Preprint Cambridge Centre for Computational Chemical Engineering ISSN 1473 – 4273

Semantic 3D City Inferences - multi-domain reasoning on Dynamic Geospatial Knowledge Graphs

Arkadiusz Chadzynski [1], Heidi Silvennoinen [2], Ayda Grišiūtė [2], Feroz Farazi [3], Sebastian Mosbach [1,3], Martin Raubal [2,4], Pieter Herthogs [2], Markus Kraft [1,2,5,6]

released: 15 February 2023

[1] CARES
Cambridge Centre for Advanced Research and Education in Singapore
1 Create Way
CREATE Tower, #05-05
Singapore, 138602

[2] Singapore-ETH Centre,
Future Cities Lab Global Programme,
CREATE campus,
1 Create Way,
#06-01 CREATE Tower,
Singapore, 138602

[3] Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

[4] Institute of Cartography and Geoinformation
Department of Civil, Environmental and Geomatic Engineering
ETH Zurich
Stefano-Franscini-Platz 5
8093 Zurich
Switzerland

[5] School of Chemical and Biomedical Engineering
Nanyang Technological University
62 Nanyang Drive
Singapore, 637459

[6] The Alan Turing Institute
2QR, John Dodson House
96 Euston Road
London, NW1 2DB
United Kingdom

Preprint No. 303

UNIVERSITY OF CAMBRIDGE

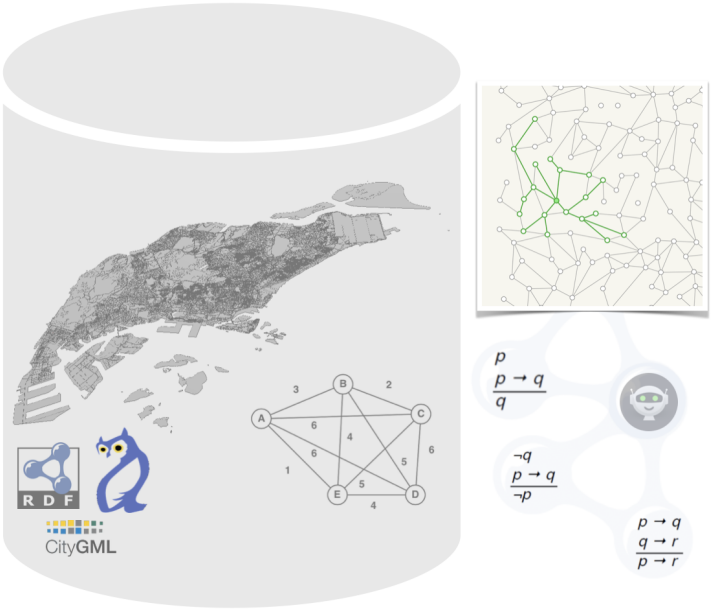
Keywords: Sustainability, Digitisation, Urban Planning, Cognitive Architecture, Artificial Intelligence, Knowledge Graph, Automation, City Modelling, Geospatial, CityGML, Semantic Web, Ontology

Edited by
Computational Modelling Group
Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

E-Mail: mk306@cam.ac.uk
World Wide Web: https://como.ceb.cam.ac.uk/

Abstract

This paper presents revised novel semantic web systems reference architecture for inferences and components that can store and operate on knowledge in the form of a fully dynamic graph to infer new statements by an intelligent autonomous agent capable of making informed choices based on long-term memories about its tasks that implement inference algorithms of all currently known classes. An *Owlconverter* tool was designed and developed as a new component which can produce fully dynamic knowledge graphs without information loss that otherwise occurs while attempting to store complex concept definitions in existing open-source dynamic RDF stores. An *Inference Agent* with extended cognitive capabilities of making informed choices based on long-term knowledge was designed and developed to act as an extended inference engine supporting all currently known classes of knowledge graphs inference algorithms. This capability is supported by the newly developed *OntoInfer* ontology that encodes the taxonomy of those algorithms linked to instances of the agent's tasks allowing the agent to make choices based on the knowledge stored in the knowledge graph. This extended architecture can demonstrate the implementation of tasks designed to work as independently executed threads containing examples of known inference algorithms using existing libraries and reasoning engines (Jena Jung and HermiT). Multi-domain reasoning capabilities on city object descriptions in terms of *OntoCityGML*, *OntoZoning* and *OntoBuildableSpace* were showcased on plot data provided by Urban Redevelopment Authority of Singapore converted into OWL 2 compliant knowledge base.



Highlights

- Revised novel semantic web systems architecture for inferences.
- Inference Agent capable of deriving new knowledge by applying graph and ontology based inference algorithms.
- OntoInfer ontology that encodes taxonomy of existing knowledge graph reasoning algorithms linked to the agent's tasks.

Contents

1 Introduction **3**

2 Dynamic Geospatial Knowledge Graphs **5**

3 Multi-domain Reasoning on Dynamic Geospatial Knowledge Graphs **10**

3.1 Knowledge Graph Inference Algorithms 12

3.2 Cognitive Capabilities of Inference Agent 13

3.2.1 Graph Based Inferences 16

3.2.2 Taxonomic Reasoning 20

4 Research Summary and Future Work **23**

A Inference Agent - UML Activity Diagram **27**

B OntoInfer ontology expressed in a DL syntax **28**

References **31**

1 Introduction

General context and problem space

Cities, taken as complex macro structures that consist of many heterogeneous subsystems of different scales, are widely recognised as one of the largest contributors to the global net positive CO_2 emissions causing global warming that results in climate change of the entire planet [75]. Many coastal cities face its effects in the form of rising sea levels as an *existential threat* [55] acting as a *crisis multiplier* [84] at a country level. At the same time, standard-based digitisation [87] is seen by the global governing bodies, such as the UN, G20, and the World Bank [66], as a means to achieve comparability [45] and data interoperability [15] at a global scale. Such efforts should eventually deepen the understanding of those complex structures and optimise [62] their designs [54] to minimise any adverse effects and get closer to the discovery of solutions [6] mitigating this 'biggest threat ever faced by modern humanity as a whole' [84].

Semantic Web Technologies (SWTs) [85] and Dynamic Geospatial Knowledge Graphs (DGKGs) [18] that implement them and adhere to the other well established geospatial standards at the same time are believed to be a modern technological answer to facilitate such interoperability using sustainable digitisation practices [83]. Cities Knowledge Graph (CKG) [22], built upon the Semantic 3D City Database [18], Cognitive Agents System Architecture [20], and GeoWeb interfaces [19], is a working prototype of such information systems. It is capable of storing city-related knowledge, spanning multiple domains of interest [76] and relating to each other, analysing it and discovering new facts as well as providing collaborative interfaces [36] that allow to blend human and artificial intelligence [42], amongst others.

As a part of The World Avatar (TWA) [52], a more general dynamic knowledge graph designed for multi-domain representation of knowledge related to the entire world, the system can take advantage of existing tools and frameworks as well as reuse knowledge of the micro-scale structures [90] that could be found as subsystems of cities. For instance, one of the critical TWA components, the J-Park Simulator (JPS) [27], is designed to work with representations of built environments to simulate emissions dispersion from various types of air pollution sources. It was already used for research on optimising designs of Eco-Industrial Parks (EIPs) [89] with respect to their carbon footprint [57], optimal site selection of modular nuclear power plants [26], simulations of chemical kinetic reaction mechanisms [29], quantum chemistry calculations [49] and combustion chemistry simulations [30]. A Parallel World Framework of TWA [28], designed for complex scenario analysis, could be utilised for multi-domain optimisation simulations. In summary, tools providing functionalities of analysis, simulation, and presentation [14] as well as the overall design orientation towards providing a semantic representation of anything that could be conceptualised, allow calling such an information system a Universal Digital Twin (UDT) [73].

Gaps in the current system architectures

Augmentation of existing CityGML [35] data transformation tools [88] allowed to base TWA on a dynamic data source - Semantic 3D City Database. Apart from improving data interoperability [81] further than relational database management systems [77] and en-

abling CityGML models for concurrency, this architectural component allows to perform simulations on City Information Models (CIMs) [32] and adds flexibly to hot swap certain information in city object representations. The addition of a layer of autonomous agents [72] implemented upon cognitive architecture principles [51] of JPS Agent Framework solved problems of large CIM creation, analysis, and visualisation [3], that were mostly manual and error-prone in more traditional information systems adopting this modelling standard [16]. GeoWeb interfaces [71] that allow for collaborative blending of artificial and human intelligence move TWA even further away from traditional GIS [31]. At the same time, semantic foundations of the overall architecture minimise risks of deliberate misinformation on geospatial web systems [34] that are very often used for modelling critical infrastructures.

Although the Semantic 3D City Database, built on Blazegraph™ [7], has already inherited support for the automated inference engine [8], there are limits on its capabilities in the case of DGKGs. First, the Blazegraph's™ engine does not support inferences with knowledge structures utilising named graphs [37]. Being able to store knowledge about separate objects in named graphs is essential for large knowledge bases because of the possibility to easily limit query scopes only to the objects of interest and improve performance [5]. Named graphs are also required to work with Object Graph Mapper (OGM) engine [19] - an Object Relational Mapper (ORM) [53] equivalent for graph databases that allows to automate the instantiation of objects stored in the knowledge base into Java objects as well as automation of data persistence related routines, like inserts, updates and deletes. Moreover, the existing inference engine materialises all the newly inferred statements as new triples. That leads to growing the knowledge base quickly [1] and, without any invalidation mechanisms in place, may lead to incorrect inferences in case of ontology changes [40]. Lastly, the off-the-shelf engine is limited to RDFS+ [2] inferences and does not support graph inference algorithms.

HermiT reasoner [33] supports Web Ontology Language (OWL) 2 [38] ontologies and was identified as a good candidate for the replacement of the default inference engine within TWA. It is based on a novel hypertableau reasoning algorithm and outperforms other reasoning engines in taxonomic reasoning tasks on benchmarks [13]. However, the reasoner was originally designed to work with static files via OWLAPI interfaces [86]. In its publicly available codebase [39], there are no examples of how to make it work with RDF triple stores and, in this way, use it as an inference engine in more novel types of semantic web system architectures. Therefore additional tools development is required to use it in such cases. Moreover, HermiT also does not support graph inference algorithms that are one of the knowledge graph algorithms [82] needed for DGKGs to provide full coverage of different types of inference techniques. However, it can be complemented by a set of separate inference algorithms integrated into one engine accessible via uniform interfaces. Results of gap analysis of existing semantic web knowledge graph systems show that it is possible to design a modern semantic web system as capable of providing a variety of known inference algorithms via a uniform interface by integration of ontology inference engines and graph inference specific software libraries to even enhance novelty of existing reference architectures, like the one presented by Allemang and Hendler [1]. Identifying such gaps in system integration is worthwhile as it can be regarded as one of the ways of measuring costs in the form of the lost opportunity of not doing integration [65].

The purpose of this paper is to present a revised novel reference architecture for DGKGs that integrates existing software components capable of applying known inference algorithms via a uniform interface. Implementation of this layer in TWA required the design and development of new software components that are also presented in this paper. This revised inference layer is also portable to a wider range of dynamic knowledge graphs based on semantic web standards, including those without geospatial capabilities. Presented reference architecture and new integration components aim to close the gaps in currently known semantic web-based knowledge graph systems and DGKGs in particular.

A general description of DGKGs and their main components is presented in Section 2. The components needed to support above mentioned capabilities in the revised novel reference architecture are presented in Section 3. Subsection 3.1 includes a more detailed explanation of currently known classes of knowledge graph algorithms. Subsection 3.2 presents an Inference Agent designed and developed to support inferences via two broad classes of currently known algorithms. Agent's tasks that implement sample graph-based inference algorithms are presented in subsection 3.2.1 whereas taxonomic reasoning-specific tasks are presented in subsection 3.2.2. The final section (Section 4) includes closing remarks and future research outlooks.

2 Dynamic Geospatial Knowledge Graphs

At present, there are no well-established and widely agreed upon definitions of Dynamic Geospatial Knowledge Graphs (DGKGs). For the purpose of this paper, Knowledge Graphs (KGs) are defined as knowledge bases [4] of a graph structure. Geospatial Knowledge Graphs (GKGs) are KGs with built-in interfaces that can store geospatial information and search for any objects described in terms of geospatial coordinates within certain regions [48]. Dynamic Knowledge Graphs (DKGs) are KGs with dynamic data storage that allows for concurrent adding, retrieving, updating and deleting of represented knowledge [43]. DGKGs are KGs that are GKGs and DKGs at the same time. An additional advantage of DGKGs that contain knowledge about objects described in terms of existing standards is they can provide comparability without additional middleware and, because of that, improved data interoperability. In general, there are two classes of standards that are applicable to the content of DGKGs: geospatial standards and knowledge representation standards.

