF).

[13] http://www.pms.ifi.lmu.de/rewerse-wga1/otn/OTN.owl

- Node**:** A point in the Network at which Arcs are connected. A node as a unique identifier; for example, as defined in the EMME NCS11.
  A Node may contain different types of controls: Network Transfer, Signal Control, and Flow Control.
  A Node may be associated with specific location information (e.g. coordinates). Note that this may be subject to change. The physical location of a node (generally larger than a single point) may be inferred based on the locations of the transportation complexes which it connects.
  A Node accesses some TransportationComplex, such as an Intersection. In the future, it may be useful to define other specific types of TransportationComplexes that are accessed by nodes, (e.g. bus stops).
- Network Transfer: Enables transfer between networks at a given Node.
- Signal Control: Controls the flow of transportation between some of the incoming and outgoing arcs that the Node connects. Signal Controls have specialized attributes such as the number of phases, phase length, signal timing, type of signal. Note that the phases and/or the phase length may vary as a function of time of day or other triggers (e.g. ground sensors, traffic sensors).
- Flow Control: Controls the flow of traffic at a given Node.
  A Flow Control may be operative/inoperative at different times. For example, "no left turns from 4-6pm".
  A Flow Control may be a generalization of Signal Control.
- Mode: A mode of transportation is a **means of** performing travel within the urban system. There are various types (instances) of Mode: Foot, Bike, PersonalVehicle, PublicTransit, Cab, CommercialVehicle, Plane, Boat, Train.
- LoopDetector: A Loop Detector is a kind of Sensor that detects vehicle presence at some point on a road segment. A Loop Detector is owned by some Organization; it has some location, and is associated with (has a feature of interest) the particular part of the transportation network (i.e. a transport:Arc) that it is located on.
  A Loop Detector makes observations about the vehicle presence on the road segment that is its feature of interest.

The vehicle presence is a proxy for the occupancy of the road segment and the average vehicle speed on the road segment.

- TTI
- MeanTTI

The physical Infrastructure of the transportation system is defined, as required, at different levels of detail (LOD). Specific types of Transportation Complex (a term we adopt from the CityGML schema) may be defined according to the Arcs that access them. We define the following types of Transportation Complex.

- Road
- Rail
- Waterway
- Airway
- Bike Trail
- Footpath
- Parking

Each Transportation Complex may be further defined as follows:

- **Road**: An aggregation of Road Segments with the same name.
- **RoadSegmentPD**: accessed only by Links that are not accessible by water or air modes. Different RoadSegments Perdurants will be accessed by Arcs that are accessible by various other Modes, not necessarily *everything* else. A Road Segment Perdurant is comprised of Road Segments that exist over time.
- **RoadSegment**: A RoadSegment has variant attributes.
  A RoadSegment has an owner, access restrictions, and is accessed by some Arc(s) -- all of which may change over time.
  A RoadSegment has some location, which is co-located with (contains the locations of) the Arcs and Nodes it contains.
- **Rail**: An aggregation of Rail Segments with the same name.
- **RailSegmentPD**: Accessed only by Arcs that are accessible by rail modes.
  A RailSegment Perdurant has an invariant location, which is co-located with (contains

the locations of) the Arcs and Nodes it contains. A Rail Segment Perdurant is comprised of Rail Segments that exist over time.

- **RailSegment**: A RailSegment has an owner, access restrictions, and is accessed by some Link(s).
- Note that the location of a RoadSegment is variable (e.g. road widening or other activities do not change the identity of the road element), whereas a RailSegment's is not.
- **IntersectionPD:** Accessed only by NodePDs. An Intersection Perdurant captures the physical entity of an intersection, which is co-located with various other transportation complexes (e.g. roads, paths) that pass through it. An Intersection Perdurant is comprised of Intersections that exist over time.
- **Intersection:** An Intersection exists at some time. It has some location. It may have some owner and is accessed by some Node. In the future, it may be useful to extend this class and relate it to certain aspects of the physical infrastructure such as signs, signals, etc.

Classes may be defined for footpaths, bicycle lanes/trails, and so on. Should it be useful, this representation could be extended to define individual traffic lanes, (e.g. the transportation complex that is accessed by a single arc).

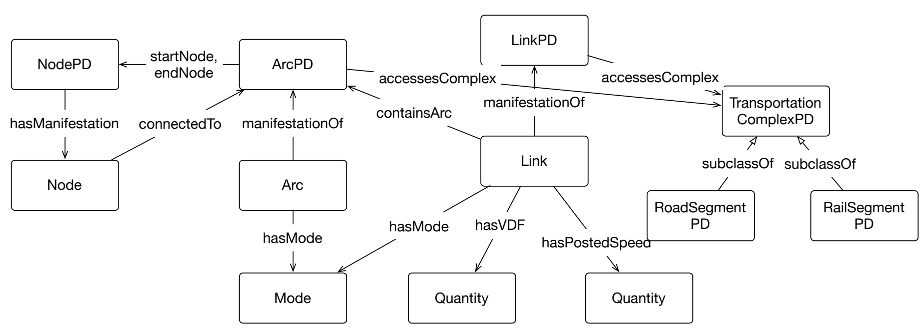


Figure 16: Structure of the Transportation Network (some omissions).

Object	Property	Value
NetworkPD	subclassOf	change:TimeVaryingConcept

	equivalentClass	change:hasManifestation some Network and change:hasManifestation only Network
	change:existsAt	exactly 1 time:Interval
Network	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some NetworkPD and change:manifestationOf only NetworkPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasNetworkComponent	only Arc or Node
NodePD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Node and change:hasManifestation only Node
	change:existsAt	exactly 1 time:Interval
	hasNodeID	max 1 NodeId
Node	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some NodePD and change:manifestationOf only NodePD
	change:existsAt	exactly 1 TemporalEntity
	inverse (hasNetworkComponent)	only Network
	connectedTo	min 1 Arc
	hasControl	only (NetworkTransfer or SignalControl or FlowControl)
	associatedLocation	only spatial_loc:Feature
LinkPD	subclassOf	change:TimeVaryingConcept

	equivalentClass	change:hasManifestation some Link and change:hasManifestation only Link
	change:existsAt	exactly 1 time:Interval
	startNode	exactly 1 NodePD
	endNode	exactly 1 NodePD
	accessesComplex	only TransportationComplexPD
Link	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some LinkPD and change:manifestationOf only LinkPD
	change:existsAt	exactly 1 time:TemporalEntity
	containsArc	min 1 ArcPD
	inverse (hasNetworkComponent)	only Network (variant or invariant?)
	associatedLinkLength	exactly 1 om:length
	supportsMode	min 1 Mode
	hasNumLanes	exactly 1 xsd:integer
	hasVDF	max 1 om: Quantity
	hasLinkCapacity	max 1 (om:Quantity and om:'has value' only (om:'has unit' only (om:'has numerator' only om:CardinalityUnitPerTime) and (om:'has denominator' only (om:'Cardinality Unit' and inverse(om:'has unit') only (inverse(om:'has value') only (gci:cardinality_of only (gci:defined_by only Arc))))))

	hasFreeFlowSpeed	max 1 om:speed
	hasPostedSpeed	max 1 om:speed
	hasToll	only MonetaryValue
	inMunicipality	exactly 1 Municipality
	inPlanningDistrict	exactly 1 PlanningDistrict
ArcPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Arc and change:hasManifestation only Arc
	startNode	exactly 1 NodePD
	endNode	exactly 1 NodePD
	change:existsAt	exactly 1 time:Interval
	accessesComplex	only TransportationComplexPD
	containedInLink	exactly 1 LinkPD
Arc	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some ArcPD and change:manifestationOf only ArcPD
	change:existsAt	exactly 1 time:TemporalEntity
	accessesComplex	only TransportationComplex
	inverse (hasNetworkComponent)	only Network
	hasControl	only AccessRestriction
	supportsMode	min 1 Mode
	hasLaneCapacity	exactly 1 om:CapacityRate
	hasVDF	max 1 om:quantity
	hasFreeFlowSpeed	max 1 om:speed
	hasPostedSpeed	max 1 om:speed
	hasToll	only MonetaryValue
	inMunicipality	only Municipality
	inPlanningDistrict	exactly 1 PlanningDistrict

NetworkTransfer	controlFor	only Node
	connectsNetworks	min 2 Network
FlowControl	controlFor	only Node
	hasInflow	min 1 Arc
	hasOutflow	min 1 Arc
SignalControlPD	subClassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some SignalControl and change:hasManifestation only SignalControl
	change:existsAt	exactly 1 time:Interval
	controlFor	only Node
	hasInflow	min 1 Arc
	hasOutflow	min 1 Arc
SignalControl	subClassOf	change:Manifestation
	equivalentClass	change:manifestationOf some SignalControlPD and change:manifestationOf only SignalControlPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasPhase	only SignalPhase
SignalPhase	signalLength	only time:DurationDescription
TransportationComplexPD	subClassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some TransportationComplex and change:hasManifestation only TransportationComplex
TransportationComplex	subclassOf	change:Manifestation

	equivalentClass	change:manifestationOf some TransportationComplexPD and change:manifestationOf only TransportationComplexPD
	spatial_loc:hasLocation	only spatial_loc:Feature
otn:Road	hasRoadId	only RoadId
	aggregationOf	only RoadSegment
RoadSegmentPD	subclassOf	TransportationComplexPD
	equivalentClass	change:hasManifestation some RoadSegment and change:hasManifestation only RoadSegment
	hasRoadSegmentId	only RoadSegmentId
	change:existsAt	exactly 1 time:Interval
RoadSegment	equivalentClass	otn:RoadElement
	subClassOf	TransportationComplex
	equivalentClass	change:manifestationOf some RoadSegmentPD and change:manifestationOf only RoadSegmentPD
	change:existsAt	exactly 1 time:TemporalEntity
	spatial_loc:hasLocation	only spatial_loc:Feature
	inMunicipality	only Municipality
Mode	equivalentClass[14]	{C,E,F,H,I,J,B,G,L,M,P,Q,R,S,A,K,T,U,V,W,Y}
Municipality		

[14] More options may be added as required. This list comes from the options specified in the EMME NCS11.

PlanningDistrict		
LoopDetector	sosa:detects	{vehicle_presence}
	sosa:observes	{road_occupancy}
	sosa:observes	{vehicle_volume}
	sosa:observes	{mean_travel_speed}
	sosa:madeObservation	only (sosa:Observation and sosa:hasFeatureOfInterest only transport:Arc and sosa:wasOriginatedBy {vehicle_presence} and sosa:hasResult RoadOccupancy or VehicleVolume or MeanTravelSpeed)
{vehicle_presence}	a	ssn:Stimulus
{road_occupancy}	a	ssn:ObservableProperty
{vehicle_volume}	a	ssn:ObservableProperty
{mean_travel_speed}	a	ssn:ObservableProperty
VehicleVolume	subClassOf	uom:Quantity
	uom:hasValue	only (uom:hasUnit only CardinalityUnitPerTime)
	gci:cardinalityOf	only LocVehiclePopulation
LocVehiclePopulation* *precise definition only possible for a particular location	gci:definedBy	only (Vehicle and hasLocation some Feature)
RoadOccupancy	subClassOf	uom:Quantity
	uom:hasValue	only (uom:hasUnit only RoadOccupancyUnit)
RoadOccupancyUnit	subClassOf	uom:UnitDivision
	uom:hasNumerator	only uom:TimeUnit
	uom:hasDenominator	only uom:TimeUnit
MeanTravelSpeed	subClassOf	uom:Speed
	uom:hasAggregateFunction	value {uom:average}

LaneCapacity_unit	subClassOf	uom:Unit
LinkCapacity_unit	subClassOf	uom:Unit
{vehicles_per_hour}	a	LaneCapacity_unit
{vehicles_per_hour_per_lane}	a	LinkCapacity_unit

1. Note that the classes of observable properties are primarily introduced for consistency with the SSN representation as a means of capturing the semantics of a class of Sensors (in this case, Loop Detectors). Any instance of, e.g. RoadOccupancy simply corresponds to a RoadSegment occupied by some thing, or occupied by nothing:
   RoadOccupancy(x) ⇔ isPropertyOf(x,y) & RoadSegment(y) & [exists (t) occupiedBy(y,t) | -exists(t) occupiedBy(y,t)]
   As a consequence of the 4D representation, an instance of the observable property RoadOccupancy refers to a property of a road segment at some time, t.
2. Additional semantics of sensors and observations: temporal restrictions, connection to object properties
3. A RoadSegment's vehicle count can be calculated based on the KB, but can we formalize the relationship between the count and the KB? i.e. number of observations in a given interval?

IntersectionPD	subclassOf	change:TimeVaryingConcept
	subclassOf	TransportationComplexPD
	equivalentClass	change:hasManifestation some Intersection and change:hasManifestation only Intersection
	inverse(accessesComplex)	only NodePD
	change:existsAt	exactly 1 time:Interval
Intersection	equivalentClass	otn:RoadElement
	subclassOf	change:Manifestation

	subClassOf	TransportationComplex
	equivalentClass	change:manifestationOf some RoadSegmentPD and change:manifestationOf only RoadSegmentPD
	change:existsAt	exactly 1 time:TemporalEntity
	spatial_loc:hasLocation	only geosparql:Feature
	inverse(accessesComplex)	only Node

Ontologies Reused:

- Change
- SpatialLoc
- SSN: Semantic Sensor Network ontology to capture sensors and their observations. These observations are processed or used directly as attributes of the network.

Notes:

- We observe that the properties *inMunicipality* and *inPlanningDistrict* may apply to other areas of the domain (e.g. land use, building ontologies), in which case they will be better defined at a lower (more foundational) level within the ontology. However, as they are currently only required for the Transportation System sub-ontology, it is currently not clear where and how this should be done. For now, we define these properties within the Transportation Network System ontology and leave the final organization for a future iteration if and when requirements for their widespread use are identified.

Future Work:

- Define lane and link capacity units in greater detail (e.g. with numerators and denominators).
- There is a relationship between the modes of access of a link and those of the arcs it contains that should be captured in a more detailed representation.
- The Municipality class should be defined in greater detail

6.8.1 Travel Costs

http://ontology.eil.utoronto.ca/icity/TravelCost.owl

Namespace: icity-travelcost

An extension of the transportation network (and other generic ontologies) is required in order to represent the different costs associated with accessing and travelling on the networks. These may take the form of direct costs such as tolls and fares, or possible indirect costs such as vehicle wear and tear, gas, etc. In addition, there may be non-monetary costs associated with travel such as pollution and travel time. Costs are associated with Network access, but also with individual Arcs. They may also be dependent on situational factors such as time of day, or age of traveler. Travel Costs define the costs associated with accessing the transportation system; a travel cost is a property of an arc or its network. We define a separate extension of Trip Costs to capture other, indirect costs that may vary between individual trips; a trip cost is a property of some instance of travelling.

- Travel Cost: There are different types of Travel Costs which are derived from different factors, and may be defined in different ways. Travel Costs apply to Arcs and / or Networks.
- Distance Fee is a type of Travel Cost
  Distance Fee has an associated Cost
  It applies for a certain distance (between nodes, or per km)
  It applies to some Arc
  It may have an associated time-of-day applicability
  It may be associated to specific modes of transport
- Access Fee is a type of Travel Cost
  Access Fee has an associated Cost
  It may have an associated time-of-day applicability
  It may be associated to specific modes
  It applies to some Network

Object	Property	Value
TravelCost	travelCostOf	only (transportation:Arc or transportation:Network)
	applicableFor	only time:TimePeriod or time:CalendarPeriod

	applicableTo	only transportation:Mode
	hasMonetaryCost	only monetary:MonetaryValue
transportation:Arc	hasTravelCost	only TravelCost
transportation:Network	hasTravelCost	only TravelCost
DistanceFee	subclassOf	TravelCost
	forDistance	only om:length
	travelCostOf	only transportation:Arc
AccessFee	subClassOf	TravelCost
	travelCostOf	only transportation:Network

Property	Characteristic	Value (if applicable)
travelCostOf	inverseOf	hasTravelCost

Ontologies Reused:

- iCity-Transportation Network

6.9 Parking Ontology

http://ontology.eil.utoronto.ca/icity/Parking.owl

Namespace: parking

- Parking Area: Parking Area refers to some area that enables parking of Vehicles.

  A Parking Area may contain **sub-Parking Areas**, the area of which may change.

  A Parking Area has some Parking **Policy**

  A Parking Area may provide **car changing** stations.

  A Parking Area has some **Location**.

  A Parking Area has some vacancy (or occupancy) at some point in time.

  A parking area may be owned by some Person or Organization, and it may be allocated for some Building, Location, Person, or Organization. Note that ownership and allocation of a Parking Area are distinct: an organization may own a parking area, but it may be allocated (e.g. rented) to some other organization or individual(s).

  A Parking Area may have some hours of operation.

  A Parking Area may have some limit on the dimensions of allowed vehicles (height/width/length)

associated location information (e.g. nearby crossroads, landmark, etc)
Different types (subclasses) of Parking Area may be defined as required, such as Street Parking Area, Lot Parking Area, Garage Parking Area, Illegal Parking Area, Loading/Unloading Zone Parking Area,…

- Parking Space: A Parking Space is a Parking Area with the capacity for a single vehicle. (hasCapacity 1, hasVacancy 0 or 1). Specializations of parking space may be defined based on accommodated vehicle type (e.g. small vehicles, commercial vehicles, electric vehicles,…).
  has Reservations?
  has Schedules?
  A Parking Space may or may not be occupied by some vehicle at a particular point in time. If a space is occupied, its availability may be determined (or approximated) based on the scheduled/purchased time by its current occupant.
- A Parking Facility is a parking area that is not contained by any other parking area. A Parking Facility may be owned by some organization and have some hours of operation; it may have a name, contact phone number, address, and possibly an associated website.
- Accessible Parking Space: A type of parking space reserved for users with disabilities
- EV Space: A type of parking space that provides access to some EV Charger(s)
- Parking Policy: A Parking Policy dictates under what terms some Parking Area is accessible for parking.
  A Parking Policy may have a **Rate**.
  A Parking Policy may have a **max duration**.
  A Parking Policy may have **allowable periods** *(these periods must be during the hours of operation of the parking area)*.
  A Parking Policy may apply only to a particular class of users.
  Different sorts of parking policies (subclasses) may be defined: e.g. free parking, policies for EVs, persons with accessibility needs.
- Rate: A Rate has a **monetary value** and an **associated duration**.
  A Rate has a **ParkingPaymentMethod** (e.g. mobile, license plate entry, cashier, meter).
  A Rate may have some minimum charge, specified as either a monetary value or duration (e.g. regardless of the time parked, the customer will be charged at least $5, or the rate

will be applied for at least 30 min). A maximum cost may also be specified; for example, the rate may be $5 per hour, with a maximum of $20 to park for the remainder of the policy's hours of operation. It is not always the case that the maximum cost coincides with the maximum time-based rate of the hours parked.

- EV charger: A charger for electric vehicles is an amenity which may be provided by some parking spaces.
  An EV charger has some model and is capable of charging certain classes of vehicles. An EV charger may be available or unavailable at a given time. This availability may be predetermined based on the scheduled duration of a vehicle's occupancy, and the time left to charge the vehicle.

Object	Property	Value
ParkingAreaPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some ParkingArea and change:hasManifestation only ParkingArea
	change:existsAt	exactly 1 time:Interval
	spatial_loc:hasLocation	exactly 1 spatial_loc:SpatialFeature
	spatial_loc:hasAssociatedLocation	only spatial_loc:SpatialFeature
	parkingPartOfBuilding	only Building
	maxAdmittableHeight	exactly 1 om:length
	maxAdmittableWidth	exactly 1 om:length
	maxAdmittableLength	exactly 1 om:length
	has Address	only icontact:Address
ParkingArea	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some ParkingAreaPD and change:manifestationOf only ParkingAreaPD

	change:existsAt	exactly 1 time:TemporalEntity
	hasSubParkingArea	only ParkingArea
	hasVehicleCapacity	only (CapacitySize and gci:cardinality_of only (gci:defined_by only Vehicle))
	hasParkingPolicy	only ParkingPolicy
	hasChargingStations	exactly 1 xsd:integer
	resource:ownedBy	some Person or Organization
	occupiedBy	only Vehicle
	isOpen	exactly 1 xsd:boolean
	hasParkingService	only ParkingService
	parkingAllocatedTo	only (Person or Building or Organization or Feature)
ParkingFacilityPD	subclassOf	park:ParkingAreaPD
	equivalentClass	change:hasManifestation some ParkingLot and change:hasManifestation only ParkingLot
ParkingFacility	subClassOf	ParkingArea
	subParkingAreaOf	exactly 0 ParkingArea
	foaf:name	only xsd:string
	icontact:hasWebsite	only xsd:string
	icontact:hasAddress	only contact:Address
	icontact:hasOperatingHours	only rec:HoursOfOperation
	icontact:hasTelephone	only icontact:PhoneNumber
ParkingSpace	subclassOf	ParkingArea
	hasVehicleCapacity	some (om:hasValue some (om:has_numerical_value value 1))
AccessibleSpace	subclassOf	ParkingSpace

	hasParkingPolicy	only AccessibilityParkingPolicy (to define)
EVSpace	subclassOf	ParkingSpace
	hasParkingPolicy	only EVParkingPolicty (to define)
ParkingService	*may be defined in greater detail in the future	
Valet	subclassOf	ParkingService
Carwash	subclassOf	ParkingService
ParkingPolicy	hasParkingRate	only ParkingRate
	maxDuration	only time:DurationDescription
	appliesDuring	only contact:HoursOfOperation
	appliesTo	only person:Person
	appliesFor	only vehicle:VehicleType
	hasGracePeriod	max 1 time:DurationDescription
	excludesPublicHoliday	exactly 1 xsd:boolean
ParkingRate	hasMonetaryCost	only om:MonetaryValue
	forDuration	only time:DurationDescription
	hasPayment	only ParkingPaymentMethod
	appliesTo	only person:Person
	minParkingCharge	only (om:MonetaryValue or time:DurationDescription)
	maxParkingCost	only om:MonetaryValue
FreeParkingPolicy	hasParkingRate	only (ParkingRate and hasMonetaryCost only (om:MonetaryValue and om:numerical_value [29]))

Property	Characteristic	Value (if applicable)
hasSubParkingArea	subPropertyOf	mer:hasProperPart
	domain	ParkingArea

	range	ParkingArea
	inverse	subParkingAreaOf
subParkingAreaOf	subPropertyOf	mer:properPartOf
	domain	ParkingArea
	range	ParkingArea
	inverse of	hasSubParkingArea

Ontologies Reused:

- Mereology
- Change
- Time
- OM
- Person
- Vehicle
- Contact

Future work:

Constraints may be defined to relate the hours of operation with the parking lot's associated parking policies and their hours of operation: a parking lot should have policies defined during all of its hours of operation.

