School of Earth and Planetary Sciences

Utilising Semantic Web Technologies for Improved Road Network Information Exchange

Michael Georg Niestroj

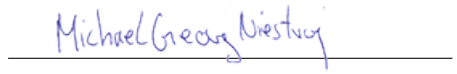
This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

September 2021

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: 

Michael Georg Niestroj

September 2021

'*Few are those who see with their own eyes and feel with their own hearts.*'

— Albert Einstein

Statement of Contributors

Supervisors:

Dr David A. McMeekin (Primary Supervisor)

Dr David Belton (Secondary Supervisor)

Dr Petra Helmholz (Secondary Supervisor)

Funding:

This research is supported by the Australian Government through the Australian Research Council LP160100524, Curtin University and the Sustainable Built Environment National Research Centre Project 2.33 and its partners. The core partners are Swinburne University of Technology, Queensland University of Technology, the University of Melbourne, the New Zealand Transport Agency, Main Roads Western Australia, Roads and Maritime Services and Aurecon Australasia PTY LTD.

Signature: 

Michael Georg Niestroj

Date: 10^{th} September 2021

Signature: 

Dr David A. McMeekin

Date: 10^{th} September 2021

Statement of Contributions

The following peer-reviewed publications are referenced in the thesis.

Paper #1:

Niestroj, M. G., McMeekin, D. A., and Helmholz, P.: Overview of standards towards road asset information exchange, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLII-4, 443–450, https://doi.org/10.5194/isprs-archives-XLII-4-443-2018, 2018.

Paper #2:

Niestroj, M. G., McMeekin, D. A., Helmholz, P., and Kuhn, M.: A proposal to use Semantic Web Technologies for improved road network information exchange, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4, 147–154, https://doi.org/10.5194/isprs-annals-IV-4-147-2018, 2018.

Paper #3:

Niestroj, M. G., McMeekin, D. A., and Helmholz, P.: Introducing a framework for conflating road network data with Semantic Web Technologies, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-2/W5, 231–238, https://doi.org/10.5194/isprs-annals-IV-2-W5-231-2019, 2019.

Candidate Attributions: Michael Georg Niestroj

Task / Publication	Paper #1	Paper #2	Paper #3
Conception & design	X	X	X
Acquisition of data & method	X	X	X
Data conditioning & manipulation	X	X	X
Analysis & statistical method	X	X	X
Interpretation & discussion	X	X	X
Final approval	X	X	X

Co-Author Attributions: Dr David McMeekin

Task / Publication	Paper #1	Paper #2	Paper #3
Conception & design	X	X	X
Acquisition of data & method			
Data conditioning & manipulation			
Analysis & statistical method			
Interpretation & discussion	X	X	X
Final approval	X	X	X

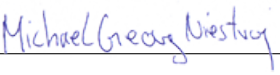
Co-Author Attributions: Dr Petra Helmholz

Task / Publication	Paper #1	Paper #2	Paper #3
Conception & design	X	X	X
Acquisition of data & method			
Data conditioning & manipulation			
Analysis & statistical method			
Interpretation & discussion	X	X	X
Final approval	X	X	X

Co-Author Attributions: Associate Professor Michael Kuhn

Task / Publication	Paper #2
Conception & design	X
Acquisition of data & method	
Data conditioning & manipulation	
Analysis & statistical method	
Interpretation & discussion	X
Final approval	X

I acknowledge that these represent my contributions to the above research output:

Signature: 

Michael Georg Niestroj

Date: 10^{th} September 2021

Signature: 

Dr David A. McMeekin

Date: 10^{th} September 2021

Signature: 

Dr Petra Helmholz

Date: 10^{th} September 2021

Signature: 

Associate Professor Michael Kuhn

Date: 10^{th} September 2021

Acknowledgements

The completion of my dissertation would not have been possible without the support and nurturing of my supervisor, Dr David McMeekin, and my co-supervisors, Dr Petra Helmholz and Dr David Belton. I have learnt a lot from you, and your effort and your guidance will be a foundation of my future career path from now on. Special thanks to Emeritus Professor Geoff West, who was part of the team that applied for the overall research proposal of the LP160100524-*CONie open standard* research project, which made this research possible.

Many thanks to Professor Robin Drogemuller, who pointed out the ontology achievements of this thesis in a practical use case of a route planner driven by Semantic Web technologies. I am also grateful to my thesis chair, Associate Professor Michael Kuhn, who supported me with the completion of my research proposal.

I would like to acknowledge the assistance of Professor Tele Tan, who motivated me at the right moment to begin the writing of this dissertation. I am also grateful to Dr Ivana Ivanova, who spent her time working with me on the data provenance concept, to Dr Jeremy Siao Him Fa, who introduced me to Semantic Web technologies, and to Tristan Reed, who helped me with technical printing issues.

A special thanks goes to Dr Marcin Wolski, Dr Claudia De Los Rios Perez, Dr Ashty Saleem, Dr Yousif Mousa, Tong Ding, Yuchen Liu, Eman Albalawi, Dr Dimitri Bulatov and Dr Rebecca Ilehag for their friendship, which is priceless to me. A shout-out to my night buddies, Dr Heng Zhou and Dr Keone Kelobonye;

no comments here, as the night sessions were always very productive. I also had the great pleasure of sharing an office with Dr Richard Palmer, Dr Hoang Long Nguyen, Dr Mustafa Hamoodi, Dr Mortaza Rezae, Roula Zougheibe, Jayita Chakraborty, Saseeka Wijesekera and Cui Hengyang.

I would also like to extend my deepest gratitude to Professor Thomas Felderhoff, who prepared me well for this successful journey, and to Professor Burkhard Igel, who motivated me to contact Professor Tele Tan and Emeritus Professor Geoff West, which helped me achieve the milestone of studying at Curtin.

I would like to thank my friends, relatives, family members and everyone who wishes to be mentioned here. A special thanks goes to my mother, who always supported me and was with me through the highs and lows. None of this would have been possible without the mental support of my wife, Xi, my son, Bosie Jacob, and my daughter Bobbi Yvonne; they were always there for me, and their love is a fundamental driver of my life.

Abstract

Road asset data harmonisation is a challenge for the Australian road and transport authorities considering their heterogeneous data standards, data formats and tools. Classic data harmonisation techniques require huge databases with many tables, a unified metadata definition and standardised tools to share data with others. In order to find a better way to harmonise heterogeneous road network data, this dissertation will use Semantic Web technologies to investigate fast and efficient road asset data harmonisation. The corresponding findings will lead to the development of a new route planning framework driven by Semantic Web technologies.

Therefore, new ontologies are designed for the management of various road asset and road-asset-related datasets from Main Roads Western Australia (MRWA), the Western Australian Land Information Authority (Landgate), Western Power and OpenStreetMap (OSM), which include road sections, overhead power lines, regulatory signs, intersections, speed limits, traffic signal sites and road stopping places. The ontologies are populated with a newly developed script that is customised for each dataset individually. The script processes the data entries, analyses relations to other data entries and writes the data into the Resource Description Framework (RDF) ontology format. From this, separate data individuals can be created for the location of each road asset in the form of multi-line strings, line strings and points. As part of the data creation process, a novel approach is introduced that integrates data provenance for each data entry to in-

form about the origin of the data entry, the individual use case and the associated location entities.

This research includes three studies that investigate heterogeneous road network data, Semantic Web technologies and route planning.

The first study compares road network centreline data from MRWA and Landgate to give information about the location discrepancies of the same road networks provided by different authorities. The comparison is supported by a newly developed algorithm that applies seven carefully formulated translation methods that enable the translation of MRWA intersections and road sections to the layout of the Landgate road network. The translated MRWA road network harmonises with the Landgate road asset data, which enables a numeral distance evaluation of the translated road section vertex and intersection features.

The second study adapts a newly designed ontology that acts as a data warehouse for the newly created MRWA, Landgate and OSM ontologies. Part of the ontology is a set of new integrated semantic rules that enable data conflation among different datasets and activate a trust score for road asset data sources. The data conflation approach with provenance support is a first step towards road asset data harmonisation; it introduces novel data access for machines to understand the same meaning of road asset data. The trust score is an important factor for data on the web, as it allows the classification of datasets with respect to their lineage.

The third and final study uses the road asset and road-asset-related individuals of the road network ontologies to populate a new route planning framework to demonstrate the efficiency of the Semantic Web for the road network in a possible industrial application. The development provides an interface to save planned routes in RDF format as well as to read a route back from RDF format into the route planner. The route planner supports the configuration of customisable constraints for route planning, e.g. turning circles, regulatory signs, traffic signal sites, rail crossings, road stopping places and overhead power lines. Although

overhead power lines are well known for being a dangerous and deadly obstacle and are often involved in accidents with heavy vehicles, their integration in a route planning framework has not been considered by others before.

The evaluation of each study uses various tests and road network sample selections to demonstrate each approach's accuracy and industrial usage.

Contents

List of Figures

List of Tables

List of Source Codes

List of Algorithms

List of Abbreviations

Chapter 1

Introduction

1.1 Introduction and Context of the Study

As the main objective, this dissertation investigates road asset data management from a new perspective utilising Semantic Web technologies that enable linking, sharing and reusing data with other datasets effortlessly on the Web in a machine-readable format (Berners-Lee et al., 2001; Bizer et al., 2009; Qi et al., 2013). The data-linking feature of the Semantic Web is a key component to conflate (by means of merging) datasets with the same meaning from heterogeneous data sources. Road network centreline data will be analysed for the same road network selections of different authorities to identify possible discrepancies in the road network representation. The findings will be tested with case studies, which will lead to the development of a new route planning framework for the road network in Western Australia driven by Semantic Web technologies.

The proposed route planner contributes to the second objective with an approach to overcome the problem of crossing overhead power lines, which is a well-known high risk obstacle for heavy vehicles (Koustellis et al., 2011; Crow, 2009; Cawley & Homce, 2003) and, to the best of the author's knowledge, has not been examined by other researchers in the literature.

Two of the most used research methods in science include qualitative and quantitative methods (Borrego et al., 2009). Qualitative research pursues an objective analysis of collected data-sets, where variables are complex and analysed by a researcher to find information within a dataset (Johnston, 2010). Quantitative research applies a statistical analysis to numerical datasets, whereby tools can be used to evaluate the data (Dörnyei, 2007). A combination of both research methods is called a mixed research method and combines the two mentioned methods at either the data collection or the data analysis levels (Borrego et al., 2009; Dörnyei, 2007).

Case studies can be processed with either a qualitative or a quantitative research method (Johnston, 2010). This research will undertake a mixed research method, as the case study that compares Main Roads Western Australia (MRWA) road network data with related Western Australian Land Information Authority (Landgate) data uses a quantitative research method that evaluates the translation data with a computer-supported numerical data analysis, and the route planner case study is a qualitative approach in which the benefits and disadvantages of the Semantic Web for the road network will be identified.

1.2 Statement of the Problem

Road asset data harmonisation is necessary for Australian road and transport agencies, as each authority uses its own data formats, data standards and asset management tools (Martin et al., 2019). The problem of not having a unified road asset management system in Australia was identified at least 17 years ago by Li & Kumar (2003) yet an accepted solution for handling the problem of heterogeneous road asset data has not been adopted (Martin et al., 2019; Perumpilly et al., 2019; Kenley et al., 2014; Austroads, 2016).

A recent governmental trial included the release of a data standard for road management and investment in Australia and New Zealand by Austroads, the main Australasian authority for road transport and traffic (Martin et al., 2019).

Once an overall accepted solution is in place, annual cost savings of $65 to $130 million can be achieved with a unified road asset data standard that enables data transfer between different software systems (Gardner, 2015). Harmonising Australia's road asset data to a unified data format is challenging, as this has been in demand for many years (Li & Kumar, 2003; Martin et al., 2019). Data harmonisation approaches often consider a common valid metadata specification to bypass the problem of heterogeneous metadata (Agarwal et al., 2013). This sounds simple but is, in reality, associated with difficulties considering that each authority is built on its own developed standards, tools and formats (Martin et al., 2019). For instance, the Integrated Road Information System (IRIS) has been MRWA's road asset management system since 2003 (Mihai & Robertson, 2003; Goodlet, 2020). While visiting the MRWA head office and being introduced to their internal asset management and work flow, it was mentioned that their asset management is based on IRIS and that transferring and migrating to a new unified standard is critical from a stakeholder perspective. It was also highlighted that such an effort is very hard to realise and that a solution can cost several million dollars just to migrate MRWA's road assets to a new standard, such as the Austroads road asset data harmonisation standard. Road and transport authorities operated by Australia's and New Zealand's state and territory governments are confronted with similar challenges (Martin et al., 2019; Gardner, 2015).

1.3 Thesis Statement and Objectives

The utilisation of Semantic Web technologies could bridge the gap of a lack of unified data standards, tools and data formats for Australian road and transport authorities; the Semantic Web is a key component for efficient and fast road asset data harmonisation in Australia.

To demonstrate the correctness of the thesis statement above, the following research objectives will be addressed:

Objective 1: Identification of data standards and ontologies for the representation of road networks with the target of delivering both a data model and ontology for route planning.

Objective 2: Conceptualisation of a road network data model and a road network ontology.

Objective 3: Development of a data processing approach to integrate road network data into the Resource Description Framework (RDF) format.

Objective 4: Utilisation of Semantic Web technologies to identify road assets that mean the same thing from different data sources, to classify a trust score for a road asset data source, and to connect surrounding road assets.

Objective 5: Comparison of road networks to identify similarities and differences in the location representation, of the same road networks provided from different sources.

Objective 6: Development of a Semantic Web technology-driven route planning framework for heavy vehicles considering hurdles, such as overhead power lines.

1.4 Scope of This Thesis

The scope of this thesis includes the identification of Semantic Web technologies in relation to road network data in Australia, with a focus on the datasets that are provided by the Western Australian data portal.[1] To demonstrate the ability of the Semantic Web to merge the data of independent data sources, datasets from MRWA, Landgate and OpenStreetMap (OSM) will be used with a data conflation approach.

[1] `https://data.wa.gov.au`

Overall, this thesis will explain the findings and approaches of three novel contributions, which include the migration of road network assets into newly designed ontologies, the translation of MRWA road sections and intersections into Landgate road assets and finally, the development of a novel route planner driven by Semantic Web technologies.

To migrate the road assets into the newly designed ontologies, arbitrary road network selections will be chosen. The ontology integration of road asset types will be limited to road sections, overhead power lines, stop and give way signs, traffic signals, speed limits, roundabouts, intersections, rail crossings and road stopping places. Road network selection datasets are prepared with the software QGIS[2] and saved in a unified coordinate reference system and data format, i.e. European Petroleum Survey Group projection 4326 – World Geodetic System 1984 (WGS84)[3] and GeoJSON[4]. An explanation of how the datasets were loaded into QGIS, the areas of interest were selected and the data were saved as a GeoJSON dataset will not be given, as this dissertation will focus on methods that explain what can be done with a given dataset and not on how to save data with a Geographic Information System (GIS) such as QGIS.

The translation of MRWA road network data into the Landgate data will inform about the differences in datasets that describe the same road centreline network provided by independent road authorities. The translated data will be integrated into ontologies for a data conflation approach with semantic rules. The translation results will not be used to inform about the map accuracy of the road assets, as the employed road asset datasets do not contain information about measurement uncertainties.

The route planner of this thesis will use Dijkstra (1959) as a base algorithm for the route planning between a start and a target node. Dijkstra's algorithm has been chosen for the route planning because it is commonly referred to as a base

[2] `https://qgis.or`
[3] `https://spatialreference.org/ref/epsg/wgs-84/`
[4] `https://geojson.org`

algorithm to determine the shortest path (Bast et al., 2016; Cherkassky et al., 1996; Gallo & Pallottino, 1988). The aim of this thesis is not to develop a better or faster route planning algorithm but to demonstrate an ability to combine available methods to form a new route planner for heavy vehicles considering overhead power lines, speed limits, turning circles, traffic signals, regulatory signs, rail crossings, road stopping places and road closures driven by Semantic Web technologies and to demonstrate the advantages of the Semantic Web for the road network.

To support the significant need for data harmonisation, novel semantic rules will be designed and applied to the newly designed road network ontologies so that data interoperability can be achieved with the road asset data of different sources.