TWA is a working prototype of DGKG that contains knowledge representations adhering to both classes of standards. It is a KG adhering to the Resource Description Framework (RDF) standard [67], at the most fundamental level. All knowledge is stored in the form of **S**ubject-**P**redicate-**O**bject (SPO) statements that link to one another, forming a graph of Linked Data [9]. OWL 2, that is fully representable in RDF [59], is used as a more specific knowledge representation standard. Both standards were originally conceptualised when dynamic RDF graph stores were unavailable, and the most commonly used data storage and exchange format was a static file. As a data exchange format, it is inefficient as any change to such a knowledge base requires the transfer, replacement and reload of the whole file any time such change occurs within it. It is also unsuitable in distributed and

collaborative environments because it is impossible to open and change such a knowledge base concurrently [64].

In terms of the history of science and scientific knowledge representation, KGs founded upon static files would be most suitable to store knowledge about Platonic ideas. In ancient times mathematics and logic were given the highest status and were regarded as pretty much complete and thought to be treating of a realm of eternal and ideal objects accessible intellectually. Plato regarded everyday phenomena as a mere illusion and reflection of those ideal objects in his famous Allegory of the Cave [63]. They seemed not to belong to science, which was meant to be concerned only with that what is true and not with ephemeral illusions. More modern times witnessed the marriage between science and engineering, forming closer ties between theory and practice. Scientists started to be motivated by solving practical problems of their time, and mathematical methods were regarded as most suitable to describe everyday phenomena and discover the laws governing them behind the scenes [25]. Ancient views on the nature of mathematics did not change until the early 20th century when Kurt Gödel famously proved that mathematical knowledge is incomplete in essence [41]. Moreover, more recent theories of science started to emphasise the revolutionary and non-accumulative nature of scientific discoveries as opposed to views that newer and better theories are discovered by better generalising more and more new observations that give more comprehensive evidence. Change from geocentric to heliocentric theories, for instance, is described by Thomas Kuhn as a *paradigm shift* that results in waking up to an entirely new world [50].

KGs designed to aid scientists with solving problems of the present time using computational tools and methods also need to consider those new discoveries. Fortunately, progress in computer systems design and engineering has been fast enough since the initial RDF and OWL conceptualisation to allow to base TWA on a dynamic RDF store that already supports concurrency and transactions. Evaluation of triple stores conducted by Chadzynski et al. [18] resulted in choosing Blazegraph™ as a data store for TWA. It is an active open-source project, released under a GPL-2.0 License, and a triple store that is W3C compliant. It is also used at Fortune 500 companies such as EMC, Autodesk as well as Wikimedia Foundation's Wikidata Query Service. Scale-out and High Availability features are available in Enterprise editions of Blazegraph™, which also supports GPU query optimisation, amongst many other features. Transactions, high concurrency, and high aggregate I/O rates are supported in all of its editions. Blazegraph™ supports partitioning data into namespaces for parallel query optimisation [10]. Using named graphs for different parts of namespaces allows querying smaller graphs independently, ensuring the efficiency of larger knowledge bases [17]. This design makes TWA a DKG capable of dynamically storing empirical scientific knowledge.

Moreover, Blazegraph™ provides interfaces for geospatial search via SPARQL queries and updates and, in this way, a full set of Create, Read, Update, Delete (CRUD) operations on objects with associated geospatial coordinates [11]. This capability of TWA's backend allows it to operate on empirical knowledge about geospatially located objects of the real world as a GKG. As an innovative platform designed to provide means to computationally aid understanding and planing of cities, TWA contains components, interfaces and methods to describe all city objects in terms of CityGML standard by OGC. OntoCityGML is an improved and logically validated version of CityGML ontology [23] that provides a

vocabulary to describe those objects in a way compliant with CityGML 2.0 standard [35] and serialised in OWL 2. Triple statements concerning city objects described in OntoCityGML form a graph-structured equivalent of 3D CityDB [21] by Technische Universität München (TUM) - Semantic 3D City Database that is capable of storing city objects in Levels Of Detail (LOD) from 0 to 4 as well as link them to objects described in terms of other ontologies, and in this way, connect knowledge about cities to knowledge about other domains. The process of creating a Semantic 3D City Database is automated via City Import Agent, designed within JPS Agent Framework. The cognitive agent includes a full set of operations necessary to instantiate city objects via an augmented TUM Importer/Exporter tool, originally developed to manually create 3D CityDB from CityGML 2.0 models serialised in static XML files [81].

The TWA system architecture described above has been used to create two proof of concept city models: one for Berlin and another for Singapore. The city model of Berlin buildings in LOD2 showcases the scalability of the platform and its capabilities to handle the so-called ***five V*** problems in smart city data management [3]. Using this system architecture, it was possible to create semantic web standards compliant representation of buildings in OntoCityGML that contains 419 909 661 Subject-Predicate-Object triples in total, partitioned into named graphs that separate from and link to each other 3 475 683 city objects consisting of 9 558 218 geolocated surface geometries, described in EPSG:25833 coordinate reference system equivalent to the original CityGML 2.0 model. The Singapore city model demonstrates how the TWA architecture enables enriching CityGML objects with further information. First, master planning data by the Singapore Urban Redevelopment Authority (URA) was loaded in the knowledge graph. Each plot was represented as a generic city object with an associated geolocated shape and additional attributes such as the plot's zoning type. Then, another graph based on the OntoZoning [76] ontology was added, containing information on what land uses each zoning type allows. The ontology represents relationships between zoning types, land uses and programmes that are more specific types of land uses to aid common tasks in urban planning and urban development processes. This description made it possible to query not only the zoning type of plots but the land uses allowed on each plot and connect those two different aspects in one graph.

OntoCityGML and OntoZoning were designed using the Protégé ontology editor that was also used for their evaluation [56]. The editor is designed to store ontologies as static files and does not contain built-in interfaces that can store them in dynamic data sources. OWL ontologies can be viewed as consisting of at least two distinguishable parts, referred to as TBox and ABox, that are names for concepts borrowed from description logics [4]. The TBox part is what could be regarded as a schema containing vocabulary and descriptions of relations between vocabulary items. It corresponds to scientific theories with well-defined sets of concepts and relations between those concepts, written in a form of SPO triples. Because of the web 1.0 legacies in the default design of the web of data, this theoretical part is meant to be shareable as a document. Although in principle, there isn't anything to stop this document from being dynamically generated, existing tools are designed under the assumption that such a document is a file. At the same time, converting OWL 2 TBox into RDF using existing tools is a process that leads to some information loss, which is elaborated more on in the next three paragraphs. However, from the history of science as well as theories of scientific progress, it is known that scientific theories

change while describing the same sets of facts. Therefore, even if the presented DGKG architecture is able to store a large ABox that contains empirical facts described within terms of existing theories, those theories could not be uploaded into it without a loss of linkage between concept descriptions. To make this architecture fully dynamic, in terms of TBox and ABox dynamism, an additional software component, replacing blank nodes with links, was required. Without making use of this component, and storing concept and property definitions in OWL 2 files by default, KG are only partially dynamic because they comprise two types of linked graphs - a static TBox RDF graph and a dynamic ABox RDF graph.

OWL 2 provides a relatively rich and flexible set of lexical and logical means used in concept definitions. In OntoZoning, some of those features were utilised to create definitions of land uses that allow or may allow certain programmes that, in turn, group concepts describing potential functional utilisation of land. An example of *AirportUse* definition from OntoZoning, showing a relatively complex concept definition as a result of saving it to OWL in RDF/XML syntax [60] using Protégé ontology editor, is included in the listing 1. Structural properties of XML, like tag nesting etc., allow to easily represent the order of relations between concepts in such definition.

```
1
2       <owl:Class rdf:about="/OZ.owl#AirportUse">
3           <rdfs:subClassOf rdf:resource="/OZ.owl#LandUse"/>
4           <rdfs:subClassOf>
5               <owl:Restriction>
6                   <owl:onProperty rdf:resource="/OZ.owl#
                        allowsProgramme"/>
7                   <owl:allValuesFrom>
8                       <owl:Class>
9                           <owl:unionOf rdf:parseType="Collection">
10                              <rdf:Description rdf:about="/OZ.owl#
                                    AirportFacility"/>
11                              <rdf:Description rdf:about="/OZ.owl#
                                    AirportTerminal"/>
12                              <rdf:Description rdf:about="/OZ.owl#
                                    LandingSite"/>
13                          </owl:unionOf>
14                      </owl:Class>
15                  </owl:allValuesFrom>
16              </owl:Restriction>
17          </rdfs:subClassOf>
18          <rdfs:subClassOf>
19              <owl:Restriction>
20                  <owl:onProperty rdf:resource="/OZ.owl#
                        mayAllowProgramme"/>
21                  <owl:allValuesFrom rdf:resource="/OZ.owl#
                        AircraftTakeoff,Landing,Taxiing,Parking"/>
22              </owl:Restriction>
23          </rdfs:subClassOf>
```

```
24      </owl:Class>
```

Listing 1: *Definition of AirportUse land use with allowed programmes and additional programmes allowed under certain conditions as OWL restrictions in RDF/XML syntax.*

OWL 2 specification also defines the language as fully representable in an SPO RDF graph. Although existing conversion tools produce an SPO graph after converting the definition included in the listing 1 into a definition expressed in *N*-Triples syntax [69], such definition and a resulting RDF graph, that are included in the listing 2, are less precise and less informative because of replacements in XML nesting structure with blank nodes made during conversion. Although some of the existing implementations use blank node identifiers, those identifiers are always locally scoped, not persistent and not portable [68]. In SPARQL, those labels cannot be used as references to specific blank nodes in the data being queried [78] and cannot be understood as identifying nodes in the active graph of the dataset [79]. Therefore, it is no longer possible to read such definitions in an intended order while attempting to retrieve them from an RDF store using SPARQL as a query language. This loss of linkage results in a less complete and more vague TBox graph while attempting to store scientific theories expressed in OWL 2 and produced with existing tools in DKGs.

```
1
2       </OZ.owl#AirportFacility> <rdf:type> <owl:Class> .
3       </OZ.owl#allowsProgramme> <rdf:type> <owl:ObjectProperty> .
4       _:blank <rdf:rest> _:blank .
5       _:blank <rdf:first> </OZ.owl#AirportTerminal> .
6       </OZ.owl#LandingSite> <rdf:type> <owl:Class> .
7       _:blank <rdf:rest> _:blank .
8       _:blank <rdf:first> </OZ.owl#AirportFacility> .
9       _:blank <owl:unionOf> _:blank .
10      _:blank <rdf:type> <owl:Class> .
11      _:blank <owl:allValuesFrom> </OZ.owl#AircraftTakeoff,Landing
            ,Taxiing,Parking> .
12      _:blank <owl:onProperty> </OZ.owl#mayAllowProgramme> .
13      _:blank <rdf:type> <owl:Restriction> .
14      </OZ.owl#AirportTerminal> <rdf:type> <owl:Class> .
15      </OZ.owl#mayAllowProgramme> <rdf:type> <owl:ObjectProperty>
            .
16      _:blank <rdf:type> <owl:Ontology> .
17      _:blank <rdf:rest> <rdf:nil> .
18      _:blank <rdf:first> </OZ.owl#LandingSite> .
19      </OZ.owl#AirportUse> <rdfs:subClassOf> _:blank .
20      </OZ.owl#AirportUse> <rdfs:subClassOf> _:blank .
21      </OZ.owl#AirportUse> <rdfs:subClassOf> </OZ.owl#LandUse> .
22      </OZ.owl#AirportUse> <rdf:type> <owl:Class> .
23      </OZ.owl#LandUse> <rdf:type> <owl:Class> .
24      _:blank <owl:allValuesFrom> _:blank .
25      _:blank <owl:onProperty> </OZ.owl#allowsProgramme> .
```

```
26      _:blank <rdf:type> <owl:Restriction> .
27      </OZ.owl#AircraftTakeoff,Landing,Taxiing,Parking> <rdf:type>
            <owl:Class> .
```

Listing 2: *Definition of AirportUse land use with allowed programmes and additional programmes allowed under certain conditions as OWL restrictions in N-Triples syntax produced from 1 using existing tools.*

Therefore, a tool that can store complex OWL 2 definitions without information loss in DKGs has been developed as a part of the proof of concept described in this paper. The *owlconverter* tool uses a skolemization technique that traces blank nodes during conversion between OWL serialisation formats and replaces them with unique IRIs [70]. This way of converting can preserve the intended order of concepts in such definitions and results in dynamic TBox RDF graphs without information loss that are linked to corresponding dynamic ABox RDF graphs by concept definitions and factual statements utilising defined concepts. Additionally, the tool can store the TBox in a separate named graph uploaded to a SPARQL endpoint, exposing a dynamic data source on the web.