If required, parking services may be defined in greater detail.

6.10 Public Transit Ontology

http://ontology.eil.utoronto.ca/icity/PublicTransit.owl

Namespace: transit

- TransitSystem: A TransitSystem is a **collection of Routes**.
  A TransitSystem may be **accessed by** some Fare or Transit Pass.
- Route: A Route consists of a series of Route Links and may be divided into Route Sections.
  A Route has some directionality (captured by the route links).

- Route Section: A Route Section is part of some Route and consists of Route Links. A Route Section begins and ends at a Stop Point.
- Route Link: A Route Link is part of some Route. It is a primitive element of a route, operating on single Arc or Link within the transportation system.
- Stop Point: A Stop Point marks the **start or end of** a Route Link (e.g. a subway stop or bus stop).
  A Stop Point is a subclass of a Node, as defined in the Transportation System ontology.
  Like a Node, a Stop Point **has an associated Location**.
  A Person may enter or exit the transit vehicle at a Stop Point.
  (to do: Station subclass of StopPoint)
- StationStopPoint: A StationStopPoint is a specialized type of Stop Point that contains multiple Stop Points. This is distinct from the Station itself (the building).
- Transit Incidents, broadly, are events of interest that occur on a particular transit trip. Typically, they are problematic, unplanned issues resulting in some delay.
  A TransitIncident is a subclass of Activity.
  It is associated with some station or stop point.
  An incident may be described (and so classified) by a predefined code: hasCode only xsd:String.
  An incident will have some resulting caused gap (i.e. the time from the incident until the next train arrives at the station).
- TransitTrip is a subclass of Trip.
  Transit Trips have specific restrictions and specialized properties. A Transit Trip occurs on some predefined route. A Transit Trip may also describe a trip on some smaller part of a Route, i.e. a Route Link. In exceptional cases, is possible that a TransitTrip may occur off-route (e.g. detours). The start and destination of a Transit Trip must be a Stop Point, and all Transit Trips must be performed with a Transit Vehicle.
- ScheduledTransitTrip is a type of RecurringEvent that only has TransitTrips as occurrences. A ScheduledTransitTrip is scheduled on some Route, RouteLink, or RouteSection, however it is not necessarily the case that the trip is accessible to travelers at the beginning stop point. It is possible that the scheduled trip will not pick up any passengers, or that passengers must pre-arrange in order to be picked up by the scheduled

trip. A Scheduled Transit Trip may have a pick-up type and/or drop-off type as defined by some Trip Access Arrangement Type: as scheduled, not available, arranged with agency, or arranged with driver.

ScheduledTransitTrips may be used to specify route and stop timetables. Like a TransitTrip, a ScheduledTransitTrip may be described as inbound or outbound with the isOutbound data property. Scheduled trips may be defined to require only the assignment of vehicles that accommodate a wheelchair rider(s); this property may be captured with the isWheelchairAccessible data property.

The start and end times of scheduled (recurring) transit trips may be used to specify route and stop timetables.

- TransitVehicle is a subclass of Vehicle.
  A TransitVehicle has a transit vehicle id. This refers to the identifier assigned by the transit authority, as opposed to a serial number.
  Transit Vehicles are owned and operated by some transit authority. There are specialized types of transit vehicles (e.g. different types of streetcars), and a restricted set of modes. Transit Vehicles typically only operate on pre-defined routes, however there are exceptions (e.g. detours, travel for maintenance, etc).
- AccessMethod: An Access Method is the means of **access to** a Line
  An AccessMethod **has a Monetary Value**.
  An AccessMethod may be **valid for** a specific distance or time.
- RouteTimetable: A Timetable represents schedule information for a particular Route, or Route Link.
  A RouteTimetable has an **expected travel time (Duration)** for the Route, or Route Link.
- A StopTimetable has an **expected arrival time (Time Instant)** for some Stop Point.
- VehicleBlock: A Vehicle Block represents a grouping of transit trips to be allocated to a particular vehicle. A transit trip is part of a single block and each block may contain multiple transit trips, therefore the allocatedFor property relating vehicle blocks and transit trips is inverse functional. Each block may be allocated multiple vehicles, but only one vehicle at a given point in time therefore the allocatedTo property which relates vehicle blocks to vehicles is functional.

- Two complementary properties (one object and one data property) have been added to capture information regarding transit passes. The data property provides a simply Boolean value to capture whether a person (at some time) has a transit pass; whereas the object property provides the ability to associate a particular transit pass (with some properties regarding, for example, its access, cost, and balance).

Object	Property	Value
TransitSystemPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some TransitSystem and change:hasManifestation only TransitSystem
	change:existsAt	exactly 1 time:Interval
	operatedBy	org:OrganizationPD
TransitSystem	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some TransitSystemPD and change:manifestationOf only TransitSystemPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasRoutes	only Route
	accessBy	only AccessMethod
AccessMethod	hasMonetaryCost	only monetary:MonetaryValue
	validFor	only (time:DurationDescription or om:length)
Fare	subclassOf	AccessMethod
TransitPass	subclassOf	AccessMethod
RoutePD	~~hasRouteId~~	~~exactly 1 RouteId~~
	subClassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Route and

		change:hasManifestation only Route
	change:existsAt	only time:Interval
	hasGTFSRouteType	exactly 1 {0,1,2,3,4,5,6,7}
Route	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some RoutePD and change:manifestationOf only RoutePD
	change:existsAt	only time:TemporalEntity
	routeShortName	max 1 xsd:string
	foaf:name	max 1 xsd:string
	hasSection	only RouteSection
	operatesOn	only ArcPD
	hasDisplayColor	max 1 xsd:string
	hasRouteTextColor	max 1 xsd:string
	icontact:hasOperatingHours	some rec:HoursOfOperation
RouteSection	mereology:contains	only RouteLink
	beginsAtStop	exactly 1 StopPoint
	endsAtStop	exactly 1 StopPoint
	operatesOn	only ArcPD
RouteLink	operatesOn	exactly 1 ArcPD
StopPoint	subclassOf	transport:Node
	spatial_loc:hasLocation	exactly 1 spatial_loc:Feature
	transit:hasStopCode	exactly 1 xsd:string
	foaf:name	min 1 xsd: string
	transit:wheelchairBoarding	exactly 1 xsd:boolean
AccessibleStopPoint	equivalentClass	StopPoint and transit:wheelchairAccessible value true

StationStopPoint	subclassOf	StopPoint
	mereology:contains	min 1 StopPoint
	spatial_loc:associatedLocation	some spatial_loc:Feature
TransitIncident	subclassOf	activity:Activity
	associatedWithStop	only StopPoint
	hasIncidentCode	min 1 xsd:string
	causedGap	only time:Interval
	associatedWithTrip	only TransitTrip
TransitTrip	subclassOf	trip:Trip
	transit:occursOn	only transit:Route or transit:RouteSection or transit:RouteLink or transport:TransportationComplex
	transit:viaVehicle	exactly 1 transit:TransitVehicle
	transit:isOutbound	only xsd:boolean
ScheduledTransitTrip	subclassOf	rec:RecurringEvent
	rec:hasOccurrence	only transit:TransitTrip
	transit:scheduledOn	only transit:Route or transit:RouteSection or transit:RouteLink
	transit:isOutbound	only xsd:boolean
	transit:isWheelchairAccessible	only xsd:boolean
	hasPickupType	max 1 TripAccessArrangement
	hasDropoffType	max 1 TripAccessArrangement
TripAccessArrangement	equivalentClass	{AccessAsScheduled, AccessNotAvailable, AccessArrangedViaAgency, AccessArrangedViaDriver}
TransitVehicle	subclassOf	vehicle:Vehicle
	hasTransitVehicleId	exactly 1 xsd:string

VehicleBlock	assignedTo	only transit:TransitVehicle
	assignedFor	min 1 ScheduledTransitTrip
person:Person	transitPass	only TransitPass
	hasTransitPass	only xsd:boolean

Ontologies Reused:

- Activity
- Change
- Spatial Location
- TransportationSystem
- Trip
- Organization

Future work:

Though not applicable for the TTC, future work should consider a representation of zone or similar information that may be used in some systems to calculate fare cost.

There is some potential to incorporate detailed constraints on the types of routes (bus, rail, etc) and the arcs in the network that the routes access, according to the mode supported by the arcs.

Constraints may also be enforced on the times of trips as compared to the hours of operation for a particular route (i.e. a trip should occur within the defined hours of operation).

With additional information, the stops associated with a particular trip may be validated against its direction id to confirm that the sequence is capturing either an inbound or outbound path.

Constraints may be added to enforce the types of vehicles that perform a particular transit trip, based upon the specifications of the scheduled trip of which the transit trip is an occurrence. For example, if the scheduled trip is wheelchair accessible, then any vehicle that performs the transit trips (or is assigned a block containing the scheduled trip) should accommodate a wheelchair. On the other hand, it may be the case that vehicle assignments sometimes conflict with the scheduled trip type and so such constraints may not be accurate/desirable.

We may also be able to infer whether a stop offers wheelchair boarding based on the associated routes and trips.

Rules may be added to express the relationship between the Arc that a Route Link operates on and the set of Arcs that a Route Section or Route operate on (i.e. the sum of all Arcs operated on by all Route Links contained in the Route Section/Route).
In extensions beyond OWL, it may be useful to formalize the relationship between transitPass and hasTransitPass properties.

6.11 Land Use Ontology

http://ontology.eil.utoronto.ca/icity/LandUse

Namespace: landuse

- Parcel: A Parcel is a way of defining some area in an urban system.
  A Parcel has a Location and an area that do not change over time.
  A Parcel may be associated with some type(s) of Land Use; this may change over time.
  There may be other types (subclasses) of Parcel, defined in more precise or different ways, such as a Zone.
  Alternatively, a Parcel may have some *associated* Area. This is a variant property as there may be various values with different accuracy from different sources.
  A Parcel may have some population that is subject to change over time.
  A Parcel may have a number of employed residents that is subject to change over time.
- LandUseClassification: Land Use Classifications provide a means of describing the land cover/use in a standard way. Various classification systems are used to identify types of land use. Currently, we include LBCS, CLUMP, and AAFC.
- The LBCS recognizes different dimensions of Land Use: Activity, Function, Structure, Site, and Ownership Classifications. Each dimension is further defined by a taxonomy of specialized classifications. For each dimension, we introduce an equivalent class name for disambiguation, e.g. to distinguish between the Activity dimension of land use (we refer to this as ActivityClassification) and the notion of an Activity in icity.
  - Activity Classification: An Activity Classification identifies the **activity use** of some Land Parcel.
    - Residential Activities
    - Shopping Activities
    - Industrial Activities

        - ...
    - Function Classification: A Function Classification identifies the **economic function of** some Land Parcel,
    - Structure Classification: A Structure Classification identifies the **type of structure(s) on** some Land Parcel.
    - Site Classification: A Site Classification identifies the **state of the site development on** some Land Parcel (e.g. is it developed or not?)
    - Ownership Classification: An Ownership Classification identifies any **constraints on the use of the land and its ownership for** some Land Parcel.
- CLUMPClassification: Canada Land Use Monitoring Program Classification is a type (subclass) of Land Use classification. CLUMP identifies 15 different types of land use, each with an associated code used in datasets. We have made the design decision that the code need not be unique to a particular land use classification, as a classification from one system may correspond to multiple classifications in CLUMP. CLUMP introduces the following land use classifications:
    - B - Urban built-up area
    - E - Mines, quarries, sand and gravel pits
    - O - Outdoor recreation
    - H - Horticulture
    - G - Orchards and vineyards
    - A - Cropland
    - P - Improved pasture and forage crops
    - K - Unimproved pasture and range land
    - T - Productive woodland
    - U - Non-productive woodland
    - M - Swamp, marsh or bog
    - S - Unproductive land - sand
    - L - Unproductive land - rock
    - 8 - Unmapped areas (technically not a CLUMP classification but it is used in the land use data)

- Z - Water areas (technically not a CLUMP classification but it is used in the land use data)

- AAFCClassification: Agriculture and Agri-Foods Canada Classification is a type (subclass of) land use classification. The codes are based on the IPCC (International Panel on Climate Change) protocol. We have made the design decision that the code need not be unique to a particular land use classification, as a classification from one system may correspond to multiple classifications in AAFC. AAFC uses the following land use classifications:
    - Unclassified
    - Settlement
    - Roads
    - Water
    - Forest
    - Forest Wetland
    - Trees
    - Treed Wetland
    - Cropland
    - Grassland Managed
    - Grassland Unmanaged
    - Wetland
    - Wetland Shrub
    - Wetland Herb
    - Other land
- TrafficZone: traffic zone is a kind of (subclass of) Parcel. It may be identified with a predefined set of identifiers, corresponding to its centroid node ID.

Object	Property	Value
ParcelPD	subclassOf	change:TimeVaryingConcept
	equivalentClass	change:hasManifestation some Parcel and

		change:hasManifestation only Parcel
	change:existsAt	exactly 1 time:Interval
	hasParcelSize	exactly 1 om:area
	spatial_loc:hasLocation	exactly 1 spatial;Feature
Parcel	subClassOf	lcbcs:Parcel
	subClassOf	spatial_loc:Feature
	subclassOf	change:Manifestation
	equivalentClass	change:manifestationOf some ParcelPD and change:manifestationOf only ParcelPD
	change:existsAt	exactly 1 time:TemporalEntity
	hasLandUse	Only LandUseClassification
	associatedArea	only om:area
	hasPopulation	only Population
ResidentPopulation	subclassOf	govstat:Population
EmployedPopulation	subclassOf	ResidentPopulation
LBCSClassification	subclassOf	LandUseClassification
ActivityClassification	subclassOf	LBCSClassification
	equivalentClass	lbcs:Activity
FunctionClassification	subclassOf	LBCSClassification
	equivalentClass	lbcs:Function
StructureClassification	subclassOf	LBCSClassification
	equivalentClass	lbcs:Structure
SiteClassification	subclassOf	LBCSClassification
	equivalentClass	lbcs:Site
OwnershipClassification	subclassOf	LBCSClassification
	equivalentClass	lbcs:Ownership
CLUMPClassification	subclassOf	LandUseClassification

	equivalentTo	hasCLUMPCode min 1 xsd:string
AAFCClassification	subclassOf	LandUseClassification
	equivalentTo	hasAAFCCode min 1 xsd:string
Unclassified	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "11"
Settlement	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "21"
Roads	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "25"
Water	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "31"
Forest	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "41"
ForestWetland	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "42"
Trees	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "45"
TreedWetland	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "46"
AAFCCropland	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "51"
GrasslandManaged	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "61"
GrasslandUnmanaged	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "62"
Wetland	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "71"
WetlandShrub	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "73"

WetlandHerb	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "74"
OtherLand	subclassOf	AAFCClassification
	equivalentTo	hasAAFCCode value "91"
UrbanBuiltUp	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "B"
MinesQuarriesSandGravelPits	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "E"
CLUMPCropland	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "A"
CLUMPWater	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "Z"
Horticulture	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "H"
ImprovedPasture	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "P"
NonProductiveWoodland	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "U"
OrchardsVineyards	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "G"
OutdoorRecreation	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "O"
ProductiveWoodland	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "T"
SwampMarshBog	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "M"
UnimprovedPasture	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "K"
Unmapped	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "8"

UnproductiveRock	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "L"
UnproductiveSand	subclassOf	CLUMPClassification
	equivalentTo	hasCLUMPCode value "S"

Reused Ontologies:

- lbcs: Land Based Classification Standards (LBCS) Ontology[15] presented by [30].
- iCity-Foundation

Future Work:

- In future versions of the ontology, it may be desirable to include an optional relationship for Parcel that identifies its associated organization (e.g. municipal / federal government, transit agency, etc.)
- Future work may extend the population representation to capture various sorts of populations (employed, students, etc)

6.12 Trip Ontology

http://ontology.eil.utoronto.ca/icity/Trip.owl

Namespace: trip

- Trip: A Trip is a kind of **Activity** wherein a Person(s) is transported from one location to another **via some** Mode(s). As with activities, trips may have participants; they may also be described with specialization of the **has participant** property: **hasDriver** and/or **hasPassenger**.

  A Trip **starts at** some **Location** and **ends at** some **Location**.

  A Trip **occurs during** some Interval.

  A Trip **occurs in** some Network(s).

  A Trip **occurs via** some Arc(s).

  A Trip **occurs on** some Transportation Complex. (e.g. a road or a rail)

  A Trip contains some Trip Segments.

  A Trip may incur some cost (monetary or otherwise).

[15] Not available online

- A Trip Segment describes part of a trip. It may be used, for example, to identify different parts of the Trip by Mode.
  The restrictions on the Mode and possibly Vehicle used will become more complicated as we begin to incorporate restrictions based on a Persons access to a vehicle (age, household).
  A Trip Segment is a specialization of a Trip that is subactivity of some Trip.
  A Trip Segment **occurs during** some Interval.
  A Trip Segment **occurs in** some Network(s).
  A Trip Segment occurs via some Arc(s).
  A Trip occurs on some Transportation Complex.
  A Trip Segment may incur some cost (monetary or otherwise).
- Tour: A sequence of Trips made by one Person.
  A Tour **starts and ends at** the same Location.

Object	Property	Value
Trip	subclassOf	activity:Activity
	startLoc	only spatial_loc:SpatialFeature
	endLoc	only spatial_loc:SpatialFeature
	~~during~~	~~exactly 1 time:Interval~~
	accessesNetwork	min 1 transportation:Network
	accessesArc	min 1 transportation:Arc
	occursOn	min 1 transportation:TransportationComplex
	viaMode	min 1 transportation:Mode
	viaVehicle	only Vehicle
	hasDriver	only change:Manifestation
	hasPassenger	only change:Manifestation
TripSegment	subclassOf	Trip
	inverse (hasSubactivity)	min 1 Trip
	viaMode	~~min~~ exactly 1 vehicle:Mode

	viaVehicle	only vehicle:Vehicle
Tour	subClassOf	Trip
	startLoc	startLoc only (inverse (endLoc) Self)

Reused Ontologies:

- iCity-TransportationSystem
- iCity-Vehicle

6.12.1 Trip Costs

http://ontology.eil.utoronto.ca/icity/TripCost.owl

Namespace: tripcost

Different costs are associated with the performance of Trips. These may take the form of direct costs such as those presented in the Travel Cost Ontology, but there may be non-monetary costs associated with travel over different arcs such as pollution and travel time. Trip Costs capture these indirect costs that may vary between individual trips; a trip cost is a property of some instance of travelling.

- A Duration Cost is a Trip Cost.
  A duration cost has an associated cost in terms of duration; e.g. the length of time to perform the trip or trip segment
  A duration cost may have an associated monetary cost (valuation); e.g. the monetary cost applied to the length of time taken to perform the trip or trip cost.
- A Distance is a Trip Cost
  A distance has an associated cost in terms of the distance travelled.
  It may also have an associated monetary cost (valuation)
- An Environmental Cost is a Trip Cost
- A Vehicle Cost is a Trip Cost

Object	Property	Value
TripCost	hasMonetaryCost	only om:MonetaryValue

	tripCostOf	only (trip:Tour or trip:Trip or trip:TripSegment)
DurationCost	subclassOf	TripCost
	hasDurationCost	only time:DurationDescription
DistanceCost	subclassOf	TripCost
	hasDistanceCost	only om:length or om:MonetaryValue
EnvironmentalCost	subclassOf	TripCost
	hasEnvironmentalCost	only CarbonEmissions
VehicleCost	subclassOf	TripCost

Reused Ontologies:

- iCity-Trip

6.13 Urban System Ontology

http://ontology.eil.utoronto.ca/icity/UrbanSystem.owl

Namespace: urbansys

Earlier in this report, we recognized that the urban system covers many different concepts, thus motivating the design of the preceding, so-called generic ontologies. However, it must be recognized that in isolation, these concepts do not effectively capture the urban system. The urban system not only includes these concepts, but relationships between them. The Urban System Ontology extends all of the previously defined ontologies in order to capture the relationships between them, in the context of the urban system.