1.5 Significance of the Study

The first significant outcome of this study is that the findings will identify the advantages and disadvantages of the Semantic Web for the contribution towards road asset data harmonisation within the Western Australian road network. The results can be then adapted for the road network data of other road and transport authorities. The second outcome contributes to the design of an ontology framework for road network data that can be used in the newly designed route planner. Furthermore, the third outcome of the study is to use the route planner to create routes on the fly and integrate live data, such as road closures and suburb information. A set of various constraints can be considered by the route planner, such as overhead power lines, left turns, right turns, stop signs, give way signs, traffic signal sites, rail crossings and road stopping places. The concept of taking overhead power lines into account in route planning is new and has not yet been investigated by other researchers. This demonstrates the integration of Semantic Web technologies for the road network with a possible industrial application implementation. Finally, this work will further support other researchers

in the field of Semantic Web technologies who are considering using the Semantic Web for road network and route planning approaches.

1.6 Thesis Structure

This thesis contains seven further chapters to deliver background information, to select datasets, to design provenance and ontology models, to prepare elementary content, to implement the methodology, to evaluate the findings and to conclude this study, as indicated in Figure 1.1.

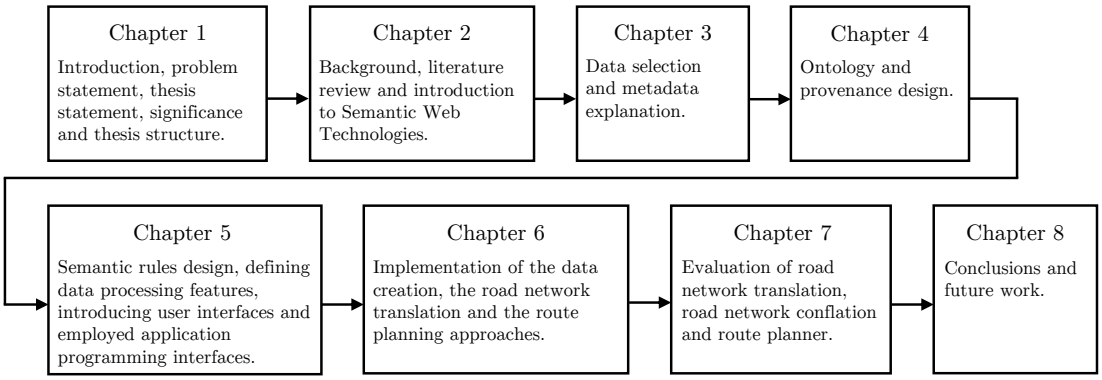


Figure 1.1: Structure of this thesis.

In Chapter 2, an introduction to data harmonisation and Semantic Web technologies will be given to provide the reader with the knowledge required to understand the technical details of this thesis and to present the area of expertise this work fits into. Furthermore, literature regarding the Semantic Web for road networks will be reviewed to inform about practices and activities closely related to this study that positively influenced this research.

In Chapter 3, the datasets will be introduced, and the related metadata information will be explained. Then, in Chapter 4, the newly designed ontologies for the road network conflation and the route planning approaches will be shown, and the provenance of each dataset will be visualised.

In Chapter 5, the semantic rules for the road network, the route planner programme features and the mock-ups to operate the route planner will be defined. The road network processing features will be explained, including the road network conflation approach, the road network translation features and the route planner. Related to the route planner's development, two application programming interfaces (APIs) will be shown that helped with the ontology and the road network processing.

In Chapter 6, the implementation of the methodology will be explained, which includes the ontology data creation of GeoJSON datasets, the translation of road network assets and the development of a route planner driver by Semantic Web technologies. The data creation process will use flowcharts to explain the data individuals' creation process for original and translated datasets. In addition, elementary ontology writing approaches that are processed in the data creation process for each data entry, such as to write features, entities and metadata individuals, will be explained. The road network translation approach will be introduced with a flowchart, and the route planner will be explained in detail with flowcharts, algorithms, source codes and supporting illustrations.

After that, in Chapter 7, the results of this study will be evaluated. In the scope of that, road network ontologies will be reasoned to evaluate the road network conflation process. Translated road networks will be analysed interactively on a digital map, and the processed translation distance information will be evaluated. The route planner will be tested for usability, and the resulting routes will be compared with route planners from Google and Microsoft Bing.

Finally, in Chapter 8, this study will conclude with an outlook that identifies further possible research activities.

Chapter 2

Background

2.1 Chapter Introduction

Acceptance and awareness of the Semantic Web has been steadily moving forward since its introduction by Berners-Lee et al. (2001). It is a key technique in solving data harmonisation problems. Few have tried to harmonise road network data with Semantic Web technologies. Although the use of Semantic Web technologies seems to be a popular tool for Intelligent Transportation Systems (ITS) (e.g. Fernandez et al., 2015; Sirin et al., 2007; Zhao et al., 2015, 2017; Provine et al., 2004b; Schlenoff et al., 2004; Marasovic & Marasovic, 2010; Hülsen et al., 2011; Niaraki & Kim, 2009; Provine et al., 2004a), its adoption for route planning approaches is rare (e.g. Niaraki & Kim, 2009; Szwed et al., 2012; Houda et al., 2010; De Oliveira et al., 2013; Corsar et al., 2015; Faaß, 2015). For heavy vehicles, no route planner that considers hazardous overhead obstacles, such as power lines, exists.

This chapter includes an introduction to data harmonisation and its use by governments. The Semantic Web with its relevant technologies will be explained regarding its use for road network applications; this significant contribution will provide critical knowledge about what the Semantic Web is and will further address Thesis Objective 1. This will lead the reader to route planning and an

introduction to Dijkstra's shortest path algorithm. Finally, the risk of overhead power lines for heavy vehicles will be discussed.

2.2 Data Harmonisation

The concept of data harmonisation has existed for more than 20 years (Patel & Sharma, 2017). It is a process to merge different datasets into one large dataset (Agarwal et al., 2013; Fichtinger et al., 2011), and is often supported by defining common valid metadata specifiers before the merge is conducted (Agarwal et al., 2013). A data harmonisation concept can also be seen as a data warehouse, as data from various heterogeneous sources come together into a single place (Patel & Sharma, 2017). It can be a challenge to find unified metadata definitions, as various data aspects need to be considered, such as completeness, life span, origin, quality and use case. Data harmonisation approaches are commonly limited to applications that share data that mean the same thing. For instance, the main organisation of Australia's road traffic and transport agencies, Austroads is in the process of implementing a data standard for road management and investment in Australia and New Zealand. The standard aims to harmonise the national road asset data across state borders (Martin et al., 2019).

2.2.1 Open Government Data

Many governments worldwide provide Open Government Data (OGD) that are available via online data portals. In 2012, four of the largest OGD portals were hosted by the United States (`www.data.gov`), the United Kingdom (`www.data.gov.uk`), France (`www.data.gouv.fr`) and Singapore (`www.data.gov.sg`) (Hendler et al., 2012). The OGD portal in Australia (`www.data.gov.au`) was established in 2013. What open data portals have in common, according to Parycek & Sachs (2010), is that their products are transparent, as information is available to everyone and users can interact with the service in collaboration

with the government. Nascimento et al. (2018); Linders & Wilson (2011) stated that access to OGD is a human and civil right.

As OGD providers can contribute their data in any data format, as no recommendation exists about a unified format (OECD, 2017). According to Tauberer (2014), one of the principles of OGD is that data should be provided in a format that can be processed by machines. The problem is that machines can process data in various common data formats, such as CSV, JPEG, JSON, GeoJSON, XLS, XLSX, XML, KML, ZIP, SHP and RDF without having a unified structure. For instance, two governmental authorities in Western Australia enable access to their road network centreline data through the Australian OGD, namely MRWA and Landgate. Both authorities enable public access to their road centreline datasets, but their underlying data structure is not in a unified format.

With the use of Semantic Web technologies, the problem of different data structures can be treated with semantic rules, as long as the same meaning of data can be assigned (Niknam & Karshenas, 2017).

2.3 Semantic Web

The Semantic Web is an extended version of the World Wide Web (WWW), with well-defined structured data with which computers can interpret the meaning of the contained information. Semantics is about understanding the meaning of something using language (Löbner, 2013). Since its public adoption, the Web has been accessible to everyone and is used by scientists especially on a daily basis to keep up to date with current global research activities (Allemang & Hendler, 2011).

For a working Semantic Web, access to a collection of information and inference rules must be available, as this can be further used to conduct automated reasoning. This kind of artificial intelligence is recognised as 'knowledge representation' and was researched before the Web was created (e.g. (Genesereth & Nilsson, 1987)). Therefore, the tasks of the Semantic Web are to express con-

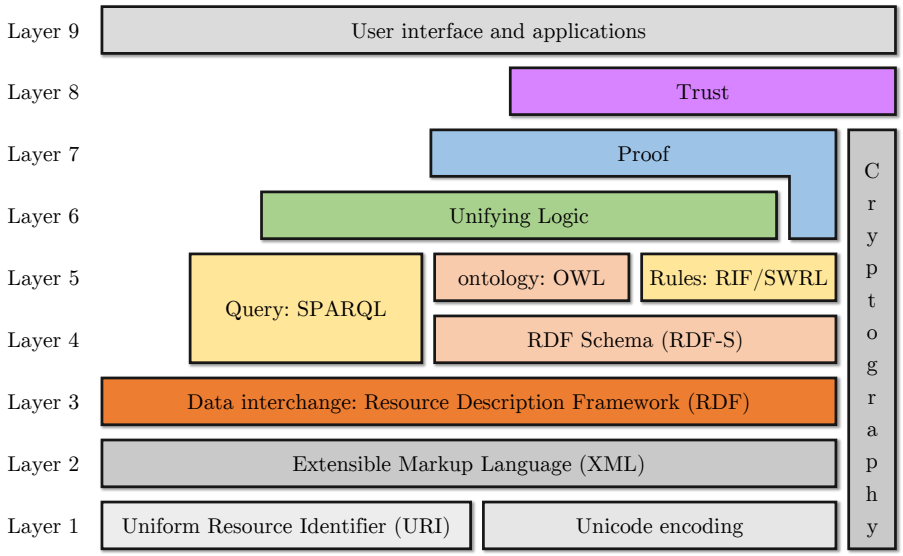


Figure 2.1: The Semantic Web technology stack (Berners-Lee, 2006; Gerber et al., 2008; Curé & Blin, 2014).

tent with a clear logic behind the data model and to be adaptable for external updates from any knowledge representation system. This means that one major challenge for the Semantic Web community is to define rules, which are used to make inferences, implement decisions and respond to questions. The logic has to cover many types of decisions, which include complicated and science-related tasks, and it must be able to describe complex object properties without getting stuck (Berners-Lee et al., 2001).

The Semantic Web Stack is used for a layer-based visualisation of recommended technologies for the Semantic Web, such as Uniform Resource Identifier (URI), Web Ontology Language (OWL), RDF, RDF Schema (RDF-S) and semantic rules (see Figure 2.1). The above-mentioned technologies will be explained in detail for the reader in the following sections of this thesis.

At the bottom of the stack Llayer 1), URIs are used to link to other documents on the Web, and guidance is given to display characters in the Unicode character set. In this dissertation, a sub-set of Unicode will be employed (American Stan-

dard Code for Information Interchange) that includes in total 127 characters, e.g. a–z, A–Z, 0–9, \, /, *, # and other standard characters.

Then, the Extensible Markup Language (XML) (Layer 2) and the RDF syntax (Layer 3) can be used to write RDF data. In practice, XML is barely used for ontology design, as it is cumbersome to apply. In the literature (e.g. Alexander, 2008), users prefer to write RDF in more efficient ways, such as with the Terse RDF Triple Language (Turtle).

Layer 4 introduces RDF-S, which, for instance, can be used to describe subclasses and sub-properties. Layer 5 is employed by both ontologies and semantic rules. With OWL, ontologies can be described more expressively than with RDF-S. Semantic rules can be written in the Rule Interchange Format and in the Semantic Web Rule Language (SWRL) and are used to add, change and remove data attributes.

The opportunity to query RDF data is given by the SPARQL Protocol and RDF Query Language (SPARQL) across Layers 4 and 5 (Curé & Blin, 2014). Layer 6 recommends the adoption of a unifying logic, which means that available RDF vocabularies can be included so that data with the same information can be shared seamlessly. Data provenance can be integrated at Layer 7 to keep track of the data's origin, updates and use case. Layer 8 is about trust, which can be information about who provided a dataset and how the data were collected (e.g. measured by governmental authorities or collected by community effort) as well as how trusted the ontology-delivered data are. Throughout Layers 1–8, cryptographic data encoding and decoding ensure the communication between each layer (Stinson, 2005). Layer 9 is on top of the Semantic Web technology stack and includes applications as well as user interfaces to interact with the given technologies.

The next section will provide further information about key elements of the Semantic Web, such as URIs, ontologies, OWL, RDF, RDF-S, linked data, ontology reasoning, SPARQL and data provenance.

2.3.1 URI

A URI is an overall valid declaration for any kind of resource on the Web, which can be abstract or physical. This could be, for instance, a document, an email address or a phone number. In the scope of this thesis, URIs will always contain a Hypertext Transfer Protocol (HTTP)[1] scheme and a host (e.g. example.org, w3.org, purl.org or dbpedia.org) and will refer to documents on the Web. This combination of scheme, host and path is also known as a namespace (see Figure 2.2). In the context of this thesis, a URI path will be provided in a hierarchical order (e.g. '/ontology/roadNetwork' and '/ontology/roadNetwork/roundabouts'), and a fragment will be used optionally to refer to specific content (e.g. '/ontology/roadNetwork/roads#RoadWithId1' and '/ontology/roadNetwork/roads#RoadWithNameA').

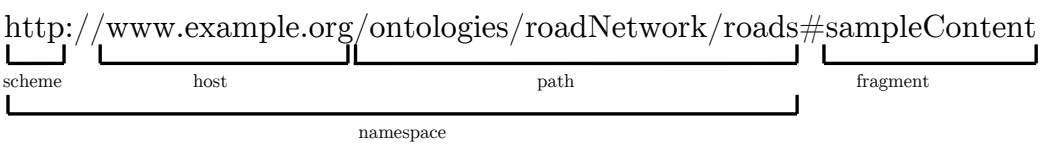


Figure 2.2: The structure shows a URI divided into the scheme, host, path, fragment and namespace.

2.3.2 Ontology

In this thesis, ontologies are related to computer science and describe an information system with logical relations, as defined by Guarino et al. (2009). In the first sense, an ontology is a logical theory that gives an 'explicit, partial account of a conceptualisation', and in the second sense, it is a 'synonym for conceptualisation' (Giaretta & Guarino, 1995, p. 6, p. 3). Borst (1997) introduced the shared aspect of a conceptualisation. Studer et al. (1998) merged Gruber's (1995)

[1] The HTTP protocol is an Internet Engineering Task Force (IETF) standard to retrieve documents online. A client requests information using an HTTP document address which is processed by an HTTP server (Berners-Lee, 1991; Belshe et al., 2015). HTTP documents are accessed by internet users on a daily base while navigating the internet with an internet browser (e.g. Mozilla Firefox, Google Chrome, Microsoft Edge and Apple Safari).

and Borst's (1997) definitions into 'an ontology is a formal, explicit specification of a shared conceptualisation' (p. 184), as *shared* reflects that an ontology is meant to be accessible by other external instances. To understand the above-mentioned definitions of 'conceptualisation' and 'formal, explicit specification', their meanings will be further explained.

Conceptualisation: A conceptualisation was defined by Genesereth & Nilsson (1987) in the context of artificial intelligence as defining a structure, $< D, \mathbf{R} >$m with a domain, D, and a set of (conceptual) relative relations, $\mathbf{R}$, on D (Guarino, 1998). A conceptualisation can be simplified and expressed as an approach to describing world objects, functions and relations (Nilsson, 1991).

Formal, explicit specification: *Formal* means here that an ontology has to be in a machine-readable format, and *explicit specification* stands for an accurate definition of concepts and their related constraints (Studer et al., 1998).

Ontologies are used to model a knowledge domain with the use of classes, properties and the relations between the classes. They are usually written in languages that allow one to define an abstract data model, such as OWL.

2.3.3 OWL

Web Ontology Language is a computational logic-based declarative language for the Semantic Web recommended by the WWW Consortium (W3C) for ontology development on the Web. The language supports the design of Semantic Web documents (known as ontologies) and can be written along with RDF information. Web Ontology Language 2 is an extended version of OWL 1 and inherits the language features, design decision and use cases. With OWL 2, it is possible to express knowledge about things, groups of things and relations between things (Golbreich & Wallace, 2012).

Reasoners can be used to infer additional related ontology information (also called knowledge), respecting that OWL statements can be either true or false. For instance, if a *set* of statements $\mathbf{A}$ includes a statement B that requires that

all statements of **A** are true, then B is also true (see Algorithm 2.1). The OWL statements are usually applied to assign world objects into categories (e.g. 'a give way sign is a sign') and to express relations (e.g. '*road a*, *road b* and *stop sign* are part of the same intersection') (Hitzler et al., 2012).