According to Pollock and Hodgson [65], "semantic interoperability is a dynamic enterprise capability derived from the application of special software technologies (such as reasoners, inference engines, ontologies, and models) that infer, relate, interpret, and classify the implicit meanings of digital content without human involvement, which in turn drive adaptive business processes, enterprise knowledge, business rules, and enterprise application interoperability". Blazegraph™ is equipped with an inference engine [12]. However, it does not support full OWL. Moreover, the built-in inference engine materialises all the inferred triples, which may lead to growing KG quickly and negatively impact the performance of it in return. Without any inference invalidation mechanisms in place, knowledge stored in the KG could also become inconsistent when TBox changes. Such invalidation mechanisms or inference provenance tracking that would invalidate obsolete statements do not exist within Blazegraph™. The existing engine also does not provide means to infer new knowledge using graph inference algorithms. Improvements to the TWA system design and the DGKG reference architecture, presented in this paper, require revision of existing novel inference architecture presented by Allemang and Hendler [1] to provide even better support for semantic interoperability and automated inference capabilities via a variety of existing KG inference algorithms. This revised reference architecture is presented in Figure 1. A proof of concept showcasing those new capabilities is described in more detail in the next section.

3 Multi-domain Reasoning on Dynamic Geospatial Knowledge Graphs

DGKGs, based on the revised novel system architecture discussed in Section 2, are capable of storing, retrieving, updating and removing knowledge about complex and geospatially located objects. Moreover, thanks to being capable of inferring new knowledge from the existing one, they are equipped with tools allowing for the automatic enrichment of knowledge concerning such objects without the necessity of painstaking data collec-

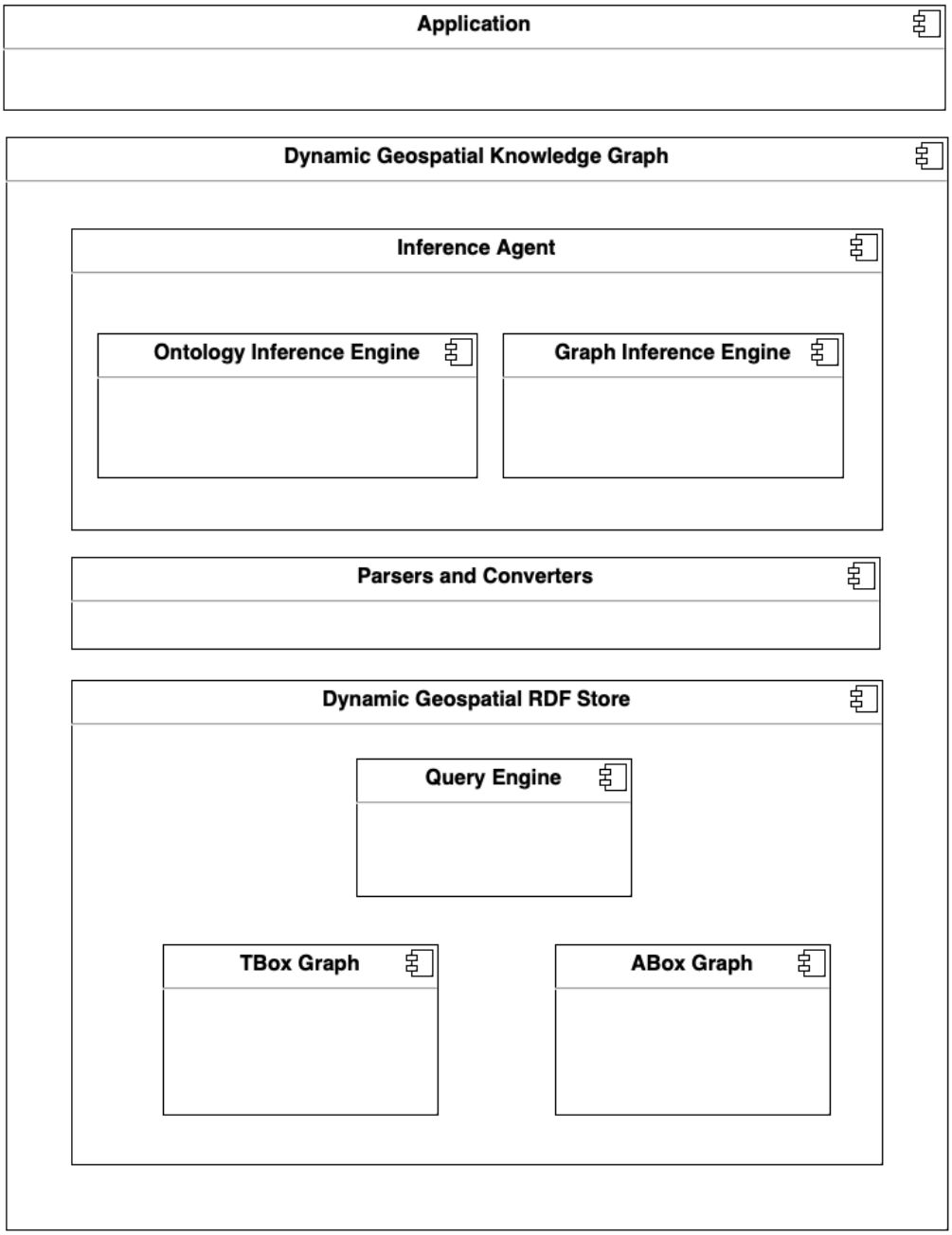


Figure 1: *Component diagram of revised novel system architecture for inferencing in DGKG. TBox and ABox graphs are both contained within a dynamic RDF store with enabled geospatial features and queries facilitated by its integrated query engine. The materialisation of triples contained in both graphs is supported by necessary data passers and converters. The inference agent, also described within the DGKG, plays the role of an inference engine capable of deriving new statements via ontology and graph inference algorithms. The backend allows to build applications utilising inferred and explicitly stated knowledge.*

tion and examination. TWA is a working prototype of DGKG with inference capabilities implemented as a proof of concept showcasing the potential of such system architectures while staying open to extensions, improvements and other types of modifications and integrations. TWA provides means to link complex object descriptions in terms of domain-specific ontologies into physical locations on the surface of the Earth. In this way, it is possible to arrive at a dynamic and holistic representation of a real-life object by linking representations of different aspects of such an object together in a form of SPO statements resulting in an RDF graph.

A query engine built into Blazegraph™ allows for knowledge access and for performing knowledge persistence routines via SPARQL query and update languages designed to work with RDF graphs. The newly designed Inference Agent (IA) presented in this paper provides KG inference capabilities over HTTP. The cognitive agent, like all other agents built upon the JPS Agent Framework, listens to HTTP requests as an input and responds with HTTP responses as an output. Request payloads sent as inputs to the IA are extracted, validated and processed by the agent's tasks before being sent back to the party that initiated communication with the agent. The agent responds with new knowledge that is not explicitly stated in the KG to the requests that specify a type and object of desired new insights. The agent can produce responses by choosing an appropriate task to a desired type that implements one of the currently known types of inference algorithms, described in the next subsection 3.1. Because of this design, the agent could also be regarded as an extended inference engine component that is capable of not only performing ontology inferences, as it was in the unmodified novel reference architecture but also inferences based on currently known graph inference algorithms.

3.1 Knowledge Graph Inference Algorithms

Inference algorithms allow to reliably make new conclusions from existing knowledge. In the revised novel system architecture for DGKGs, the query engine provides interfaces for knowledge retrieval and the IA, playing a role of an inference engine, provides interfaces for reliably arriving at new insights that are not explicitly stated in the RDF store. The reliability and validity of those conclusions are ensured by utilising existing and well-known inference algorithms by the agent's tasks. The agent demonstrates the implementation of two broad classes of inference algorithms. Due to the graph structure of the linked SPO statements, it is possible to apply graph inference algorithms. This class of algorithms was not included in the unmodified novel reference system architecture. In general, this class of algorithms is usually omitted in the literature devoted to inferences on the semantic web that focuses on the ontology inference algorithms, which is the second broad class of algorithms supported by the agent's tasks dedicated to making new conclusions from existing knowledge.

When graph inference algorithms are applied to get new insights, semantics are disregarded, and only the graph structure, i.e. number and depth of branches of each node, is taken into account to complete this task. In this way, they are more general than ontology inference algorithms because they do not take into account any interpretation of the statements that form a particular graph structure. Therefore the same set of conclusions, arrived at by using this class of algorithms, would be applicable to all isomorphic

KGs with the same numbers of nodes and relations between those nodes but representing knowledge about heterogeneous domains. From the point of view of this class of algorithms, such KGs are, in fact, one and the same KG, regardless of whether statements that form those graphs refer to different objects, similar objects or the same objects described from different points of view.

Graph inference algorithms can be grouped into three classes of algorithms, focused on different aspects of the relationship between nodes in the graph. Community detection algorithms find groups of nodes that are most similar to each other in terms of how they are related to all the other nodes within a graph. Centrality detection algorithms analyse the roles of nodes of a graph. Here belong algorithms that weigh nodes by their influence on the overall graph structure in terms of how they are related to them within a graph. Path-finding algorithms measure distances within a graph and score paths between nodes depending on how many other nodes are on those paths. The proof of concept presented in this paper describes an application of sample algorithms belonging to each class of graph inference algorithms. The DGKG inference capabilities could be extended beyond this proof of concept by implementing more of the known algorithms for each class.

Due to the usage of semantic web standards for knowledge representation within DGKGs, apart from algorithms applicable to all graphs in general, it is also possible to utilise ontology inference algorithms for deriving new knowledge from existing and explicitly stated knowledge. OWL 2 ontologies provide means to group individual objects of a given universe of discourse into named sets, denoted as classes. Such classes are arranged within hierarchies that express relationships of individual objects as belonging to larger or smaller groups of objects or subsets of larger sets, denoted as subclasses of superclasses. In other words, such ontologies create taxonomies. Individuals that belong to a certain named set are said to be instances of a class in terms of OWL 2. Apart from the ability to express such unary relationships of individuals, it is possible to express binary relationships between individuals belonging to those named sets while adopting this knowledge representation standard. Binary relations between individuals correspond to named sets of ordered pairs of individuals, denoted as object properties. This level of language expressivity makes it possible to utilise two classes of ontology-based reasoning known from the broader discipline of logic. Namely, it is possible to apply taxonomic reasoning as well as rule-based reasoning to arrive at new knowledge based on existing knowledge represented in OWL 2. Due to the fact that there are more convenient semantic web standards and languages of expressing rules than those provided by raw OWL 2 that could be utilised while working with existing reasoning engines, only taxonomic reasoning capabilities have been included in the proof of concept presented in this paper. This makes IA capable of checking for consistency, class membership, class specialisation, class disjointness, value and property restriction, as well as cardinality restriction on classes defined in the ontology.

3.2 Cognitive Capabilities of Inference Agent

IA, capable of answering with new insights to requests specifying objects described within the KG and types of desired reasoning methods, is built upon the cognitive architecture of the JPS Agent Framework introduced by Chadzynski et al. [20]. Within this architecture, a dynamic RDF store provides means of storing long- and short-term memories about

objects organised into larger mental structures described in a form of OWL 2 compliant SPO statements. Above described extended novel reference architecture places long-term knowledge corresponding to scientific theories (TBox), as well as knowledge about facts described in terms of those theories (ABox), in a dynamic store that allows for concurrent access and modifications to any functional processes that may operate on those structures by intelligent autonomous agents. This dynamism makes it close to what actually happens during a cognitive process. It reflects on a constant interplay between general and particular or between longer-term theories and an influx of new facts that eventually leads to modifications of theories and makes cognitive subjects 'seeing certain things in a new light' [44].