- A Person may be a **member of** a Family and/or a Household.
  A Person may work for another Person, or some Organization, or be enrolled at some Educational Institution.
  A Person may **have access to** some **Vehicle**.
  A Person may **have access to** some **Bicycle**.
  A Person has a **Schedule** for a given point (period) in time.
- A Schedule is a plan for some Activity to occur at/over some point in time.
- A Family **has members** who are Persons, and who are related via the has-spouse or has-child properties.

- A Household has one or more Persons as **members**. We do not make any commitment regarding the identity of the Persons, and in fact a Person may belong to more than one Household.
- A Dwelling Unit is **located in some Building** (e.g. House, Apartment,...)
- An Organization must have at least 2 Person(s) as members(s).
- A Firm or a Business Establishment may have a Person as an employee
- An Employee is a type of Person(s).
- Occupation: An Occupation **is performed by** some Person.
  An Occupation has a type (e.g. sales, skilled trades)
- A Building may be located on some Parcel of land (this is an invariant property of any building).
  A Building **has an owner**, which may be a Persons or some Organization.
  A Building **has occupants**, which may or may not be the same Persons or Firm who own it.
  A Building may **provide** some **Parking**.
- A Building Unit may be **occupied by** some Persons or Organization.
  A Building Unit may be **provide some Parking**.
- A Vehicle may be **occupied by** at least one Person, and some cargo.
  A Vehicle is **owned by** some Person(s) or Firm.
  A Vehicle has some associated **Mode**.
- Occupant: An occupant is a Person who is occupying a Vehicle during transit.
  An Occupant may be a Driver or a Passenger
- Cargo: A Cargo is some Thing that is not a Person and is occupying a Vehicle during transit.
- An *entire* Arc is accessible by a single set of Mode(s).
- A Road Segment is accessed by some Arc(s) with modes that are not water, air, or rail.
- A Parking Area has some **owner**.
  A Parking Area may be **occupied by** some Vehicle (however, it might also be occupied by some debris or activities such as construction).
- A Parking Policy may **apply to** a specific group of Persons or Organizations.
  A Parking Policy may have a **vehicle type restriction.**

- A TransitSystem may be **owned by** some Organization.
- A Route is **executed by** various Vehicles at different points in time.
- A Trip is **made by** a Person to **facilitate participation in some Activity**.

Object	Property	Value
person:Person	memberOf	min 1 household:Family
	memberOf	min 0 household:Household
	schema:worksFor	some (person:Person or org:Organization)
	hasAccess	some (vehicle:Vehicle or Bicycle)
	hasSchedule	some Schedule
Schedule	hasActivity	only activity:Activity
	scheduledFor	exactly 1 time:Interval
household:Family	hasMember	only person:Person
household:Household	hasMember	min 1 (household:Family or person:Person)
household:DwellingUnitPD	locatedIn	some building:Building
org:Organization	org:hasOrgMember	min 2 person:Person
org:Firm	hasEmployee	only person:Person
org:BusinessEstablishment	hasEmployee	only person:Person
org:Employee	equivalentClass	person:Person and employedBy some (tove:Organization or person:Person)
Occupation	performedBy	some person:Person
	hasOccupationType	only OccupationType
building:BuildingPD	locatedOn	only landuse:Parcel
building:Building	hasOwner	min 1 (person:Person or org:Organization)
	hasOccupant	some person:Person or org:Organization or org:BusinessEstablishment

	hasParking	only parking:ParkingArea
vehicle:Vehicle	occupiedBy	only (Occupant or Cargo)
	hasOwner	only (person:Person or org:Organization)
	hasMode	only transportation:Mode
Occupant	equivalentClass	person:Person and occupies some vehicle:Vehicle
Cargo	equivalentClass	not(person:Person) and occupies some vehicle:Vehicle
transit:TransitSystem	hasOwner	only org:Organization
transit:Route	executedBy	only vehicle:Vehicle
trip:Trip	subClassOf	activity:Activity
	performedBy	some person:Person
	associatedWith	only activity:Activity

Future work:

- May be useful to add a generalized ‘hasPass’ relationship to capture various possible passes a person may have (transit and otherwise)
- Many extensions of this ontology are possible. Additional axioms and properties may be added in future work as new use cases are identified.

7 Extra-logical Design Practices

Here, we summarize and explain the design practices that were adopted in the creation of the ontologies. These practices do not pertain to the semantic definitions, but rather are adopted to address pragmatic concerns regarding the organization and maintenance of the ontologies.

- Organizational terms for reuse (full import) of ontologies: e.g. owl-time. In order to create the required groupings under organizational subclasses, it is easiest to merge the imported ontology into the iCity container (e.g. icity/Time/). This allows for the addition of organizational subclass assertions (e.g. TemporalEntity subclassOf TimeOntologyThing) and also ensures that the appropriate version is captured/reused as a snapshot. This prevents any issues should versioning IRIs not be used by the ontology’s author.

- IRI reference instead of import for large vocabularies (e.g. schema.org)
- View on expressive limitations: if semantics cannot be precisely captured in OWL, it should at least be represented in the OWL ontology, defined in natural language, and ideally in the future they may be captured in some more expressive extension.

8 Evaluation

Throughout development, the iCity ontologies were presented to the iCity-ORF researchers and other stakeholders for review and feedback. These activities served as a kind of informal evaluation that helped to inform, improve, and validate the design of the ontology. In addition, the ontology has been formally evaluated against the requirements described in Section 4. In this section, we review the results of the ontology evaluation with respect to consistency and competency.

8.1 Consistency

A fundamental requirement for any ontology is consistency. If the axioms in the ontology are inconsistent, then the classes are unsatisfiable and any data that is mapped into the ontology will be inconsistent. This inhibits the application of the ontology for data verification. In addition, any sentence may be deduced from an inconsistent set of axioms, so this is also problematic for any reasoning applications. From a basic ontology design perspective, if the axioms are inconsistent then there is something wrong with the way the domain has been formalized; the ontology contains some set of statements that in some way contradict each other. Similarly, it is important to check for (and avoid) any unsatisfiable classes. In a consistent ontology it is still possible that select classes may not be satisfiable. In such cases it is impossible to instantiate the class with any data and maintain consistency. The ontology has been evaluated for both consistency and (absence of) unsatisfiable classes using the Pellet OWL reasoner. One unsatisfiable class was identified, however the class (time:January[16]) is not reused in the extensions and is deprecated in the version of the W3C Time Ontology that is imported by the iCity ontologies.

[16] http://www.w3.org/2006/time#January

8.2 **Competency**

The Requirements stage of ontology development resulted in the identification of five motivating scenarios and ## associated competency questions. Competency questions provide guidance for ontology design, as well as a clear set of criteria against which the ontology may be evaluated. The evaluation focuses on determining whether the ontology is sufficient to formalize the identified competency questions. It is straightforward to demonstrate that the requirements are satisfied by formalizing each of the competency questions using the ontology. Since the ontology has been formalized in OWL 2, the usual mechanism of accessing the data it encodes with be with the SPARQL query language[17]. Therefore, the ontology has been evaluated with the use SPARQL to formalize each of the identified competency questions. Implicit in each formalism is a mapping between the natural language used in the requirement and the terms defined in the ontology. This mapping will be made explicit in the application of the ontology, which addresses the mapping models required to encode information in datasets as instances in the ontology.

In the following, we demonstrate the results of evaluation by formalizing each of the identified competency questions in SPARQL. The following namespaces will be used in addition to the namespaces defined in the previous section:

Commented [MK4]: Review & confirm namespaces defined for ontologies

- rdf:
- rdfs:
- owl:

```
• PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
•   PREFIX time: <http://www.w3.org/2006/time#>
•   PREFIX transit: <http://ontology.eil.utoronto.ca/icity/PublicTransit/>
•   PREFIX change: <http://ontology.eil.utoronto.ca/icity/Change/>
•   PREFIX spatial: <http://ontology.eil.utoronto.ca/icity/SpatialLoc/>
•   PREFIX trip: <http://ontology.eil.utoronto.ca/icity/Trip/>
•   PREFIX re: <http://ontology.eil.utoronto.ca/icity/RecurringEvent/>
•   PREFIX geo: <http://www.opengis.net/ont/geosparql#>
•   PREFIX icontact: <http://ontology.eil.utoronto.ca/icontact.owl#>
•   PREFIX bif: <http://www.openlinksw.com/schemas/bif#>
```

-

[17] https://www.w3.org/TR/sparql11-overview/

Many of the competency questions pertain to some given individual of interest (e.g. a particular household or traffic zone). We capture such cases with a placeholder denoted in curly brackets (e.g. {household-1}) to illustrate where the individual or individuals of interest would be substituted.

On the role of GeoSPARQL Functions: In practice, the spatial relationships between objects may not be encoded in the knowledge base. In such cases, GeoSPARQL functions may be employed in the query to compute these relationships using the coordinate data defined for the geo:Geometry objects. As the implementation of these functions is subject to some variation between triple stores, for the purposes of evaluation we design the queries under the assumption that the spatial relationships between geo:Feature objects are given. This allows us to maintain a consistent, triple store-independent formalization.

8.2.1 CQs for Land Use and Transportation Simulation

CQ1-1: What trips originated/ended[18] in a given zone?

```
SELECT ?trip WHERE {
?trip rdf:type trip:Trip.
?trip trip:startLoc ?sloc.
{zone} a landuse:TrafficZone.
loc:hasLocation ?zloc.
?zloc geo:contains ?sloc.
}
```

CQ1-2: What is the occupation breakdown of the travelers whose trips originated/ended in a given zone?

```
SELECT ?occupation (COUNT ?trip as ?trips) WHERE {
?trip rdf:type trip:Trip.
?trip trip:startLoc ?sloc.
{zone} a landuse:TrafficZone.
```

[18] This and subsequent queries may be easily repurposed to retrieve trips with a particular end zone by replacing trip:startLoc with trip:endLoc.

```
loc:hasLocation ?zloc.
?zloc geo:contains ?sloc.
?trip urban:tripPerformedBy ?p.
?o org:performedBy ?p.
?o org:hasOccupationType ?occupation.
} GROUP BY ?occupation
```

CQ1-3: What were the purposes of the trips that originated/ended in a given zone?

```
SELECT ?trip ?activitytype WHERE {
?trip rdf:type trip:Trip.
?trip trip:startLoc ?sloc.
{zone} a landuse:TrafficZone.
loc:hasLocation ?zloc.
?zloc geo:contains ?sloc.
?trip trip:associatedwith ?activity.
?activity rdf:type ?activitytype.
}
```

CQ1-4: In a particular time period, how many trips originated/ended in a given zone?

```
SELECT (COUNT ?trip as ?trips) WHERE {
?trip rdf:type trip:Trip.
?trip trip:startLoc ?sloc.
{zone} a landuse:TrafficZone.
loc:hasLocation ?zloc.
?zloc geo:contains ?sloc.
} GROUP BY ?zloc
```

CQ1-5: What were the transportation mode(s) taken by trips that originated/ended in a given zone?

```
SELECT DISTINCT ?mode WHERE {
?trip rdf:type trip:Trip.
?trip trip:startLoc ?sloc.
```

```
{zone} a landuse:TrafficZone.
loc:hasLocation ?zloc.
?zloc geo:contains ?sloc.
?trip trip:viaMode ?mode.
}
```

CQ1-6: Who are the members of a particular household?

The following query returns all persons who are or have been members of a household, the change:existsAt property would need to be used to constrain the results to household members at a particular point in time.

```
SELECT ?person WHERE {
{household} rdf:type household:HouseholdPD.
{household} change:hasManifestation ?hhld.
?hhld household:hasHouseholdMember ?person_at_t.
?person_at_t change:manifestationOf ?person.
}
```

CQ1-7: What trips were performed, by which members of a particular household?

```
SELECT ?person ?trip WHERE {
{household} rdf:type household:HouseholdPD.
{household} change:hasManifestation ?hhld.
?hhld household:hasHouseholdMember ?person_at_t.
?person_at_t change:manifestationOf ?person.
?trip urbansys:tripPerformedBy ?person_at_t.
}
```

CQ1-8: What were the purposes of the trips performed by members of a particular household?

```
SELECT ?person ?trip ?activity WHERE {
{household} rdf:type household:HouseholdPD.
{household} change:hasManifestation ?hhld.
?hhld household:hasHouseholdMember ?person_at_t.
```

```
?person_at_t change:manifestationOf ?person.
?trip urbansys:tripPerformedBy ?person_at_t.
?trip urbansys:associatedWith ?occ.
?occ rdf:type ?activity.
}
```

CQ1-9: What is the age, sex, and occupation of the traveler who performed a particular trip?

```
SELECT ?age ?sex ?occ WHERE {
{trip} urbansys:tripPerformedBy ?person_at_t.
?person_at_t change:manifestationOf ?person.
?person_at_t person:hasAge ?d.
?d om:hasValue ?m.
?m om:has_numerical_value ?age.
?occ org:performedBy ?person_at_t.
?person person:hasSex ?sex.
}
```

CQ1-10: What land use classification is associated with a particular parcel?

```
SELECT ?class WHERE {
{parcel} change:hasManifestation ?parcel_at_t.
?parcel_at_t landuse:hasLandUse ?landuse.
?landuse rdf:type ?class.
}
```

8.2.2 CQs for Transit Research

CQ2-1: What date and time has a subway incident occurred?

```
SELECT ?datetime WHERE{
{incident} rdf:type transit:TransitIncident.
{incident} activity:beginOf ?t.
?t time:inXSDDateTimeStamp ?datetime.
}
```

CQ2-2: What are the locations of vehicles on a particular route after the occurrence of a subway incident?

```
SELECT ?x ?t ?td1
WHERE {
  ?x a transit:TransitVehicle.
  ?x transit:onRoute ?route.
  ?x change:existsAt ?t.
  ?t time:inside ?t1.
  ?t1 time:inXSDDateTimeStamp ?td1.
  ?x spatial:hasLocation ?f.
  ?f geo:hasGeometry ?g.
  ?g geo:asWKT ?gwkt.

   {incident} rdf:type transit:TransitIncident.
   {incident} activity:beginOf ?t_incident.
   ?t_incident time:inXSDDateTimeStamp ?dt_i.

  FILTER(?route = {route})
  FILTER(?td1 > dt_i)
}
```

CQ2-3: Are any buses located more than a certain distance from their assigned route at a given point in time?[19]

```
SELECT ?x ?g ?route_pd ?g_trip ?d
WHERE
{
  ?x a transit:TransitVehicle.
  ?x transit:onRoute ?route.
```

[19] Note: the precise formalism of this query will vary depending on the triple store and how it has implemented the required GeoSPARQL functions. If no GeoSPARQL or other similar spatial functions have been implemented, then this query may not be successfully answered. In some cases it may be possible that the spatial relations of interest are pre-computed and populated in the triple store.

```
  ?x spatial:hasLocation ?f.
  ?f geo:hasGeometry ?g.
  ?g geo:asWKT ?g_wkt.

  ?route icontact:hasOperatingHours ?ho.
  ?ho re:hasSubRecurringEvent ?trip.
  ?trip spatial:hasLocation ?f_trip.
  ?f_trip geo:hasGeometry ?g_trip.
  ?g_trip geo:asWKT ?g_trip_wkt.

  FILTER(bif:st_distance(?g_wkt,?g_trip_wkt) <= {distance})
}
```

8.2.3 CQs for Smart Parking Applications

Note that parking information is one example of a scenario where the majority of the data of interest is subject to change, (currently) at a low frequency. In such cases, rather than formulate queries for specific points in time, it is sufficient to organize the results for time-variant properties by their associated timepoint or interval. In the future as more real-time data becomes available, the nature of this may change and there will be more queries oriented toward data associated with specific timepoints (i.e. "now").

CQ3-1 What is the address of the parking lot P?

The following query returns the street number and name for a particular parking lot. Other attributes of address exist and may be referenced as required. As it is possible for the address to change over time, the query returns all distinct values for the lot's address.

```
SELECT DISTINCT ?num ?street MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingFacility;
change:existsAt ?t
icontact:hasAddress ?a.
?a icontact:hasStreetNumber ?num.
```

```
?a icontact:hasStreet ?street.
}
```

CQ3-2 What is the capacity of parking lot P?

The capacity of a parking lot may change over time (e.g. as a result of layout changes), thus this query returns all distinct capacities of the parking lot.

```
SELECT DISTINCT ?capacity MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingArea;
change:existsAt ?t
parking:hasVehicleCapacity ?c.
?c om:has_value ?c_measure.
?c_measure om:has_numerical_value ?capacity.
}
```

CQ3-3 Is it accessible by disabled people, and if so how many parking spots are for disabled vehicles?

The allocation of accessible parking spaces may change over time thus a temporal dimension is also included in the query. The result will return the number of accessible parking spaces (if any) on record for a parking lot, including changes made to this figure over time.

```
SELECT ?t (COUNT(?p AS ?accessible_spot)) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingArea;
change:existsAt ?t
parking:hasSubParkingArea ?p.
?p rdf:type parking:AccessibilityParkingSpace.
} GROUP BY ?t
```

CQ3-4 Is there a height limit for vehicles for a parking lot P?

```
SELECT ?hlimit WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
parking:maxAdmittableHeight ?hquantity.
?hquantity om:has_value ?hmeasure.
?hmeasure om:has_numerical_value ?hlimit.
}
```

CQ3-5 What are the geographic coordinates for parking lot P?

In this query we return the geocoordinates associated with the parking lot's address. An alternative approach might query for the associated spatial feature (i.e. the region occupied in space) instead.

```
SELECT DISTINCT ?coord MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingFacility;
change:existsAt ?t
icontact:hasAddress ?a.
?a icontact:hasGeoCoordinates ?coord.
}
```

CQ3-6 What building is a particular parking lot located in?

```
SELECT ?building WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
parking:parkingPartOfBuilding ?building.
}
```

CQ3-7 Is a particular parking lot open to the public at a given time?

```
ASK {
SELECT DISTINCT ?coord MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingFacility;
```

```
icontact:hasOperatingHours ?hours.
?hours recurring:startTime ?open;
recurring:endTime ?close.
FILTER({time} >= ?open && {time} <= ?close)
}
```

CQ3-8 How much does it cost to park in a particular parking lot?

In this query we return the cost and the duration at which it is applied; for example, **5** dollars per **1** hour. It is also possible to retrieve more detail on the cost such as the currency.

```
SELECT DISTINCT ?cost ?perhour MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot change:existsAt ?t;
parking:hasParkingPolicy ?policy.
?policy parking:hasParkingRate ?rate.
?rate parking:hasMonetaryCost ?mval.
?mval om:has_value ?mmeasure.
?mmeasure om:has_numerical_value ?cost.
?rate parking:forDuration ?d.
?d time:hours ?perhour
}
```

CQ3-9 What types of payment are accepted at a particular parking lot?

```
SELECT DISTINCT ?paymethod MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot change:existsAt ?t;
parking:hasParkingPolicy ?policy.
?policy parking:hasPaymentMethod ?paymenthod.
}
```

CQ3-10 How many parking spots are designated for electric vehicles in a particular parking lot?

```
SELECT ?t (COUNT(?p AS ?ev_spot)) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot rdf:type parking:ParkingArea;
change:existsAt ?t
parking:hasSubParkingArea ?p.
?p rdf:type parking:EVParkingSpace.
} GROUP BY ?t
```

CQ3-11 What types of electric vehicle chargers are available in a particular parking lot?

```
SELECT DISTINCT ?chargetype MAX(?t) WHERE {
{lotpd} rdf:type parking:ParkingAreaPD;
change:hasManifestation ?lot.
?lot change:existsAt ?t;
parking:hasEvCharger ?charge.
?charge rdf:type ?chargetype.
}
```

8.2.4 CQs for ATIS via IT-SoS

CQ4-1: What are the averages of the TTI_Max values that have been observed over some period of time?

```
SELECT  ?x ?wayID
WHERE {
?y a transport:MeanTTI_Max.
?y om:hasValue ?measure.
?measure om:numerical_value ?x.
?y om:aggregateOver ?t_interval.
?t_interval time:hasBeginning ?t1.
?t1 time:inXSDDateTime ?dt1.
?t_interval time:hasEnd ?t2.
?t2 time:inXSDDateTime ?dt2.
```

```
   ?y om:aggregateOf ?y_2.
   ?y_2 om:aggregateOver ?wayID.
   FILTER(?dt <= {time}^^xsd:dateTime && ?dt2 >=
      {time}^^xsd:dateTime)
   }
```

CQ4-2: What are the averages of the TTI_Max values that have been observed at some location?

```
SELECT DISTINCT ?x ?t_interval ?dt1 ?dt2
WHERE {
?y a transport:MeanTTI_Max.
?y om:hasValue ?measure.
?measure om:numerical_value ?x.
?y om:aggregateOver ?t_interval.
?t_interval time:hasBeginning ?t1.
?t1 time:inXSDDateTime ?dt1.
?t_interval time:hasEnd ?t2.
?t2 time:inXSDDateTime ?dt2.
?y om:aggregateOf ?y_2.
?y_2 om:aggregateOver {location id}.
}
```

CQ4-3: What are the averages of the TTI_Max values that have been observed at some location, over some period of time?

```
SELECT DISTINCT ?x ?dt1 ?dt2
WHERE {
?y a transport:MeanTTI_Max.
?y om:aggregateOf ?y_2.
?y_2 om:aggregateOver {location id}.
?y om:hasValue ?measure.
?measure om:numerical_value ?x.
?y om:aggregateOver ?t_interval.
?t_interval time:hasBeginning ?t1.
```

```
?t1 time:inXSDDateTime ?dt1.
?t_interval time:hasEnd ?t2.
?t2 time:inXSDDateTime ?dt2.
FILTER(?dt1 <= {time}^^xsd:dateTime && ?dt2 >=
  {time}^^xsd:dateTime)
}
```

8.2.5 CQs for ArcGIS Query Support

CQ5-1: What neighbourhood(s) does a particular route go through?