Algorithm 2.1: Example of a nested OWL statement (pseudo code).

```
1 A ⟵ set of A statements
2 B ⟵ statement (included in A)
3 if all A statements in A true then
4 |   B ⟵ true
5 end
```

All atomic entries, e.g. *intersections*, *roads*, *road a*, *road b*, *give way sign* and *stop sign*, are called entities. Objects are individuals, e.g. *road a*, *road b*, *give way sign* and *stop sign*. Categories are classes, e.g. sign and road, and relations are properties, e.g. part of the same intersection. Properties are further separated into object properties for object-to-object relations, e.g. as an intersection is connected to a road. Data-type properties are used to assign a data value to an object, e.g. assigning a road name to a road, and annotation properties are used to encode information about the ontology, e.g. to read the metadata of an individual (Hitzler et al., 2012).

A powerful OWL feature is the use of constructors to merge entity names into expressions. For example, the classes 'sign' and 'intersection' can be combined into the class expression 'signed intersections'. The resulting expression can then be used to classify intersections with road signs.

2.3.4 RDF

The RDF is recommended by the W3C to encode structured metadata that can be shared and reused (Miller, 2001). The RDF infrastructure can be employed for interoperability between different metadata elements. Vocabularies are used in RDF to describe a set of metadata elements or a set of properties. Anyone can

provide RDF vocabularies. Predefined vocabularies are available to be reused for large-scale compatibility (Miller, 2001).

The Linked Open Vocabularies[2] data portal evaluates and counts the use of ontologies among other ontologies. The following list summarises some of the most commonly used RDF vocabularies:

- Dublin Core Metadata Element Set:[3] the specification contains 15 metadata properties (title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage and rights) and is standardised as IETF RFC 5013, American National Standards Institute/National Information Standards Organization Standard Z39.85-2007, and International Organization for Standardization (ISO) Standard 15836:2009.

- Dublin Core Metadata Initiative (DCMI) Metadata Terms:[4] the specification contains all DCMI-maintained vocabularies (e.g. properties, vocabulary-encoding schemas, syntax-encoding schemas, and classes) and also includes the Dublin Core Metadata Element Set.

- Friend of a Friend:[5] the specification contains vocabularies to link people and information on the Web using classes (e.g. agent, document, online account, organisation, person and project) and properties (e.g. account name, age, current project, family name, first name, knows, member, past project, primary topic, status and title).

- Simple Knowledge Organization System:[6] the specification is a W3C recommendation to share and link organisation systems on the Web using classes (e.g. collection, concept, schema and ordered collection) and properties

[2] `https://lov.linkeddata.es/dataset/lov/`
[3] dce: `http://dublincore.org/documents/dces`
[4] dcterms: `http://dublincore.org/documents/dcmi-terms`
[5] foaf: `http://xmlns.com/foaf/spec/20140114.html`
[6] skos: `http://www.w3.org/2009/08/skos-reference/skos.html`

(e.g. alternative label, change note, definition, example, member, notation and semantic relation).

- Creative Commons Rights Expression Language:[7] the specification is used to describe copyright licenses using classes (e.g. work, license, jurisdiction, permission, requirement and prohibition), license properties (e.g. permit, legal code and deprecation date) and work properties (e.g. attribution name, attribution URL and use guidelines for the work).

- Schema.org vocabulary:[8] the specification aims to provide metadata for structured data on the Web (e.g. creative work, audio/video/image objects, events, health and medical types, organisations, persons, places, products, reviews, and actions) and was founded by the companies Google, Microsoft, Yahoo and Yandex.

- Provenance (PROV) ontology:[9] the specification is recommended by the W3C to interchange and represent provenance on the Web using classes (e.g. entity, activity, agent, collection, bundle, person, organisation and location), properties (e.g. was generated by, was derived from, was attributed to, used, had a primary source and was influenced) and restrictions (e.g. ended at time, was invalidated by and was ended by).

- Basic WGS84 Geo Positioning:[10] the specification provides the opportunity to define geographic coordinates using WGS84 in longitude and latitude on the Web.

In the later sections of this thesis, examples will be shown with the use of some of the described RDF vocabularies, e.g. 'DCMI Metadata Terms', 'PROV ontology' and 'Basic WGS84 Geo Positioning'.

[7] cc: `http://creativecommons.org/ns`
[8] schema: `http://schema.org/docs/about.html`
[9] prov: `http://www.w3.org/TR/prov-o/`
[10] geo: `http://www.w3.org/2003/01/geo/`

2.3.4.1 RDF Basics

To understand the principles of RDF, we will introduce knowledge for ontology design. The RDF requires expressions of simple statements called RDF triples that are written in the form <subject><predicate><object>. For instance, triples regarding a road network can be as follows:

<Kent Street>	<is a>	<road>
<Kent Street>	<is located in>	<Bentley>
<Kent Street>	<has part>	<Intersection A>
<University Boulevard>	<is a>	<Road>
<Intersection A>	<is an>	<Intersection>
<Intersection A>	<has latitude>	<–32.001275°>
<Intersection A>	<has longitude>	<115.887242°>
<Intersection A>	<is part of>	<University Boulevard>

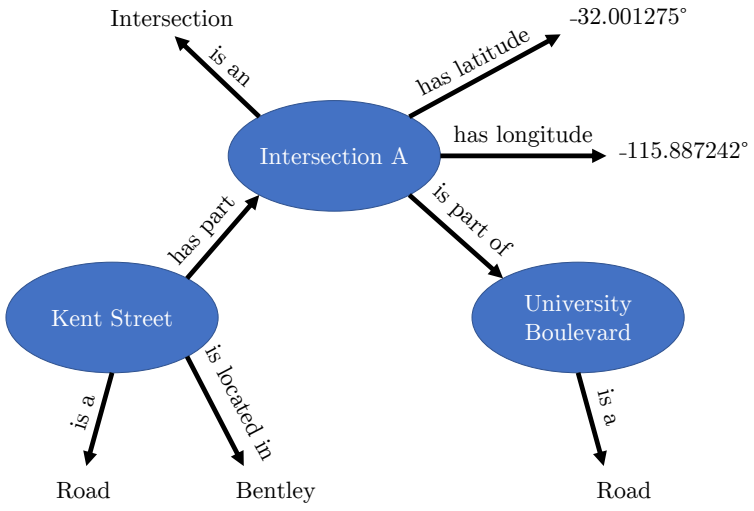


Figure 2.3: This graph shows a visualisation of eight sample triples.

Triples are simple structures combined using the subject, predicate and object notation. Each triple can be visualised with a graph. For example, the sample triples above regarding a road network are indicated in Figure 2.3, showing that

'Kent Street' is a road in 'Bentley' and connected through 'Intersection A' with the road 'University Boulevard'. Another piece of information here is that the intersection is located at '–32.001275°' latitude and '115.887242°' longitude.

The RDF supports linking to other external sources using URIs. For example, instead of the literal[11] 'Kent Street', the *DBpedia* entry '`http://dbpedia.org/resource/Kent_Street,_Perth`' can be used as a resource. Other triples from the example above can also be replaced to enable machines to understand the meaning of the data, such as what a 'location' is, what an 'intersection' is, what a 'road' is, what 'has part' or 'is part of' means and what the coordinates in 'longitude' and 'latitude' are. The triples from above will now be rewritten with URIs instead of literals where possible so that a connection to other documents on the Web can be established:

```
@prefix  dcterms:  <http://purl.org/dc/terms/> .
@prefix  dbr:      <http://dbpedia.org/resource/> .
@prefix  dbo:      <http://dbpedia.org/ontology/> .
@prefix  ex:       <http://example.org/> .
@prefix  geo84:    <http://www.w3.org/2003/01/geo/wgs84_pos#> .
dbr:Kent_Street,_Perth   a                  dbo:road ;
                         dcterms:Location   dbr:Bentley,_Western_Australia ;
                         dcterms:hasPart    ex:Intersection_A .
ex:University_Boulevard  a                  dbo:road .
ex:Intersection_A        a                  dbr:Intersection_(road) ;
                         geo84:lat          –32.001275 ;
                         geo84:long         115.887242 ;
                         dcterms:isPartOf   ex:University_Boulevard .
```

The previous example was written in the Turtle notation and introduced further URI prefixes with a leading at ('@') sign, which is commonly used for better

[11] A literal is a value that is not expressed as a URI, such as '–32.001275'.

readability; this enables RDF statements to be written in a compact form (Beckett et al., 2014). In this thesis, developed ontologies will be written in Turtle notation, and, therefore, important Turtle syntax are introduced next:

- A number sign (‘#’) is used for comments.

```
# this is a comment
```

- An at prefix (‘@prefix’) is used to abbreviate a URI.

```
@prefix ex: <http://example.org/> .
```

- Angle brackets (‘<’ and ‘>’) are used to enclose URIs.

```
<http:www.example.org>
```

- A point (‘.’) is used to complete a statement.

```
ex:Intersection rdf:type owl:Class .
```

- A semicolon (‘;’) is used to define a series of predicates and objects.

```
ex:Unit rdf:type owl:DatatypeProperty ;
        rdfs:subPropertyOf ex:Measurement .
```

- A comma (‘,’) is used to define a series of objects.

```
ex:Intersection rdfs:label "Intersection"@en ,
                           "Straßenkreuzung"@de .
```

2.3.4.2 RDF Vocabulary

With RDF, the opportunity exists to define classes, properties and types. RDF-S mainly adds the opportunity to express sub-classes, sub-properties, domains and ranges (Brickley & Guha, 2014; Schreiber & Raimond, 2014; Hitzler et al., 2012). It is a common practice to use a camel-case notation for better readability with a leading lowercase letter for properties and a leading capital letter for classes, as RDF properties and classes need to be defined as one term. For instance, in camel-case notation, the class 'sampleclass' can be written as 'SampleClass', and the property 'sampleproperty' can be written as 'sampleProperty'.

Prefixes

Prefixes are used in two different forms. The first form uses a slash ('/') as the final URI character. This means that an abbreviated URI can lead to another document, such as 'ex:test', which will be processed as '<`http://example.org/test`>'. The second form uses a number sign ('#') as the final URI character. This means that the abbreviated URI will refer to specific content of a given document. For instance, 'owl:Class' will be processed as '<`http://www.w3.org/2002/07/owl#Class`>'. The following prefixes will be used in the examples in this section:

```
@prefix  ex:   <http://example.org/> .
@prefix  owl:  <http://www.w3.org/2002/07/owl#> .
@prefix  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix  rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
```

Classes and sub-classes

A class is a container to categorise things, e.g., in the context of this thesis, roads, road signs, traffic signals, road stopping places and power lines. Sub-classes are used to categorise things into sub-categories; for example, a 'road edge' can be a sub-class of a 'road', and a 'stop sign' can be a sub-class of a 'road sign'. The

declaration of a ‘road’ that is a class and a ‘road edge’ that is a sub-class of a ‘road’ is introduced next:

```
ex:Road       rdf:type          owl:Class .
ex:RoadEdge   rdfs:subClassOf   ex:Road .
```

Properties and sub-properties

In ontology design, it is typical to define two types of properties, *object properties* and *datatype properties*.

Object properties describe relations, such as ‘intersection A has road sign stop’. Object properties can be categorised into sub-properties, e.g. ‘has stop sign’ is a sub-property of ‘has road sign’. Data-type properties are used to define literals, such as road names, object ids, geographic coordinates and dates.

The description of object property ‘has road sign’, which has the sub-property ‘has stop sign’ to declare that a stop sign is subordinate to a road sign, is defined in the next example:

```
ex:hasRoadSign   rdf:type             owl:ObjectProperty .
ex:hasStopSign   rdfs:subPropertyOf   ex:hasRoadSign .
```

Furthermore, the definition that a ‘road name’ of a road is declared as a data-type property is indicated next:

```
ex:RoadName   rdf:type   owl:DatatypeProperty .
```

Domains and ranges

Domains are used to restrict subjects to the definition of properties. For instance, we can define that the object property ‘has stop sign’ is part of the domain ‘inventory’. Ranges are used for the restriction of property values. The following

shows an example of the property 'has stop sign' in the domain 'inventory' and specifies that the range of the property is within the range 'road signs':

```
ex:hasStopSign   rdfs:domain   ex:Inventory .
                 rdfs:range    ex:RoadSigns .
```

2.3.5 Linked Data

The term linked data describes a set of methods for sharing and publishing structured data on the Web. This means that well-defined data are stored on the Web in a machine-readable format and are linked to and from other external datasets. The use of URIs and the HTTP is an elementary practice of linked data (Bizer et al., 2009). Berners-Lee (2009) released a set of rules for publishing linked data on the Web, which is interpreted next:

1. URIs are used to name things, such as intersections.
   Example: *www.example.com/data/roadNetwork/intersections/*

2. Use an HTTP schema so that users can retrieve information on the Web.
   Example: *http://www.example.com/data/roadNetwork/intersections/*

3. Useful information can be provided at a URI using the RDF, RDF-S and SPARQL standards.

4. Other URIs can be linked so that users get access to more information. For instance, with the use of the OWL attribute 'same as' the information of two documents can be compounded.
   Example: subject: *http://en.wikipedia.org/wiki/Eyre_Highway*
   predicate: *http://www.w3.org/2002/07/owl#sameAs*
   object: *http://wikitravel.org/en/Eyre_Highway*

The rules for publishing data on the Web are also known as the 'linked data principles' (Bizer et al., 2009).

2.3.6 Ontology Reasoning

The Semantic Web supports that knowledge can be received after reasoning an ontology (Ding et al., 2005). An ontology reasoner solves ontology problems, evaluates semantic rules and delivers information as knowledge (Parsia et al., 2017); commonly used ontology reasoners are HermiT,[12] Konclude,[13] ontop[14] and Pellet.[15] Semantic rules can be defined in an ontology to map individuals to properties and classes. However, semantic rules are not processed by default. This means that a reasoner needs to process (reason) an ontology first in order to be able to make changes to an ontology, e.g. set properties and classes. In contrast, if not explicitly saved, a reasoned ontology will be set to its initial state once a reasoner is deactivated.

A repository to test ontology reasoners, for consistency, classification and realisation, with many different problems from various domains, such as logic, mathematics, computer science, science and engineering, social sciences, and arts and humanities, is available with the Thousands of Problems for Theorem Provers (TPTP)[16] problem library (Sutcliffe, 2017). The TPTP library defines 217 Semantic Web problems (as of October 2020) to be solved against the Friend of a Friend [17] ontology.

2.3.6.1 Semantic Web Rule Language

An ontology reasoner can process semantic rules that are written in SWRL to edit an ontology, such as by moving individuals from one class to another, setting properties and classes and changing literal values. For example, a semantic rule for a road network could include the following statement: if a literal 'roadtype' contains the value 'road', then its object will be set to the class 'road'. The

[12] http://www.hermit-reasoner.com
[13] http://derivo.de/en/products/konclude/
[14] http://ontop.inf.unibz.it
[15] https://github.com/stardog-union/pellet
[16] http://tptp.org
[17] http://www.foaf-project.org

SWRL notation of the example is shown next, assuming that 'ex:' is a prefix for '<http://www.example.org/>' and 'swrlb:' is a prefix for <http://www.w3.org/2003/11/swrlb#>:

ex:roadtype(?a, ?b) ∧ swrlb:matches(?b, "road") -> ex:Road(?a)

The SWRL rules can be compared to conditional statements from common programming languages (e.g. Java, Python, C++ and others), with the limitation that only a true condition can be defined. Thus, if a SWRL rule is true, then an action, such as setting a class or a property, will be conducted. The SWRL language provides a set of approximately 79 built-ins that can be used for the definition of semantic rules, e.g. to compare dates, times, durations, numbers and strings, and to evaluate mathematical expressions (Horrocks et al., 2019). A selection of SWRL built-ins that will be used for the creation of new semantic rules for the road asset data conflation approach of this thesis is introduced next:

- *swrlb:abs(?a, ?b)* provides in the first argument the absolute value of the second numeric argument.

$$swrlb{:}abs(?a, -5.1) \longrightarrow ?a = 5.1$$

- *swrlb:contains(?a, ?b)* returns *true* if the first argument contains the case-sensitive value of the second argument.

$$swrlb{:}contains(\texttt{"}teststring\texttt{"}, \texttt{"}ststt\texttt{"}) \longrightarrow true$$
$$swrlb{:}contains(\texttt{"}teststring\texttt{"}, \texttt{"}sTsT\texttt{"}) \longrightarrow false$$

- *swrlb:containsIgnoreCase(?a, ?b)* returns *true* if the first argument contains the case insensitive value of the second argument.