The proof of concept presented in this paper expands on the cognitive capabilities of JPS Agents by introducing an ontology that encodes a taxonomy of existing inference algorithms as well as particular cognitive agent's tasks that contain functions allowing the agent to apply those algorithms on the other knowledge structures stored within the KG. Newly developed for this purpose *OntoInfer* ontology contains all of the classes and sub-classes of previously mentioned inference algorithms. It also contains object properties that relate those classes to tasks implemented by the IA. In this way, the agent reads knowledge about relations between algorithms and its own tasks from the KG and makes intelligent choices based on this knowledge about which task to operate in order to satisfy certain requests, instead of having this 'choice logic' implemented in a form of programming language code. It opens a possibility for an agent to make more dynamic, context-based choices and extends the flexibility of the overall autonomous intelligent agents system architecture.

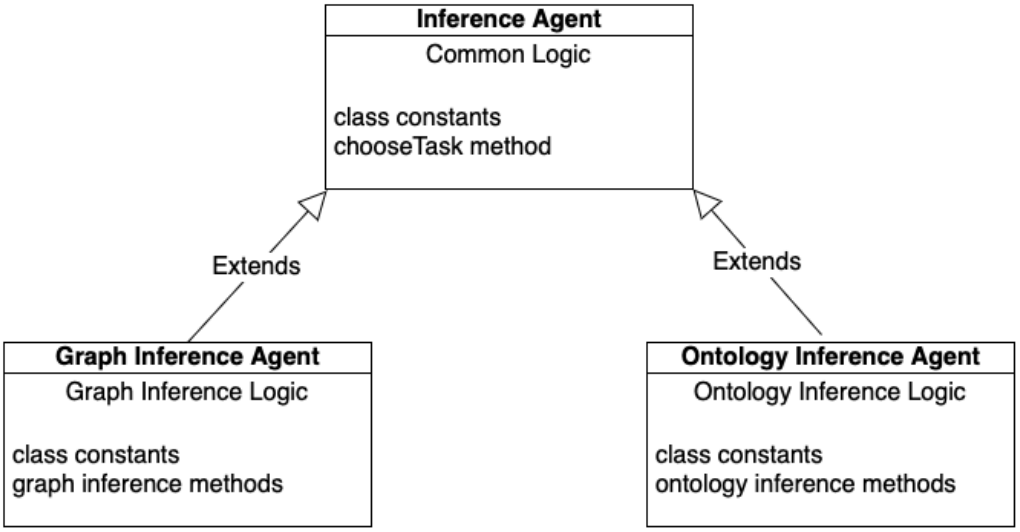


Figure 2: *Class diagram of Inference Agent designed as an abstract class in Java that groups common inference agent logic and subclasses into Graph Inference Agent and Ontology Inference Agent that contain operations specific to two types of inference algorithms within DGKG.*

Because the two classes of the inference algorithms utilised by the agent operate on different parts of the knowledge base, the agent is implemented as an abstract class in Java that groups common functionality required to make all inferences. This abstract class could not be instantiated as a Java object. However, instances of its concrete subclasses, namely Graph Inference Agent and Ontology Inference Agent, listen to and intercept requests pointing to either the whole graphs or into identifiers of classes and individuals within a graph, accordingly. Knowledge about different parts of the globe, which are geospatially described within separate and widely adopted Coordinate Reference Systems (CRS), is stored in separate namespaces within TWA. Every namespace has got automatically assigned a unique International Resource Identifier (IRI). Different parts of namespaces are separated into named graphs that describe distinct aspects of geospatial objects. Graph inference algorithms are provided with IRIs of named graphs containing knowledge in a form of TBox and ABox as an input for analysis. Ontology inference algorithms operate on IRIs of classes, individuals or both to analyse their relationships within the knowledge base. The only exception is consistency checking which, similarly to the graph inference algorithms, requires whole graphs as an input. The relationship between IA and its two incarnations responsible for applying different classes of inference algorithms is depicted in Figure 2.

Autonomous agents based on the JPS Agent Framework are deployed as microservices [61] that listen to HTTP requests and produce HTTP responses after processing request parameters. Input validation is compulsory for every agent at the very first step after request interception to stay in accordance with cybersecurity best practices and to prevent agents from processing potentially corrupted information, causing system malfunctions further downstream [58]. Input for the IA consists of request payload in JSON format that contains one or more KG object IRIs that resolve to either whole graphs, classes or individuals as well as an IRI of a particular inference algorithm, described in *OntoInfer*, to be applied to those objects. The ontology axioms and assertions used by the agent are included in the Appendix B.

Upon successful input request payload validation, the agent queries for the information stored in the KG for which of its tasks are suitable to perform the operations allowing it to apply the requested algorithm. This agent's long-term knowledge is dynamic and allows for further extensions of the agent, making it able to make more sophisticated choices depending on the operational context. For instance, the agent could potentially choose between ontology inference and graph inference tasks while being asked for class specialisation checking, if there were many threads of any of those tasks currently busy with calculating other answers. In this way, if the ontology inference thread pool was already exhausted and the number of questions on the queue was exceeding a certain threshold, the agent could check what is the shortest path between the two classes. If it turned out that there is a shortest path with no nodes in between, that would mean that the two given classes form a subclass-superclass relationship. By choosing between different algorithms to find the same answer, depending on the context and self-observation, the agent would demonstrate a certain degree of cognitive flexibility with regard to achieving a given goal by its currently available means. This design also can change the agent's knowledge dynamically and vary it independently of the core agent's code that requires recompilation and the agent's redeployment every time any change occurs within it.

After the agent's decision about the most appropriate task for the requested algorithm is made, the agent queries the KG for knowledge about objects to which the algorithm is going to be applied. Upon retrieval of it from the KG, the agent places it on a data queue in a form of a map that associates an agent's task IRI with the fetched data. All agent's tasks run in separate threads and extend Java Runnable interface [46]. This allows the tasks to work independently of the main agent's thread while observing the queue for new data processing maps, as well as putting the results of their operations back on the queue for the agent to pick up and return as its output. The agent executes tasks that wait until any data appears on the queue. If there is data associated with a particular task, a task applies an algorithm that is dedicated to handling it via an inference engine. In the case of graph inference-specific tasks that operate on the whole graphs and may require even a few hours to complete large graph analysis on modern hardware, tasks insert results of applied inference algorithms in separate named graphs in the KG. This allows for faster retrieval of new knowledge as well as the use of inferred knowledge for further inferences, comparisons and analyses. Moreover, this design also provides a simple invalidation mechanism for inferred knowledge that could be easily identified and removed from the KG by graph drop operation [80] in case of DGKG changes requiring reapplication of an inference algorithm. Regardless of that whether a task caches any inferred knowledge this way in the KG or not, it puts the inferred knowledge on a data queue from which it is picked up by the IA, encapsulated in JSON response payload and returned back to the party that initiated the request to the agent. A detailed activity diagram illustrating described information flow is included in the Appendix A.

3.2.1 Graph Based Inferences

To demonstrate **community detection** capabilities of the IA on multi-domain knowledge representations as a proof of concept, the Edge Betweenness Clusterer from Jena Jung [47] library was integrated into an agent's task. The algorithm was applied to the semantic representation of land plot data of Singapore, where geolocated plot shapes were described in terms of OntoCityGML and plot zones and associated land use information in terms of OntoZoning. The graph was compressed before being passed onto the algorithm for clustering. The initial RDF graph fetched from the TWA was mapped to a Jena Jung Undirected Sparse Graph of integers, where each subject, predicate and object was mapped to a uniquely identifying numerical value. This operation reduced the overall graph memory footprint and significantly shortened the computing time required to complete cluster detection by the algorithm. The mapping information was stored by the task in the internal class member variable for the purpose of remapping clustered integers back to the original subjects, predicates and objects and restoring the original RDF graph with each plot assigned to a certain cluster discovered by the algorithm. The results of applying the clustering algorithm that contained the information about the cluster number of each plot as assigned by the algorithm were stored in the TWA as a separate named graph within the same namespace and allowed the visualisation of the distribution of geolocated plot clusters and their shapes on a map of Singapore included in the Figure 3.

The algorithm correctly assigns most of the plots to clusters that correspond to each zoning type and match the subjects and objects of appropriate *PlotIRI hasZone ZoningTypeIRI*

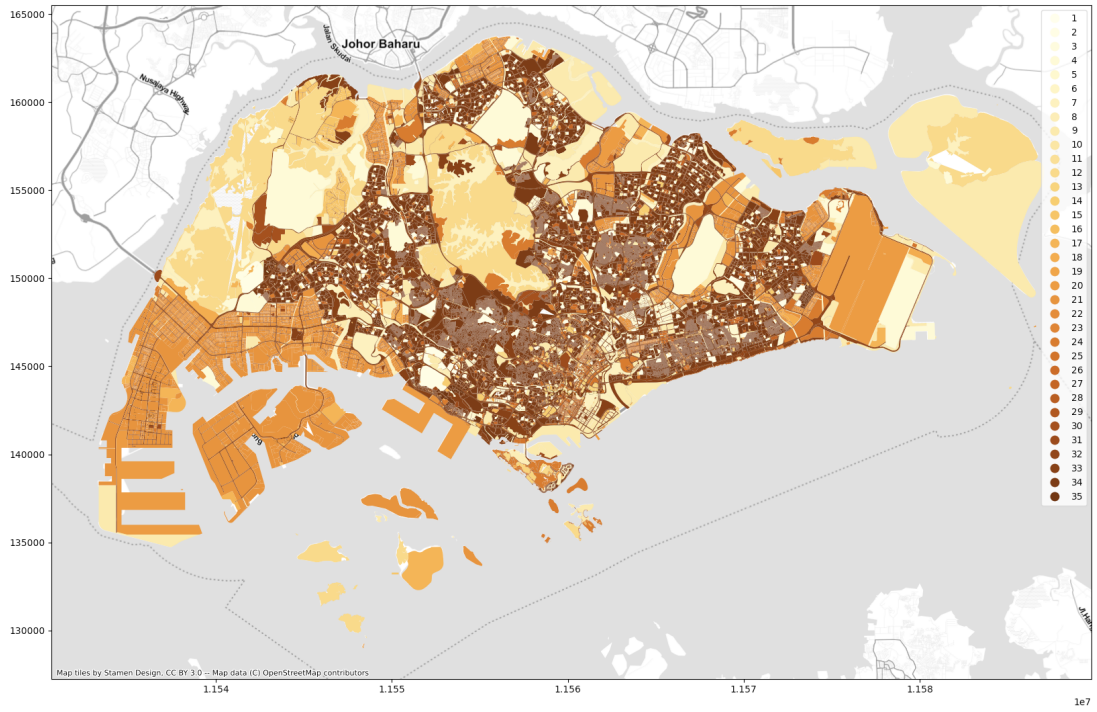


Figure 3: *The result of applying the community detection algorithm on a TWA subgraph consisting of Singapore plots data described as generic city objects using the OntoCityGML ontology and functionally described as land plots with zones using the OntoZoning ontology. Plots of the same colour belong to the same cluster. The colour gradient represents sequential numbers, with clusters having the lowest sequential number in light yellow and the highest sequential number in dark brown.*

SPO statements stored in the TWA before applying the algorithm, where *PlotIRI* is a city object IRI described in the OntoCityGML and *ZoningTypeIRI* is an IRI of an instance of a concept defined within OntoZoning ontology. The clustering algorithm detects which plots are linked to the same zoning type and allocates these plots to the same cluster. Relating the city objects to more complex data, such as nodes representing city citizens' visits to the areas within plots, would allow for clustering plots based on the number of visits that took place in each plot. Applying the algorithm to find clusters that correspond to how frequently each plot is visited could also find new insights, as the knowledge graph does not contain this information stated explicitly.