CQ5-2: What types of land use does a particular route go through?

CQ5-3: What types of land cover does a particular route go through?

CQ5-4: What points of interest does a particular route pass by?

CQ5-5: What types of road does a particular route travel on?

CQ5-6: What (if any) parts of a route travel on a road segment that is above grade?

CQ5-7: What (if any) parts of a route travel on a road segment that is below grade?

9 Application

Application of the ontology serves to ground its evaluation by demonstrating how it may be used in practice -- in particular, how its capacity to represent the domain may be used to address some motivating scenarios. Applying the ontology for a demonstrates how the ontology can be used to represent the data of interest and produce answers for the competency questions. It also provides insight into the required architectures for ontology-based solutions to the motivating scenarios.

Applications of the iCity ontology were explored as case studies derived from the motivating scenarios identified during the iCity-ORF project. These case studies represent a small subset of possible applications of ontologies for urban informatics. Beyond serving as concrete examples for how the ontology may be used, these projects serve to demonstrate the sufficiency of the ontology to integrate, capture, and retrieve the data of interest.

In the sections below, we provide an overview of each of the case study applications of the iCity Ontology. Mappings from the data sources into the ontology are described in detail in the appendices, and the R2RML files are available in the project's GitHub repository.

9.1 Exploration of Travel Model Data

Based on the motivating scenario described in Section 5.1, one possible application for the ontology would be to support the exploration of simulation results. In this case study, we focus on data generated by the TASHA travel model. We leverage the ontology as a means of understanding and exploring its output.

Rather than simply provide access to a SPARQL endpoint to evaluate the CQs of interest, in this application a data access tool, the Linked Data Reactor[20] (LD-R), was implemented as an additional layer to support easy exploration of the model output. The resulting architecture is depicted in Figure 18. The LD-R tool provides a layer between the user and the triple store; rather than design and implement queries directly, a user is able to explore the data by browsing through a pre-designed selection of "facets". A screenshot of the implementation in Figure 17 illustrates some of the configured facets and the display that results from interacting with them. The facets defined for this application are described in more detail below, and the configuration files are available in GitHub at:
https://github.com/EnterpriseIntegrationLab/icity/tree/master/applications/TASHA/configs .

[20] http://ld-r.org

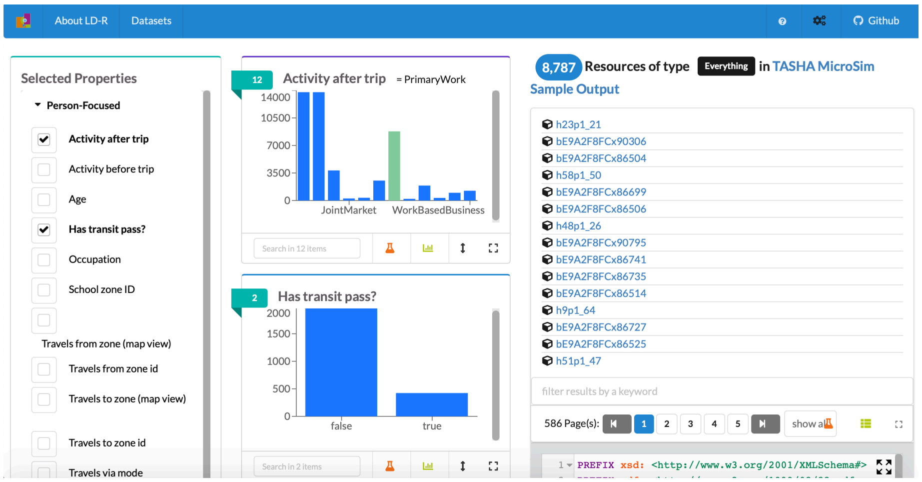


Figure 17: Screenshot of the LD-R interface implemented to explore the TASHA output data.

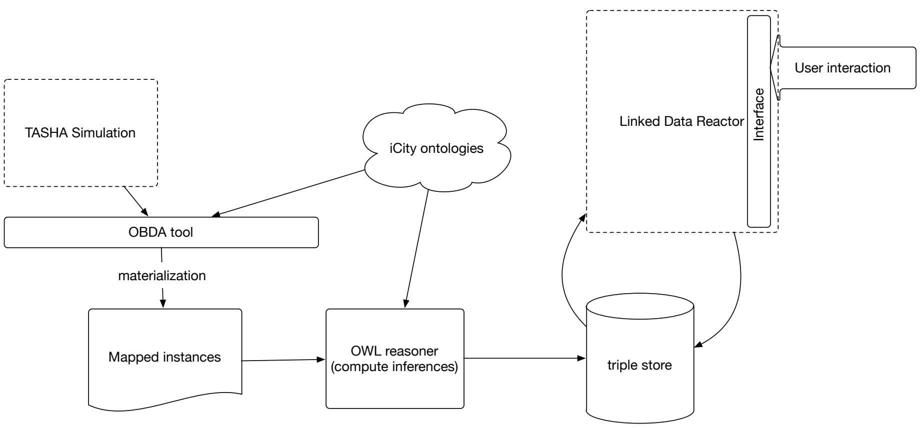


Figure 18: Architecture with LD-R supported data access.

9.1.1 Summary of Facets

Facets of the data are displayed based on the properties of the objects in the triple store. Selecting the values displayed for a particular facet enables the user to constrain the results displayed in order to further explore other facets of the data.

For example, if a user selects the “Start zone ID” property then the LD-R will display a facet showing the distribution of trips starting at different zones. The user may select additional properties, *or* they can select one or more zone ids to narrow the scope of the analysis. If Zone X is selected, then any other facets that are displayed will only display properties for trips that started at Zone X. This allows the user to explore answers to questions such as: “what is the distribution of modes for trips going from zone X to zone Y”? or “What is the most common activity at origin for trips that are taken with transit?”

In order to support the sort of analysis required, it is useful to group the properties into the particular object types for which they apply; in this case, we have created “Trip-focused” and “Person-focused” categories.

Trip-focused:

- Start zone map, End zone map: displays the start and end zone of trips on a map. Start and end zones may be selected to narrow the scope of trips of interest.
- Start zone ID, End zone ID: display the zone IDs of the trips’ origins and destinations.
- Start time, End time: displays the start and end times of a trip, encoded in xsd:dateTimeStamp format. The times are assigned a default date of 01-01-2019.
- Trip Mode: displays the mode type used for the trip.
- Activity at origin, Activity at destination: displays the activity types performed at start and end of the trip (i.e. the activities directly preceding and following the trip).
- Traveler Type: displays the type (class) of person who performed the trip.
- Traveler Age: displays the age of the person who performed the trip.
- Traveler has transit pass: a boolean value indicating whether the person who performed the trip has a transit pass.
- Traveler’s school zone: displays the zone id of the school where the person performing the trip is enrolled (if applicable).
- Traveler’s work zone: displays the zone id of the location where the person performing the trip is employed (if applicable).
- Traveler’s occupation: displays the class of occupation of the traveler (if employed).

Person-focused:

- Age: displays the age range of persons in the TASHA output.

- Transit pass: displays a Boolean value indicating the number of persons with (true) and without (false) transit passes.
- School zone ID: displays the distribution of the zone ids of the schools where people are enrolled.
- Work zone ID: displays the distribution of zone id of the locations where people are employed.
- Occupation: displays the distribution of occupation types for people in the simulation.
- Travels from zone, Travels to zone (map view): displays a map view of the zones where the selected persons travel from and to.
- Travels from zone ID, Travels to zone ID: displays the distribution of zone IDs of the locations where the selected persons travel to and from.
- Trip start times, Trip end times: displays the distribution of trip start times and end times for trips performed by the selected persons.
- Travels via mode: displays the mode of travel for trips performed by the selected persons

9.1.2 Data Mappings

In addition to providing a mechanism to query the simulation output, defining mappings to the ontology serves to formalize the data such that its semantics is clear. This provides a level of documentation not previously available for TASHA output.

The mappings that were designed to define the simulation output data in terms of the iCity Ontology are described in detail in Appendix A.The Karma mapping files are available online in the GitHub project repository at: https://github.com/EnterpriseIntegrationLab/icity/tree/master/mappings/TASHA. The mappings were formalized in the W3C standard R2RML [31] and designed and implemented using the Karma [32] data transformation tool. The data was transformed into RDF (i.e., through OBDA *materialization*) and then uploaded to a Virtuoso triple store.

9.1.3 Future Work

LD-R provides the ability to explore the results of a particular facet by enabling data pivots.[21] While this is a potentially useful tool, initial performance was rather poor so we have not included this capability at this time. It may be a tool to consider at a later date should the tool be upgraded beyond the EC2 t2 micro instance. Similarly, the "restrictAnalysisToSelected" tag may be useful in filtering results, however documentation notes that this option may result in slowed performance so we have opted to exclude this for the time being. These features may be explored in the future, taking performance requirements into account.

LD-R documentation also notes a timeline view as a desired future enhancement. In the meantime, it might be useful to consider manipulating the data in order to view the associated timestamps at a higher level of granularity (e.g. hourly).

Finally, as a follow-up to this work it would be interesting to consider the capture and integration of results from other travel model simulations and tools. This would provide useful insights into the potential value of semantic integration in the context of simulation results.

9.2 Analysis of TTC Data for Bus Bridging Study

This case study was derived from the motivating scenario described in Section 5.2. Similar to the previous case, the main goal of this application is to support researchers in navigating and exploring data of interest. In this case, the relevant datasets are those provided by the local transit authority. No specialized architecture was designed; rather, standard Semantic Web tools of an RDF triple store and a data mapper were implemented in order to formalize, integrate, and provide a mechanism to access this data.

9.2.1 Data mapping

The transit research CQs were motivated by a work on bus bridging that was being conducted as part of the iCity-ORF Project 2.3. The CQs required data from several sources: (1) the gtfs specification of vehicle routes, (2) reports on subway incidents, and (3) data on the real-time locations of transit vehicles. The mappings that were designed to define this data, in terms of the iCity Ontology are described in detail in Appendix B. The Karma mapping files are available online in the GitHub project repository at:

---

[21] http://ld-r.org/docs/configFacets.html

https://github.com/EnterpriseIntegrationLab/icity/upload/master/mappings/TTC. The mappings were formalized in the W3C standard R2RML and designed and implemented using the Karma data transformation tool. The data was transformed into RDF (i.e., through OBDA *materialization*) and then uploaded to the Virtuoso[22] triple store.

9.2.2 Queries

The result of the data mapping process was an RDF triple store, containing all of the data of interest, formalized in the language of the iCity Ontology. This triple store provides a point of access (SPARQL endpoint) for the queries – including, but not limited to, those identified by the motivating scenario – to be put forward and answered.

A note on the use of GeoSPARQL functions: The CQs identified for this application involve spatial relationships, defined by GeoSPARQL, between various spatial regions. Two approaches are possible to obtain the desired result: (1) the spatial relationships might be pre-computed by some external service given the specified geometries, and transformed into RDF and uploaded along with the other data to the triple store; or (2) the spatial relationships might be determined as part of the SPARQL query, by employing the GeoSPARQL functions supported by the triple store. The latter approach was adopted in the design of the CQs used for this application, however it is important to note that this approach will be highly dependent on the triple store used. Currently, triple stores provide varying degrees of support for GeoSPARQL functions. Those that do provide support employ their own specialized vocabulary to call the GeoSPARQL functions in a SPARQL query. The application and CQs described in this report are specific to the Virtuoso triple store, and would require revision for implementation with other triple stores.

9.2.3 Future Work

This use case demonstrated a very basic application of the ontology. There are many opportunities for future work to improve its functionality to support the motivating scenario. In particular, the architecture could be developed to streamline the data mapping process, for example to automate the addition of data to the triple store when new/updated data becomes available. Further, usability should be considered. As was explored in the previous application, an interface to support both access to and presentation of the data should be considered in the

[22] https://virtuoso.openlinksw.com

future. A simple next step might be the creation of query templates to avoid the need for transportation researchers to interact directly with the SPARQL endpoint.

9.3 Ontology for ATIS in the IT-SoS Architecture

The purpose of the architecture is to support the creation of tools capable of dynamic data discovery, information management, and interoperability between data sources and services. The architecture focuses on three areas: the storage of the data in a data lake, the semantic representation of the data, and the services layer. As described in Section 5.4, the iCity Ontology plays a key role in the IT-SoS architecture, enabling an integrated representation of domain knowledge and supporting semantic interoperability through different tools and across systems. In the iCity project, this architecture was demonstrated with an example implementation of an Advanced Traveler Information System (ATIS) application that incorporates data from loop detectors. The objective of this application is to integrate real-time data with the Online Trip Planner (OTP) tool. Researchers from the project completed two case study implementations of the ATIS application. The first utilized a database to capture processed loop detector data sets, while the second designed the application according to the IT-SoS architecture and thus leveraged the ontology to process the data. The purpose of this work was to showcase the differences between a "status quo" application and one developed according to the IT-SoS architecture. This work is presented in [ref CTRF paper] and relevant code is available on a Github repository here: <url>. Here, we focus on the use of the iCity ontology in the context of the IT-SoS Architecture.

9.3.1 Project 1.2: IT-SoS Architecture

The IT-SoS architecture proposed in iCity Project 1.2 is combined of the following major components, as illustrated in Figure 19.

- **Infrastructure**: a multi-cloud strategy has been adopted to host the above layers (data, Servicers and Applications). The right selection of the appropriate cloud resources and hosting is based on the data and services requirements. The cloud infrastructure provides dynamic recourse allocation and better cost management.
- **Data Lake**: provides a storage repository that host a vast amount of data (structured and unstructured).
- **Ontology Engine:** provides access to a semantic representation for the data.

- **Services Layer:** provides a platform to develop the services. The services consume the data which is provided by the Data Lake layer or through integration with the Ontology Engine.
- **Application Layer:** uses one or more services to create a specific application, e.g., the Advanced Traveler information system (ATIS).

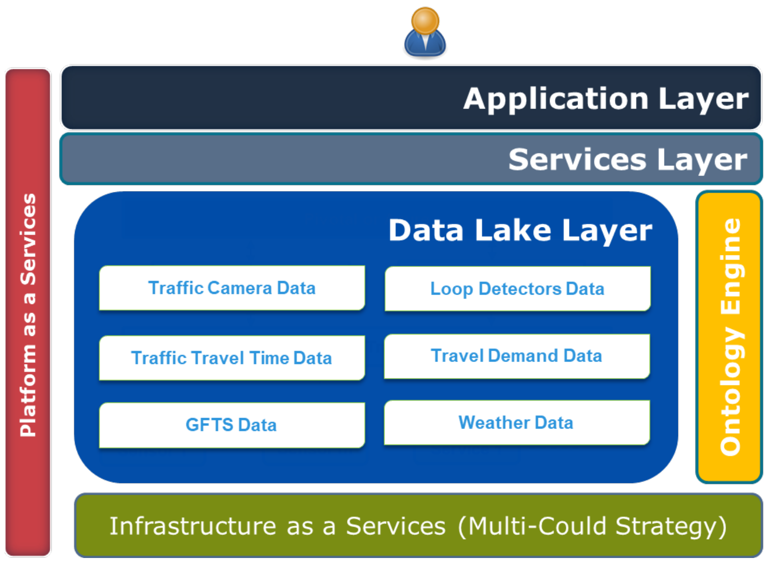


Figure 19: IT-SoS Architecture

The iCity Ontology Engine supports the IT-SoS by providing access to semantically integrated data. In order for the ontology engine function, the data sources must somehow be interpretable in the language of the ontology(s). This requires some form of semantic annotation or mapping of the data in its original form; ontology will be used to explicitly describe the semantics of each entry in the data sources. We propose the adoption of R2RML (RDB to RDF Mapping Language) to specify these mappings. R2RML is a W3C Recommendation that has been developed specifically for this purpose.

Using the appropriate ontologies, these mappings will be defined for each type of data source. The inclusion of new data sources will require the definition of new R2RML mapping definitions, but will not impact any of the existing data sources or mappings.

The mappings and the data serve as input to a tool which converts the data into information represented using the terminology of the ontology. This data is formatted according to Semantic Web Standards (i.e. it is serialized in RDF) such that it may then be loaded into a knowledge graph, (i.e., a triple store).

The resulting data store houses all of the data. As a result of the mapping process, this data is now semantically annotated and integrated. In other words, the relationships between the various data stores are now explicit according to the concept definitions in the ontology. This data may be accessed via SPARQL queries: these are queries that are specified using the terminology defined by the ontology. In particular, the data store provides APIs that may be called by a variety of applications to access the data of interest via these queries. This perspective of the architecture is illustrated in Figure 20.

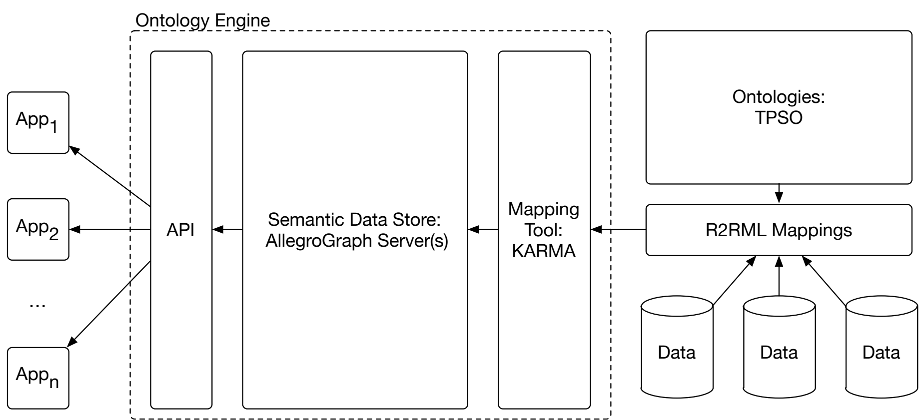


Figure 20: Ontology Engine - interface between Project 1.1 and Project 1.2

An alternative architecture is possible in which the data is maintained solely in its original databases and is retrieved on-demand via Ontology-Based Data Access (OBDA) tools. This approach employs the same R2RML mappings, the main difference being that the data is not stored centrally, but assumed to be distributed in pre-existing relational databases. There are additional challenges related to this approach, thus it is a consideration for future work.

9.3.2 ATIS Application

The transformed data is stored an implementation of the AllegroGraph semantic graph database running on a remove server. There exist a number of other data stores that might have been used and would provide similar API functionality. AllegroGraph was chosen due to its popularity and its unique implementation of a quintuple representation that includes an identifier for each statement in the data store, as opposed to typical approaches which only provide identifiers at the dataset level. This functionality is required for future extensions to this work that address issues such as provenance and confidence in the facts in the data store.

In the context of the ATIS application, the mapped data, stored in AllegroGraph, shall serve as input for the trip planning tool. Based on the road segments used in a trip from one location to another, the ATIS will use the AllegroGraph API to query the ontology to retrieve TTI data for the route. This data can then be used as input to the OTP in order to advise the user of potential delays.

9.3.3 Data Mapping

The application of the ontology engine architecture for the semantic augmentation of sensor data was straightforward. Some minor cleaning of the datasets was required: primarily this involved some reformatting of data values in order to comply with standard datatypes (e.g. for date-time encodings). This cleaning was done using additional functionality provided by the mapping tool, but could also have been accomplished with some other preprocessing mechanism.

The KARMA[23] [32] tool was selected to support the transformation of the original sensor datasets into a semantic representation. This choice was motivated by several factors including: ease of use – the tool is straightforward to use and includes a GUI to support the R2RML specification process; range of acceptable data formats – the tool supports the transformation of not only data in relational databases, but also data in .csv and .json formats, among others; batch transformation – the tool easily enables the transformation of batches of files given the R2RML mappings and thus should easily scale to larger use cases.

Commented [MK5]: Move this discussion elsewhere?

The mappings that were designed to define this data, in terms of the iCity Ontology are described in detail in Appendix C. The Karma mapping files are available online in the GitHub project repository at: https://github.com/EnterpriseIntegrationLab/icity/upload/master/mappings/ITSoS.

[23] http://usc-isi-i2.github.io/karma/

9.3.4 Future Work

To-date, the ontology engine has been used to facilitate the semantic formalization of loop detector data for the ATIS application. Future work will be pursued in two different directions:

(1) Additional data sources may be added to extend the scope of the ATIS application to cover other locations. This will be straightforward as the existing semantic mappings may be reused for other datasets of the same type.

(2) The ATIS application may be extended, or a new application may be explored altogether, to incorporate a broader range of dataset types. This might include data on the weather, road closures, concerts and sporting events, safety indices, and so on.

These extensions will provide new opportunities to improve the traveler's experience, and serve to demonstrate the utility of the IT-SoS framework and the value of the ontology engine as an easily extensible tool to support semantic integration.

9.4 Integration with ArcGIS

Based on the motivating scenario described in Section 5.5, a prototype application was developed to investigate the potential use of ontologies to support semantic integration of data in ArcGIS. The application functions as a simple shortest path finder that is augmented with contextual information about the resulting route. A subset of GFX data is mapped into RDF using the vocabulary defined by the TPSO to create an integrated, semantically annotated dataset. This supports a streamlined query process: the data is stored in a triple store that is accessed by SPARQL queries to obtain information about a route from multiple data sources, a process that would otherwise have required a number of complex queries in ArcGIS.