$swrlb{:}containsIgnoreCase(\texttt{"}teststring\texttt{"}, \texttt{"}stst\texttt{"}) \longrightarrow true$
$swrlb{:}containsIgnoreCase(\texttt{"}teststring\texttt{"}, \texttt{"}sTsT\texttt{"}) \longrightarrow true$

- *swrlb:equal(?a, ?b)* returns *true* if the first argument is equal to the second argument.

$swrlb{:}equal(\texttt{"}teststring\texttt{"}, \texttt{"}teststring\texttt{"}) \longrightarrow true$
$swrlb{:}equal(\texttt{"}teststring\texttt{"}, \texttt{"}string\texttt{"}) \longrightarrow false$
$swrlb{:}equal(1000, 1000) \longrightarrow true$
$swrlb{:}equal(1000, 1000.1) \longrightarrow false$

- *swrlb:notEqual(?a, ?b)* returns *true* if the first argument is not equal to the second argument.

$swrlb{:}notEqual(\texttt{"}teststring\texttt{"}, \texttt{"}teststring\texttt{"}) \longrightarrow false$
$swrlb{:}notEqual(\texttt{"}teststring\texttt{"}, \texttt{"}string\texttt{"}) \longrightarrow true$
$swrlb{:}notEqual(1000, 1000) \longrightarrow false$
$swrlb{:}notEqual(1000, 1000.1) \longrightarrow true$

- *swrlb:lessThanOrEqual(?a, ?b)* returns *true* if the first numeric argument is less than or equal to the value of the second argument.

$swrlb{:}lessThanOrEqual(10, 50) \longrightarrow true$
$swrlb{:}lessThanOrEqual(100, 50) \longrightarrow false$

2.3.7 Open Geospatial Consortium Simple Features Access

The Open Geospatial Consortium (OGC) Simple Features Access standard specifies names and definitions of simple geographic features, e.g. points, curves, surfaces and geometry collections, and is standardised as ISO Standard 19125:2004

(Herring et al., 2011). A feature can be anything in the real world with a defined spatial location, such as points, line strings, intersections and traffic lights (Battle & Kolas, 2012). For instance, we assume that an arbitrary intersection has an area of 200 m^2 but that location is defined by a simple features point in longitude and latitude. For this thesis, the relevant features are points, line strings and multi-line strings that can be written as well-known text (WKT),[18] The examples in this section will use the prefixes 'ex' to link to a sample ontology, 'sf' to link to the OGC simple features ontology and 'geo' to link to the GeoSPARQL ontology, as indicated next:

```
@prefix  ex:   <http://example.org#> .
@prefix  sf:   <http://www.opengis.net/ont/sf#> .
@prefix  geo:  <http://www.opengis.net/ont/geosparql#> .
```

Points are zero-dimensional objects that have an x-coordinate and a y-coordinate, i.e. longitude and latitude, respectively (Herring et al., 2011). The definition of a simple features WKT point for the sample coordinates '115.7°' longitude and '–31.6°' latitude is shown next:

```
ex:PointCoordinate_1 a sf:Point ;
    geo:asWKT "Point(115.7 –31.6)"^^geo:wktLiteral .
```

In the example above, the predicate 'geo:asWKT' was used to describe the WKT point object that followed, which was parsed as a 'geo:wktLiteral'[19] in the '^^' notation that defined a literal relation to the GeoSPARQL ontology. A line string is a linear interpolated curve with line segments at each vertex pair. A line is defined if a line string has exactly two points. A collection of multiple line

[18] A textual representation within OGC Simple Features for a unified definition of geometries, such as for points, line strings, multi-points, multi-line strings, multi-polygons, geometry collections, polyhedrons and tetrahedrons.

[19] As mentioned earlier, a literal is a value that is not represented as a URI, such as a piece of text or a number.

strings within a single object is called a multi-line string (Herring et al., 2011). The following example shows both, the definition of a WKT line string and the definition of a WKT multi-line string with arbitrary sample coordinates:

```
ex:LineString_1 a sf:LineString ;
    geo:asWKT "LineString(115.7 -31.6, 115.9 -31.5, 115.7 -31.4)"
    ^^geo:wktLiteral .
ex:MultiLineString_1 a sf:MultiLineString ;
    geo:asWKT "MultiLineString((115.7 -31.6, 115.9 -31.5), (115.7
    -31.4, 115.4 -31.2))"^^geo:wktLiteral .
```

2.3.8 SPARQL and GeoSPARQL

Sometimes, it is necessary to retrieve specific information from an ontology. With SPARQL, it is possible to query ontologies and to retrieve information in a syntax that is similar to Structured Query Language (SQL) statements using 'SELECT' and 'WHERE' expressions (Battle & Kolas, 2012). Moreover, GeoSPARQL is an OGC standard for the Semantic Web that extends SPARQL for unified geospatial data modelling, indexing and querying. With such queries, a user can ask for certain things, such as whether an individual has a geometry (e.g. points and line strings). The related query statement is as follows:

```
SELECT ?element
WHERE { ?element geo:hasGeometry ?location . }
```

The OGC GeoSPARQL provides more RDF vocabulary to define OGC Simple Features (e.g. 'geo:asWKT') (Battle & Kolas, 2012). For instance, if RDF individuals contain OGC Simple Features and OGC GeoSPARQL attributes, then topological relations (e.g. equals, disjoint, intersects, touches, within, contains, overlaps and crosses) can be identified with queries (Perry & Herring, 2012).

The next example declares two geospatial individuals (‘ex:individuals_A’ and ‘ex:individual_B’) that are defined as OGC Simple Features line strings, and a GeoSPARQL query to determine if individual ‘ex:individual_A’ has any crossing line strings:

1. Design an ontology that declares two line strings in Turtle notation:

```
@prefix ex: <http://www.example.org/> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix sf: <http://www.opengis.net/ont/sf#> .

ex:individual_A rdf:type owl:NamedIndividual;
                geo:hasGeometry ex:LineString_1 .

ex:individual_B rdf:type owl:NamedIndividual;
                geo:hasGeometry ex:LineString_2 .

ex:LineString_1 a sf:LineString ;
                geo:asWKT "LineString(117.1 -31.2, 117.3 -31.1,
                  117.4 -31.0)"^^geo:wktLiteral .

ex:LineString_2 a sf:LineString ;
                geo:asWKT "LineString(117.1 -31.0, 117.4 -31.2,
                  115.7 -31.4)"^^geo:wktLiteral .
```

2. Define an GeoSPARQL query statement to identify crossing line strings:

```
PREFIX ex: <http://www.example.org/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
```

```
PREFIX sf: <http://www.opengis.net/ont/sf#>

SELECT ?p
WHERE {
   ex:individual_A geo:hasGeometry ?feature .
   ?feature a sf:LineString .
   ?feature geo:asWKT ?mgeo .
   ?p a sf:LineString .
   ?p geo:asWKT ?pgeo .
   FILTER(geof:sfCrosses(?mgeo, ?pgeo))
}
```

Prefixes in SPARQL do not require a leading at sign ('@') compared to the Turtle notation. The prefix 'geof:' is used to enable GeoSPARQL filter functions; as the example above, 'geof:sfCrosses' was adopted to determine if the line string of 'ex:individual_A' was crossed by another line string, as visualised in Figure 2.4. The example query from above can be interpreted as follows:

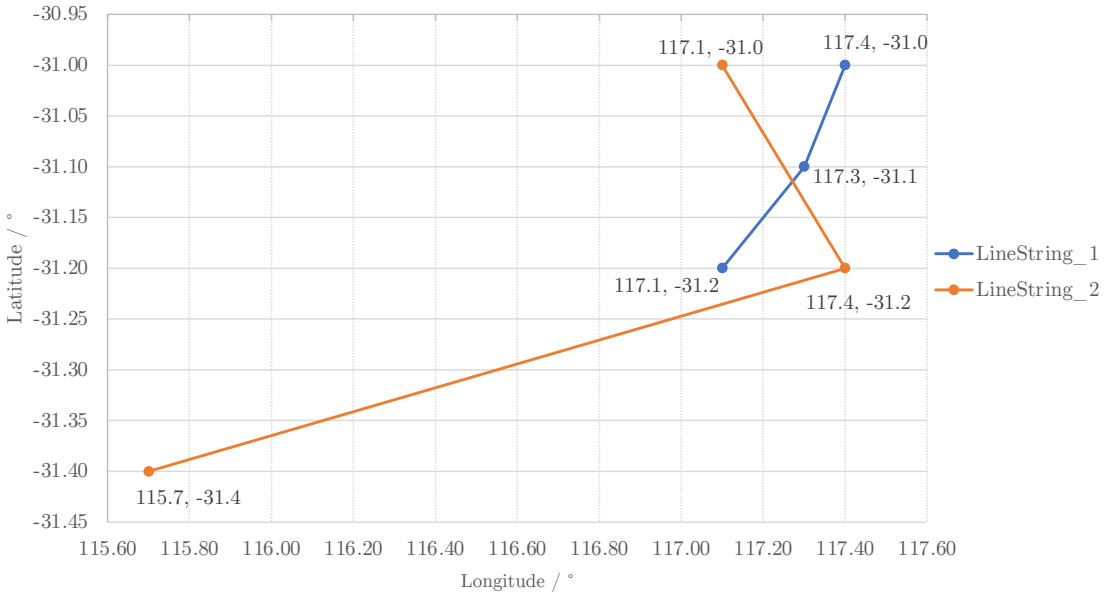


Figure 2.4: Graph of the line strings 'LineString_1' and 'LineString_2'.

1. Select value the variable '?p' as the return value.

2. The variable '?feature' is related to 'ex:individual_A' and requires 'ex:individual_A' to be a line string, with its WKT value available in the variable '?mgeo'.

3. The variable '?p' is a line string individual with its WKT individual available with the variable '?pgeo'.

4. If the GeoSPARQL crosses function ('geof:sfCrosses') returns a value for the WKT line strings '?mgeo' and '?pgeo', then a crossing line string exists between the individual 'ex:individual_A' and a related individual that will be returned in the variable '?p'.

As seen in the example, a SPARQL query can be used, for instance, to identify whether an overhead power line crosses a road.

2.3.9 Data Provenance

Provenance information provides context and motivation, which leads to the production of data. More specifically, provenance informs users about various data life cycle information, such as the origin, updates, handling, use cases, validations and life span. The W3C recommends the *PROV-O* ontology for linking provenance to the Semantic Web (Lebo et al., 2013). The *PROV-O* data model uses three base classes to define data provenance, as visualised in Figure 2.5, namely: agent, activity and entity.

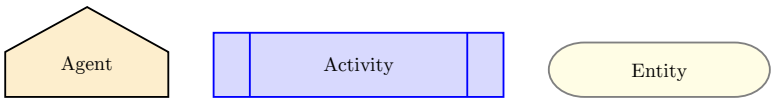


Figure 2.5: Standard visualisation of a PROV-O agent, activity and entity.

An agent is an instance that can be in charge of the activation of activities as well as the creation of entities and activities from other agents. For instance,

if a dataset is provided by an organisation, then the organisation can act as an 'agent', as they are responsible for the data quality (Moreau & Missier, 2013).

An activity describes the occurrence of something that collaborates with entities in a period. This can be a process or algorithm that is activated to use, generate, update or modify entities. For example, if a regulatory sign is a part of an intersection, then a property can be set to assign the sign to the intersection.

An entity refers to a defined thing[20] whose provenance has to be described. An entity's lifetime is limited by its creation and invalidation attributes. For instance, assume a regulatory sign entity can be created as a result of a broken traffic signal and can be invalidated after the traffic signal has been repaired.

The following prefixes will be used in this section:

```
@prefix   :       <http://example.org/> .
@prefix   prov:   <http://www.w3.org/ns/prov#> .
```

The *PROV-O* ontology contains approximately 30 classes and 50 properties (Lebo et al., 2013). Not all of those classes and properties can be described in the scope of this thesis. Therefore, only the properties relevant to this dissertation and their inverses will be explained. For instance, a simplified provenance expression can be 'A -> provenance -> B', with its inverse as 'B -> inverse provenance -> A'. The provenance attributes required to understand the developed provenance models of this thesis are explained next:

- *prov:used* indicates the usage of an entity by an activity. The inverse of this property is *prov:usedBy*. The following example defines the activity ':createRoadSign', which uses the entity ':RoadSignData' and its inverse.

[20] A thing is a kind of a subject that exists in the real world (e.g. roads, roundabouts, street lights and road signs), in digital form (e.g. datasets and graphs) or as a concept (e.g. planned road works).

```
:createRoadSign a prov:Activity ;
                prov:used :RoadSignData .
:RoadSignData a prov:Entity ;
              prov:wasUsedBy :createRoadSign .
```

- *prov:wasGeneratedBy* indicates the creation of an entity. The inverse of this property is *prov:generated*. The next example defines the activity ':createRoadSign', which creates the entity ':RoadSign' and its inverse.

```
:createRoadSign a prov:Activity ;
                prov:generated :RoadSign .
:RoadSign a prov:Entity ;
          prov:wasGeneratedBy :createRoadSign .
```

- *prov:wasAssociatedWith* describes which agent is responsible for an activity. The inverse of this property is *prov:wasAssociateFor*. The next example defines the activity ':createRoadSign', which is associated with the agent ':Authority' and its inverse.

```
:createRoadSign a prov:Activity ;
                prov:wasAssociatedWith :Authority .
:Authority a prov:Agent;
           prov:wasAssociateFor :createRoadSign .
```

- *prov:wasAttributedTo* describes the relation of an entity to an agent. The inverse of this property is *prov:contributed*. The next example defines the entity 'RoadSignData', which is attributed to the agent ':Authority' and its inverse.

```
:RoadSignData a prov:Entity ;
                    prov:wasAttributedTo :Authority .
:Authority a prov:Agent;
                prov:contributed :RoadSignData .
```

- *prov:hadPrimarySource* refers to another entity to describe that significant content has been used from that source. The inverse of this property is *prov:wasPrimarySourceOf*. The next example defines the entity ':RoadSign', which uses data from the entity 'RoadSignData' and its inverse.

```
:RoadSign a prov:Entity;
            prov:hadPrimarySource :RoadSignData .
:RoadSignData a prov:Entity ;
                    prov:wasPrimarySourceOf :RoadSign .
```

- *prov:atLocation* describes the geographical position of an identifiable place, which can be either a line string, a multi-line string, a point, an address, a landmark or a non-geographical place (e.g. directory, row or column). The following example defines a location with the entity ':Location' of an entity ':RoadSign', which is generated by the entity ':createRoadSign'.

```
:Location a prov:Entity;
            prov:hadPrimarySource :RoadSignData ;
            prov:wasGeneratedBy :createRoadSign .
:createRoadSign prov:generated :Location .
:RoadSign prov:atLocation :Location .
```

A graph is a simple visualisation to understand the provenance of Semantic Web documents, as a text representation can quickly become complicated. Figure 2.6 shows a *PROV-O* graph that contains the previously introduced provenance.

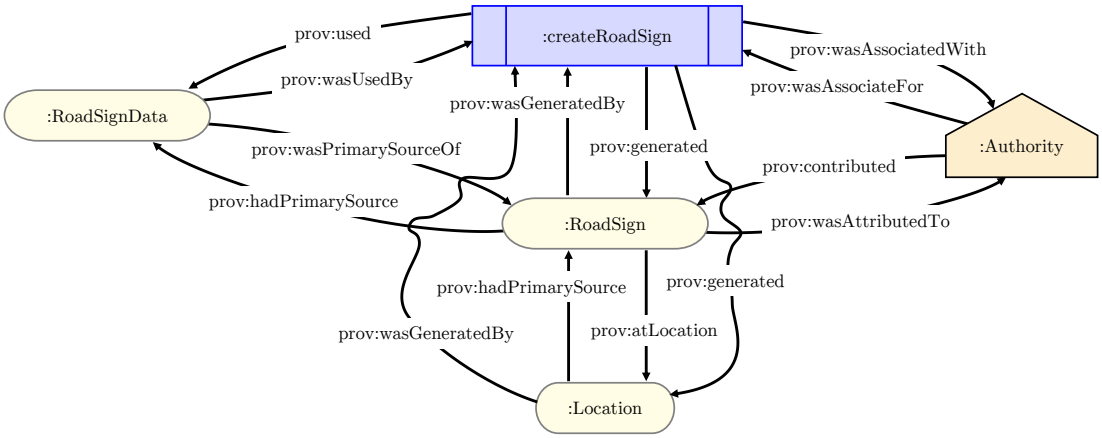


Figure 2.6: Graph to indicate the relation of the introduced provenance classes and properties.