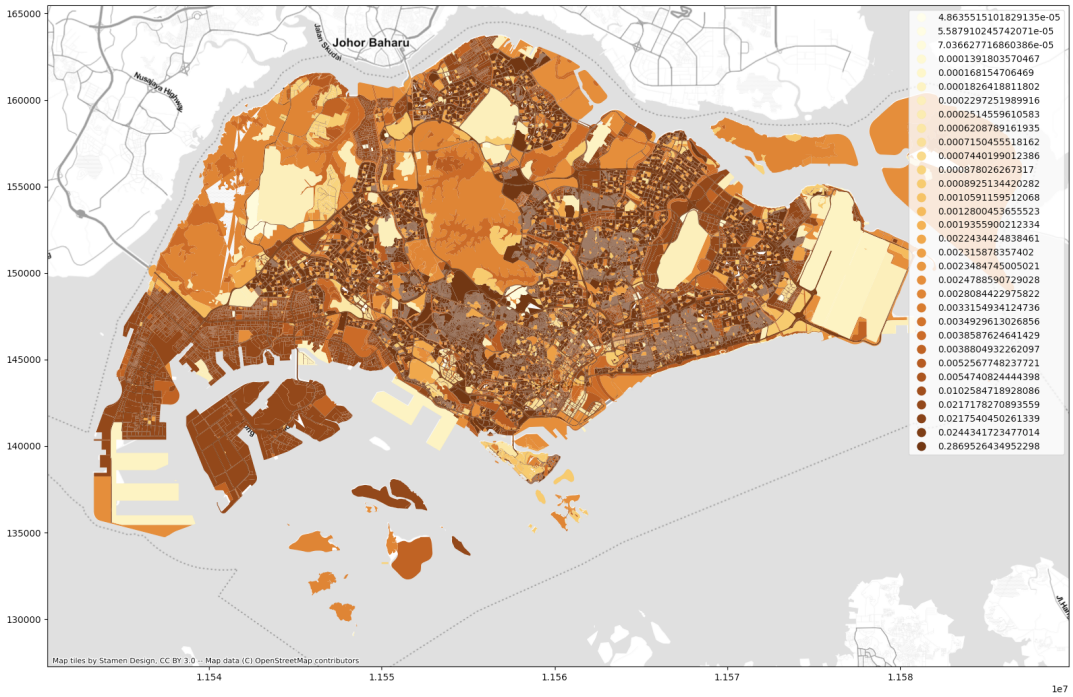


Figure 4: *The result of applying the centrality detection algorithm on a TWA subgraph consisting of Singapore plots data described as generic city objects using the OntoCityGML ontology and functionally described as land plots with zones using the OntoZoning ontology. The colour gradient represents the PageRank score of their zones, with plots having the lowest PageRank score of their zones in light yellow and the highest PageRank score of their zones in dark brown.*

An implementation of a **centrality detection** algorithm found in Jena Jung was also integrated into an IA's task to demonstrate the agent's capabilities related to this subclass of graph inference algorithms. The implementation of the Page Rank algorithm from this library was computationally efficient enough to not require graph remapping on the same dataset. Straightforward RDF graph conversion to Jena Jung graph implementation allowed to apply the algorithm and score each graph element accordingly to its transitive influence on the whole graph. The information about what rank each plot was assigned by the algorithm was stored in the TWA as a separate named graph within the same namespace and allowed to visualise the distribution of geolocated zoning types' ranks on plots

belonging to those zoning types with their shapes on a map of Singapore included in the Figure 4.

The page rank algorithm results in 33 distinct ranks for all the plots. Each rank corresponds to one zoning type, with the exception of one rank that contains plots with 'Light rapid transit' and 'Mass rapid transit' zoning types. Plots with higher ranks have more common zoning types. This is because plots with the same zoning type are all linked to each other with one node (the zoning type) in between, and hence plots with a common zoning type are more closely linked to many nodes than plots with a rare zoning type. This example again shows how graph inferencing algorithms can provide meaningful and interpretable results. With more complex data in the knowledge graph (and more links between plots and other nodes), it would be possible to come up with more surprising findings.

Path finding capabilities of the IA in this proof of concept were demonstrated by the means of integrating Unweighted Shortest Path algorithm implementation from the Jena Jung library. Similarly, for community detection, this task required graph compression to be computationally efficient enough on the Singapore plot dataset described in terms of OntoCityGML and OntoZoning ontologies. The same technique of mapping RDF statement parts into integers was reused between those two tasks. The inferred knowledge was also stored in a form of SPO statements in a separate named graph within the same namespace to eliminate the necessity of repeating this computationally intensive process every time the agent is asked to compute the distance between the exact same set of nodes on an unchanged graph. The named graph storing the inferred statements could be deleted or versioned and archived in case of KG changes that may affect previously computed paths. The agent was able to answer correctly when asked about epistemic distances between objects described in the following subgraph of the TWA:

$$\begin{aligned} \mathit{Agriculture} &\sqsubseteq \mathit{ZoningType} \\ \mathit{Business} &\sqsubseteq \mathit{ZoningType} \\ \mathit{Agriculture} &: \mathit{Agriculture} \\ \mathit{Business} &: \mathit{Business} \\ (\mathit{plotA}, \mathit{Agriculture}) &: \mathit{hasZone} \\ (\mathit{plotB}, \mathit{Agriculture}) &: \mathit{hasZone} \\ (\mathit{plotC}, \mathit{Business}) &: \mathit{hasZone}. \end{aligned}$$

It identified the distance between *plotA* and *plotB* as having a value of 1 because both plots are related to the same instance of the *Agriculture* class via *hasZone* object property. The agent also answered with the correct value of the epistemic distance between the *plotA* and the *plotC* and the same value between the *plotB* and the *plotC*. The shortest path between both plots to the *plotC* contains 2 nodes as they are related to instances of *ZoningType* class via its subclasses by the *hasZone* object property, namely the *Business* instance of the *Business* class, in case of the *plotC* and the *Agriculture* instance of the *Agriculture* class, in case of the *plotB* and the *plotA*.

3.2.2 Taxonomic Reasoning

For the purpose of demonstrating taxonomic reasoning capabilities of the IA, an abstract Taxonomic Reasoning task was created to group all methods required to work with a dedicated reasoning engine. All other subtypes of taxonomic reasoning are handled by separate concrete task classes that extend the abstract task. The HermiT reasoner engine was integrated to those tasks as a well tested and outperforming other engines of this type on benchmarks. It is based on the well known version of the tableau algorithm with improvements, namely novel hypertableau algorithm. The reasoner is capable of working with OWL 2 knowledge bases. However there are no examples of using it with DKGs in its public codebase. Therefore the Taxonomic Reasoning task required methods of converting JSON SPARQL query results that were returned while fetching TBox and ABox from the dynamic RDF store to the *N*-Triples format that could be parsed by OWL API library and used with HermiT. This process also required a method reverse to the skolemization technique used to upload TBoxes into the RDF store with the newly developed *owlconverter* tool. Without it, the OWL API was not able to correctly restore more complex concept definitions that contained restrictions and caused HermiT to respond with incorrect answers.

Both OntoCityGML and OntoZoning were previously evaluated for the consistency and the IA confirmed results of the evaluation. Additional consistency checks were done with OntoBuildableSpace ontology that makes use of OntoZoning and Units of Measure ontology to represent measurable characteristics of buildable spaces on plots in Singapore. As the ontology consists of more complex class definitions and property relationships than OntoZoning ontology, it served as a good base to illustrate IA Class Disjointness Checking, Value Restriction Checking, Property Restriction Cardinality Restriction Checking assertion tasks. The **Consistency Checking** task was separately tested to ensure that it also works correctly on inconsistent knowledge bases. A unit test developed for the task correctly asserted consistency of the following knowledge base:

$$\begin{aligned} (A, Class) &: type \\ (B, Class) &: type \\ (C, Class) &: type \\ (C, B) &: disjointWith \\ A &\sqsubseteq B \\ A &\sqsubseteq C \end{aligned}$$

It also correctly detected the inconsistency of a knowledge base obtained from it by adding the following two more statements:

$$\begin{aligned} (a, NamedIndividual) &: type \\ a &: A \end{aligned}$$

because it is not possible for an individual a to be an instance of two disjoint classes. Similar unit tests were developed for all other tasks dedicated to work with all the other methods of taxonomic reasoning to ensure that the IA is able to handle them correctly.

The task handling **Class Membership Checking**, given the following TWA fragment:

$$(CityObjectA, NamedIndividual) : type$$
$$(CityObjectA, CommercialZone) : hasZoningType$$
$$(hasZoningType, Plot) : domain$$
$$(hasZoningType, ZoningType) : range$$

is able to infer the following statements that were not explicitly stated in the KG:

$$(CityObjectA, Plot) : type$$
$$(CommercialZone, ZoningType) : type.$$

It is also possible to assert the truthfulness of the following statements:

$$SideSetback \sqsubseteq Length$$
$$FrontSetback \sqsubseteq Distance,$$

that are not explicitly stated in the following fragment of the TWA:

$$FrontSetback \sqsubseteq Setback$$
$$SideSetback \sqsubseteq Setback$$
$$Setback \sqsubseteq Distance$$
$$Distance \sqsubseteq Length$$

via the **Class Specialisation Checking** task of the IA.

The task dedicated to the **Class Disjointness Checking** is able infer the following statement:

$$(FrontSetback, AbsoluteHeight) : disjointWith$$

not explicitly stated in the following fragment of the TWA:

$$FrontSetback \sqsubseteq Setback$$
$$(Setback, AbsoluteHeight) : disjointWith.$$

The **Value Restriction Checking** task of the IA asserts if two classes with a given IRIs are bound as domain and range of an object property with another IRI. Using the following fragment of the TWA:

$$(hasBuildableSpace, ObjectProperty) : type$$
$$(hasRequiredSetback, ObjectProperty) : type$$
$$(hasSource, ObjectProperty) : type$$
$$(hasRequiredSetback, BuildableSpace) : domain$$
$$(hasBuildableSpace, BuildableSpace) : range$$
$$(hasSource, Setback) : domain$$
$$(hasRequiredSetback, Setback) : range$$
$$(BuildableSpace, Class) : type$$
$$(Setback, Class) : type$$

it is possible to assert that the following statements, not explicitly included in the KG, are always false:

$$(Setback, Setback) : hasRequiredSetback$$
$$(Setback, BuildableSpace) : hasRequiredSetback$$
$$(BuildableSpace, BuildableSpace) : hasBuildableSpace$$
$$(Setback, BuildableSpace) : hasSource.$$

Using the **Property Checking** task it is possible to assert if two individuals with given IRIs are bound in a relationship by an object property with another IRI. Given the following fragment of the TWA:

$$(FrontSetback, Class) : type$$
$$(BuildableSpace, Class) : type$$
$$(Setback1, NamedIndividual) : type$$
$$(Setback1, FrontSetback) : type$$
$$(BuildableSpace1, NamedIndividual) : type$$
$$(requiresSetback, ObjectProperty) : type$$
$$(requiresSetback, BuildableSpace) : domain$$
$$(requiresSetback, FrontSetback) : range$$
$$(BuildableSpace1, Setback1) : requiresSetback$$
$$FrontSetback \sqsubseteq Setback$$
$$Setback \sqsubseteq Distance$$
$$Distance \sqsubseteq Quantity$$

and the assertion of the truthfulness of the following explicitly stated statement by the task:

$$(BuildableSpace1, Setback1) : requiresSetback$$

it is possible to infer new knowledge about inheritance by combining these tasks with Class Specialisation Checking and to conclude that the following statements:

$$(Setback1, FrontSetback) : type$$
$$(Setback1, Setback) : type$$
$$(Setback1, Distance) : type$$
$$(Setback1, Quantity) : type$$

are true within this knowledge base.

Lastly, the **Cardinality Restriction Checking** task allows to infer that a *BuildableSpace can have StoreyAggregate with no more than 5 Storeys*, given the following fragment of

the TWA:

$$
\begin{aligned}
(StoreyAggregate, Class) &: type \\
(Storey, Class) &: type \\
(containsStorey, ObjectProperty) &: type \\
(containsStorey, StoreyAggregate) &: domain \\
(containsStorey, Storey) &: range \\
(CardinalityRestriction, Restriction) &: type \\
(CardinalityRestriction, containsStorey) &: onProperty \\
(CardinalityRestriction, Storey) &: onClass \\
(CardinalityRestriction, 5) &: maxQualifiedCardinality.
\end{aligned}
$$

4 Research Summary and Future Work

DGKGs adhering to geospatial standards have the potential to open existing data silos to one another and, by increasing comparability, improve geographical data interoperability. By turning data into knowledge, DGKGs based on semantic web standards are also capable of deriving new and not explicitly stated knowledge from such multi-domain descriptions by applying inference algorithms. The resulting graph nature of the underlying RDF knowledge representations in such KGs encourages the design of an inference engine capable of automatically producing new statements by examining structural relationships using graph inference algorithms alongside taxonomic reasoning algorithms. The linked data approach enables information systems to arrive at such knowledge inflexions easier way than before. The dynamic graph stores allow to collaborate and work with knowledge concurrently and also by blending human and artificial intelligence of autonomous cognitive agents. Such systems could be utilised to automatically discover new structural and taxonomic relationships by linking various classes of city objects to each other together with multi-domain descriptions that were not considered by anyone before. Thus, the revised architecture and inference agent could support efficient urban data management practices and land use allocation [74] by providing novel semantic and structural methods for evaluating cities' land use and built form regulatory data, as illustrated by the examples (see section 3). On a larger scale, those capabilities taken together could enable a better understanding of cities and aid their future planning in ways that take into account current threats faced by humanity as a whole, like those imposed by climate change to which cities, taken as macro structures, are one of the most significant contributors.