9.4.1 Initial Implementation

At the time of this report, an initial prototype has been implemented and the development of a second version is ongoing. The design adopted for the initial prototype is illustrated in Figure 21. The relevant datasets are extracted from the GFX into a PostgreSQL database in ArcGIS Enterprise. This database is accessed by the ontop ontology mapping tool. Using a predefined set of mappings, RDF data is generated from the database by ontop and stored in an SQLite database

– the knowledge graph. This data may then be accessed as required using Python, in particular the Owlready2[24] and rdflib[25] libraries.

When a user accesses the system, they specify an origin and destination. The system then leverages ArcGIS functionality to calculate the shortest path from the origin to the destination, according to the road segments defined in the network. The then system queries the knowledge graph for contextual information about the road segments in the shortest route. These results are collected and aggregated for presentation to the user.

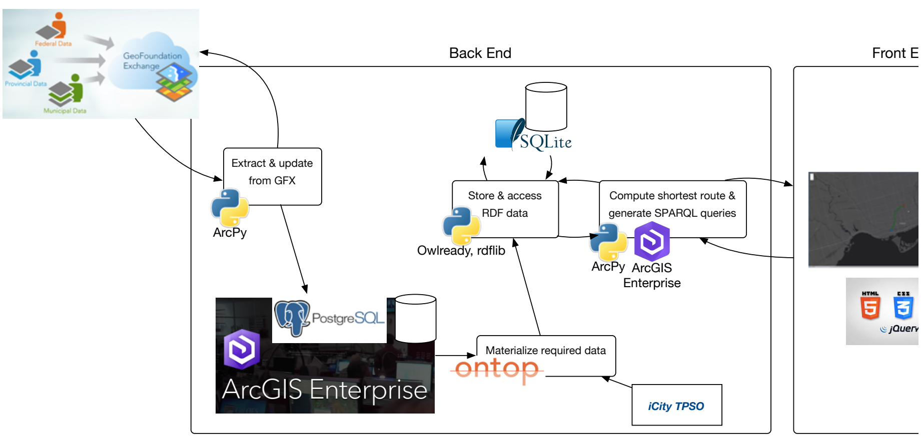


Figure 21: Esri prototype design

Currently, the queries and the results display are hard-coded. However, with the query flexibility supported by the ontology it will be a straightforward extension to enable a more interactive interface in future iterations.

9.4.2 Data Mapping

The scope of the initial prototype was restricted to five key GFX datasets, and focused only on a subset of the fields in each:

- Neighbourhood
- Land Use

[24] https://owlready2.readthedocs.io/en/latest/

[25] http://rdflib.readthedocs.io

- Land Cover
- Point of Interest
- Road Segment

In addition, new tables were generated by ArcGIS processes to capture the spatial relationships between the features defined in the Neighbourhood, Land Use, Land Cover, and Point of Interest datasets, and those in the Road Segment dataset. This was done in advance for efficiency as the ArcGIS processes are highly performant and specialized for such tasks.

The resulting datasets were mapped into RDF using the Ontop OBDA tool. The mappings are described in Appendix D; the Ontop files are available online at https://github.com/EnterpriseIntegrationLab/icity/tree/master/mappings/Esri_GSX.

This application serves as a good example of a scenario where application-specific extensions to the ontology are required. The GFX defines its own set of concepts for its datasets. Instead of defining data mappings according to the generic iCity concepts, it is more precise to use the definitions adopted by the GFX. Therefore, we extend the generic iCity terms such as Road Segment and Land Use with specializations according to the GFX standard. This enables a clear specification of the relationship between GFX-specific terms, general concepts, as well as overlapping concepts from other data sources. For example, a Road Segment defined in the GFX will have some associated Road Class code that indicates the type of road, e.g. a Freeway segment. This still inherits properties of a generic road segment, and a subset of the generic icity road segments will in fact be freeway segments, however a road segment – in general – need not have a GFX road class code. These GFX specific extensions are described along with the mappings in Appendix D.

9.4.3 Future Work

Future work on this project will incorporate the ability to automatically update the knowledge base to incorporate updates to the GFX. It will also explore the inclusion of additional fields as well as datasets that are external to the GFX. Beyond this, there will also be a focus on improving the user interface, both in terms of the visualization of information as well supporting user interaction for decision-making. These extensions will be driven by investigations into more detailed requirements for the NextGen-911 use case. Another application of this prototype that should be explored in future work is the use of the ontology to verify data according to the GFX

standard. The ontology could be extended with the elicitation of more precise, intended semantics from the owners of the GFX standard. Based on these extensions, incoming datasets could then be assessed automatically against the definitions in the ontology. Such an application could contribute to improved efficiencies and data quality.
There are other opportunities for future development of the GFX Ontology extension as well. More meaningful taxonomies could be imposed on the existing land use, land cover, and road classes that are currently defined. In addition, an implicit relationship exists between land use and POI classes that should be explored.

10 Workflows

The activities required in the design and application of an ontology may vary greatly from case to case, however there are also likely to be commonalities. Recognizing and designing workflows around these common tasks improves efficiency and enables repeatability of past work as well as consistency of future work. In this section we provide an overview of the workflows that have been employed for the following key tasks that were involved in the design, maintenance, and application of the iCity TPSO:

1. Data Mapping and Materialization
2. Data Storage and Access
3. Versioning
4. Documentation generation

10.1 Data Mapping

Data mapping refers to the process by which existing data sets are defined according the vocabulary of the ontology. These definitions serve to disambiguate data sets and make their semantics explicit. They are specified in such a way that the data sets may be automatically transformed into, or accessed with Semantic Web technologies through an approach referred to as Ontology Based Data Access (OBDA) [33]. While other approaches are possible, the de facto standard for defining such mappings on the Semantic Web is the RDF to RDF Mapping Language (R2RML)[26].

[26] https://www.w3.org/TR/r2rml/

10.1.1 Alternative approaches

Here, we focus on the triple store architecture, wherein the data sources are transformed (materialized) into triples and uploaded to a triple store(s). This triple store may then be accessed via SPARQL queries (including applications using the Apache Jena framework). It should be noted that another possible architecture involves applying the semantic augmentation to access the data in a database, this is referred to as virtual access.

This guide focuses on the use of the Karma Data Integration Tool[27] for semantic augmentation and data transformation, however it should be noted that several similar tools exist, with varying capabilities and limitations. These tools are often referred to as R2RML processors or OBDA tools; examples are Mastro[28] and Ontop[29], among others.

10.1.2 Basic data mapping/import workflow with Karma and Virtuoso

(1) Design mappings to capture the data using ontology. This step is performed offline and shall be done only once for a particular data source (i.e. all data of like format may be accessed/transformed with the same mapping). Karma provides a GUI to support this process. Note that some cleaning may be required in order to transform the data into an appropriate form.

   i. Open Karma, load dataset and relevant ontology files (in current Karma implementation, imports are not directly applied so uploading only the main ontology file may not capture all of the necessary terms).

   ii. Data cleaning: transform the data as required (reformatting, separation of cell contents, etc).

   This may require some use of Python. For example, in the TTS data we want to transform 3d coordinates to 2d coordinates, and format them according to the WKT format.

   Simple reformat as WKT:

   return "POLYGON(" + getValue("coordinates") + ")"

   Reformat to remove 0-valued 3rd dimension from coordinates:

---

[27] http://usc-isi-i2.github.io/karma/

[28] http://www.obdasystems.com/mastro

[29] https://ontop-vkg.org

```
import re
line = getValue("coordinates")
line = re.sub(',',' ',line)
line = re.sub(' 0 ',',',line)
return line
```

The specification of IRIs is also a good step to take here. In some cases, this may require reformatting of some of the data. It will also likely require the introduction of some base namespace, e.g. "https://w3id.org/icity/TTC_srt_delays/..."

iii. Specify ontology mappings in Karma.
iv. Export R2RML model (ttl or rdf) file. This model is a representation of the mapping of the data into the ontology.
v. At this point, for a one-off transformation the transformed data may also be exported and saved for upload into the desired triple store. However, if the mappings are to be generated and uploaded at a later date, only the R2RML model is required.

10.1.3 Repeated Data Mappings

For multiple datasets with the same mapping, we can automate the above process once an initial mapping has been defined. This is possible using the batch mode in Karma[30].

Example: let's download a bunch of TTC incident files and try to map them with a single command, using the R2RML mapping that we defined for the first dataset.
Beginning with data files:

- SubwayDelay201706.csv
- SubwaySRTLogs201707.csv
- SubwaySRTLogs201708.csv

And a pre-defined mapping file

- SubwaySRT_Mapping

[30] https://github.com/usc-isi-i2/Web-Karma/wiki/Batch-Mode-for-RDF-Generation

There are 2 ways to do this: offline or online through the API. The API may eventually be useful should the mappings be incorporated into part of some larger process (e.g. a reaction to something: a file being uploaded or stream data being received). Note that a different process would need to be implemented for each file type in order to account for the different mapping files. For now, we have employed the offline implementation.

10.1.4 Offline Batch Mapping

Batch mapping is useful for large quantities of files, or large file sizes. Note that for large mappings, the JVM memory may need to be increased when the commands are run.

First-time setup: To build the offline jar, go to the karma-offline subdirectory and execute the following:

```
cd karma-offline
mvn install -P shaded
```

```
java -cp karma-offline-0.0.1-SNAPSHOT-shaded.jar
edu.isi.karma.rdf.OfflineRdfGenerator --sourcetype CSV --filepath
"./files/SubwayDelay201706.csv" --modelfilepath "./files/SubwaySRT_Mapping.ttl" --
outputfile "./files/ttc-subway-delay-201706.n3" --sourcename "ttc"
```

10.1.4.1 A basic script to map a directory of files of the same type

Given:

- One or more files of the same type (i.e. with the same ontology mapping), in the directory "`./karma-offline/target/files`".
- A predefined mapping file (`SubwaySRT_Mapping.ttl`), stored in the same directory.

Execute from ./karma-offline/target directory:

```
for file in ./files/*.csv; do java -cp karma-offline-0.0.1-SNAPSHOT-
shaded.jar edu.isi.karma.rdf.OfflineRdfGenerator --sourcetype CSV --
filepath "$file" --modelfilepath "./files/SubwaySRT_Mapping.ttl" --
outputfile "${file/%csv}ttl" --sourcename "ttc"; done
```

For large files the default memory limit may need to be adjusted, e.g.:

```
java -Xmx6000m -cp karma-offline-0.0.1-SNAPSHOT-shaded.jar
edu.isi.karma.rdf.OfflineRdfGenerator --sourcetype CSV --filepath
"files/trip_stations.csv" --modelfilepath
"files/trip_stations_model.ttl" --outputfile
"files/trip_stations.ttl" --sourcename "tasha_microsim"
```

Result:

- A translated set of triples for each input file (`<filename>.ttl`)

Notes:

- The Karma installation (one-click install) doesn't come with karma-offline, this requires installation of the full version from Github.
- To run Karma (gui app) from the full installation:
  `>cd Web-Karma/karma-web`
  `>mvn jetty:run`
  Karma should be accessible at: http://localhost:8080
- File paths are relative to the target directory that the command is executed from

10.2 Data Storage and Access

Once the ontology mappings have been created and the RDF triples have been materialized, typically the data must be stored somewhere that is accessible with Semantic Web tools. As mentioned in the previous section, one option is to use OBDA tools to provide virtual access to the data. In this scenario, data is stored in its native form in some relational database(s) and accessed through SPARQL queries that are transformed into SQL queries.

To-date, we have focused on the alternative approach: generating RDF triples from the data, based on a mapping to the ontology, and uploading the data into a triple store. Many different triple stores are available. All triple stores provide the same core functionality (that is, to store and provide access to RDF triples), with different characteristics. Factors in choosing a triple store may include cost, capabilities (in terms of speed and storage), as well as other tools that may be packaged with the store.

A triple store will provide a SPARQL endpoint that can be used to evaluate SPARQL queries against the uploaded data. Most also provide a SPARQL API that can be used to support integration the store directly with some application(s).

All triple stores provide some mechanism(s) to upload data. This may vary slightly between implementations.

10.2.1 Upload to triple store

Karma includes an option to configure upload to a triple store ("publishing data"), therefore it's possible that the mapping and upload process may be combined into a single step. However, it is not clear from the documentation whether this is possible in batch mode. It may be more appropriate to use the upload functionality provided by the chosen triple store.
Allegrograph supports data upload through the WebView tool, but also provides a tool for more efficient, server-side, command-line uploads. This process is outlined below.

Uploading large datasets server-side on Allegrograph:

1. Transfer files to server where Allegrograph instance is running, e.g.
   ```
   scp -r -i katsumi-key.pem <local location of files to upload>
   ec2-user@ec2-35-183-119-164.ca-central-1.compute.amazonaws.com:
   <remote location of files to upload on aws>
   ```
2. Access server, e.g.
   ```
   ssh -i katsumi-key.pem ec2-user@ec2-35-183-119-164.ca-central-
   1.compute.amazonaws.com
   ```
3. Run agtool to load file(s) onto specified graph:
   ```
   agtool load http://test:xyzzy@ec2-35-183-119-164.ca-central-
   1.compute.amazonaws.com:10035/repositories/gtfs_test
   ./gtfs_to_upload/*.ttl
   ```

10.3 ***Ontology Documentation***

In addition to the publication and maintenance of the detailed report provided here, it is a good practice to provide detailed documentation at the individual ontology level. The iCity ontologies use Widoco[31] [34] to automatically generate HTML documentation pages based on the metadata specified in each ontology's OWL file. When an ontology's IRI is accessed via a web browser, rewrite rules (discussed at further length in the following section) will automatically return the HTML documentation rather than the native OWL file. This helps to ensure availability of the

[31] https://zenodo.org/badge/latestdoi/11427075

documentation and thus usability of the ontologies. A useful guide for the specification of ontology-level metadata is provided by the authors of Widoco here: http://dgarijo.github.io/Widoco/doc/bestPractices/index-en.html. Future work should focus on the extension and elaboration of metadata that is currently encoded in the ontology in order to enrich the resulting HTML documentation.

Two simple commands can be run in the directory where the ontologies are stored to automatically update all of the documentation files based on the content of the github repository.

To update the documentation for the latest (default) version of the icity ontologies:

```
for file in *.owl; do java -jar /Applications/widoco-1.4.6-jar-with-
dependencies.jar -ontFile $file -outFolder "${file/%.owl}" -
getOntologyMetadata -rewriteAll -lang en -includeImportedOntologies -
htaccess -webVowl -licensius; done
```

To update all of the version-specific documentation files:

```
for file in */*.owl; do java -jar /Applications/widoco-1.4.6-jar-with-
dependencies.jar -ontFile $file -outFolder "${file/%.owl}" -
getOntologyMetadata -rewriteAll -lang en -includeImportedOntologies -
htaccess -webVowl -licensius; done
```

10.4 Ontology Versioning

OWL2 includes a version IRI annotation at the ontology-level, however a detailed approach for releasing of new versions of a particular ontology is not prescribed. In the continuing the development of the iCity ontologies, it is important to approach the task of versioning in a standard manner. In addition to considering the relevant suggestions and requirements as described in the OWL2 specification, we have identified the following set of key requirements that must be met by the versioning process:

1. Latest versions of the ontologies should be easily found and identifiable as such. In this way users will be aware when improvements and/or corrections are available.
2. Permanent IRIs must be used to name the ontologies. These IRIs must dereference to the ontologies' locations thus ensuring that they are easily accessible.
3. Updated versions of the ontologies should not be *unknowingly* updated in ontologies that import/reuse them.
4. The task of incorporating updates into ontologies that are using previous versions must be addressed.

The last two points are related to an important challenge regarding potential issues arising from the update of imported ontologies. When a new version of some iCity TPSO ontology – Ontology X – is issued, it will be desirable to update any ontologies that import Ontology X (including and in particular those other iCity ontologies that may Ontology X). One consequence of this update is that the importing ontology (the ontology that imports Ontology X) is changed in that it is now using a new version of Ontology X, so it too should be identified as a new version. Clearly tracking and reflecting such changes is critical for ensuring the usability of the ontologies.

10.4.1 Versioning Principles

The following principles must be adhered to in order to avoid issues with the versioning process (detailed in the subsequent section).

1. All iCity TPSO ontologies must employ an ontology IRI and version IRI to make the series of ontology versions explicit.
2. Any change (addition, removal, modification) to an ontology's axioms, including those of any ontologies it (directly or indirectly) imports is considered a revision to the ontology. When such revisions are officially released, they must be distinguished as such using the Version IRI attribute.
3. According to best practices, the ontology IRI and version IRI locations should be defined and de-referenceable with persistent URLs. The current version of an ontology should be available at the ontology IRI to ensure that it is identified as the current version and that it is easily discoverable to the public.
4. All imports must directly reference the imported ontology's *Version IRI* (if available) to make the reuse explicit; this is necessary as any update to an imported version should result in an update to the importing ontology's version as well. Failure to do so potentially violates one of the conventions for versioning described by the W3C. If the Ontology IRI is used, then the imported ontology will always be the most recent version. While it is undoubtedly desirable to ensure that the most correct, up-to-date version of a resource is used, when a new version is created, by definition of import the ontology that is importing this new version is now also changed. Thus, according to the W3C guidelines, this ontology should also be recognized with a new Version IRI. Importing the Ontology IRI makes this distinction difficult to recognize and impossible to maintain.

10.4.2 Process to Update Ontology-x.owl

The following process describes the steps necessary to release an updated version of some iCity Ontology.

<table>
<tr><td>Process:</td><td>Update Version</td></tr>
<tr><td>Input:</td><td>New version of an ontology: ontology-x.owl</td></tr>
<tr><td>Output:</td><td>New version of TPSO ontologies, with updated ontology-x.owl incorporated</td></tr>
<tr><td colspan="2">Steps:</td></tr>
<tr><td colspan="2">1. Define new Version IRI to distinguish the ontology (reflect that it is a new version). Common practice for naming adopted by the iCity Ontologies is <Version IRI> = <Ontology IRI>+<Version number>. For example, if we have Ontology IRI = "http://w3id.org/icity/Change/" then the Version IRI should be of the format " http://w3id.org/icity/Change/1.0/"</td></tr>
<tr><td colspan="2">2. Rename the owl:versionIRI attribute value to the new Version IRI. Do not rename rdf:about, this tag defines the Ontology IRI.</td></tr>
<tr><td colspan="2">3. Upload the ontology and create a persistent url for the new Version IRI to redirect to the new ontology's location, according to the predefined rewrite rules in the .htacess file.[32]</td></tr>
<tr><td colspan="2">4. Update rewrite rules for the persistent url for the Ontology IRI to redirect to this new, most recent version.</td></tr>
<tr><td colspan="2">5. As required, update ontologies that import Ontology-x.owl. Specifically:
5.1. Create a new version of the importing ontology (perform Process to Update Ontology-x).
5.2. Update the <import> tag to reflect the import of the new VersionIRI of Ontology-x.
5.3. Update (replace) all instances of the old Version IRI to the new Version IRI (ideally, this will simply require an update to the prefix name definition).
5.4. Perform the Update Version process on the importing ontology.</td></tr>
</table>

In our experience, implementing these updates may be problematic with ontology editors. The interface may obscure dependencies between ontologies thus making the required changes unclear. At the time of this report, we recommend modifying the xml files directly. Currently, the most straightforward way to implement these updates is by replacing all of the old version IRIs at once. The imports structure of the ontology must be considered to determine which ontologies have been impacted by the revision. In some cases, a revision may require changes to the design of the importing ontologies; at a minimum, since the importing ontologies now import

[32] The exact procedure will vary depending on the storage / purl set-up.

a new version of Ontology-X, this must be reflected with the release of a new version as discussed earlier in this section.

10.4.3 Versioning infrastructure

The following are required components of an infrastructure that supports the versioning process described previously.

10.4.3.1 File Storage

Versions of the ontology may be developed and maintained via a version control system such as Github, however this is not required. The granularity enabled by version control system may be useful but is not necessary for the kind of versioning described here. For example, in git every change that is made creates a new version of the ontology (file), but each of these changes should not necessarily correspond to the issuing of a new version IRI for the ontology. Each version IRI instead corresponds to a new release of the ontology.

The chosen file repository has two key roles:

1. To host the ontology and its version history and make it available and understandable to the public.
2. To support continued development of the ontology (i.e. between versions).

10.4.3.2 Permanent URL Redirect

Several organization schemes are possible with respect to the storage of versions of each owl file (and corresponding html documentation), and the approach to redirect a persistent url to the appropriate version of the file. Currently, we employ a .htaccess file with rewrite rules to redirect web requests for the ontology IRIs to their actual location (or documentation, as appropriate). The purpose of the .htaccess file is to facilitate the appropriate redirect from the permanent urls to the files maintained in the Github repository. The rewrite rules for new versions of the documentation (discussed in the previous section) are similarly accounted for in the .htaccess file.

Since the OWL files may be versioned independently (a change to one file doesn't *necessarily* mean a new version of all of the files), the Ontology IRIs may redirect to different versions, so the rewrite rules would need to be custom for each path. One option would be to maintain folders for each version of the UrbanSystem ontology (resulting in some duplication of files). Instead, we maintain each sub-ontology's version history in its own file; Ontology Version IRIs are directed here. Then, we also maintain a collection of "latest" ontology files in the main directory (without any version suffix). The rewrite rule for this is straightforward and results in only a single duplicate file for each sub-ontology. However, it does require an additional step for the update process: that the "latest" file in the main directory be replaced.