2.4 ITS

Most Semantic Web technologies that include road infrastructure are matched to ITS, whose definition is standardised by the ISO/TC 204 standard. Overall, the application of ITS focuses on the task of improving road safety by integrating wireless communication technologies into road infrastructures while enabling information exchange between sensor-based measurement units and transport authorities. A typical ITS application wirelessly informs road users about hazards in their direct environment (Desertot et al., 2012). For example, an authorised road user sends enabled device information about an oil slick at a highway exit to a responsible transport authority through an ITS. The contacted transport authority can interact in real time and inform car users in the related danger zone through an ITS message about the oil slick. Additionally, an external contractor can be ordered automatically via a separate ITS message to remove the oil slick.

2.4.1 Overview of ITS Developments

The following overview of current ITS research delivers an introduction to the closely related activities of this dissertation. Although this thesis will not employ sensor-based systems that can enable wireless communication with transport authorities, the findings often contain traffic situations and route-planning-related content and, therefore, are included in this section.

Lorenz et al. (2005) presented an ontology for transport networks (OTN) that aimed to encode a given Geographic Data File (GDF)[21] ontology into the OWL format. Their ontology model supported a variety of road network classes, such as roads, roundabouts, intersections and junctions. For instance, they used their ontology system to simplify the visualisation of traffic network elements into Scalable Vector Graphics.

Fernandez et al. (2015) introduced an ontology-based system for driving assistance in different traffic situations. They considered the connections between a vehicle's travel route, the weather and traffic regulations in Japan. Semantic rules were written in SWRL, and the ontology was reasoned with Pellet.[22] With SPARQL queries, they were able to determine the next action a vehicle driver should take to reach the target, such as reduce their speed, turn left/right or go straight. Also, advice was given to the driver in certain situations to turn on the headlights avoid puddles or flowing water.

Fernandez et al. (2016) proposed a system that used sensors installed at vehicles, bridges, roads and road signs for a safer driving experience. Their ITS used an ontology model based on Fernandez et al. (2015) to describe and classify vehicles (e.g. public, private and priority), road infrastructure assets (e.g. roads, road markings, lanes, signs and parking locations) and an interconnected sensor network (e.g. measurement properties, observations and sensor input/output). For

[21] The GDF format is used in ITS applications for the specification of road and road-related information (e.g. conceptual data models, logical data and physical encoding formats) (International Organization for Standardization, 2011).

[22] Pellet is an open-source OWL description logics reasoner written in Java (Sirin et al., 2007).

example, their dynamic system was able to calculate the arrival time of a vehicle considering multiple aspects, such as car speed and traffic signal duration.

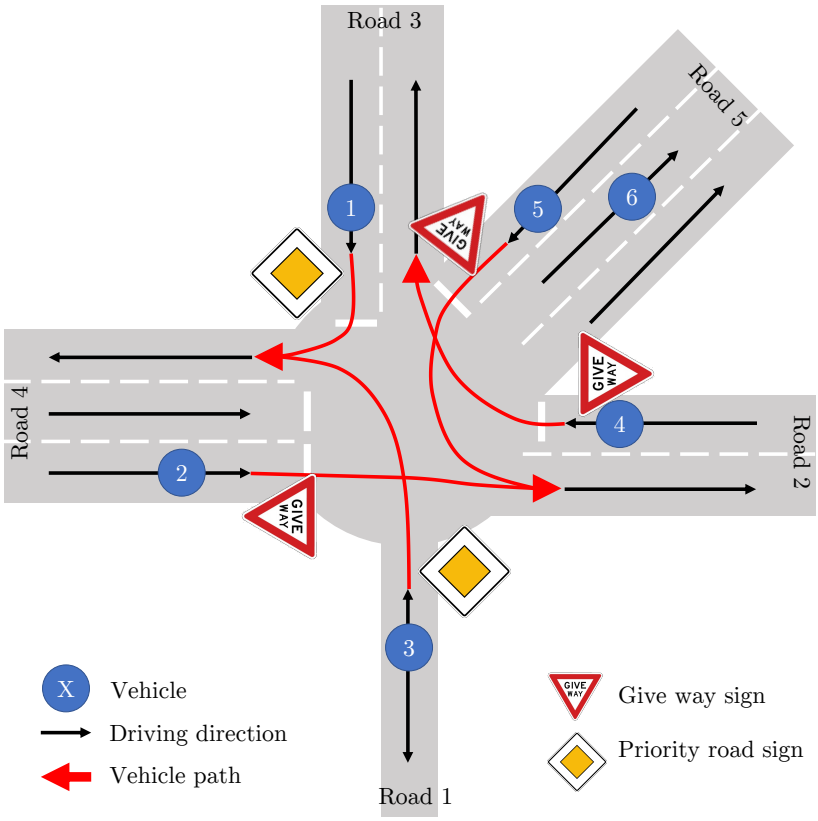


Figure 2.7: Example of a traffic situation at a complex intersection with multiple roads, lanes, vehicles and regulatory signs (Hülsen et al., 2011).

Hülsen et al. (2011) modelled an ontology to reason out right-hand traffic situations for cars, motorbikes and heavy vehicles at complex intersections. The example in Figure 2.7 shows a complex intersection with five roads, 11 lanes, six vehicles and five regulatory signs. They managed to query their ontology to avoid possible accidents at an intersection. For instance, in the context of this example, identifying vehicle 4 has a right-turn conflict with vehicle 5; vehicle 5 has the right of way before vehicle 4; vehicle 3 has to give way to vehicle 1; vehicle 1 has no yield; and vehicles 1 and 3 both have an existing future connection to lane 3 of road 4 once they have turned.

Zhao et al. (2015, 2017) created an ontology-based advanced driver assistance system for autonomous vehicles to increase driving safety. Their system was able to detect speeding while taking into account speed limit information and velocity sensor data converted into RDF streams. The use of Continuous SPARQL (C-SPARQL)[23] queries enabled Zhao et al. (2015, 2017) to evaluate the maximum vehicle's speed continuously in a range of 500 ms. Their knowledge-based approach included the automated driving decisions stop, move to left and give way at uncontrolled intersections following the right-of-way rule. According to their data repository in 'Toyota Technological Institute: Core ontologies for safe autonomous driving' (TTI, 2015) they did not use a geographical localisation data standard for the vehicles and road infrastructure as is available with the OGC simple features, as specified by Herring et al. (2011).

Marasovic & Marasovic (2010) proposed a concept to design service routes on a daily basis with Semantic Web technologies. Their road segment domain ontology was similar to the model of Niaraki & Kim (2009) and it contained the same quantitative criteria of road, vehicle and unit and slightly differed qualitative criteria that added service and season but removed tourist, user, safety and facility. Marasovic's system informed road users through wireless devices about road situations to prevent accidents and traffic jams. For instance, if a natural event, such as a landslide, tree on the road or deep snow, caused a road closure, then a road authority would be informed by a road user about the circumstances and would send the information to a news feed so that the nearest proper service provider would receive the information and handle the problem. Unfortunately, no further known publications have been produced by this group regarding the implementation of this system.

Provine et al. (2004b) and Schlenoff et al. (2004) introduced an ontology for autonomous vehicles to estimate the damage after a collision with surrounding objects. Provine et al. (2004a) showed that ontology reasoning can inform about

[23] C-SPARQL is a language that extends SPARQL to query continuously changing RDF stream data (Barbieri et al., 2010).

collision consequences using a cost function that determines the best path as the path with the cheapest cost to reach the target. Lane changes and damage risk objects increase the path's cost with different weights. For instance, their ontology reasoning approach determined that the cost of a lane change is small concerning the cost of a collision between a motorcycle and a brick. In comparison, if a large vehicle with high rigidity (e.g. an off-road vehicle) hit the same brick the motorcycle hit, then the collision would cause no damage and, therefore, no extra costs after striking the object.

The reviewed literature of this section summarised various use cases with ontologies involved in ITS research activities. It has been identified that common practices include the use of Protégé to model ontologies, Pellet to reason ontologies and SWRL to define semantic rules. The road network ontologies described in the reviewed literature often shared a similar layout, which principles (e.g. model the road network in a hierarchical asset order) can be adapted for the road network ontology design of this thesis. To the best of the author's knowledge none of the reviewed scientific documents described the use of semantic rules for the identification of same road asset data from heterogeneous data sources. The significant contribution of this work will include an ontological road network representation that enables a trust score to weight the trust of a road asset data source. The aspect of a trusted road network data source is new and has not been investigated by others. The findings of the ontological road network representation will be further employed as a solid base for route planning.

2.5 Route Planning

In their simplest form, route planning approaches identify the shortest route between a starting point and a destination. This section will review the literature regarding route planning that positively influenced the route planning approach of this thesis. Niaraki & Kim (2009) worked on a personalised route planning approach with a multi-criteria decision technique. The technique they used followed

a hierarchical structure and used a pairwise criteria comparison to create a road segment impedance model that took into account distance, velocity or traffic and also considered personal information, tourist preferences, road segment length and path-allowed vehicle types. They covered a road network of approximately 55,000 km and included additional data, such as gas stations, seaside areas, jungles, weather and road traffic information. Although their approach compared the shortest route with a route that was identified by their multi-decision system, which supported various features, such as current speed, weight, fuel capacity and shipment, they did not mention whether a certain vehicle type (e.g. motorcycle, car or tourist bus) influenced the path selection.

Szwed et al. (2012) presented a route planning method that was able to use ontologies and semantic rules to select the best route planning algorithm, such as exact, approximate and greedy regarding a current traffic situation. For instance, if speed discrepancies were observed on a track, then their ontology suggested an alternative route to reach the target. In harsh winter conditions, they selected a greedy algorithm that suggested driving on main roads.

Houda et al. (2010) and De Oliveira et al. (2013) worked on a public transportation ontology regarding journey planning. Their approach was able to consider buses, trams, metros and trains as transportation methods and passengers' points of interest to reach the final destination. They implemented a rule-based journey pattern with the use of SWRL that enabled path planning for weekdays, festival days, and travel behaviours considering shopping, leisure activities, walking time and covered station areas. Their SWRL-based ontology reasoning system was able to classify these patterns into passengers' relevant requirements, such as connection points with shelter, no stopovers, leisure and interesting facilities in the surrounding area.

Corsar et al. (2015) conceptualised a transport disruption ontology framework[24] to describe disruptive impacts on travel mobility, such as road closures.

[24] `http://purl.org/td/transportdisruption`

Their framework enabled modelling traffic-related events and supported the input parameters event type, location, duration, relation to other events and custom impacts entered by agents. For instance, while planning a journey, a user would receive information about road work and related road closures regarding their journey from A to B and how much longer the journey would take if a disruption occurred.

Faaß (2015) developed, in the scope of his master's thesis, an ontology-based route planning system for OSM data that considered facilities for drinking, shopping, food, banks, entertainment, health, hotels and miscellaneous locations and their opening hours. For instance, if a user needed to use a petrol station on their way to their destination, then it would be a hard requirement that the petrol station be open upon arrival.

The in this section reviewed literature summarised available route planners driven by Semantic Web Technologies. None of the reviewed scientific documents described to include live data into their route planning approach, such as possible with the route planner of this thesis with road closures and suburb information. A significant contribution with the route planner of this dissertation includes the consideration of overhead powerlines as no one has treated that before.

2.5.1 Route Planning Algorithm

To reach a target destination from a start point is a simple task for humans when looking at a map. In computer science, optimal route planning algorithms are a good example of merging algorithms with the real world (Delling et al., 2009). Many different algorithms for route planning exist, such as breadth-first, depth-first, A*, D*, ALT, contraction hierarchies, arc flags, Bellman Ford Moore, Floyd Warshall, Greedy, global route planning and dynamic highway-node routing. Bast et al. (2016) presented a comprehensive overview of the algorithms for route planning in transportation networks. It bundled research from scientists that worked for global companies, such as Apple, Esri, Google, MapBox,

Microsoft, Nokia, PTV, TeleNav, TomTom and Yandex, on route-service-related projects. The most basic idea of a route planning algorithm is to find the shortest path. One of the most famous shortest path algorithms is the approach of Edsger Wybe Dijkstra that sums the length of road edges to find the shortest path (see (Dijkstra, 1959)).

2.5.1.1 Dijkstra's Shortest Path Algorithm

The identification of the shortest path is a basic problem in large-scale network applications (Cherkassky et al., 1996; Gallo & Pallottino, 1988). Dijkstra's (1959) shortest path algorithm is the reference algorithm for route planning and is referenced in most of the reviewed literature (e.g. (Lauther, 2006; Wang et al., 2005; Goldberg et al., 2007; Geisberger et al., 2008; Schultes & Sanders, 2007; Delling et al., 2011; Bast et al., 2016; Szwed et al., 2012; Madduri et al., 2006; Edmonds et al., 2006; Dibbelt et al., 2015)). It is a node-based approach with non-negative edge lengths. The algorithm provides the shortest path from start node s to every other node j in a graph. Each node must be iterated once in an order depending on the edge length. A priority queue (Q) is maintained for the nodes, sorted by their temporary distances from s. We assume that distance λ from source node 1 to each node j is infinity, except for the source itself. The statement

$$\lambda(j) = \begin{cases} \infty & \text{for all } j \neq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

can be used in this context for a better visualisation of the initial situation. No nodes are labelled 'visited' before the first iteration, i. In the first iteration, distance 0 (node s to s) is added to Q, and the algorithm extracts from node s the minimum distance λ to every neighbour of s. Then, node s will be set to visited, and node j with the nearest distance to s will move up Q to the second position. In each further iteration, a not-visited node j with the nearest distance to s will be nominated, from which the distances to its neighbours will

be calculated. Once a node is scanned, its distance to start node s, dist(*node*, *length*), is true, and Q will be updated (Bast et al., 2016).

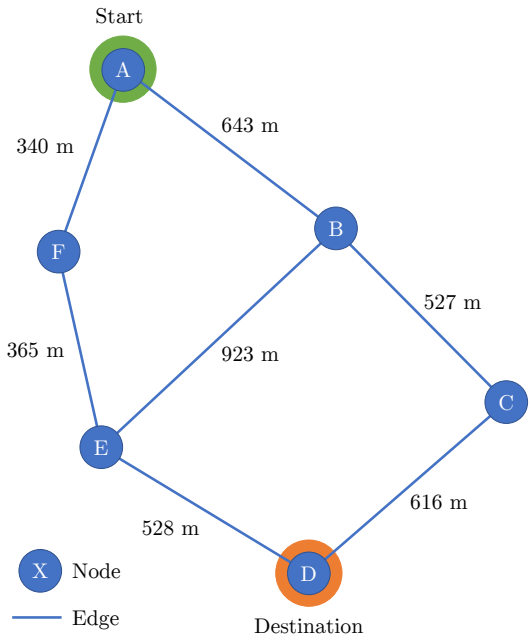


Figure 2.8: Graph that shows six nodes (A–F), a start node, an end node, seven edges and the resulting edge distance between connected nodes.

Consider the graph in Figure 2.8 that shows six nodes (A–F), with start node A and destination node D highlighted in green and orange, respectively. The distances between connected nodes are written next to each edge. One can see, that it is simple to identify that the shortest path is in the order A-F-E-D. In comparison, machines have to consider all possibilities to determine the shortest path efficiently. The processing of Dijkstra's algorithm to determine the shortest path from node A to all other nodes requires seven stages (see Figure 2.9). The stages are as follows:

Stage 1: The distance from node A to itself is set to zero, and the other distances from node A are set to infinity.

Stage 2: Node A is labelled as visited, and the distances from its neighbours (nodes B and F) are identified. Node F has a distance of 340 m from node

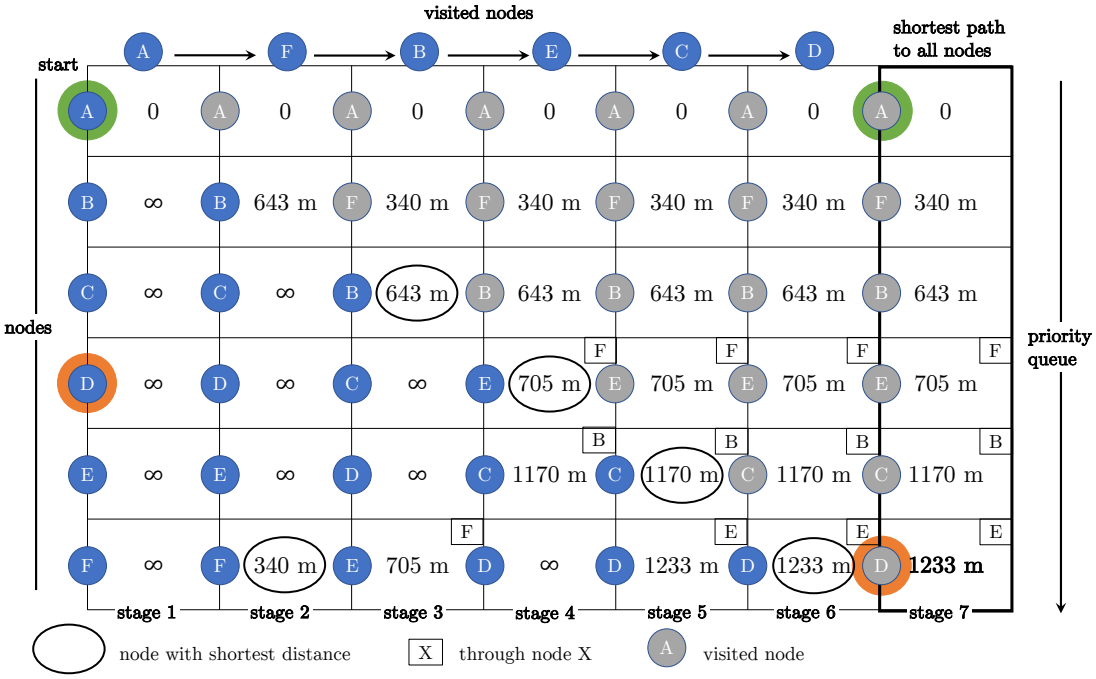


Figure 2.9: Dijkstra algorithm applied to a graph with six nodes (A–F) with determined distances from start node A. The destination node is D, and the graph shows all distances from node A ordered in a priority queue.