Integrated building blocks at the maturity level of the research prototype for the above capabilities have been described in this paper on the example of TWA. The main forces driving the need for those capabilities, namely the present lack of information systems capable of facilitating highly interoperable and dynamic representations of cities that would allow for complex analysis and inferences to support better planning to mitigate risks of multiplying crises imposed by the external environmental factors, are presented in the Section 1. Apart from that, the section contains a brief description of current systems' gap analysis and challenges that could be overcome by the extended novel architecture

containing uniform interfaces that allow for working with fully dynamic graph representation by an inference engine capable of applying different classes of known inference algorithms. A working definition of DGKG has been introduced in Section 2. It was followed by a more detailed description of the gaps in existing information systems of this type and an explanation of the disadvantages of partially static KGs as they are unable to reflect on the non-accumulative and dynamic nature of scientific knowledge that those types of KGs are aiming to utilise to better understand geographical objects, and cities in particular. Those gaps were illustrated in the example of the building blocks of TWA introduced before including components presented in this paper and believed to be commonly found in similar KGs built on publicly available open-source software. The section also includes a description of *owlconverter* tool designed and developed for the purpose of producing fully dynamic knowledge representations in such systems without any information loss that would otherwise occur if more complex definitions were included in ontologies. The high level overview of the extended novel reference architecture for inferences is presented in the Section 3 together with a more detailed description of supported classes of KG inference algorithms discussed in Subsection 3.1. It is followed by an explanation of that how inferences are facilitated by an intelligent autonomous agent with extended cognitive capabilities of choice based on its long term memories encoded in the newly introduced for this purpose *OntoInfer* ontology in the Subsection 3.2. Examples of multidomain reasoning using three broad classes of graph inference algorithms are included in Subsection 3.2.1 that is followed by the Subsection 3.2.2 including examples of taxonomic reasoning on such multidomain descriptions that include concepts of *OntoCityGML*, *OntoZoning* and *OntoBuildableSpace* within TWA.

Apart from reusing existing components, this work required design and development of the following novel elements:

- A reference architecture for inferences that extends novel architecture previously presented by Allemang and Hendler [1] that consists of the following additions:
  - *owlconverter* that allows to produce fully dynamic knowledge graphs compliant with semantic web standards by W3C and OWL 2 without information loss that otherwise occurs while attempting to store complex concept definitions in existing open source dynamic RDF stores;
  - Inference Agent (IA) - an autonomous intelligent agent with extended cognitive capabilities of making informed choices supported by long term knowledge stored in the KG that acts as an extended inference engine supporting all currently known classes of KG inference algorithms;
  - *OntoInfer* ontology that encodes taxonomy of existing KG reasoning algorithms linked to instances of IA tasks allowing the agent to make choices based on the knowledge stored in the KG;
  - 10 agent's tasks designed to work as independently executed threads implementing examples of known inference algorithms using existing libraries and reasoning engines (Jena Jung and HermiT) and opening an agent for cognitive flexibility while choosing between different implementations in order to achieve a set goal depending on self observation about its thread and computational resources availability.

DGKGs open for collaborative interfaces and citizen science knowledge acquisition and inferencing by neogeographers [31] are also part of cities and similar considerations apply towards their impact on the external environment as it is in case of any other city objects. As it was discussed before, cognitive capabilities of the IA could be extended even further and beyond capabilities of making informed choices based on long term knowledge stored in the TWA. It would be possible for the agent to consider energy usage minimisation factors while choosing between different tasks in a cognitively flexible way to achieve its inference goals. The current inferencing capabilities could be extended by implementation of many more algorithms of the discussed classes as well. This would expand agent's pool of choices and extend flexibility by allowing to compare their respective results on larger scales. This future work could be regarded as a next brick on paving the road towards self-sustainable knowledge graphs that are conceptualised as city objects capable of modelling and understanding their operational environment forming cognitive cities [24].

Acknowledgements

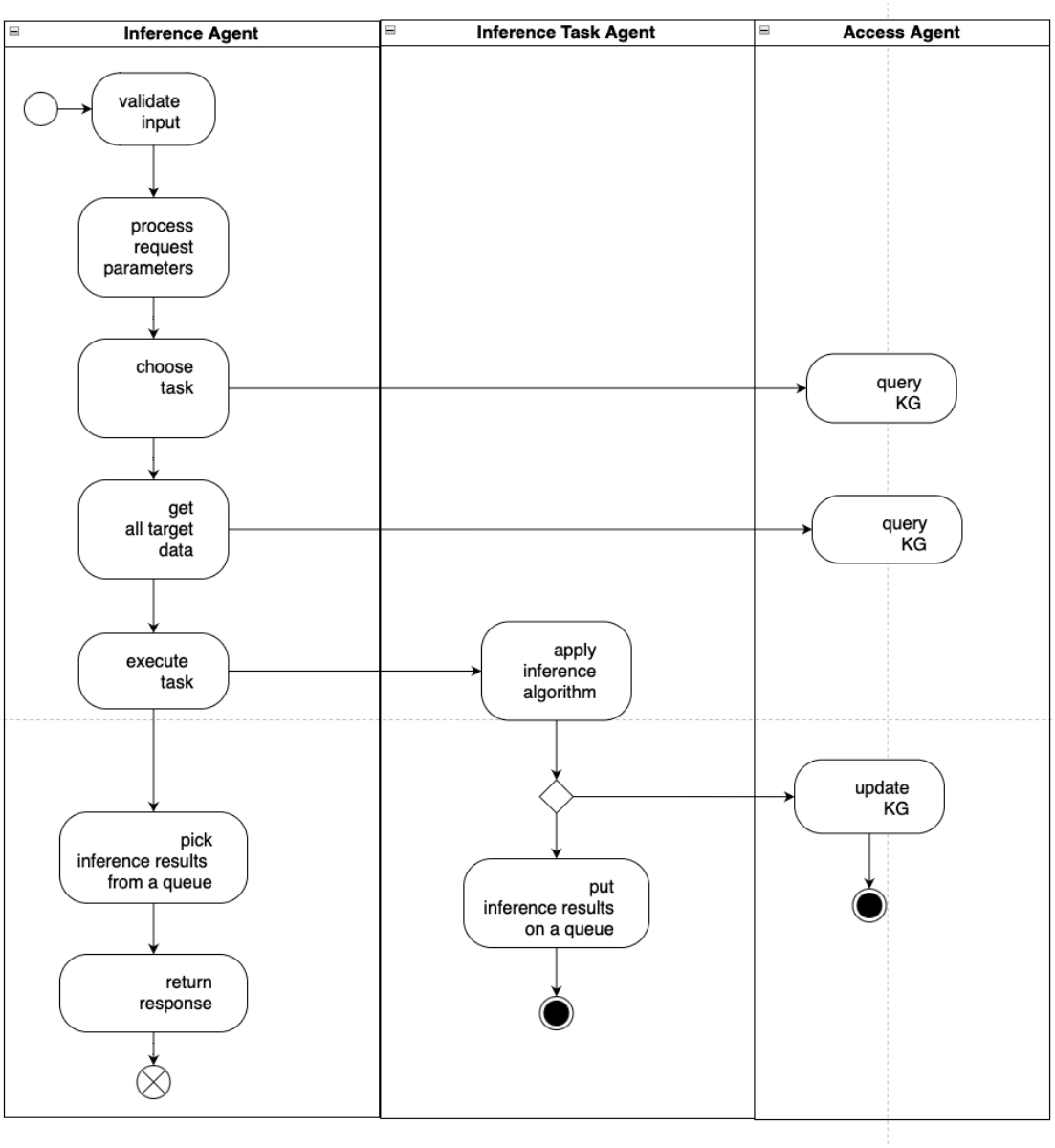
This research is supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. Markus Kraft gratefully acknowledges the support of the Alexander von Humboldt foundation.

The research was conducted as part of an Intra-CREATE collaborative project involving CARES (Cambridge Centre for Advanced Research and Education in Singapore), which is the University of Cambridge's presence in Singapore, and the Future Cities Lab Global at Singapore-ETH Centre and ETH Zurich, which is supported and funded by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme and ETH Zurich (ETHZ), with additional contributions from the National University of Singapore (NUS), Nanyang Technological University (NTU), Singapore and the Singapore University of Technology and Design (SUTD). The authors would like to thank Andrew Breeson for help with proofreading and language correction of the paper.

List of abbreviations

RDF	**R**esource **D**escription **F**ramework
OWL	**W**eb **O**ntology **L**anguage
UN	**U**nited **N**ations
SWT	**S**emantic **W**eb **T**echnology
KG	**K**nowledge **G**raph
GKG	**G**eospatial **K**nowledge **G**raph
DKG	**D**ynamic **K**nowledge **G**raph
DGKG	**D**ynamic **G**eospatial **K**nowledge **G**raph
CKG	**C**ities **K**nowledge **G**raph
TWA	**T**he **W**orld **A**vatar
JPS	**J**-**P**ark **S**imulator
EIP	**E**co-**I**ndustrial **P**ark
UDT	**U**niversal **D**igital **T**win
CIM	**C**ity **I**nformation **M**odel
GIS	**G**eographic **I**nformation **S**ystem
OGM	**O**bject **G**raph **M**apper
ORM	**O**bject **R**elational **M**apper
RDFS	**R**esource **D**escription **F**ramework **S**chema
SPO	**S**ubject-**P**redicate-**O**bject
GPU	**G**raphics **P**rocessing **U**nit
I/O	**I**nput/**O**utput
SPARQL	**S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage
CRUD	**C**reate, **R**ead, **U**pdate, **D**elete
OGC	**O**pen **G**eospatial **C**onsortium
TUM	**T**echnische **U**niversität **M**ünchen
XML	E**x**tensible **M**arkup **L**anguage
GML	**G**eography **M**arkup **L**anguage
W3C	**W**orld **W**ide **W**eb **C**onsortium
LOD2	**L**evel **O**f **D**etail 2
IRI	**I**nternationalized **R**esource **I**dentifier
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
IA	**I**nference **A**gent
CRS	**C**oordinate **R**eference **S**ystem
JSON	**J**ava**S**cript **O**bject **N**otation
UML	**U**nified **M**odeling **L**anguage

A Inference Agent - UML Activity Diagram



B OntoInfer ontology expressed in a DL syntax

Classes:

$$KnowledgeGraph \sqsubseteq owl : Thing$$
$$Agent \sqsubseteq KnowledgeGraph$$
$$GraphStore \sqsubseteq KnowledgeGraph$$
$$GraphStore \equiv Namespace$$
$$Inference \sqsubseteq KnowledgeGraph$$
$$Task \sqsubseteq Agent$$
$$Graph \sqsubseteq GraphStore$$
$$Algorithm \sqsubseteq Inference$$
$$Triple \sqsubseteq Graph$$
$$OntologyBased \sqsubseteq Algorithm$$
$$GraphBased \sqsubseteq Algorithm$$
$$Subject \sqsubseteq Triple$$
$$Predicate \sqsubseteq Triple$$
$$Object \sqsubseteq Triple$$
$$CentralityDetection \sqsubseteq GraphBased$$
$$CommunityDetection \sqsubseteq GraphBased$$
$$PathFinding \sqsubseteq GraphBased$$
$$Evaluation \sqsubseteq OntologyBased$$
$$RuleBased \sqsubseteq OntologyBased$$
$$TaxonomicReasoning \sqsubseteq OntologyBased$$
$$CardinalityConsraint \sqsubseteq TaxonomicReasoning$$
$$ClassDefinition \sqsubseteq TaxonomicReasoning$$
$$ClassDisjointness \sqsubseteq TaxonomicReasoning$$
$$ClassMembership \sqsubseteq TaxonomicReasoning$$
$$ClassSpecialisation \sqsubseteq TaxonomicReasoning$$
$$Inheritance \sqsubseteq TaxonomicReasoning$$
$$NumberConstraint \sqsubseteq TaxonomicReasoning$$
$$ValueRestriction \sqsubseteq TaxonomicReasoning$$