The .htaccess file currently in use is available for review in the /docs folder of the iCity Github repository[33]. The version of the file that is in use is currently stored in the /ontologies/icity subdirectory of the EIL server; this is the file path that has been chosen for the ontologies' IRIs. Alternatives such as w3id.org exist for the definition of permanent urls, however issues with security certificates were encountered that could not be resolved. To address this and avoid future issues we have elected to define the IRIs according to a University-owned url.

10.4.4 Future Work

Guidelines on semantic version numbering[34] should be adopted for future revisions.

11 Future Work

- Standardization: mapping to concepts in other areas of transportation & city services, consider alignment to foundational ontology
- Expand on transportation planning domain: Freight, Complete Streets
- Expand use cases: Simulation management, Survey management, Provenance
- Explore opportunities for inference
- Implementation with GUDR (data maintenance strategy implementation)
- Naming conventions for individuals (to be incorporated into mappings)
- Alignment / mapping between reused ontologies (e.g. owl-time and om concepts of duration)
- Improve design by generalizing more concepts as possible (e.g. parking rates – define rate representation in general for reuse in other areas)

[33] https://raw.githubusercontent.com/EnterpriseIntegrationLab/icity/master/docs/.htaccess

[34] http://semver.org/

- FOL extensions

11.1 Research

- Ontology versioning
- Ontology visualization (navigation of terms, understanding / querying data)
- Ontologies for simulation
- Ontologies for data collection

Ontologies present the opportunity to concurrently address multiple challenges for urban modelling and simulation, such as:

1. the incomparability between models and results,
2. the need for bespoke query design, and
3. the opaqueness/complexity of models and results.

In this application, we focus on the use of the ontology to formalize the simulation results. Future work should extend this to focus on the models and the simulation runs.

Additional aspects of the simulation (e.g. inputs, run dates, models used) are relevant, and should be defined in future work. They are discussed in Section **Error! Reference source not found.**.

- *Complex mappings and the need for (and definition of) shortcut properties*
-

Future iterations of the iCity Ontology should develop a deeper semantics for the concepts identified here, in addition to an expansion of scope. This will be dictated largely by use cases identified by the various project groups, which will not only determine additional requirements for representation, but potential applications for additional functionality that may be supported by the ontology.

Acknowledgements

This project was supported by the Ontario Ministry of Research and Innovation through the ORF-RE program.

References

[1] E. J. Miller and P. A. Salvini, "The integrated land use, transportation, environment (ILUTE) microsimulation modelling system: Description and current status," *Travel behaviour research: The leading edge,* pp. 711-724, 2001.

[2] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," ed: Stanford knowledge systems laboratory technical report KSL-01-05 and …, 2001.

[3] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The knowledge engineering review,* vol. 11, no. 2, pp. 93-136, 1996.

[4] E. J. Miller, J. Vaughan, D. King, and M. Austin, "Implementation of a "next generation" activity-based travel demand model: the Toronto case," in *Presentation at the Travel Demand Modelling and Traffic Simulation Session of the 2015 Conference of the Transportation Association of Canada*, 2015.

[5] M. Elshenawy, B. Abdulhai, and M. El-Darieby, "Towards a service-oriented cyber–physical systems of systems for smart city mobility applications," *Future Generation Computer Systems,* vol. 79, pp. 575-587, 2018.

[6] R. Battle and D. Kolas, "Linking geospatial data with GeoSPARQL," *Semant Web J Interoperability, Usability, Appl. Accessed,* vol. 24, 2011.

[7] *Time Ontology in OWL*, O. W3C, 2017. [Online]. Available: https://www.w3.org/TR/owl-time/

[8] J. R. Hobbs and F. Pan, "Time ontology in OWL," *W3C working draft,* vol. 27, p. 133, 2006.

[9] C. Welty, R. Fikes, and S. Makarios, "A reusable ontology for fluents in OWL," in *FOIS*, 2006, vol. 150, pp. 226-236.

[10] H.-U. Krieger, "Where temporal description logics fail: Representing temporally-changing relationships," in *Annual Conference on Artificial Intelligence*, 2008: Springer, pp. 249-257.

[11] M. Katsumi and M. Fox, "A Logical Design Pattern for Representing Change Over Time in OWL."

[12] L. Obrst *et al.*, "The 2006 Upper Ontology Summit Joint Communiqué," *Applied Ontology,* vol. 1, no. 2, pp. 203-211, 2006.

[13] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with DOLCE," in *International Conference on Knowledge Engineering and Knowledge Management*, 2002: Springer, pp. 166-181.

[14] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology*. Mit Press, 2015.

[15] M. Katsumi and M. Fox, "Defining Activity Specifications in OWL," in *WOP@ ISWC*, 2017.

[16] R. Kowalski and M. Sergot, "A logic-based calculus of events. NewGeneration Computing 4," ed, 1986.

[17] M. S. Fox, "Constraint-Directed Search: A Case Study of Job-Shop Scheduling," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1983.

[18] A. Sathi, M. S. Fox, and M. Greenberg, "Representation of activity knowledge for project management," *IEEE Transactions on pattern analysis and machine intelligence,* no. 5, pp. 531-552, 1985.

[19] M. S. Fox and M. Gruninger, "Enterprise modeling," *AI magazine,* vol. 19, no. 3, pp. 109-109, 1998.

[20] M. Grüninger, "Using the PSL ontology," in *Handbook on Ontologies*: Springer, 2009, pp. 423-443.

[1] E. J. Miller and P. A. Salvini, "The integrated land use, transportation, environment (ILUTE) microsimulation modelling system: Description and current status," *Travel behaviour research: The leading edge,* pp. 711-724, 2001.

[2] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," ed: Stanford knowledge systems laboratory technical report KSL-01-05 and …, 2001.

[3] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *The knowledge engineering review,* vol. 11, no. 2, pp. 93-136, 1996.

[4] E. J. Miller, J. Vaughan, D. King, and M. Austin, "Implementation of a "next generation" activity-based travel demand model: the Toronto case," in *Presentation at the Travel Demand Modelling and Traffic Simulation Session of the 2015 Conference of the Transportation Association of Canada*, 2015.

[5] M. Elshenawy, B. Abdulhai, and M. El-Darieby, "Towards a service-oriented cyber–physical systems of systems for smart city mobility applications," *Future Generation Computer Systems,* vol. 79, pp. 575-587, 2018.

[6] R. Battle and D. Kolas, "Linking geospatial data with GeoSPARQL," *Semant Web J Interoperability, Usability, Appl. Accessed,* vol. 24, 2011.

[7] *Time Ontology in OWL*, O. W3C, 2017. [Online]. Available: https://www.w3.org/TR/owl-time/

[8] J. R. Hobbs and F. Pan, "Time ontology in OWL," *W3C working draft,* vol. 27, p. 133, 2006.

[9] C. Welty, R. Fikes, and S. Makarios, "A reusable ontology for fluents in OWL," in *FOIS*, 2006, vol. 150, pp. 226-236.

[10] H.-U. Krieger, "Where temporal description logics fail: Representing temporally-changing relationships," in *Annual Conference on Artificial Intelligence*, 2008: Springer, pp. 249-257.

[11] M. Katsumi and M. Fox, "A Logical Design Pattern for Representing Change Over Time in OWL."

[12] L. Obrst *et al.*, "The 2006 Upper Ontology Summit Joint Communiqué," *Applied Ontology,* vol. 1, no. 2, pp. 203-211, 2006.

[13] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with DOLCE," in *International Conference on Knowledge Engineering and Knowledge Management*, 2002: Springer, pp. 166-181.

[14] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology*. Mit Press, 2015.

[15] M. Katsumi and M. Fox, "Defining Activity Specifications in OWL," in *WOP@ ISWC*, 2017.

[16] R. Kowalski and M. Sergot, "A logic-based calculus of events. NewGeneration Computing 4," ed, 1986.

[17] M. S. Fox, "Constraint-Directed Search: A Case Study of Job-Shop Scheduling," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1983.

[18] A. Sathi, M. S. Fox, and M. Greenberg, "Representation of activity knowledge for project management," *IEEE Transactions on pattern analysis and machine intelligence,* no. 5, pp. 531-552, 1985.

[19] M. S. Fox and M. Gruninger, "Enterprise modeling," *AI magazine,* vol. 19, no. 3, pp. 109-109, 1998.

[20] M. Grüninger, "Using the PSL ontology," in *Handbook on Ontologies*: Springer, 2009, pp. 423-443.

[21] M. S. Fox, "A foundation ontology for global city indicators," *University of Toronto, Toronto, Global Cities Institute,* 2013.

Appendix A. TASHA Data Mapping

- Currently looking at modeling "microsim" output from TASHA
- What about representation of the model and / or simulation itself? (e.g. parameters, other model attributes)

The TASHA Microsim results are output into 5 csv files: persons.csv (basic demographic attributes of the people taking the trips), trips.csv (description of the trips taken: by which person, and from what origin to what destination), trip_modes.csv (description of the modes used to make the trip), trip_stations.csv (identifies intermediate stations used to change modes – e.g. the station at which the trip changes from auto to transit), and facilitate_passenger.csv (indicates a relationship between two trips when one trip – by the driver – facilitates another – by the passenger).

Mapping

Note that each instance represented by the output files should be distinguished from instances in the real world, as instances of some simulation output.

- Rather than an ontology of the urban system, this data should be formalized by an ontology of simulations of the urban system. This requires an extension of the urban system ontology to capture the notion of simulation, and to formalize the relationship between an instance of a simulation and various instances of domain specific classes such as persons, trips, etc.
- Propose an extension of UrbanSystem.owl: UrbanSystemSimulation.owl
    - Introduce classes: model, simulation run etc…
    - Key relationship: SimulationRun hasSimulationOutput some UrbanSystemOntologyThing

The result of an urban system simulation is essentially an instance of some part(s) of the urban system and can be formalized by the urban system ontology. In addition, we need a way to distinguish such instances from real-world data. To accomplish this, we extend the Urban System Ontology with an ontology for simulations: the Urban System Simulation Ontology. The following concepts are required for the Simulation extension:

- Simulation: A Simulation is an execution of some model system. It has some input and **output** data, defined by some instances of the UrbanSystemOntologyClass.
  A Simulation has a **run date**.

Object	Property	Value
Simulation	hasSimulationOutput	some UrbanSystemOntologyThing
	hasRunDate	exactly 1 xsd:dateTime

Simulation Metadata

Each set of simulation output files should be associated with a particular model run.

Future Work:

- Determine whether output files of the simulation metadata exist (if not, request output of some basic metadata, e.g. date run, etc?).
    - <_:simulation_id> a sim:Simulation; hasRunDateTime…

Mississauga Zones

- name: zone id; Note: unclear whether this ID is specific to TASHA or intended to match up to other traffic zone ids
  -> for now, transform to ensure unique ID:
  return "trafficzone_traisi_" + getValue("name")
  -> <name_transform > a landuse:TrafficZone
- coordinates:
  -> apply transformation (used for other esri data) to format coordinates as WKT:
  `import re`
- `coord = getValue("coordinates").replace(",0", ",")`
- `coord = re.sub(r'(\d+)(,)(\d+)', r'\1 \3', coord)`
- `coord = "POLYGON(" + coord + ")"`
- `coord = coord.replace(",)",")")`
- `return coord`
- <name_transform> spatialloc:hasGeometry [a Geometry; asWKT <coordinates_transform>].

persons.csv

- -> add <simulation_id> attribute; default value "dummy_sim_iri"
- Household_id:

  -> <household_id> a household:HouseholdPD; sim:outputOfSimulation <_:simulation_id>
- Person_id:

  -> transform person_id to unique identifier:

  <person_id_transform> : return "h" + getValue("household_id") + "p" + getValue("person_id")

  -> <person_id> a person:PersonPD; sim:outputOfSimulation <_:simulation_id>; hasManifestation [a person:Person; inverse(household:hasMember) <household_id>].

- Age:

  Revised: Person add age property for Person: Person has Age exactly 1 uom:duration

  -> <_person@t> hasAge [a uom:duration; hasValue [a uom:measure; uom:hasNumericalValue <age>; uom:hasUnit uom:year]]
- Sex:

  Revised: Person add instances of Sex class (M/F for the purposes of urban studies) transform value of Sex attribute to IRI:

  -> <person_id> person:hasSex <sex_transform>
- License (Boolean)

  revised: Person add Boolean property: isLicensedDriver for Person

  -> <_person@t> isLicensedDriver <license>
- Transit_pass (Boolean)

  transit:TransitPass…

  revised: UrbanSystem rename hasPass to hasTransitPass, rename Pass to TransitPass consider definition of a boolean property

  -> need to apply a transformation to achieve: if <transit_pass> = 'true' then there is some transit pass object

  -> transform <transit_pass> to <dummy_transit_pass_id>:

  if (getValue("transit_pass")=="true"):

return "transitpass_h" + getValue("household_id") + "p" + getValue("person_id")

-> <_person@t> hasTransitPass <dummy_transit_pass_id>.

<dummy_transit_pass_id> a transit:TransitPass.

- Employment_status:

~~revised: *Org* add Employee hasEmploymentType some EmploymentStatus~~

Revised: introduced subclasses of employee (FT, PT,…)

-> need to apply transformation to convert employment_status to appropriate class name:

http://ontology.eil.utoronto.ca/icity/Organization/FullTimeRegEmployee

http://ontology.eil.utoronto.ca/icity/Organization/FullTimeHomeEmployee

http://ontology.eil.utoronto.ca/icity/Organization/PartTimeRegEmployee

http://ontology.eil.utoronto.ca/icity/Organization/PartTimeHomeEmployee

es = getValue("employment_status")

- if (es =="F"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/FullTimeRegEmployee"
- if (es =="P"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/PartTimeRegEmployee"
- if (es =="H"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/FullTimeHomeEmployee"
- if (es=="J"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/PartTimeHomeEmployee"

-> <_person@t> a <employement_status_transform>

- Occupation:

revised: Org create subclasses of Occupation to capture occupation types

-> transform occupation field to appropriate occupation subclasses:

O: not employed

G: general office / clerical

http://ontology.eil.utoronto.ca/icity/Organization/GeneralOffice

P: professional / management / technical

http://ontology.eil.utoronto.ca/icity/Organization/Professional

S: retail sales and service

http://ontology.eil.utoronto.ca/icity/Organization/Sales

M: manufacturing / construction / trades

http://ontology.eil.utoronto.ca/icity/Organization/Trades

o = getValue("occupation")

if (o == "G"):

- return "http://ontology.eil.utoronto.ca/icity/Organization/GeneralOffice"
- if (o == "P"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/Professional"
- if (o == "S"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/Sales"
- if (o == "M"):
- return "http://ontology.eil.utoronto.ca/icity/Organization/Trades"

-> <_person@t> org:employedAs [a Occupation; a <occupation_transform>]

- Free_parking (Boolean)[35]: true if free parking is available at the person's work location

Revised: introduced FreeParking subclass of parking policy; other more specific scenarios could be subclasses of the FreeParking class.

-> transform to represent free parking policy if applicable

if (getValue("free_parking") == "true"):

- return "http://ontology.eil.utoronto.ca/icity/Parking/FreeParkingPolicy"

-> <_person@t> org:EmployedBy [a org:Organization; parking:hasAllocatedParking [a park:ParkingArea; park:hasParkingPolicy [a park:ParkingPolicy; a <free_parking_transform>]]]

- Student_status

~~to do: add Student subclass of Person; enrolledAt some org:School;~~

Revised: added subclasses of Student to capture enrollment type (FullTimeStudent, PartTimeStudent). As future work, these classes may to be defined based upon some notion of course enrollment.

-> transform to capture appropriate classes (FT or PT)

s = getValue("student_status")

- if (s == "F"):

35

- return "http://ontology.eil.utoronto.ca/icity/Organization/FullTimeStudent"
- if (s == "P"):
- return http://ontology.eil.utoronto.ca/icity/Organization/PartTimeStudent

  -> <person@t> a <student_transform>
- Work_zone: work location is contained in some zone; 0 if unemployed

  assumption: zone IDs correspond to traffic zone identifiers; otherwise we can use the generic Parcel class.

  -> transform into "trafficzone_trasi_" iri

  if (getValue("work_zone") != "0"):
- return "trafficzone_traisi_" + getValue("work_zone")

  -> <work_zone_transform> a landuse:TrafficZone

  -> <employer_blank_node> spatialloc:hasLocation [a Feature; inverse(contains) <work_zone_transform>]
- School_zone: school location is contained in some zone; 0 if not a student

  -> transform into "trafficzone_trasi_" iri

  if (getValue("school_zone") != "0"):
- return "trafficzone_traisi_" + getValue("school_zone")

  -> <school_zone_transform> a landuse:TrafficZone

  -> <school_blank_node> spatialloc:hasLocation [a Feature; inverse(contains) <school_zone_transform>]
- Weight: omitted
- Additional mappings required: in order to capture the *conditional existence* of some Organization that employs the Person, or School in which the person is enrolled; we need to instantiate custom blank nodes. i.e. rather than define the blank node in the mapping, we need to create a column "employer_blank_node" which is only defined with a blank node value for entries where the person is employed. Similarly for "school_blank_node".

  -> <employer_blank_node>

  if (getValue("employment_status") != "O") & (getValue("employment_status") != ""):
- -> return "_:" + getValue("household_id") + "_" + getValue("person_id") + "_employer"

  -> <_person@t> org:employedBy <employer_blank_node>.

-> <school_blank_node>
if (getValue("student_status") != ""):
return "_:" + getValue("household_id") + "_" + getValue("person_id") + "_school"
-> <_person@t> org:enrolledIn <school_blank_node>

- Karma seems to have issues with certain types of mappings, therefore we need to manually generate some additional classes of blank nodes (which are currently being incorrectly generated by the software):
    - Feature1 (school loc)
      if (getValue("student_status") != ""):
    - return "_:" + getValue("household_id") + "_" + getValue("person_id") + "_school_loc"
    - Feature1 (work loc)
    - Duration (age)

Notes:

- Make sure to check that the fields are correctly transformed into IRIs; in some cases prefixes may need to be added (e.g. to distinguish between generic ids)

Discussion:

Note that the "free_parking" field is formalized as representing whether or not there is a parking area that is associated with the person's place of employment with a free parking policy. However, the value of this field is ambiguous; the parking area may offer free parking for employees only, it may offer free parking for the general publish, or free parking during specific times of day (at which the employee is at work)/

trips.csv

- -> add <simulation_id> attribute; default value "dummy_sim_iri"
- Household_id:
  -> <household_id> a household:HouseholdPD; sim:outputOfSimulation <_:simulation_id>
- Person_id:
  -> transform to create unique identifier:

<person_id_transform> : return "h" + getValue("household_id") + "p" + getValue("person_id")

-> <person_id_transform> a person:PersonPD; sim:outputOfSimulation <_:simulation_id>; hasManifestation [a person:Person; inverse(household:hasMember) <household_id>].

->transform to create blank node for manifestation:

return "_:h" + getValue("household_id") + "p" + getValue("person_id") + "_person_" + getValue("trip_id")

- Trip_id:
  todo: consider whether a more specific type of participation (e.g. 'performedBy') should be defined
  -> transform to create unique identifier:
  return "h" + getValue("household_id") + "p" + getValue("person_id") + "t" + getValue("trip_id")
  -> <trip_id_transform> a trip:Trip; activity:hasParticipant <_person@t>.
- O_zone: origin zone of the trip; i.e. the trip begins at a location that is contained in the o_zone.
  ->transform o_zone value to ensure uniqueness and identify provenance
  return " trafficzone_traisi_" + getValue("o_zone")
  -> <o_zone_transform> a landuse:TrafficZone.
  -> <trip_id> trip:startLoc [a spatialLoc:Feature; (inverse)(sfcontains) <o_zone_transform>]
- O_act: activity at the origin zone; i.e. activity that the traveller was performing just prior to the trip, one of: {PrimaryWork, SecondaryWork, ReturnFromWork, WorkBasedBusines, School, JointOther, IndividualOther, Market, JointMarket, Home}
  revised: Created TASHA extension of UrbanSystemSimulation ontology. add TASHA Activity subclasses (note there is overlap with TTS activities, however these activities vary depending on the TTS year, and are slightly more specific)
  *Assumption: o_act occurs directly before the trip.
  -> transform o_act into classes as defined in icity TASHA extension:
  return "http://ontology.eil.utoronto.ca/icity/TASHA/" + getValue("o_act")

-> <_oact> a activity:Activity; hasParticipant <person@t>; occursDirectlyBefore <trip_id>.

-> <_oact> a <activity_transform>.

- D_zone: destination zone of the trip

  todo: landuse:Zone subclass of Parcel

  todo: add optional relationship for Parcel that identifies its associated organization

  ->transform d_zone value to ensure uniqueness and identify provenance

  return " trafficzone_traisi_" + getValue("d_zone")

  -> <d_zone_transform> a landuse:Zone; zoneSystemFor ??.

  -> <trip_id> trip:endLoc [a spatialLoc:SpatialFeature; (inverse)(geo:contains) <d_zone_transform>]

- D_act: as above with o_act

  *Assumption: d_act occurs directly after the trip.

  -> transform d_act into classes as defined in the icity TASHA extension:

  return "http://ontology.eil.utoronto.ca/icity/TASHA/" + getValue("d_act")

- Weight

Notes:

- A note on activity ordering, in particular as it applies to the concepts of origin- and destination-activities used in TASHA: Although we are unable to fully define the semantics, some notion of an ordering on activity occurrences must be captured in some cases. To address this, we introduce the properties: “occursBefore” and “occursDirectlyBefore” in the Activity ontology.