A and will be nominated as the next to visiting node, as it has the shortest distance.

Stage 3: Nominated node F will move the priority queue up to the second position. Then, the neighbours' distances (nodes A and E) will be calculated, and node F will be set to visited. Node B has the shortest distance with 643 m and will be nominated as the next visiting node.

Stage 4: Node B remains in its position in the priority queue, as it is in the right position. Then, the distances to its neighbours (nodes A, C and E) will be calculated, and node B will be set to visited. The path from node A to node E goes through node F, which is indicated in the top-right corner of the cell. Node E will be further nominated as the next visiting node.

Stage 5: Node E remains in its position in the priority queue. Then, the distances to its neighbours (nodes B, D and F) will be calculated, and node E will be set to visited. The path from node A to node C goes through node B. Node C will be nominated as the next visiting node.

Stage 6: Node C remains in its position in the priority queue. Then, the distances to its neighbours (nodes B and D) will be calculated, and node C will be set to visited. The path from node A to node D goes through nodes E and F. Node D will be nominated as the next visiting node.

Stage 7: Node D remains in its position in the priority queue. The path to its neighbours (nodes E and C) will be calculated. As all nodes have been visited, node D will be set to visited, and the processing of the shortest path from node A to every other node is completed.

After processing Disjktra's algorithm, it has been computed that the shortest path from node A to node D is 1,233 m. The best way to read the graph is to start at node D. Here, we can see that the order D-E-F-A has been identified, which is the same (reversed) result as our initial visual inspection above.

Note: there is ongoing research on speed-up techniques regarding Dijkstra's algorithm which are up to 3 million times faster than the original code. Each of these techniques requires pre-processing. If a speed-up technique is developed, then the run-time comparison and memory usage will commonly be evaluated against Dijkstra's algorithm (Delling et al., 2009).

2.5.1.2 Edge Weight

In its simplest form, an edge's weight is just represented by the distance between two nodes. In the past, several studies have sought to identify additional constraints that can be parsed into a function that defines an edge's weight. Winter (2002) remodelled a graph by adding nodes and edges so that turning costs could be taken into account. Dean (2004) presented a framework that considered the

edge arrival time for predictable travel times, such as rush hour. Baum et al. (2014) optimised a cost function for electric vehicles that considered road edge altitude so that an electrical vehicle's battery could be charged while driving downhill. Overall, cost functions are used to specify the weight of an edge. For instance, if speed limit v and edge length s are given, then travel time

$$t = \frac{s}{v} \tag{2.2}$$

can be used instead of distance as the edge weight. In this thesis, a route will be planned concerning the edge weight configurable for distance and travel time, and additional constraints, such as overhead power lines, turning circles, traffic signals, regulatory signs and rail crossings will be taken into account in a cost function. To the best of the author's knowledge, the concept of considering overhead power lines which requires integrating data from a different agency, is new and has not yet been done by others.

2.6 Heavy Vehicles on Roads

The leading cause of fatal accidents induced by heavy vehicles is electrocution due to contact with overhead power lines (Koustellis et al., 2011; Crow, 2009; Cawley & Homce, 2003). The clearance between a road and an electrical conductor in Australia is regulated through the voltage. Below 1 kV, the clearance to the ground must be at least 5.5 m, and from 1–33 kV, the clearance must be above 6.7 m for vehicles with a maximum height of 4.6 m (Seneviratne, 2015). In Western Australia, heavy mobile plants came into contact with overhead power lines 47 times between 1995 and 1999, and two accidents (one of them with a death involved) took place in 2000 when a crane operated too close to an overhead power line (Torlach, 2001). Between 1998 and 2008, a total of 74 contacts with overhead power lines were reported (Ridge, 2008), and 26 contacts occurred between 2009

and 2018 (Department of Mines, 2019). The accidents took place in the same state.

Ongoing climate change and the resulting extreme weather conditions (e.g. heatwaves, bushfires and extensive rainfall) harm a roads' usage. Roads are being closed and are inoperable due to floods and bushfires (Chhetri et al., 2012). Therefore, this work will enable the consideration of live information on road closures and minimise the occurrence of crossing overhead power lines with roads in the route planner of this thesis.

2.7 Chapter Summary

With the availability of publicly accessible government data, the Semantic Web has the potential to become one of the key technologies for solving data harmonisation problems. By integrating data provenance, it is possible to keep track of data origin and use cases. Nevertheless, Semantic Web technologies are often used in ITS to enable communication between various interfaces. A few approaches of applied Semantic Web technologies exists regarding route planning. Considering heavy vehicles, for example, no one has ever provided avoidance of power lines for route planning.

This chapter reviewed relevant work regarding the content of this thesis. The concept of data harmonisation was identified and its use by governments was revealed. Semantic Web technologies were introduced in detail, meaning that RDF, ontologies, OWL, SWRL and SPARQL basics were explained with examples to further follow up with the arrangements in this thesis. Then, ITS and route planning literature that used Semantic Web technologies as a translator between various systems was reviewed. In the scope of route planning, Dijkstra's shortest path algorithm was further explained. Examples of heavy vehicles colliding with overhead power lines were shown, and the impact of weather conditions on route planning was indicated.

Chapter 3

Datasets

3.1 Chapter Introduction

In this chapter, the road network datasets employed in this thesis will be selected. The data selection will include datasets from MRWA, Landgate, Western Power, OSM and DBpedia.

3.2 Data Selection

A road network consists of many different road assets, e.g. road sections, roundabouts, intersections, road signs, rail crossings and traffic signals, that are linked together into a complex road network representation. The road asset datasets presented in this thesis are either based on quality-controlled ground surveys from MRWA and Landgate or collected from the OSM community on a trusted database.

MRWA data

Main Roads Western Australia provides most of the road asset data employed in this thesis, as summarised in the following list:

- Road network: the dataset contains a centreline representation of all state and local roads controlled by MRWA divided into road sections, whereby

an intersection is used as a road section separator. For instance, if a road has four intersections in total between its start and its end, then the road can be divided into five road sections. Source: `http://catalogue.data.wa.gov.au/dataset/mrwa-road-network`

- Intersections: the dataset contains intersections that are extracted from MRWA's road information system and describes the location at which a road junction with two or more roads meet or cross. Source: `http://catalogue.data.wa.gov.au/dataset/mrwa-intersections`

- Regulatory signs: the dataset is a collection of road traffic regulatory signs operated by MRWA. In this thesis, the relevant regulatory signs are 'stop' and 'give way'. Source: `http://catalogue.data.wa.gov.au/dataset/mrwa-signs-regulatory`

- Traffic signals: the dataset contains the locations of traffic signals for pedestrian traffic and vehicle control at intersections and roads that are controlled and maintained by MRWA. Source: `http://catalogue.data.wa.gov.au/dataset/mrwa-traffic-signal-sites`

- Road stopping places: the dataset contains the location of roadside stopping places for cars and heavy vehicles in Western Australia. Source: `https://catalogue.data.wa.gov.au/dataset/mrwa-road-stopping-places`

- Road closures: the dataset contains information on currently closed roads in the Western Australian road network. Source: `https://catalogue.data.wa.gov.au/dataset/mrwa-road-closures-closed`

- Rail crossings: the dataset contains the locations of rail crossings on public access roads in Western Australia. The dataset has been selected because train rails can include a traction power network to supply an electrified rail network. Source: `https://catalogue.data.wa.gov.au/dataset/mrwa-rail-crossings`

- Speed limits: the dataset contains information on the maximum travel speed allowed on a road. The speed limit information ranges from 10 km/h to 110 km/h in steps of 10 km/h, and a further data value, '50km/h applies in built-up areas or 110km/h outside built up areas', is present in the dataset. Source: `http://catalogue.data.wa.gov.au/dataset/mrwa-legal-speed-limits`

Landgate data

The Landgate 'LGATE-012' roads dataset contains extracted road centrelines, roundabouts and connectors. The data mostly cover the area of the MRWA road network dataset and have been collected independently. Source: `https://catalogue.data.wa.gov.au/dataset/roads-lgate-012`

Western Power data

The dataset 'WP-031' contains the approximate locations of the overhead power line routes of Western Power's distribution network. Source: `https://catalogue.data.wa.gov.au/dataset/distribution-overhead-power-lines-wp-031`

OSM data

The OSM dataset is a bundle of all collected OSM data entries in Australia in a layer-based representation for the categorisation of multi-line strings, line strings and points. It contains various data types, such as roads, tracks, traffic lights, buildings, fences, waterfalls, mineshafts, gas stations and many other features that have been captured as points of interest. Not all of the data within the dataset are relevant to this thesis, and therefore, only the line string layer relevant to this thesis that includes roads and roundabouts will be used. Source: `http://download.geofabrik.de/australia-oceania/australia.html`

DBpedia data

The DBpedia dataset is a snapshot of the Wikipedia dataset prepared as linked data for the Semantic Web. Wikipedia datasets are collected by community effort, as anyone can contribute data to Wikipedia. In the scope of this thesis, a town description will be retrieved from DBpedia for each passing town when planning a route from one place to another. Example (Perth, WA):
`http://dbpedia.org/resource/Perth,_Western_Australia`

3.3 Metadata Structure

The datasets from MRWA, Landgate, OSM and Western Power contain various metadata information, such as an object identifier, a road name and a kilometre point (SLK). It is not necessary to assign the data structure to a specific approach, such as road network translation, road network conflation or route planning, as the datasets can be used in multiple approaches. For instance, the MRWA datasets will be used for the road network translation, the road network conflation and the route planning approaches. The Landgate data will be employed for the road network translation and the road network data conflation approaches. The Western Power dataset and the OSM datasets will only be used for one approach each, i.e. the route planning and the road network data conflation, respectively.

Main Roads Western Australia metadata

The structure of the MRWA metadata information of traffic signal sites, legal speed limits, road stopping places, regulatory signs and road networks is shown in Table 3.1. The table view allows for the recognition of metadata identifiers among the shared MRWA datasets. For example, all MRWA datasets use an 'object identifier' for unique asset identification. Not all of the metadata are required for processing in this thesis, but they are still indicated in the table to represent the given information. The important metadata are as follows:

- Object identifier: a unique object identification.

Table 3.1: Metadata information of the MRWA datasets traffic signal sites, legal speed limits, road stopping places, regulatory signs, rail crossings, road closures, intersections and road networks.

MRWA Metadata	TSS	LSL	RSP	RS	RC	I	R	RN	MRWA Metadata	TSS	LSL	RSP	RS	RC	I	R	RN
Accessible tables			✓						Ownership			✓					
Accessible toilets			✓						Panel 01 design				✓				
Asset number				✓					Panel 01 design meaning				✓				
Commemoration way			✓						Panel 02 design				✓				
Common usage name		✓	✓	✓			✓	✓	Panel 02 design meaning				✓				
Constructed shelter			✓						Panel 03 design				✓				
Carriageway		✓	✓	✓				✓	Panel 03 design meaning				✓				
Closure type					✓				Panel 04 design				✓				
Date approved				✓					Panel 04 design meaning				✓				
Date inspected							✓		Panel count				✓				
Date installed				✓					RA name (region name)		✓	✓	✓			✓	✓
Datum NE ID								✓	RA number (region identifier)		✓	✓	✓			✓	✓
Effluent dump site			✓						Regulatory sign type				✓				
End node name								✓	Region					✓			
End node number								✓	Rest area name			✓					
End SLK		✓	✓					✓	Rest area type			✓					
End true distance		✓	✓					✓	Road		✓	✓	✓	✓		✓	✓
Entry date					✓				Road name		✓	✓	✓			✓	✓
Geoloc	✓	✓	✓	✓		✓		✓	Route NE identifier		✓	✓	✓			✓	✓
Geoloc ST length		✓						✓	Scenic lookout			✓					
Identifier					✓				Service status	✓							
IIT Protection							✓		Shape					✓			
Incident level					✓				Shape ST length								
Incident type					✓				Signal type	✓							
Information board			✓						Site reference identifier	✓							
Latitude				✓					SLK				✓				
LG name (town name)		✓	✓	✓			✓	✓	Speed limit		✓						
LG number (town identifier)		✓	✓	✓			✓	✓	Start node name								✓
Lighting present			✓						Start node number								✓
Location					✓				Start SLK		✓	✓				✓	✓
Longitude				✓					Start true distance		✓	✓					✓
Natural shade			✓						Stay 24 hour			✓					
Network element								✓	Suburb								
Network type		✓	✓	✓			✓	✓	Surface			✓					
NM begin MP (start measure)								✓	Surface area			✓					
NM end MP (end measure)								✓	Surface type			✓					
Node description	✓					✓			Traffic impact					✓			
Node identifier	✓					✓			True distance				✓				
Node name	✓					✓			Upload date time					✓			
Number of bins			✓						Xing No							✓	
Number of tables			✓						Xing Type							✓	
Number of toilets			✓						XSP (cross selection position)			✓	✓				
Object identifier	✓	✓	✓	✓	✓	✓	✓	✓									

Datasets: Traffic Signal Sites (TSS); Legal Speed Limits (LSL); Road Stopping Places (RSP); Regulatory Signs (RS); Road Closures (RC); Intersections (I); Rail Crossings (R) and Road Network (RN)

- Start SLK: the kilometre start point of a road asset.
- End SLK: the kilometre end point of a road asset.
- SLK: the kilometre point of a road asset.
- Road: a unique road identifier.
- Route network identifier: a unique identifier of a road that can consist of one or more road sections.
- Local government authority (LG) number: a unique identifier of a town.
- Regional authority (RA) number: a unique identifier of a region.
- Common usage name: a road name.
- Speed limit: the maximum allowed driving speed on a road.
- Regulatory sign type: the type of regulatory sign (e.g. give way and stop).
- Rest area type: the type of road stopping place (e.g. heavy vehicle rest area).

Landgate metadata

The structure of the Landgate road network metadata[1] is shown in Table 3.2. The dataset uses the entity 'object identifier' for unique asset identification. The required metadata entities are as follows:

- Object identifier: a unique object identification.
- FC sub-type: the classification of a road network asset, including roads, roundabouts and roundabout connectors.
- Genoma identifier: the Landgate data index entity.

[1] A detailed explanation of Landgate metadata is available at `https://catalogue.data.wa.gov.au/dataset/roads-lgate-012/resource/85ec9f86-17ee-4946-abbf-08b2cb0c1ccf`.

- MRWA classification: a match to MRWA road assets.
- MRWA road number: a match from a Landgate road to an MRWA road by using the MRWA metadata entity 'road'.
- Road name: road name without its type (e.g. Manning Road -> Manning).
- Road type: a road type extension, such as avenue, circuit, crest, drive, esplanade, heights, lane, parade and way.

Table 3.2: Metadata information of the Landgate road network dataset.

Landgate Metadata (LGATE-012)		
Access level	FC sub-type	Road access right
Attribute reliability data	Genoma identifier	Road custodian
Capture method	Geographic name	Road name
Data custodian	Geometric length	Road status
Data feature modified	Lane count	Road suffix
Data feature retired date	Map classification	Road surface
Data source	Metadata identifier	Road type
Date feature created	MRWA classification	Road usage
DEC category	MRWA road number	Spatial reliability date
DEC type	Object identifier	Speed limit
Direction category	OGC feature identifier	Target display
Elevation accuracy	Plan accuracy	Track hierarchy

Western Power metadata

The Western Power overhead power lines metadata contains the data entries 'object identifier', 'pick id', 'shape' and 'voltage'. Although Western Power provides the 'object identifier' attribute, as seen before in the MRWA and Landgate metadata descriptions, the official Western Power asset identifier metadata 'pick id' will be used for unique asset allocation. The entities 'voltage' and 'shape' will not be used, as it is sufficient to know that an overhead power line is present at a given location for the route planning approach.