Properties:

$$(applicableTo, ObjectProperty) : type$$
$$(applicableTo, Algorithm) : domain$$
$$(applicableTo, Namespace) : range$$
$$(appliedBy, ObjectProperty) : type$$
$$(appliedBy, Algorithm) : domain$$
$$(appliedBy, Task) : range$$
$$(performedBy, ObjectProperty) : type$$
$$(performedBy, Task) : domain$$
$$(performedBy, Agent) : range$$
$$(hasInferenceAlgorithm, ObjectProperty) : type$$
$$(hasInferenceObject, ObjectProperty) : type$$
$$(hasInferredValue, DataProperty) : type$$
$$(hasInferredValue, xsd : double) : range$$

Assertions:

$$singaporeEPSG4326 : Namespace$$
$$OntologyInferenceAgent : Agent$$
$$GraphInferenceAgent : Agent$$
$$ClassDisjointnessCheckingAlgorithm : ClassDisjointness$$
$$ClassDisjointnessCheckingTask : Task$$
$$(ClassDisjointnessCheckingAlgorithm, ClassDisjointnessCheckingTask) : appliedBy$$
$$(ClassDisjointnessCheckingTask, OntologyInferenceAgent) : performedBy$$
$$ClassMembershipCheckingAlgorithm : ClassMembership$$
$$ClassMembershipCheckingTask : Task$$
$$(ClassMembershipCheckingAlgorithm, ClassMembershipCheckingTask) : appliedBy$$
$$(ClassMembershipCheckingTask, OntologyInferenceAgent) : performedBy$$
$$ClassSpecialisationCheckingAlgorithm : ClassSpecialisation$$
$$ClassSpecialisationCheckingTask : Task$$
$$(ClassMembershipCheckingAlgorithm, ClassSpecialisationCheckingTask) : appliedBy$$
$$(ClassSpecialisationCheckingTask, OntologyInferenceAgent) : performedBy$$
$$ConsistencyCheckingAlgorithm : Evaluation$$
$$ConsistencyCheckingTask : Task$$
$$(ConsistencyCheckingAlgorithm, ConsistencyCheckingTask) : appliedBy$$
$$(ConsistencyCheckingTask, OntologyInferenceAgent) : performedBy$$

Assertions (continue):

$$PropertyCheckingAlgorithm : TaxonomicReasoning$$
$$PropertyCheckingTask : Task$$
$$(PropertyCheckingAlgorithm, PropertyCheckingTask) : appliedBy$$
$$(PropertyCheckingTask, OntologyInferenceAgent) : performedBy$$
$$ValueRestrictionCheckingAlgorithm : ValueRestriction$$
$$ValueRestrictionCheckingTask : Task$$
$$(ValueRestrictionCheckingAlgorithm, ValueRestrictionCheckingTask) : appliedBy$$
$$(ValueRestrictionCheckingTask, OntologyInferenceAgent) : performedBy$$
$$InheritanceChckingAlgorithm : Inheritance$$
$$(InheritanceChckingAlgorithm, ClassMembershipCheckingTask) : appliedBy$$
$$(InheritanceChckingAlgorithm, PropertyCheckingTask) : appliedBy$$
$$EdgeBetweennessAlgorithm : CommunityDetection$$
$$ConsistencyCheckingTask : Task$$
$$(EdgeBetweennessAlgorithm, EdgeBetweennessTask) : appliedBy$$
$$(EdgeBetweennessTask, GraphInferenceAgent) : performedBy$$
$$PageRankAlgorithm : CentralityDetection$$
$$PageRankTask : Task$$
$$(PageRankAlgorithm, PageRankTask) : appliedBy$$
$$(PageRankTask, GraphInferenceAgent) : performedBy$$
$$UnweightedShortestPathAlgorithm : PathFinding$$
$$UnweightedShortestPathTask : Task$$
$$(UnweightedShortestPathAlgorithm, UnweightedShortestPathTask) : appliedBy$$
$$(UnweightedShortestPathTask, GraphInferenceAgent) : performedBy$$

References

[1] D. Allemang and J. Hendler. Chapter 6 - RDF and inferencing. In D. Allemang and J. Hendler, editors, *Semantic Web for the Working Ontologist (Second Edition)*, pages 113–123. Morgan Kaufmann, Boston, second edition edition, 2011. ISBN 978-0-12-385965-5. doi:10.1016/B978-0-12-385965-5.10006-8.

[2] D. Allemang and J. Hendler. Chapter 8 - RDFS-Plus. In D. Allemang and J. Hendler, editors, *Semantic Web for the Working Ontologist (Second Edition)*, pages 153–185. Morgan Kaufmann, Boston, second edition edition, 2011. ISBN 978-0-12-385965-5. doi:10.1016/B978-0-12-385965-5.10008-1.

[3] M. Amović, M. Govedarica, A. Radulović, and I. Janković. Big data in smart city: Management challenges. *Applied Sciences*, 11(10), 2021. doi:10.3390/app11104557.

[4] F. Baader, I. Horrocks, C. Lutz, and U. Sattler. *An Introduction to Description Logic*. Cambridge University Press, 2017. doi:10.1017/9781139025355.

[5] M. Barbieri. Working with RDF database named graphs, 2021. URL https://medium.com/agnos-ai/working-with-rdf-database-named-graphs-a5ddab447e91. Accessed 24th November, 2022.

[6] M. Batty and W. Yang. A digital future for planning, February 2022. URL https://digital4planning.com/wp-content/uploads/2022/02/A-Digital-Future-for-Planning-Full-Report-Web.pdf. Accessed 24th November, 2022.

[7] B. Bebee. Blazegraph, 2020. URL https://github.com/blazegraph/database/wiki/Main_Page. Accessed 24th November, 2022.

[8] B. Bebee. Inference and truth maintenance, 2020. URL https://github.com/blazegraph/database/wiki/InferenceAndTruthMaintenance. Accessed 23rd November, 2022.

[9] T. Berners-Lee. Linked data, 2006. URL https://www.w3.org/DesignIssues/LinkedData.html. Accessed 23rd December, 2022.

[10] Blazegraph - About, 2021. URL https://github.com/blazegraph/database/wiki/About_Blazegraph. Accessed 23rd December, 2022.

[11] Blazegraph - GeoSpatial, 2021. URL https://github.com/blazegraph/database/wiki/GeoSpatial. Accessed 23rd December, 2022.

[12] Blazegraph - Inference and Truth Maintenance, 2020. URL https://github.com/blazegraph/database/wiki/InferenceAndTruthMaintenance. Accessed 23rd December, 2022.

[13] J. Bock and R. Volz. Benchmarking OWL reasoners. In *Conference: Workshop on Advancing Reasoning on the Web: Scalability and Commonsense*, 2007.

[14] H. Boyes and T. Watson. Digital twins: An analysis framework and open issues. *Computers in Industry*, 143:103763, 2022. doi:10.1016/j.compind.2022.103763.

[15] A. Buccella and A. Cechich. Towards integration of geographic information systems. *Electronic Notes in Theoretical Computer Science*, 168:45–59, 2007. doi:10.1016/j.entcs.2006.08.023. Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006).

[16] M. Buyukdemircioglu and S. Kocaman. Reconstruction and efficient visualization of heterogeneous 3d city models. *Remote Sensing*, 12(13), 2020. doi:10.3390/rs12132128. URL https://www.mdpi.com/2072-4292/12/13/2128.

[17] J. J. Carroll, C. Bizer, P. Hayes, and P. Stickler. Named graphs. *Journal of Web Semantics*, 3(4):247–267, 2005. doi:10.1016/j.websem.2005.09.001. World Wide Web Conference 2005——Semantic Web Track.

[18] A. Chadzynski, N. Krdzavac, F. Farazi, M. Q. Lim, S. Li, A. Grisiute, P. Herthogs, A. von Richthofen, S. Cairns, and M. Kraft. Semantic 3D City Database — an enabler for a dynamic geospatial knowledge graph. *Energy and AI*, 6:100106, 2021. doi:10.1016/j.egyai.2021.100106.

[19] A. Chadzynski, S. Li, A. Grisiute, J. Chua, J. Yan, H. Y. Tai, E. Lloyd, M. Agarwal, J. Akroyd, P. Herthogs, and M. Kraft. Semantic 3D City Interfaces – intelligent interactions on dynamic geospatial knowledge graphs, 2022. Submitted for publication. Preprint available at https://como.ceb.cam.ac.uk/preprints/297/.

[20] A. Chadzynski, S. Li, A. Grisiute, F. Farazi, C. Lindberg, S. Mosbach, P. Herthogs, and M. Kraft. Semantic 3D City Agents — an intelligent automation for dynamic geospatial knowledge graphs. *Energy and AI*, 8:100137, 2022. doi:10.1016/j.egyai.2022.100137.

[21] K. Chaturvedi, Z. Yao, and T. H. Kolbe. Web-based exploration of and interaction with large and deeply structured semantic 3D city models using HTML5 and WebGL. In T. P. Kersten, editor, *Bridging Scales - Skalenübergreifende Nah- und Fernerkundungsmethoden, 35. Wissenschaftlich-Technische Jahrestagung der DGPF*, volume 24, Köln, 2015. Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V., Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation e.V.

[22] Cities Knowledge Graph - ResearchGate, 2022. URL https://www.researchgate.net/project/Cities-Knowledge-Graph. Accessed 22th November, 2022.

[23] CityGML ontology (2.0), 2022. URL http://cui.unige.ch/isi/onto//citygml2.0.owl. Accessed 23rd December, 2022.

[24] J. Cuenca, F. Larrinaga, L. Eciolaza, and E. Curry. Towards cognitive cities in the energy domain. *Designing Cognitive Cities*, 2018.

[25] R. Descartes and D. Cress. *Discourse on Method*. Hackett Classics. Hackett Publishing Company, Incorporated, 1998. ISBN 9781603844802.

[26] A. Devanand, M. Kraft, and I. A. Karimi. Optimal site selection for modular nuclear power plants. *Comput. Chem. Eng.*, 125:339–350, 2019. doi:10.1016/j.compchemeng.2019.03.024.

[27] A. Eibeck, M. Q. Lim, and M. Kraft. J-Park Simulator: An ontology-based platform for cross-domain scenarios in process industry. *Comput. Chem. Eng.*, 131:106586, 2019. doi:10.1016/j.compchemeng.2019.106586.

[28] A. Eibeck, A. Chadzynski, M. Q. Lim, L. K. Aditya, L. Ong, A. Devanand, G. Karmakar, S. Mosbach, R. Lau, I. A. Karimi, E. Y. S. Foo, and M. Kraft. A parallel world framework for scenario analysis in knowledge graphs. *Data-Centric Engineering*, 1:e6, 2020. doi:10.1017/dce.2020.6.

[29] F. Farazi, J. Akroyd, S. Mosbach, P. Buerger, D. Nurkowski, M. Salamanca, and M. Kraft. OntoKin: An ontology for chemical kinetic reaction mechanisms. *J. Chem. Inf. Model.*, 60(1):108–120, 2020. doi:10.1021/acs.jcim.9b00960.

[30] F. Farazi, M. Salamanca, S. Mosbach, J. Akroyd, A. Eibeck, L. K. Aditya, A. Chadzynski, K. Pan, X. Zhou, S. Zhang, M. Q. Lim, and M. Kraft. Knowledge graph approach to combustion chemistry and interoperability. *ACS Omega*, 5 (29):18342–18348, 2020. doi:10.1021/acsomega.0c02055.

[31] M. Foth, B. Bajracharya, R. Brown, and G. Hearn. The Second Life of urban planning? Using NeoGeography tools for community engagement. *Journal of Location Based Services*, 3(2):97–117, 2009. doi:10.1080/17489720903150016.

[32] J. Gil. City Information Modelling: A Conceptual Framework for Research and Practice in Digital Urban Planning. *Built Environment*, 46(4):501–527, Dec. 2020. doi:10/ghqp88.

[33] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014. doi:10.1007/s10817-014-9305-1.

[34] M. F. Goodchild. Citizens as voluntary sensors: Spatial data infrastructure in the world of web 2.0. *Int. J. Spatial Data Infrastructures Res.*, 2:24–32, 2007.

[35] G. Gröger, T. H. Kolbe, C. Nagel, and K. H. Häfele. Ogc city geography markup language (citygml) en-coding standard, 2012. URL `https://www.ogc.org/standards/citygml`. Accessed 23rd November, 2022.