  An activity occursBefore another if its endOf instant is before the beginOf instant of the other activity; the occursBefore relation is transitive. An activity occursDirectlyBefore another if it occursAt an interval that meets the interval of the other activity. We cannot define this semantics in OWL, though it would be supported by an extension with rules. In OWL, we are only able to comment on the semantics, and define occursBefore as transitive and occursDirectlyBefore as a subproperty of occursBefore.

trip_modes.csv

- Trip_id:
  -> transform to create unique identifier:
  return "h" + getValue("household_id") + "p" + getValue("person_id") + "t" + getValue("trip_id")
  -> <trip_id_transform> a trip:Trip; activity:hasParticipant[36] <_person@t>.
- Mode: {Auto, Passenger, WAT, DAT, Walk, Bike, Carpool, Schoolbus, RideShare}
  *assumption: the trip is made *completely* with the specified mode
  WAT: "walk access transit"
  DAT: "drive access transit"
  -> transform into unique TASHA IRIs
  return "http://ontology.eil.utoronto.ca/icity/TASHA/" + getValue("mode")
  <mode_transform> a Mode.
  -> <trip_id_transform> trip:viaMode <mode_transform>.
- O_depart
  transform "minutes from midnight" to xsd:time
  *assumption: based on a review of the data, we assume "minutes from midnight" is minutes *after* midnight as opposed to before.
  - *add dummy date columns
    -> import time
  - c = time.strptime("00:00:00","%H:%M:%S")
  - t = time.mktime(c)
  - otime = getValue("o_depart")
  - otime = float(otime)*60
  - t = t + otime
  - tstring = time.strftime("%H:%M:%S",time.localtime(t))
  - return getValue("dummy_date") + "T" + tstring + "-5:00"

[36] Ontology was later updated to include a more specific property: http://ontology.eil.utoronto.ca/icity/UrbanSystem/tripPerformedBy

- -> <trip> activity:beginOf [a time:Instant; time:inXSDDateTimeStamp <o_depart_transform>].
- **Note**: a trip (an activity) begins and ends at a particular instant in time. In order to define this instant using the xsd:dateTime datatype, we need to provide a date. To achieve this, for now, we introduce a <dummy_date> attribute with a default date: 2018-01-01. This is consistent with the simulation data as it is intended to capture a single day of travel activity. In future applications it will be desirable to make a more informed date selection.
- D_arrive
  convert "minutes from midnight" to xsd:time
  -> as above for o_depart
- Weight

Notes:

- Based on a review of the facilitate_passenger data (in particular, the '-1' driver trip ID assigned if the driver "facilitates" the passenger from home), we assume that the trip of the driver and the passenger is identified as a single trip, rather than two individual trips. The *overall* trip of the driver (including travel before / after the trip with the passenger) then is implicit, and there are no overlapping trips captured in this dataset. Another way to view this is to consider the trip as representing the movement of the vehicle rather than the person.
  - **Note:** it's currently not clear whether there are two mode entries for a trip with a passenger – is it captured simply as a "passenger" mode, or both "passenger" and "auto"?
- A question could be raised regarding the relationship between the modes identified in the TASHA data. For example, carpool, rideshare, and passenger modes are likely all assumed to take place via auto. Similarly, we might define a relationship between rideshare, carpool and passenger modes. While it is possible to pursue such a representation, this raises the question: *are these modes classes or individuals?* Currently, in our representation there is nothing we need to say about the *class of auto modes*, the *class of bike modes*, and so on, therefore we opt to maintain a representation where the distinct modes are represented as individual members of the Mode class.

trip_stations.csv

- Observation: assuming we are uploading all of the csv files, there is no need to assert the households and persons as simulation output *each* time – once is sufficient. The same is true for the relationship between households, persons and trips.
- Household_id:
  ->omitted
- Person_id:
  -> omitted
- Trip_id:
  -> transform to create unique identifier:
  return "h" + getValue("household_id") + "p" + getValue("person_id") + "t" + getValue("trip_id")
  -> <trip_id_transform> a trip:Trip.
  omitted participant mapping too, this was also captured by previous data
- Station
  Note: this value represents the initial station "The selected stations are the places where the traveler's vehicle has been left as they switch to using public transit."
  -> transform the id to create an iri
  return "NCS16_centroid_2011_" + getValue("station")
  <station_transform> a spatialloc:Feature.
  <sub_trip1> trip:endLoc <station_transform>.
  <sub_trip2> trip:startLoc <station_transform>.
  Note: "The station zone numbers are references to centroids in the NCS16 standard." See correspondence table
  Q: is there a resource for determining the coordinates of the centroids?
  Note: for now, we assume the centroids used are according to the 2011 system. In addition, we use the supplementary Station Correspondence file to approximate the location of the station (e.g. what zone it is located in).
- Direction: auto2transit or transit2auto
  The mode of these trips (we expect) is "DAT" (drive access to transit), however the direction field provides additional information about the subtrips that comprise the trip. If

the direction is auto2transit, then the first subtrip should be identified as having mode "Auto". Similarly, for transit2auto the second subtrip has the "Auto" mode. To accomplish we transform the direction field into two separate mode attributes:

-> transform <mode1_derived>

d = getValue("direction")

if "auto2" in d:

   return http://ontology.eil.utoronto.ca/icity/TASHA/Auto

-> <sub_trip1> trip:viaMode <mode1_derived>.

-> similar transformation for mode2_derived.

-> <sub_trip2> trip:viaMode <mode2_derived>.

Note that no mode is defined for the transit case as TASHA does not have an IRI for exclusive transit modes.

- Weight: not mapped

facilitate_passenger.csv

- Household_id: omitted (relationship captured in other data sources)
- Passenger_id

  -> transform to unique identifier

  <passenger_id_transform> : return "h" + getValue("household_id") + "p" + getValue("passenger_id")

  -> <passenger_id_transform> a person:PersonPD; change:hasManifestation [a person:Person]

  Note: mapping to household, etc omitted (redundant)
- Passenger_trip_id

  -> transform to create unique trip_id

  -> transform to create unique identifier:

  return "h" + getValue("household_id") + "p" + getValue("passenger_id") + "t" + getValue("passenger_trip_id")

  -> <passenger_trip_id_transform> a trip:Trip.

  revised: create specializations of 'hasParticipant' for trip: hasPassenger and hasDriver

<passenger_trip_id_transform> trip:hasPassenger <_passenger_id@t>; trip:hasDriver <_driver_id@t>

- Driver_id
  -> transform to unique identifier
  <driver_id_transform> : return "h" + getValue("household_id") + "p" + getValue("driver_id")
  -> <driver_id_transform> a person:PersonPD; change:hasManifestation [a person:Person]
- Driver_trip_id
  -> transform to create unique identifier:
  return "h" + getValue("household_id") + "p" + getValue("driver_id") + "t" + getValue("driver_trip_id")
  -> <driver_trip_id_transform> activity:occursDirectlyBefore <passenger_trip_id_transform>
  Note: could add mapping <driver_trip_id_transform> activity:hasParticipant <_driver_id@t> however if our interpretation is correct then this should be redundant with data already defined in trips.csv
- Weight

Notes:

The specification of trip id's isn't entirely clear. On a facilitates_passenger trip, is there a unique trip_id that is generated *only* for the passenger? In other words, without referencing the facilitates_passenger data, is there no way to identify the driver making the trip (and thus we would only find the previous and subsequent trips)? Or, is a trip_id also generated for the driver in trips.csv, so there are two distinct trip_ids generated for the same trip? We assume the former, but it's not clear in the description of the data.

11.2 Future Work

- Determine whether it's possible to retrieve the intended simulation date from the simulation metadata

- Station Correspondence: We should to capture the link between the NCS16 Centroid IDs and the traffic zone they are contained in. This data could be used to identify the station name associated the centroid.
    - Centroid_2016 (or Centroid_2011?)
    - GTA2001Zone
- In future work, the “weight” attribute should be incorporated into the mapping, as it relates to a particular simulation.

Appendix B. Transit Data Mapping

Commented [MK6]: Update to include both Allegrograph & Virtuoso mappings

Overview of datasets:

- Subway delay data
- TTC schedule data (gtfs)
- Vehicle location data (nextbus / other)

Subway & SRT Logs (December 2018)

To do: match the Station and Line values with the identifiers defined in gtfs.

- Date, Time: the date/time of the incident
  Modify the date value to the xsd:date format
  date = getValue("Date")
- return datetime.datetime.strptime(date, '%m/%d/%y').strftime('%Y-%m-%d')

  Combine and transform these values into xsd:DateTimeStamp format
  `return getValue("xsddate") + "T" + getValue("Time") + "-05:00"`
  -> <_incident> a transit:TransitIncident; activity:beginOf [a time:Instant;
  time:inXSDDateTimeStamp <date_time_stamp_transform>]
- Day: the day of week of the incident
  Transform to schema.org day of week format:
- return "http://schema.org/" + getValue("Day")
  -> <_incident> a transit:TransitIncident; activity:beginOf [a time:Instant;
  time:inXSDDateTimeStamp <Day_transform>^^xsd:dateTimeStamp]
- Station
  Note: while station attribute values do match for the stop names specified in the gtfs files, they are defined at a higher level, whereas gtfs specifies multiple stop points for a given station. In the future, there may be a need to automate the formal relationship between the stations and the gtfs stops they contain. It would also be useful to integrate the subway station names with the actual associated location (shape info) of the stations, if available.
  -> <_incident> transit:associatedWithStop _:stopbnode [foaf:name <station>].

- Code
  -> <_incident> transit:hasIncidentCode <code>
- Min Delay: the delay caused by the incident in minutes, or more accurately – the length of the incident in minutes.
  -> <_incident> activity:occursAt [a time:Interval; time:hasDurationDescription [a time:GeneralDurationDescription; time:minutes <Min Delay>]]
- Min Gap: the gap following the incident in minutes. Note that this value represents the resulting gap; i.e. from the time of the delay until after it is resolved and the next train arrives at the station.
  -> <_incident> causedGap [a time:Interval; time:hasDurationDescription [a time:GeneralDurationDescription; time:minutes <Min Delay>]]
- Bound: optional route property?
  -> to do: add boundCardinalDirection as optional property (complimentary to isInbound property for routes)
  -> Future work: the value of this attribute might be applied to identify the involved stop point (e.g. which station platform) more precisely.
- Line: (specialized) route
- Transform Line names into corresponding gtfs route ids:
  - line = getValue("Line")
  - if (line == "YU"):
  -     return "55305"
  - if (line == "BD"):
  -     return "55306"
  - if (line == "SRT"):
  -     return "55307"
  - if (line == "SHP"):
  -     return "55308"
- -> <_incident> associatedWithTrip [a TransitTrip; transit:occursOn [a Route; manifestationOf <line_transform>]].
- <line_transform> a RoutePD.

Commented [MK7]: Q: do gtfs routes distinguish between input and outbound?

- Vehicle: the vehicle involved in the incident
  -> <_incident> associatedWithTrip [a TransitTrip; viaVehicle [a TransitVehicle; hasTransitVehicleId <Vehicle>]

If required, approach to modify values to define new IRIs:
Create new columns for: stationIRI, incidentIRI, transitTripIRI, transitVehicleIRI, intervalIRI, instantIRI

1. Modify station values for creation of stationIRI:
   ```
   x = getValue("Station")
   x=x.replace(" ","")
   x=re.sub('[()]','-',x)
   return x
   ```
2. Generate incident IRI based on station & dates (with ':' omitted); generate trip and vehicle IRIs as a function of the incident IRI, e.g.:
   ```
   datetime = getValue("XSDDateTime")
   datetime=datetime.replace(":","")
   return "incident" + datetime
   ```

AVL Data (TTC NVAS XML Feed)

Provides near-live, historical, and predicted transit vehicle locations.
For the purposes of mapping design, we will use the 501 Streetcar data as example, loading the following into Karma as a web service url:
http://webservices.nextbus.com/service/publicXMLFeed?command=vehicleLocations...
With the additional specification of the t parameter, we can query 15 minutes prior to some point in time (e.g. to retrieve a snapshot of the vehicles' locations shortly after some subway incident). In order to obtain a complete picture, we will need to retrieve the data for all bus routes within the time period of interest.
For example, a subway delay of 40 minutes occurred on May 1, 2018 at 10:05 AM. Therefore to begin examining vehicle locations of interest, we can request data specifying the corresponding t

parameter (latest time of update) by, let's say 10 minutes later (May 1, 2018 at 10:15AM) in epoch time: t=1525184100.

http://webservices.nextbus.com/service/publicXMLFeed?command=vehicleLocations&a=ttc&r=501&t=1525184100000

Note that the XML feed requires the specification of a route (the "r" parameter), therefore in order to retrieve and map the required location data for a particular scenario (e.g. the question of the locations of shuttle buses following a service), we will need identify all route ids of interest (e.g. bus routes) and the time window(s) of interest (e.g. after a subway delay), and script a retrieval of the required data (and its translation into an ontology-base formalism).

Detailed documentation on the feed is available at:
http://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf.The mapping with respect to the output from an arbitrary route is defined below. Note that this may change slightly with the retrieval and storage of NextBus feed data into a database.

- ~~lastTime~~ GPSTime in XML feed collection

- vehicle id
  -> <vehicle_id> a <transit:TransitVehiclePD>; change:hasManifesation [a transit:TransitVehicle>
- vehicle dirTag: omitted
- vehicle heading: omitted
  to discuss: include direction information in ontology? dirTag, heading?
- vehicle lat, vehicle lon
  transform to capture the location in a single field in WKT format
  vehicle_lat_lon_transform:
  return "POINT(" + getValue("lat") + " " + getValue("lon") + ")^^
  <http://www.opengis.net/ont/geosparql#wktLiteral>"
  -> <vehicle_id> change:hasManifestation [a transit:TransitVehicle; hasLocation [a geosparql:Feature; geosparql:hasGeometry [a sf:Point; geosparql:asWKT <vehicle_lat_lon_transform>]]]
  -> artificial blank node required for Feature (Karma bug): <loc_blanknode>
    - return "_:feature" + getValue("id") + "_" + getValue("GPStime")

- predictable

  (omitted)

- routeTag: specifies the route name that the vehicle is on: relationship with vehicle block; *as well as* existence of trip viaVehicle. Note this is distinct from the route id assigned by gtfs.

  -> <_:vehicle@t> transit:onRoute <routeTag>.

  <routeTag> a transit:Route.

  also -> _:transittrip a transit:TransitTrip.

  :transittrip trip:occursOn transit:Route; trip:viaVehicle <_:vehicle@t>

- secsSince Report: can be used in conjunction with lastTime to calculate a timestamp for the location

  transform: GPSTime – secsSince Report => epoch time of vehicle location

  ```
  import time
  diff = int(getValue("secsSinceReport"))
  gpstime= int(getValue("GPStime"))
  t = gpstime - diff
  return str(datetime.datetime.utcfromtimestamp(t).isoformat()) + "Z"
  #assumes the datetime conversion outputs UTC time
  #for local time use .fromtimestamp and append "+05:00"
  ```

  -> transform gpstime to create blank nodes for vehicle and trip intervals <v_interval_blanknode>, <trip_interval_blanknode>:

  return "_:interval_v_" + getValue("GPStime") + getValue("secsSinceReport")

  -> <_:vehicle@t> change:existsAt [a time:Interval; time:inside [a time:Instant; time:inXSDDateTimeStamp <t_transform>]].

  -> <_:transittrip> activity:occursAt [a time:Interval; time:inside [a time:Instant; time:inXSDDateTimeStamp <t_transform>]].

TTC Routes & Schedules (gtfs)

Note: see https://www.nature.com/articles/sdata201889 for an example of processing and filtering gtfs data

Revision to original mapping:

1. an incorrect generation of blank nodes in the mapping was identified. To remedy this, all future revisions to this mapping should generate regular IRIs (not blank nodes) for known but unidentified objects.
2. Required addition of location datatype[37] to accommodate AllegroGraph's geospatial reasoning properties[38].
   Datatype generated from AGraph's N-Dimensional Geospatial Datatype Designer: http://franz.com/ns/allegrograph/5.0/geo/nd#_lat_la_-9.+1_+9.+1_+1.-4_+1.-1_lon_lo_-1.8+2_+1.8+2_+1.-4
   Required format: "<lat><lon>"^^<datatype>

agency.txt

Describes the organization responsible for a particular route.

- agency_id
  -> <agency_id> a org:OrganizationPD
- <org_at_t_iri>
```
from datetime import date
return getValue("agency_id") + "_" +
date.today().strftime("%m-%d-%Y")
```
- agency_name
  -> <agency_id> change:hasManifestation [a org:Organization; foaf:name <agency_name>]
- agency_url
  -> <agency_id_manifestation> icontact:hasWebSite <agency_url>
- agency_timezone
  -> the timezone where the agency is located.
  to do: an Organization has some assocated location (a sf:Feature), and a Feature may be associated with a time:TimeZone. Alternatively, we may also associate timezones with addresses.

---

[37] http://gruff.allegrograph.com:10035/doc/geospatial-nd-tutorial.html#geo-intro

[38] http://gruff.allegrograph.com:10035/doc/magic-properties.html

- agency_lang

  (omitted)
- agency_phone

  -> <agency_id_manifestation> icontact:hasPhoneNumber <agency_phone>
- agency_fare_url

  (omitted)

calendar_dates.txt

Defines exceptions to service definitions in calendar.txt

- service_id

  -> <service_id> a recur:HoursOfOperation
- <date>

  a time:TemporalEntity

  -> <xsd_date> xmodify to xsd format: YYYY-MM-DD

  `s = getValue("date")`
- `return s[0:4] + "-" + s[4:6] + "-" + s[6:8]`
-

  -> <service_id> <exception_type> <date>.

  <date> time:inXSDDate <xsd_date>]
- exception_type: service added or removed (1: added, 2: removed)

  modify to capture implied property

  -> 1: recur:recursAddition

  -> 2: recur:recursExcept
- `s = getValue("exception_type")`
- `if ((int)(s) == 1):`
- `    return "http://ontology.eil.utoronto.ca/icity/RecurringEvent/recursAddition"`
- `if ((int)(s) == 2):`

- ```
      return
  "http://ontology.eil.utoronto.ca/icity/RecurringEvent/recur
  sExcept"
  ```

calendar.txt

Defines dates for service availability; a weekly recurring event(s).

- service_id
  -> <service_id> a recur:HoursOfOperation
- Monday:
  -> modify if 1: <service_id> schema:dayOfWeek schema:Monday
  if (int)(getValue("monday"))==1 :
- return "http://schema.org/Monday"
- Tuesday:
  -> if 2: <service_id> schema:dayOfWeek schema:Tuesday
- wednesday,thursday,friday,saturday,Sunday:
  -> as above
- <start_date> a time:Instant
  -><start_date_xsd> modify to xsd format: YYYY-MM-DD
  s = getValue("start_date")
- return s[0:4] + "-" + s[4:6] + "-" + s[6:8]
  -> <service_id> recur:beginsRecurring <start_date>.
  <start_date> time:Instant; time:inXSDDate <start_date>.
- <end_date> a time:Instant
  -> <end_date_xsd> modify to xsd format: YYYY-MM-DD
  s = getValue("end_date")
- return s[0:4] + "-" + s[4:6] + "-" + s[6:8]
  -> <service_id> recur:endsRecurring <end_date>
  <end_date> time:inXSDDate <end_date>.

routes.txt

- route_id
  -> <route_id> a transit:RoutePD
  ```

- <route_manifestation_id>:

```
from datetime import date
return getValue("route_id") + "_" +
date.today().strftime("%m-%d-%Y")
```

- agency_id
- <transit_system_iri>:

  ```
  return "transitsystem_" + getValue("agency_id")
  ```

  -> <route_id> transit:inTransitSystem <transit_system_iri>.
- <transit_system_iri> a transit:TransitSystem; transit:operatedBy <agency_id>.
- route_short_name

  -> <route_id> change:hasManifestation <route_manifestation_iri>.

  <route_manifestation_iri> a transit:Route; transit:routeShortName <route_short_name>.
- route_long_name

  -> <route_id> change:hasManifestation <route_manifestation_iri>.
- <route_manifestation_iri> a transit:Route; foaf:name <route_long_name>.
- route_desc (not filled)
- route_type

  -> <route_id> transit:hasGTFSRouteType <route_type>
- route_url (not filled)
- route_color

  -> <route_id> change:hasManifestation <route_manifestation_iri>.
- <route_manifestation_iri> a transit:Route; transit:hasDisplayColour <route_color>.
- route_text_color

  -> <route_id> change:hasManifestation <route_manifestation_iri>.
- <route_manifestation_iri> a transit:Route; transit:hasRouteTextColour <route_text_color>.

shapes.txt

Describes the shape and location of a particular trip.

- shape_id

  -> <shape_id> a spatial_loc:LineString

- ~~shape_pt_lat,shape_pt_lon,shape_pt_sequence~~
  ~~-> need to transform the set of latitudes, longitudes, and sequence identifiers to create a series of points to specify the coordinates of the linestring in the WKT format, e.g. "LINESTRING(lat1 lon1, lat2 lon2,…)~~
  ~~Two models to design and apply:~~
  ~~1 of 2: capture the line string represented by the aggregation of all of the points for a particular shape[39]~~
  ~~2 of 2: capture the individual line segments (and their lengths) as part of (sf:contains) the line string defined by <shape_id>~~
- shape_pt_lat,shape_pt_lon,shape_pt_sequence:
  Shapes are not supported by Allegrograph, therefore this transformation need only capture the individual points as lat/lon coordinates
- The geospatial encoding used by Allegrograph requires the longitude values to have 3 digits before the decimal place, meaning any 2-digit coordinates will need to be padded with a 0. Note that the zfill function doesn't work in this case because we are only concerned with the number of digits preceding the decimal place.

  <shape_pt_nD_transform> (specified nD datatype http://franz.com/ns/allegrograph/5.0/geo/nd#_lat_la_-9.+1_+9.+1_+1.-4_+1.-1_lon_lo_-1.8+2_+1.8+2_+1.-4 note that specification shouldn't be necessary when AGraph on autorecognize):

```
lon = float(getValue("shape_pt_lon"))
lat = float(getValue("shape_pt_lat"))

lon_str = str(abs(lon))
lat_str = str(abs(lat))

lon_str = lon_str.split(".")
```

[39] Note: this requires the use of the group by function, it's not clear whether the mapping file will also capture the grouping transformation, or if the mapping function will need to be applied to a "grouped" input file. We may need to transform the file a priori for automated mappings.