SOM metadata

The structure of the OSM metadata is indicated in Table 3.3. Once the dataset is loaded with QGIS,[2] then an area of interest can be selected and saved as a line string layer in a machine-readable format, such as GeoJSON. The next list defines the metadata that are required for the road network conflation approach of this thesis:

- OSM identifier: a unique asset identifier.
- Highway: the information about a road asset types, such as residential, footway, cycleway, secondary, service, step, traffic signal, trunk and turning circle.
- Other tags: can contain different information about the max speeds, one-way roads, junctions, roundabouts, bicycle use, footways, surfaces, data sources, service roads, electrified and any other tag that the person who entered the data into OSM can define.

Table 3.3: Metadata information of the OSM line strings dataset.

OSM Line Strings Metadata	
Aerial way	OSM identifier
Barrier	Other tags
Highway	Waterway
Man-made	Z order
Name	

[2] QGIS is a free and an open-source GIS available at `http://www.qgis.org`. The QGIS software is used in this thesis to select road network selections and to save these selections in the GeoJSON data format.

3.4 Chapter Summary

This chapter introduced the datasets used in this study. The data selection included data from MRWA road networks, intersections, regulatory signs, traffic signals, road stopping places, road closures, rail crossings and speed limits. From the Landgate road network data, roundabouts and roundabout connectors were selected. For the consideration of overhead power lines for the route planner, the Western Power overhead power lines dataset was taken. The introduced datasets also include OSM and DBpedia data. For each employed dataset, the metadata were shown, and important data entries that will be employed in later chapters of this thesis were explained.

Chapter 4

Road Network Ontology Design

4.1 Chapter Introduction

This chapter will introduce the newly created ontology design of the road network conflation and route planning approaches. All ontologies used in this dissertation were developed and optimised for their specific use cases. For the road network conflation approach, the MRWA, OSM and Landgate ontologies are relevant, as is a further fourth ontology that manages conflated data; the design is based on Niestroj et al.'s 2019 work. The route planning approach will employ ontologies for Western Power and route processing as well as an MRWA ontology whose principle is based on the MRWA conflation ontology. In addition, newly created ontology-based data provenance models will be introduced to keep track of the origin of a dataset and to further define how trusted a dataset from a given source is.

4.2 Road Network Data Conflation

The road data network conflation ontology is used as a data warehouse, as it imports the ontologies from MRWA, Landgate and OSM and processes the semantic rules that are applied to all datasets. The first layer of the ontology

is visualised in Figure 4.1 and indicates that the ontology classes 'osm:OSM', 'mrwa:MRWA', 'landgate:Landgate' and ':ConflatedData' are connected through 'owl:Thing'. The following prefixes will be used for the road network conflation approach:

```
osm: <http://www.example.org/OSM/>
mrwa: <http://www.example.org/MRWA/>
landgate: <http://www.example.org/LANDGATE/>
: <http://www.example.org/DATA/>
```

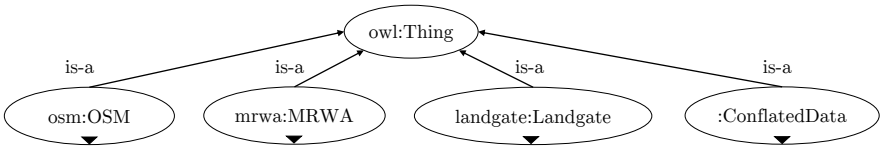


Figure 4.1: The top-level class layer of the conflated road network ontology.

Next, the content of each of the newly designed ontologies will be defined, meaning that the classes and object relations will be explained for each ontology.

4.2.1 MRWA Ontology

The MRWA ontology class is sub-divided into 'features', 'location' and 'road network' as indicated in Figure 4.2. The features class contains information on the categorisation of road network geometries. Regions and towns are sub-classes of the location class and included for the location identification of road assets. The road network class is used to classify intersections, inventories, roads, road sections and regulatory signs.

The newly created ontology contains essential object properties for the relations of MRWA data individuals so that the data relations can be processed and information that means the same thing can be reasoned. The following list describes the custom-created object properties that are required for the functionality of the data conflation process:

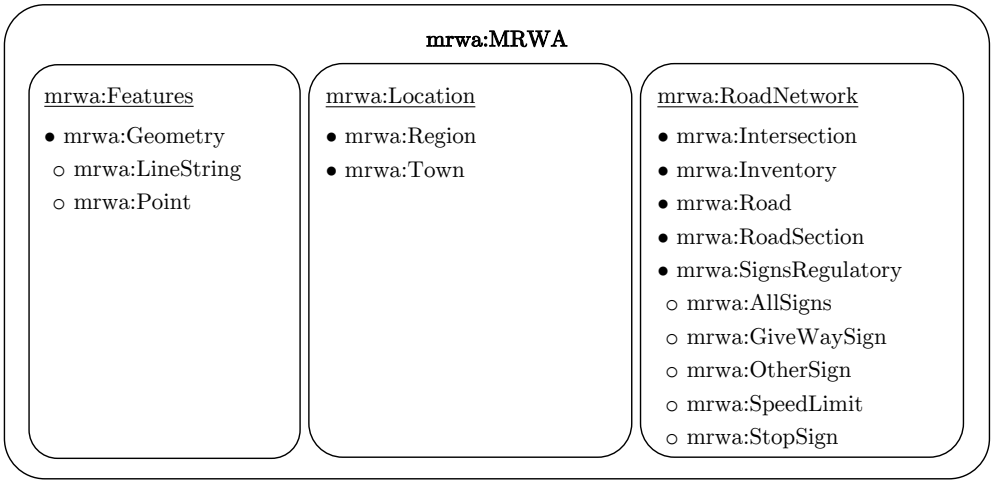


Figure 4.2: Structure of the MRWA ontology classes for the road network data conflation.

- 'mrwa:hasLineCoordinates' establishes a connection between a road section individual and its related line string. The inverse 'mrwa:isLineCoordinatesOf' is used to allocate a line string to a road section.

- 'mrwa:hasPointCoordinates' is used to assign road signs and intersections to point coordinate individuals. The object property is further used to separate line string individuals into point individuals, as point features are a requirement of the road network conflation. The inverse 'mrwa:isPointCoordinatesOf' identifies an individual of a point, which can be a sign, an intersection or a vertex of a line string.

- 'mrwa:hasRoad' is used to assign a road section to a road. The inverse is 'mrwa:isRoadOf' and classifies a road to a road section.

- 'mrwa:hasRoadSection' is an object property of intersections that allocates a road section to an intersection. The inverse is 'mrwa:isRoadSectionOf'.

- 'mrwa:hasTown' is used to assign a town to a road. Its inverse is 'mrwa:isTownOf' and describes the direction from a town to a road.

- 'mrwa:hasRegion' is an object property to assign a town to a region, e.g. the town 'Wanneroo' is in the 'Perth metropolitan' region. Its inverse is 'mrwa:isRegionOf'.

- 'mrwa:isConnectedTo' is used to connect a road section to a neighbouring road section so that a road network that is based on road section individuals can be constructed.

4.2.2 Landgate Ontology

The newly created Landgate ontology class is sub-divided into 'features' and 'road network' as shown in Figure 4.3. The features class is used for the categorisation of road network geometries, and the road network class is used to classify roundabout connectors, roads and roundabouts.

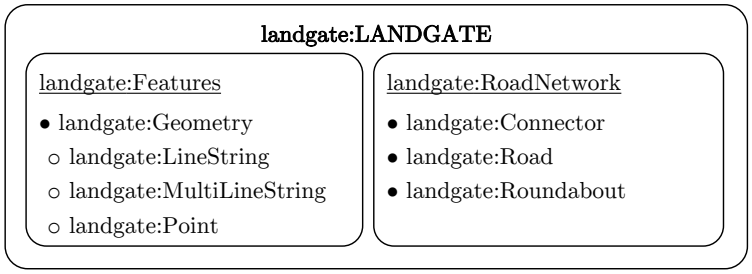


Figure 4.3: Structure of the Landgate ontology classes.

The ontology contains custom-created object properties for the relations of Landgate data individuals so that the data relations can be processed and information that means the same thing can be reasoned. The newly designed Landgate object properties are as follows:

- 'landgate:hasMultiLineCoordinates' is an object property of a road section that keeps track of the related multi-line strings, as road sections, roundabouts and roundabout connectors are represented by multi-line strings. The inverse is 'landgate:isMultiLineCoordinatesOf' and is used for the direction from a road asset to its multi-line string.

- 'landgate:hasLineCoordinates' is used for the sub-division of multi-line strings into line strings. The inverse is 'landgate:isLineCoordinatesOf' and is used for the connection of line strings to multi-line strings.

- 'landgate:hasPointCoordinates' is an object property of a line string, as each line string consists of at least two points that are separated into point individuals. The inverse 'landgate:isPointCoordinatesOf' connects a point to its line string.

- 'landgate:isConnectedToNode' is used to connect a road section to a neighbouring road section as well as to connect roundabout connectors to roundabouts and road sections.

- 'landgate:isGroupOfConnectors' is used to group the roundabout connectors of the same roundabout.

- 'landgate:isPartOfSameRoundabout' is used to group the roundabout individuals of the same roundabout.

4.2.3 OSM Ontology

The newly designed OSM ontology class is sub-divided into 'features' and 'road network' as indicated in Figure 4.4. The features class is used for the categorisation of road network geometries, and the road network class is used to set cycleways, residential roads, roads, roundabouts, service roads, tertiary roads, trunks and tracks.

Object properties are included in the ontology for the relations of OSM data individuals so that the data relations can be processed and information that means the same thing can be reasoned. The following list defines the designed object properties:

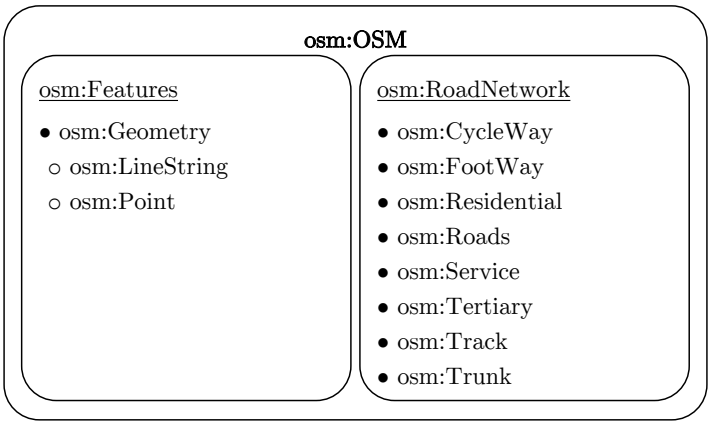


Figure 4.4: Structure of the OSM ontology classes.

- 'osm:hasLineCoordinates' is an object property of road network assets that allocates their related line strings. The inverse 'osm:isLineCoordinatesOf' is used to match a line string to its road network asset.
- 'osm:hasPointCoordinates' is an object property of line strings, as each line string consists of at least two points. The inverse is 'osm:isPointCoordinatesOf'.
- 'osm:isConnectedTo' is used to connect a road section to a neighbouring road section.

4.2.4 Conflated Data Ontology

The newly created conflated data ontology class is sub-divided into 'features', 'road network' and 'trust' as shown in Figure 4.5. The features class is used to set the road network geometries, and the road network class is used for the categorisation of intersections, roads and roundabouts. The trust class is used to rank the road asset dataset sources by a trust score, whereby a higher trust score indicates a more trusted data source (see Section 4.4.5: Trust Provenance). The classes mentioned will include data individuals from the MRWA, OSM and Landgate ontologies.

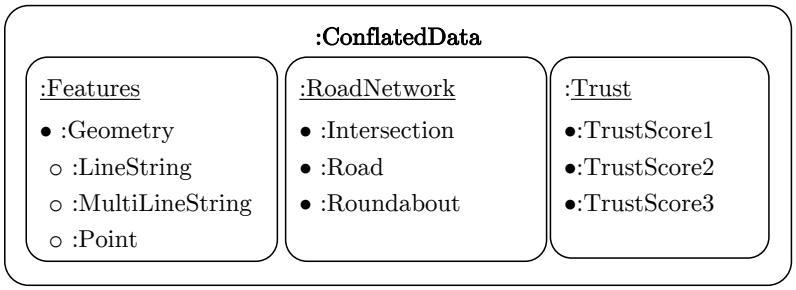


Figure 4.5: Structure of the conflated data ontology classes.

The ontology contains object properties for conflating MRWA, Landgate and OSM data so that data relations can be processed and information that means the same thing can be reasoned. The following list defines the newly defined object properties that are needed to identify data that means the same thing within the different datasets:

- ':hasIntersectionPart' is an object property that connects MRWA intersections to neighbouring road assets, such as road sections and roundabout connectors. The inverse ':isPartOfIntersection' connects road assets to MRWA intersections.

- ':hasSameLineCoordinates' is used to describe different line string individuals that share the same coordinates. The inverse ':isSameLineCoordinates' traces a line string back to the individual with the same coordinates.

- ':hasSameMultiLineCoordinates' is used to describe different multi-line string individuals that share the same coordinates. The inverse ':isSameMultiLineCoordinates' traces a multi-line string back to the individual with the same coordinates.

- ':hasSamePointCoordinates' is used to describe different individuals that share the same point coordinates. The inverse ':isSamePointCoordinates' traces a point back to its road asset.

- ':hasTrustedSource' is used to set the trust score of a dataset, which relates to its corresponding data source. The inverse ':isTrustedSource' traces a trust score back to its road asset.
- ':isSameRoadAs' is used to describe the road section data from 'Landgate', 'MRWA' and 'OSM' that refer to the same road.

4.3 Route Planner

The ontologies that will be used by the route planner are based on MRWA and Western Power datasets. The MRWA ontology is a simplified version of the ontology described in Section 4.2.1, as the unused elements of the road network conflation approach for the route planner have been removed for an efficient ontology size, meaning that the line strings are not sub-divided into points and that fewer object properties are used. In addition, road stopping places, rail crossings and traffic signal sites have been added. A 'route' ontology will be used to save a planned route so that previously calculated routes can be reloaded back into the route planner at any given time. Ontology reasoning will be not activated for the route planning, as it is currently not practical with large datasets. The following prefixes will be used in the route planning approach of this thesis for the MRWA, Western Power and route ontologies:

```
mrwa: <http://www.example.org/MRWA/>
wp: <http://www.example.org/WP/>
route: <http://www.example.org/ROUTE/>
```

4.3.1 MRWA Ontology

The MRWA class is sub-divided into 'features', 'location' and 'road network' as indicated in Figure 4.6. The features class is used for the categorisation of road network geometries. Regions and towns are sub-classes of the location class and

are used to inform about the geographic position of a road section. The road network class is used to classify roads, road sections, rail crossings, road stopping places, traffic signal sites and regulatory signs.

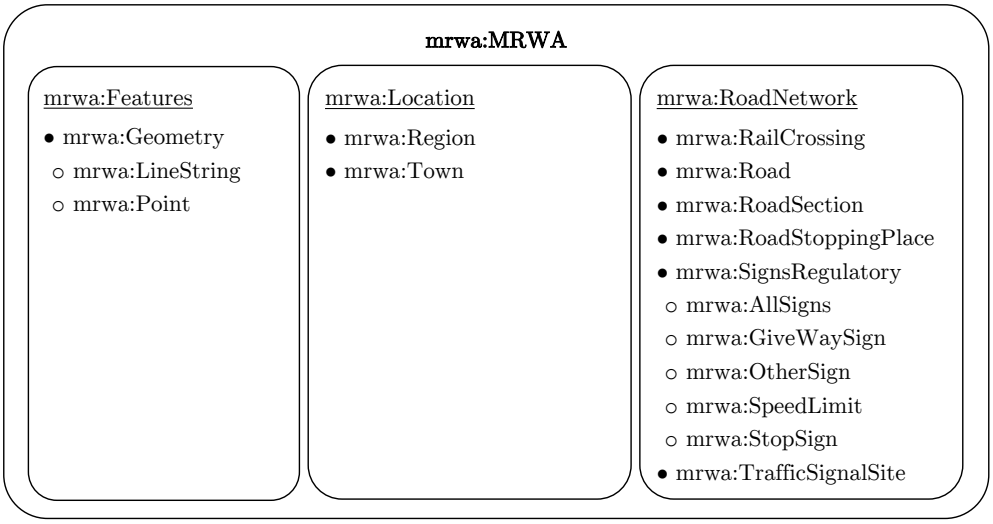


Figure 4.6: Structure of the MRWA ontology classes for route planning.