[36] G. B. Hall, R. Chipeniuk, R. D. Feick, M. G. Leahy, and V. Deparday. Community-based production of geographic information using open source software and web 2.0. *International Journal of Geographical Information Science*, 24(5):761–781, 2010. doi:10.1080/13658810903213288.

[37] I. Herman. RDF graph literals and named graphs, 2022. URL https://www.w3.org/2009/07/NamedGraph.html#named-graphs. Accessed 24th November, 2022.

[38] I. Herman. OWL 2 web ontology language, 2022. URL https://www.w3.org/TR/owl2-overview/. Accessed 24th November, 2022.

[39] HermiT. Hermit github code repository, 2022. URL https://github.com/phillord/hermit-reasoner. Accessed 24th November, 2022.

[40] P. Hitzler, M. Krötzsch, and S. Rudolph. *Foundations of Semantic Web Technologies*. Textbooks in Computing. Chapman & Hall/CRC, 2009. ISBN 978-1-4200-9050-5.

[41] D. Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books, 1999. ISBN 9780465026562.

[42] A. Hudson-Smith, R. Milton, J. Dearden, and M. Batty. Virtual cities: Digital mirrors into a recursive world, 2007. CASA Working Paper 125, available at https://www.ucl.ac.uk/bartlett/casa/publications/2007/dec/casa-working-paper-125.

[43] A. Hudson-Smith, A. Crooks, M. Gibin, R. Milton, and M. Batty. Neogeography and web 2.0: concepts, tools and applications. *Journal of Location Based Services*, 3(2):118–145, 2009. doi:10.1080/17489720902950366.

[44] E. Husserl, L. Landgrebe, J. Churchill, and K. Ameriks. *Experience and Judgment*. Northwestern University studies in phenomenology & existential philosophy. Northwestern University Press, 1973. ISBN 9780810105959.

[45] R. Jain, M. Johnson, A. Albizri, and G. M. Elias. An open system architecture framework for interoperability. *International Journal of Business Information Systems*, 41 (4):423–452, 2022. doi:10.1504/ijbis.2022.127573.

[46] Java - Interface Runnable, 2022. URL https://docs.oracle.com/javase/8/docs/api/index.html?java/lang/Runnable.html. Accessed 11th January, 2023.

[47] JUNG - Java Universal Network/Graph Framework, 2010. URL https://jung.sourceforge.net. Accessed 11th January, 2023.

[48] D. Kolas and T. Self. Spatially-augmented knowledgebase. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *The Semantic Web*, pages 792–801, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76298-0. doi:10.1007/978-3-540-76298-0_57.

[49] N. Krdzavac, S. Mosbach, D. Nurkowski, P. Buerger, J. Akroyd, J. Martin, A. Menon, and M. Kraft. An ontology and semantic web service for quantum chemistry calculations. *J. Chem. Inf. Model.*, 59(7):3154–3165, 2019. doi:10.1021/acs.jcim.9b00227.

[50] T. Kuhn. *The Structure of Scientific Revolutions*. Donald F. Koch American Philosophy Collection. University of Chicago Press, 1970. ISBN 9780226458038. URL https://books.google.com.sg/books?id=dJHuAAAAMAAJ.

[51] P. Langley, J. E. Laird, and S. Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, 2009. doi:10.1016/j.cogsys.2006.07.004.

[52] M. Q. Lim, X. Wang, O. Inderwildi, and M. Kraft. *The World Avatar—A World Model for Facilitating Interoperability*, pages 39–53. Springer International Publishing, Cham, 2022. ISBN 978-3-030-86215-2. doi:10.1007/978-3-030-86215-2_4.

[53] P. C. Linskey and M. Prud'hommeaux. An in-depth look at the architecture of an object/relational mapper. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, page 889–894, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936868. doi:10.1145/1247480.1247581.

[54] A. Malhotra, M. Shamovich, J. Frisch, and C. van Treeck. Urban energy simulations using open CityGML models: A comparative analysis. *Energy and Buildings*, 255: 111658, 2022. doi:10.1016/j.enbuild.2021.111658.

[55] Ministry of Sustainability and the Environment. Zero waste masterplan Singapore, 2019. URL https://www.mse.gov.sg/resources/zero-waste-masterplan.pdf. Accessed 22th November, 2022.

[56] M. Musen. The Protégé project: A look back and a look forward. *AI Matters*, 1(4), 2015. doi:10.1145/2757001.2757003.

[57] L. Ong, G. Karmakar, J. Atherton, X. Zhou, M. Q. Lim, A. Chadzynski, L. Li, X. Wang, and M. Kraft. Embedding energy storage systems into a dynamic knowledge graph. *Industrial & Engineering Chemistry Research*, 61(24):8390–8398, 2022. doi:10.1021/acs.iecr.1c03838.

[58] OWASP - Input Validation, 2021. URL https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html. Accessed 23rd December, 2022.

[59] OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition), 2012. URL https://www.w3.org/TR/owl-mapping-to-rdf/. Accessed 23rd December, 2022.

[60] OWL Web Ontology Language Parsing OWL in RDF/XML, 2004. URL https://www.w3.org/TR/owl-parsing/. Accessed 23rd December, 2022.

[61] C. Pautasso, O. Zimmermann, M. Amundsen, J. Lewis, and N. Josuttis. Microservices in practice, part 2: Service integration and sustainability. *IEEE Software*, 34 (2):97–104, 2017. doi:10.1109/MS.2017.56.

[62] A. Perera, K. Javanroodi, and V. M. Nik. Climate resilient interconnected infrastructure: Co-optimization of energy systems and urban morphology. *Applied Energy*, 285:116430, 2021. doi:10.1016/j.apenergy.2020.116430.

[63] Plato and I. Richards. *Plato's Republic*. Cambridge University Press, 1966. ISBN 9780521093590.

[64] P. Poinet, D. Stefanescu, and E. Papadonikolaki. Collaborative workflows and version control through open-source and distributed common data environment. In E. Toledo Santos and S. Scheer, editors, *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*, pages 228–247, Cham, 2021. Springer International Publishing. ISBN 978-3-030-51295-8.

[65] J. T. Pollock and R. Hodgson. Adaptive information - improving business through semantic interoperability, grid computing, and enterprise integration. In *Wiley series in systems engineering and management*, 2004.

[66] W. J. Radermacher. *Official Statistics 4.0: The Era of Digitisation and Globalisation*, pages 119–156. Springer International Publishing, Cham, 2020. ISBN 978-3-030-31492-7. doi:10.1007/978-3-030-31492-7_4.

[67] RDF 1.1 Concepts and Abstract Syntax, 2014. URL https://www.w3.org/TR/rdf11-concepts/. Accessed 23rd December, 2022.

[68] RDF 1.1 Concepts and Abstract Syntax - Blank Nodes, 2014. URL https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes. Accessed 23rd December, 2022.

[69] RDF 1.1 N-Triples - A line-based syntax for an RDF graph, 2014. URL https://www.w3.org/TR/n-triples/. Accessed 23rd December, 2022.

[70] RDF 1.1 Semantics - Skolemization (Informative), 2014. URL https://www.w3.org/TR/rdf11-mt/#skolemization-informative. Accessed 23rd December, 2022.

[71] S. Roche, B. Mericskay, W. Batita, M. Bach, and M. Rondeau. WikiGIS basic concepts: Web 2.0 for geospatial collaboration. *Future Internet*, 4(1):265–284, 2012. doi:10.3390/fi4010265.

[72] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3 edition, 2010.

[73] T. Savage, J. Akroyd, S. Mosbach, N. Krdzavac, M. Hillman, and M. Kraft. Universal digital twin: Integration of national-scale energy systems and climate data. *Data-Centric Engineering*, 3:e23, 2022. doi:10.1017/dce.2022.22.

[74] Z. Shi, H. Silvennoinen, A. Chadzynski, A. von Richthofen, M. Kraft, S. Cairns, and P. Herthogs. Defining archetypes of mixed-use developments using google maps api data. *Environment and Planning B: Urban Analytics and City Science*, 2022. doi:10.1177/23998083221141428.

[75] P. Shukla, J. Skea, E. C. Buendia, and V. Masson-Delmotte, H.- O. Pörtner, D. C. Roberts, P. Zhai, R. Slade, S. Connors, R. van Diemen, M. Ferrat, E. Haughey, S. Luz, S. Neogi, M. Pathak, J. Petzold, J. Portugal Pereira, P. Vyas, E. Huntley, K. Kissick, M. Belkacemi, J. Malley. IPCC, 2019: Summary for Policymakers. In *Climate Change and Land: An IPCC Special Report on Climate Change, Desertification, Land Degradation, Sustainable Land Management, Food Security, and Greenhouse Gas Fluxes in Terrestrial Ecosystems*. IPCC, 2019.

[76] H. Silvennoinen, A. Chadzynski, F. Farazi, Z. Shi, A. Grisiute, A. von Richthofen, S. Cairns, M. Kraft, and P. Herthogs. Multi-criteria site selection using an ontology: the OntoZoning ontology of zones, land uses and programmes for Singapore, 2022. Submitted for publication. Preprint available at https://como.ceb.cam.ac.uk/preprints/295/.

[77] M. Sir, Z. Bradac, and P. Fiedler. Ontology versus database. *IFAC-PapersOnLine*, 48(4):220–225, 2015. 13th IFAC and IEEE Conference on Programmable Devices and Embedded Systems.

[78] SPARQL 1.1 Query Language - Syntax for Blank Nodes, 2013. URL https://www.w3.org/TR/sparql11-query/#QSynBlankNodes. Accessed 10th January, 2023.

[79] SPARQL 1.1 Query Language - Treatment of Blank Nodes, 2013. URL https://www.w3.org/TR/sparql11-query/#BGPsparqlBNodes. Accessed 10th January, 2023.

[80] SPARQL 1.1 Update - Drop Operation, 2013. URL https://www.w3.org/TR/sparql11-update/#def_dropOperation. Accessed 23rd December, 2022.

[81] A. Stadler, C. Nagel, G. König, and T. H. Kolbe. *Making Interoperability Persistent: A 3D Geo Database Based on CityGML*, pages 175–192. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-87395-2. doi:10.1007/978-3-540-87395-2_11.

[82] Stanford University. What are some knowledge graph inference algorithms?, 2022. URL https://web.stanford.edu/~vinayc/kg/notes/What_Are_Some_Inference_Algorithms.html. Accessed 24th November, 2022.

[83] E. P. Trindade, M. P. F. Hinnig, E. M. da Costa, J. S. Marques, R. C. Bastos, and T. Yigitcanlar. Sustainable Development of Smart Cities: A Systematic Review of the Literature. *Journal of Open Innovation: Technology, Market, and Complexity*, 3 (1), Dec. 2017. doi:10/ggfr6n.

[84] United Nations Security Council. Climate change 'biggest threat modern humans have ever faced', world-renowned naturalist tells security council, calls for greater global cooperation, 2021. URL https://press.un.org/en/2021/sc14445.doc.htm. Accessed 22th November, 2022.

[85] A. von Richthofen, P. Herthogs, M. Kraft, and S. Cairns. Semantic City Planning Systems (SCPS): A literature review. *Journal of Planning Literature*, 2022. doi:10.1177/08854122211068526. In press.

[86] W3C. Owl api, 2022. URL https://www.w3.org/2001/sw/wiki/OWLAPI. Accessed 24th November, 2022.

[87] U. Winkelhake. *Roadmap for Sustainable Digitisation*, pages 127–178. Springer International Publishing, Cham, 2018. ISBN 978-3-319-71610-7. doi:10.1007/978-3-319-71610-7_6.

[88] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe. 3dcitydb - a 3d geodatabase solution for the management, analysis, and visualization of semantic 3d city models based on citygml. *Open Geospatial Data, Software and Standards*, 3(1):5, May 2018. doi:10.1186/s40965-018-0046-7.

[89] C. Zhang, A. Romagnoli, L. Zhou, and M. Kraft. Knowledge management of eco-industrial park for efficient energy utilization through ontology-based approach. *Applied Energy*, 204:1412–1421, 2017. doi:10.1016/j.apenergy.2017.03.130.

[90] X. Zhou, D. Nurkowski, A. Menon, J. Akroyd, S. Mosbach, and M. Kraft. Question answering system for chemistry—a semantic agent extension. *Digital Chemical Engineering*, 3:100032, 2022. doi:10.1016/j.dche.2022.100032.