```
        lon_str = "%s.%s" % (lon_str[0].zfill(3), lon_str[1])

        lat_str = lat_str.split(".")
        lat_str = "%s.%s" % (lat_str[0].zfill(2), lat_str[1])

        if lon>0:
            lon_str="+" + lon_str
        elif lon<0:
            lon_str="-" + lon_str

        if lat>0:
            lat_str="+" + lat_str
        elif lat<0:
            lat_str="-" + lat_str
        return lat_str + lon_str
```

Need to introduce a new data property to relate nD data points to individual geometries. Analogous to the "asWKT" property

<point_iri> as_nDLatLon <shape_pt_nD_transform>.

- shape_dist_traveled: omitted
- point_iri: unique ID for point object

- line segment points
  transform lat and lon columns to WKT point format:
  - return "POINT(" + getValue("shape_pt_lon") + " " +
  getValue("shape_pt_lat") + ")"
  set semantic type datatype: http://www.opengis.net/ont/geosparql#wktLiteral
  <shape_id> a geo:Geometry; sf:contains <point_iri>.
  <point_iri> a sf:Geometry; as_nDLatLon <shape_pt_nD_transform>.

Future work: In order to define the associated distance between points in the shape, we need to reference the points across rows (i.e. using the <shape_pt_sequence> attribute).

- *A few open questions must first be resolved*:

- How can we create a mapping between rows in Karma, without modifying the input file?
- Does the distance represent a (straight) line segment, or possibly a curved line string with multiple intermediate points but only a known beginning and end?
- What is the most appropriate property to capture the distance travelled, and where should it be defined?

stop_times.txt

Two key relationships captured in this dataset are not only between the attributes in the csv file but between the rows, according to the ordering: the arrival and departure times, and the stop_ids need to be referenced across rows in order to better capture the route links and times of the scheduled trip segments. R2RML tools are not suited for this, therefore we opt to pre-process the stop_times file to create new attributes that capture the attribute values of interest from the following row. In this case, we are interested in stop_id and arrival_time, so the result of the preprocessing is two new attributes: next_stop_id and next_arrival_time.
This preprocessing was performed using pandas (run stop_times_preprocess.py with python3).

- trip_id
- <subtrip_iri>:
- `return "subtrip_" + getValue("trip_id") + "_" + getValue("arrival_time").replace(":","") + "_" + getValue("stop_id") + "_" + getValue("next_stop_id")`
  -> <trip_id> a trip:ScheduledTransitTrip; activity:hasSubRecurringEvent <subtrip_iri>.
- <subtrip_iri> a trip:ScheduledTransitTrip.
- arrival_time: not used
- departure_time, next_arrival_time: the scheduled departure time from the current stop and arrival time at the next stop. *note need to ensure the format is correct (xsd:time)
  -> <_scheduled_subtrip > rec:startTime <departure_time>; rec:endTime <next_arrival_time>
- stop_id, next_stop_id: the origin of the trip segment
  <route_iri>:

- `return "route" + getValue("trip_id") + "_" + getValue("stop_id") + "_" + getValue("next_stop_id")`
- -> <subtrip_iri > transit:scheduledOn <route_iri>.
- <route_iri> a transit:RouteSection; transit:beginsAtStop <stop_id>; transit:endsAtStop <next_stop_id>
- stop_sequence: not used by the mapping but could be used for preprocessing to ensure correct values for next stops and next arrival times
- stop_headsign: not in use
- pickup_type: indicates whether passengers are picked up at the stop, and if so how the pickup must be arranged.
  transform pickup_type into appropriate IRI
  ```
      type = getValue("pickup_type")
      if (type == "0"):
        return "http://ontology.eil.utoronto.ca/icity/PublicTransit/AccessAsScheduled"
      if (type == "1"):
        return "http://ontology.eil.utoronto.ca/icity/PublicTransit/AccessNotAvailable"
      if (type == "2"):
        return
      "http://ontology.eil.utoronto.ca/icity/PublicTransit/AccessArrangedViaAgency"
      if (type == "3"):
        return
      "http://ontology.eil.utoronto.ca/icity/PublicTransit/AccessArrangedViaDriver"
      -> <_scheduled_ subtrip> transit:hasPickupType <pickup_type_transform>
  ```
- drop_off_type: indicates whether passengers are dropped off at the stop, and if so how the pickup must be arranged.
  transform dropoff_type into appropriate IRI
  as above with pickup type
- shape_dist_traveled: cumulative distance (from start of trip_id)
  omit for now: TBD what the best way to represent this is

stops.txt

- stop_id
  -> <stop_id> a transit:StopPoint
- stop_code
  -> <stop_id> transit:hasStopCode <stop_code>
- stop_name
  -> <stop_id> foaf:name <stop_name>
- stop_desc (not filled)
- stop_lat,stop_lon: transform into WKT format:
  -> <stop_lon_lat_wkt>: return "POINT(" + getValue("stop_lon") + " " + getValue("stop_lat") + ")^^<http://www.opengis.net/ont/geosparql#wktLiteral>"
  -> <stop_id> spatial_loc:hasLocation [a spatial_loc:Feature; spatial_loc:hasGeometry [a spatial_loc:Geometry; asWKT <stop_lon_lat_wkt>]]
  manually add blank node for Feature class (karma workaround)
  return "_:stoplocfeature" + getValue("stop_id")

  transform into nD format for allegrograph:
  <nD_transform>

```
lon = float(getValue("stop_lon"))
lat = float(getValue("stop_lat"))

lon_str = str(abs(lon))
lat_str = str(abs(lat))

lon_str = lon_str.split(".")
lon_str = "%s.%s" % (lon_str[0].zfill(3), lon_str[1])

lat_str = lat_str.split(".")
lat_str = "%s.%s" % (lat_str[0].zfill(2), lat_str[1])

if lon>0:
```

```
    lon_str="+" + lon_str
elif lon<0:
    lon_str="-" + lon_str

if lat>0:
    lat_str="+" + lat_str
elif lat<0:
    lat_str="-" + lat_str
return lat_str + lon_str
```

- <nD_transform>^^<agraph datatype>
- zone_id (not filled)
- stop_url (not filled)
- location_type (not filled)
- parent_station (not filled)

  wheelchair_boarding

  need to transform numeric value into xsd:Boolean:

```
wb = getValue("wheelchair_boarding")
if (wb == '1'):
    return 'true'
if (wb == '2'):
    return 'false'
```

  -> <stop_id> transit:wheelchairBoarding <wheelchair_boarding_transform>

trips.txt

- route_id

  -> <route_id> a transit:RoutePD; change:hasManifestation <_route_manifest>

  manual blank node for route@t:

  return "_:route" + getValue("route_id") + "_" + getValue("trip_id")
- service_id: identifies the days when service is available for a particular route. A route may have multiple service_ids defined, and each trip has a single service_id that represents the days during which the trip is provided. The properties of the service_id may be used to define the recurring days for the <trip_id>. In order to capture this, we'll

need to merge the appropriate values for each service_id from the calendar.txt dataset. OR: in lieu of merging the csv files, we define <service_id> as another RecurringEvent and define <service_id> rec:hasSubRecurringEvent <trip_id>. In other words, during this abstract event that recurs on some days of the week, the scheduled trip also occurs. This is a correct representation but is it overly complex (e.g. to query for a schedule?)
-> <service_id> a contact:HoursOfOperation.
<_route_manifest> icontact:hasOperatingHours <service_id>.
-> <service_id> rec:hasSubRecurringEvent<trip>.
<trip> a transit:ScheduledTransitTrip;
The service may also be formalized as the Hours of Operation associated with the Route (as below).
~~A route has some hours of operation, described by the service_id. Hours of operation are a kind of recurring event, thus representing the recurrence of a service being operational (or business being open).~~
~~-> <route_id> change:hasManifestation [a transit:Route; icontact:hasOperatingHours <service_id>~~

- trip_id
  -> <trip_id> a transit: ; transit:scheduledOn <route_id>]
- trip_headsign
  -> <trip_id> foaf:name <trip_headsign>
- trip_short_name (not filled)
- direction_id: inbound (1) vs outbound (0) (based on suggested values from documentation
  -> if 1: <trip_id> transit:isOutbound “true”
  -> if 0: <trip_id> transit:isOutbound “false”
  <trip_id> transit:isOutbound <direction_id_transform>
  <direction_id_transform>:

```
      d = getValue("direction_id")
      if (d=="1"):
          return "true"
      if (d=="0"):
```

return "false"

- block_id
  -> <block_id> a transit:VehicleBlock; transit:assignedFor <trip_id>; assignedTo [a transit:TransitVehicle]
- shape_id: geometry representing the location of the scheduled trip; note that the shape defines the path of the scheduled trip rather than the route because the route is more general so it may include trips of slightly different shapes.
  -> <trip_id> hasLocation [a spatialloc:Feature; spatialloc:hasGeometry <shape_id>]
  add manual blank node for Feature:
  return "_:feature" + getValue("trip_id") + "_" + getValue("shape_id")
- wheelchair_accessible
  a property of the vehicle performing the trip – but any number of vehicles could perform the scheduled trip, therefore this is a property of the scheduled trip (that restricts vehicle assignment) rather than of an *individual* vehicle that performs an occurrence of the trip.
  -> <trip_id> transit:isWheelchairAccessible <wheelchair_accessible_transform>
  0: not specified,
  1: accommodation for at least one wheelchair
  <trip_id> transit:isWheelchairAccessible "true"
  2: no accommodation for wheelchair riders
  <trip_id> transit:isWheelchairAccessible "false"
    - wheelchair_accessible_transform:
      w = getValue("wheelchair_accessible")
    - if (w=="1"):
    - return "true"
    - if (w=="2"):
    - return "false"
- bikes_allowed (not filled)
  may be addressed similar to wheelchair_accessible

Appendix C. Loop Detector Data Mapping

The loop detector data was received in a simple, tabular format with the following column headings: WayID, Mean_Value_Max, Time, and Date. In fact, this data set was an excellent example of the challenges for semantic interoperability: communication with the persons responsible for generating the data set was required in order to understand the meaning of each of the attributes. This revealed the following, informal semantics for each attribute:

- WayID: this value is the identifier of the road segment over which the loop detector reading is aggregated.
- Mean_Value_Max: this value is the average Max TTI (Maximum Travel Time Index), aggregated over the wayID readings, at one-hour intervals.
- Time: this value indicates the time of day of the start of the one-hour interval, represented using an integer value that indicates hours past midnight. For example, “0” indicates 12:00 AM, “1” indicates “1:00 AM”, and so on.
- Date: this value represents the date during which the readings were taken, formatted as: year_month_day. For example, the value “017_07_01” indicates the date July 1, 2017.

Much of the information that is embedded in these values is not clear from the attribute labels alone. In order to enable interoperability, the semantics of these values must be made explicit. As described in the previous section, this is done with mappings expressed in R2RML. Which ontology(s) is used in the mappings depends on the scope of the concepts represented in the data. Key concepts that we can recognize from the data are the notion of road segments, time, and measures.

In what follows, we describe the mappings that were defined for each attribute, with respect to the iCity TPSO, in order to support the data transformation. In particular, the Observations[40] and Transportation System[41] Ontologies play a key role. As mentioned in Section 9.3.3, some reformatting of the values into standard datatypes was also required.

- The WayID value was defined as an individual member of the RoadSegment class, as defined in the Transportation System Ontology from the TPSO.

[40] http://ontology.eil.utoronto.ca/icity/Observations

[41] http://ontology.eil.utoronto.ca/icity/TransportationSystem

- The RoadSegment class represents part of a particular Road that makes up the physical infrastructure of the transportation system.
- The Mean_Value_Max value was defined as the value of the numerical_value property of a Measure that is the value of a MeanTTI_Max Quantity. The concepts of a numerical_value, a Measure, and a Quantity are defined in the Units of Measure Ontology from the TPSO, whereas the MeanTTI_Max specialization of a Quantity is defined in the Transportation System Ontology. More specifically, the MeanTTI_Max is captured as an aggregate of a TTI_Max Quantity, that is the aggregate of a TTI_Max value over a particular RoadSegment (the WayID), that is aggregated over a particular Interval in time.
- The concept of an interval is introduced in the Time Ontology, and the Interval itself is captured using the transformed Time and Date values. The resulting value (an xsd:dateTimeStamp, as discussed above) provides a value for the start time of the Interval that the TTI_Max is aggregated over, while a value one-hour later provides the end time for the Interval.

This mapping is illustrated in Figure 22. The KARMA mapping files are available online in the GitHub project repository at:
https://github.com/EnterpriseIntegrationLab/icity/tree/master/mappings/ITSoS

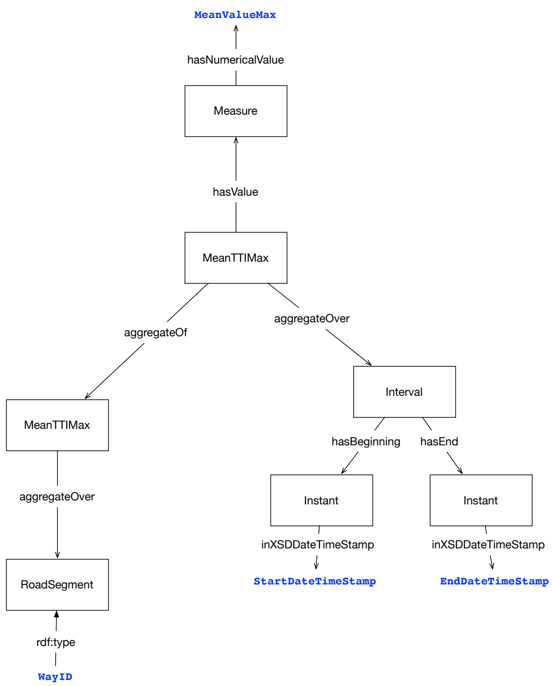


Figure 22: Mapping the data values into TPSO concepts. Data values are depicted in boldface blue and the ontology concepts are illustrated with the lines and rectangles.

Appendix D. Esri GFX Data Mapping

GFX tables used:

- Neighbourhood
- Land Use
- Land Cover
- Point of Interest
- Road Segment

Additional tables computed based on spatial relationships:

- Intersect Neighbourhood
- Near Land Use
- Near Land Cover
- Near Point of Interest

Esri Extension of TPSO

- New class: Neighbourhood subClassOf Parcel
  - subClassOf hasLandUse value 'NeighborhoodOrLocalPark'
  - subClassOf foaf:name min 1 xsd:string
- New class: GFXLandUseClassification subClassOf LandUseClassification
  - Subclasses may be defined according to the value of hasGFXLandUseCode:

GFXLandUseClassification	hasGFXLandUseCode
General / Residential	1
Government	2
Medical	3
Education	4
Transportation	5
Commercial	6
Religious	7
Recreation	8
Cultural / Heritage	9
Hotel	10
Airport	11
Industrial	12
Community Centre	13
Agricultural	14
Energy	15
Banking and Finance	16
Mail and Shipping	17
Weather	18

Water Supply and Treatment	19
Information and Communication	20
Other	

- New Class: GFXLandCover subClassOf LandUseClassification
    - Subclasses of GFXLandCover may be defined according to the value of hasGFXLandCoverCode:

0	No data	50	Shrubland	121	Annual Cropland
10	Unclassified	51	Shrub tall	122	Perennial Cropland and Pasture
11	Cloud	52	Shrub low	200	Forest/Tree classes
12	Shadow	53	Prostrate dwarf shrub	210	Coniferous Forest
20	Water	80	Wetland	211	Coniferous Dense
21	Beach	81	Wetland - Treed	212	Coniferous Open
30	Barren/Non-vegetated	82	Wetland - Shrub	213	Coniferous Sparse
31	Snow/Ice	83	Wetland - Herb	220	Deciduous Forest
32	Rock/Rubble	100	Herb	221	Broadleaf Dense
33	Exposed land	101	Tussock graminoid tundra	222	Broadleaf Open
34	Developed	102	Wet sedge	223	Broadleaf Sparse
35	Sparsely vegetated bedrock	103	Moist to dry nontussock graminoid/dwarf shrub tundra	230	Mixed Forest
36	Sparsely vegetated till-conluvium	104	Dry graminoid prostrate dwarf shrub tundra	231	Mixedwood Dense
37	Bare soil with cryptogam crust - frost boils	110	Grassland	232	Mixedwood Open
40	Bryoids	120	Cultivated Agricultural Land	233	Mixedwood Sparse

- New Class: PointOfInterest
    - PointOfInterest subclassof locatedOnParcel min 1 Parcel
    - Subclasses of POI may be defined according to the value of hasGFXPOIClassCode

1	General	16	Banking and Finance
2	Government	17	Mail and Shipping
3	Medical	18	Weather
4	Education	19	Water Supply and Treatment
5	Transportation	20	Information and Communication

6	Commercial	21	Settlement
7	Religious	22	Natural
8	Recreation	99	Other
9	Cultural / Heritage		
10	Hotel		
11	Airport		
12	Industrial		
13	Community Centre		
14	Agricultural		
15	Energy		

- New Class: GFXRoadSegment subclassOf TransportationComplex (not quite icity:RoadSegment because it also includes other modes of transport)
    - Subclassof hasRoadLevel exactly 1 RoadLevel
    - subclasses of RoadSegment may be defined according to the value of hasGFXRoadClassCode:

1	Freeway	11	Rapid Transit
2	Expressway / Highway	12	Service Lane
3	Arterial	13	Winter
4	Collector	14	Major Arterial
5	Local / Street	15	Minor Arterial
6	Local / Strata	16	Recreation
7	Local / Unknown	17	Resource
8	Alleyway / Lane	18	Lane
9	Ramp	19	Alleyway
10	Resource / Recreation	20	Local
21	4WD	22	Ferry
23	Farm	24	Freeway Ramp
25	Highway Ramp	26	Major Arterial Ramp
27	Minor Arterial Ram	28	Collector Ramp
29	Local Ramp	99	Other

- New object property: hasRoadLevel some RoadLevel
- Subclasses of RoadLevel definable wrt hasGFXRoadLevelCode values

Grade Level – Above or below

0	Ground Level

1	First Level
2	Second Level
3	Third Level
4	Fourth Level
-1	Subsurface
-2	Second Subsurface

- New object property: routeNear
    - Possibly definable wrt geo:within between locations of objects?
- New object property: routeIntersects
    - Can define wrt geo:intersects between locations of objects

12 Mappings from tables to iCity TPSO Esri Extension

Neighbourhood (neighbourhood_mun)

- Feature hash
- {feature hash} a Neighbourhood;
- Name
- {feature hash} foaf:name {Name}
- *Province or Territory*
- *Provider type*
- *Data source*
- *Creation date*
- *Revision date*

Land Use (landuse_mun)

SELECT feature_hash, lu_class, name1, desc_english from landuse_mun

Ontop mapping:
{feature hash} a Parcel;
hasLandUse [a GFXLandUseClassification;

hasGFXLandUseCode {code}

foaf:name {name 1}

Land Cover (landcover_mun)

- Feature hash
- {Feature hash} a Parcel
- Coverage type

- {Feature hash} hasLandUseClassification [a GFXLandCoverClassification; hasGFXLandCoverClassCode {code}]

Point of Interest (pointofinterest_mun)

- Feature hash
- {Feature hash} a Parcel;
- Contains / locatedOn- [a PointOfInterest; hasGFXPOIClassCode {class}; foaf:name {Name 1}]
- Point of Interest Class
- *POI type*
- Name 1

Commented [MK8]: TBD: relationship between a Parcel and some Building / Facility / etc. Define POI class?

Commented [MK9]: Next stage: can we categorize the Point of interest type according to the point of interest class?

Road Segment (roadsegment_mun)

- Feature hash
- {Feature hash} a RoadSegment;
- ~~isAboveGrade~~ roadLevelCode {Above of below grade};
- foaf:name {Official Street Name};
- hasGFXRoadClassCode {Road Class};
- hasLength [a length; measure [a Measure; numerical_value {Length}]]
- Above or below grade
- Official Street Name
- Road Class

Intersect Neighbourhood (generated via ArcGIS process)

- Roadsegment (feature hash)
- Neighbourhood feature hash

Near Land Use (generated via ArcGIS process)

- Roadsegment (feature hash)
- Parcel with some land use feature hash

Near Land Cover (generated via ArcGIS process)

- Roadsegment (feature hash)
- Parcel with some Land Cover feature hash

Near POI (generated via ArcGIS process)

- Roadsegment (feature hash)
- POI locatedOnParcel with some Parcel feature hash