The ontology contains object properties for the relation of MRWA data individuals, so that data relations can be identified. The following list defines the newly designed object properties:

- 'mrwa:hasLineCoordinates' establishes a connection between a road section individual and its related line string.
- 'mrwa:hasPointCoordinates' is used to assign regulatory signs and intersections to their related point coordinate individuals.
- 'mrwa:hasRoad' is used to assign a road section to a road.
- 'mrwa:hasTown' is used for roads to set a town of a road.
- 'mrwa:hasRegion' is an object property to assign a town to a region, e.g. the town 'Coolgardie' is in the 'Goldfields Esperance' region.

4.3.2 Western Power Ontology

The Western Power ontology class is sub-divided into 'features' and 'overhead power lines' as shown in Figure 4.7. The features class contains information for the categorisation of multi-line strings, which are used for the localisation of overhead power lines. The overhead power lines class provides information about a power lines data source, which is (in the scope of this thesis) the Western Power 'WP-031'[1] dataset from the Western Australian data portal. The ontology contains an object property for the relation of Western Power data individuals, which is defined as follows:

- 'wp:hasMultiLineCoordinates' establishes a connection between an overhead power line individual and its related multi-line string.

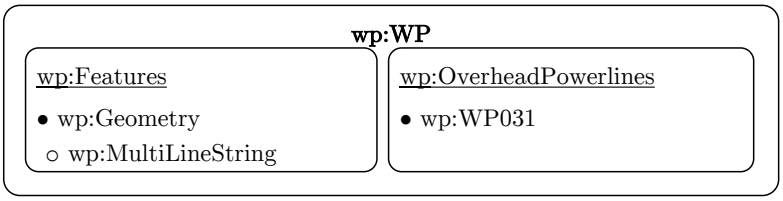


Figure 4.7: Structure of the Western Power ontology classes.

4.3.3 Route Ontology

The route ontology class is sub-divided into 'edge' and 'key node' as indicated in Figure 4.8. The edge class is used for the representation of route sections, and the key node class is used to indicate the start and end of a route.

The ontology contains object properties to set the data relations of route data individuals to establish a connection between the compounding road sections of a route. The following list defines the newly created object properties:

- 'route:hasNextRoute' points to the next route element.

[1] `https://catalogue.data.wa.gov.au/dataset/distribution-overhead-power-lines-wp-031`

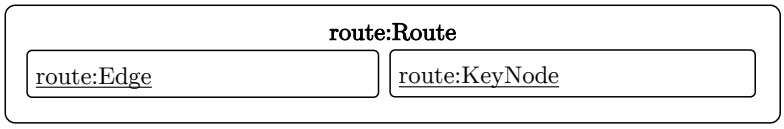


Figure 4.8: Structure of the route ontology classes.

- 'route:hasNextRoadEdge' establishes a connection between the next route element and the corresponding MRWA road edge individual.
- 'route:hasPreviousRoute' points to a previous route element.
- 'route:hasPreviousRoadEdge' establishes a connection between a previous route element and a corresponding MRWA road edge individual.

4.4 Data Provenance Models

Each of the created ontologies has a newly designed underlying *PROV-O* data model that describes the relations of the agents, activities and entities. This section will provide an introduction to the newly created provenance models for the MRWA, Landgate, OSM and Western Power datasets. The contribution of this section, i.e. creating provenance for road asset data, is novel and, to the best of the author's knowledge, has not been done by other researchers. Overall, the models share a similar approach, except for the model of the road network translation approach. A hash table will be used with hash values to simplify the explanation of the provenance models that share the same layout.

4.4.1 MRWA Provenance

The provenance of simple[2] MRWA relations is shown in Figure 4.9. The graph uses the hash values '<data>', '<process>', '<asset>' and '<location>' as placeholders for the provenance of road sections, regulatory signs, traffic signal sites,

[2] Simple means in this context that line strings are not further sub-divided into points, and that features have not been translated.

intersections, road stopping places, rail crossings and speed limits, as indicated in Table 4.1. To retrieve the right provenance model for each of the assets mentioned, e.g. road sections, one has to select the appropriate table row value and put it in place of the angle brackets ('<' and '>'). For instance, a road section 'mrwa:RoadSection' had the dataset 'govdata:98c8107c-1244-4511-8369-3b9c0d75bca6 (Road Network)' as 'prov:hadPrimarySource', which 'prov:wasUsed-By' the process 'alg:processMrwaRoadNetwork'. The process 'prov:wasAssociated-With' the agent 'mrwa:MRWA' and 'prov:generated' the road section, which 'prov:was-Attributed' to the agent 'mrwa:MRWA'. The location of the road section was at the entity 'mrwa:LineString', which was 'prov:generated' by the process 'alg:processMrwaRoadNetwork'.

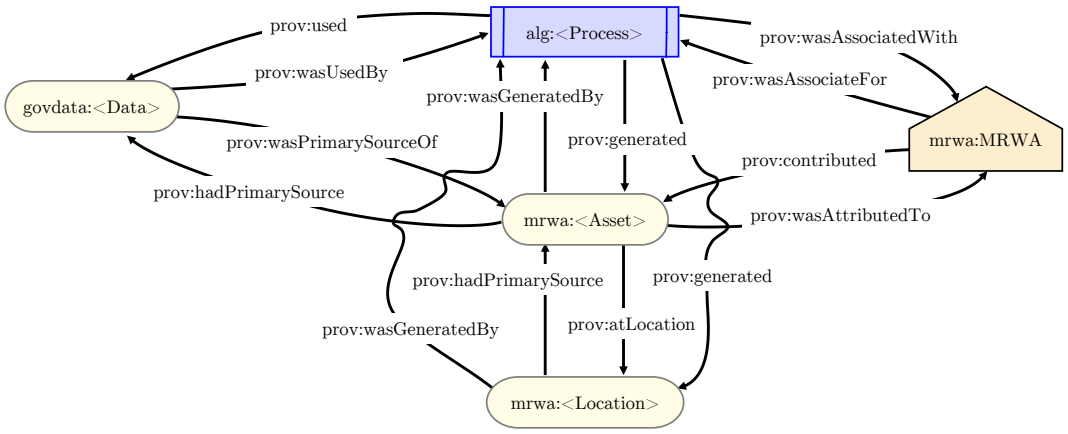


Figure 4.9: Provenance model of simple MRWA relations.

For better readability, the inverses of each of the above-mentioned provenance relations, such as 'prov:used', 'prov:wasPrimarySourceOf', 'prov:wasAssociateFor', 'prov:wasGeneratedBy' and 'prov:contributed', are not mentioned here but can be identified interactively from the graph. [3]

The MRWA provenance model designed for the translated intersections is shown in Figure 4.10. One can see that the upper part of the graph that interconnects the data source 'govdata:1f40e3a3-87a6-4d1b-8869-62bbdbe3bc3c', the road

[3] In addition, the provenance graphs that can be extracted from the hash values in Table 4.1 and the graph in Figure 4.9 are visualised in the appendix in Section A.2.

Table 4.1: Hash values for the placement in the provenance graph in Figure 4.9.

Data	Process	Asset	Location
98c8107c-1244-4511-8369-3b9c0d75bca6 (Road Network)	processMrwaRoadNetwork	RoadSection	LineString
f3336e73-38f8-4456-a8d6-e18a52527495 (Regulatory Signs)	processMrwaSigns	RegulatorySign	Point
1ab843ac-1b21-4ee8-892e-2d324a56bd78 (Traffic Signal Sites)	processMrwaTrafficSignalSites	TrafficSignalSite	Point
1f40e3a3-87a6-4d1b-8869-62bbdbe3bc3c (Intersections)	processMrwaIntersections	Intersection	Point
8ff91d94-4604-4c95-a036-76390b77aeaa (Road Stopping Places)	processMrwaRoadStoppingPlaces	RoadStoppingPlace	Point
203c7fac-14f6-45f2-affd-e04559aa150c (Rail Crossings)	processMrwaRailCrossings	RailCrossing	Point
1ab843ac-1b21-4ee8-892e-2d324a56bd78 (Legal Speed Limits)	processMrwaSpeedLimit	SpeedLimit	LineString

asset ‘mrwa:Intersection’, the agent ‘mrwa:MRWA’, the process ‘alg:processMrwa-Intersections’, which creates an intersection, and their location ‘mrwa:Point (original)’ uses the same provenance that has been described above for the example with hash values. The difference is that in addition, activity ‘alg:translationMethod’ exists and ‘prov:used’ an original point that ‘prov:generated’ an ‘mrwa:Point (translated)’; a translated point is also mapped as a ‘prov:atLocation’ from an ‘mrwa:Intersection’.

The design of the translated MRWA road sections indicated in Figure 4.11 seems complex at first glance. However, with the knowledge from the previously described translated intersection provenance models, it can seen that only two new entities have been added to the lower half of the graph, both of which concern the location of a road section. The reason for the two extra entities is that an original MRWA line string will be sub-divided into its original vertex points. The original points and the original line string of a road section are ‘prov:wasGeneratedBy’ both by the process ‘alg:processMrwaRoadNetwork’. The process ‘alg:TranslationMethod’ ‘prov:used’ original intersection points and ‘prov:generated’ the entity ‘mrwa:Point (translated)’ as well as ‘prov:generated’ the resulting ‘mrwa:LineString (translated)’ as a result of the translated points. All points and line strings in the graph are described as a ‘prov:atLocation’ from

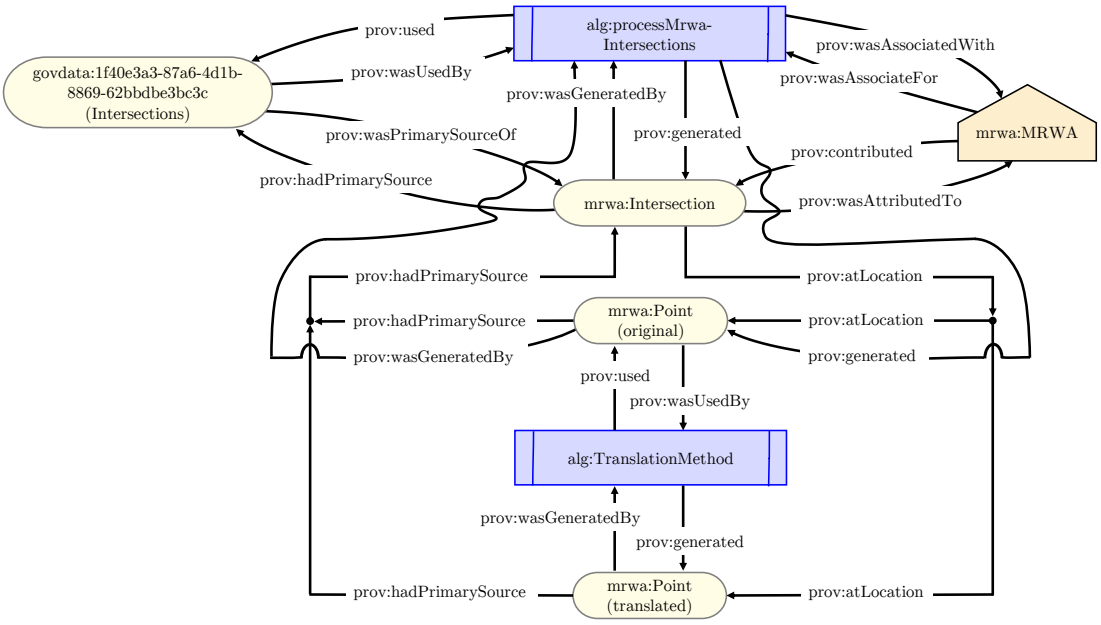


Figure 4.10: Provenance model of translated MRWA intersections.

'mrwa:roadSection, which, inversely, is the inverse a 'prov:hadPrimarySource' of each feature.

4.4.2 Landgate Provenance

The newly created Landgate data provenance is indicated in Figure 4.12 and shows the tracking of the creation of 'landgate:RoadAsset', e.g. the road sections, roundabouts and roundabout connectors, as well as the resulting features 'landyuchgate:MultiLineString', 'landgate:LineString' and 'landgate:Point'.

The process 'alg:processLandgateSlipDataSet' 'prov:generated' the road assets and features. Once a 'landgate:RoadAsset' is 'prov:wasGeneratedBy' the process, the related 'landgate:MultiLineString' was 'prov:generated' and further sub-divided into instances of 'landgate:LineString' and 'landgate:Point'. The agent 'landgate:LANDGATE' had the attribute 'prov:wasAssociateFor' for the process and 'prov:contributed' the road assets.

Each entity used a 'prov:hadPrimarySource' attribute which reffered to the 'landgate:RoadAsset' entity. The dataset 'govdata:85d59328-9eb6-4cdf-b2c0-

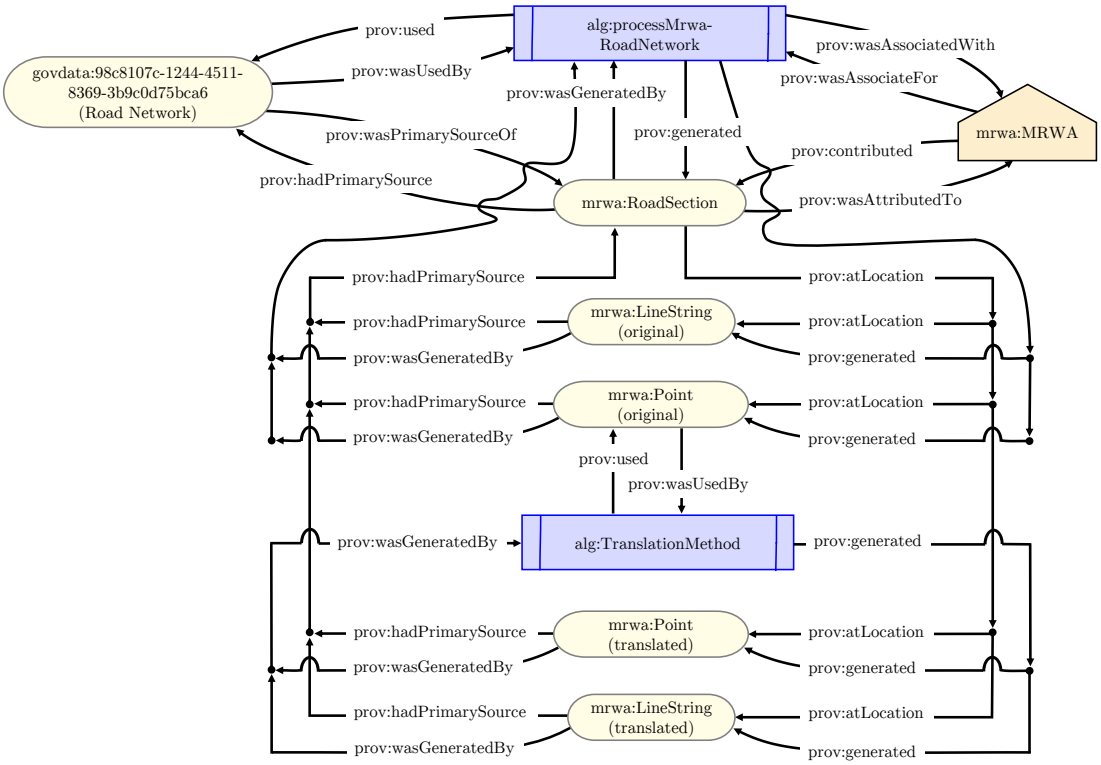


Figure 4.11: Provenance model of MRWA road sections for the road network conflation and the road network translation approaches.

358a141ccb25' was the 'prov:wasPrimarySource' of 'landgate:RoadAsset' and 'prov:wasUsedBy' the process 'alg:processLandgateSlipDataSet'.

4.4.3 Western Power Provenance

The provenance graph of the Western Power overhead power lines dataset is indicated in Figure 4.13. The power line entity 'wp:OverheadPowerline' was 'prov:generated' by the process 'alg:processWPOverheadPowerlines' and had the entity 'govdata:ca52f0ad-5705-44ae-84c0-9e5551471997' as the 'prov:hadPrimarySource'. The process 'prov:generated' the multi-line string 'wp:MultiLineString', which is a 'prov:atLocation' of an 'wp:OverheadPowerline'. The agent 'wp:WP' was the 'prov:was-AssociateFor' the process and contributed to the overhead power line asset. The process 'prov:used' the 'govdata:ca52f0ad-5705-44ae-84c0-9e5551471997'

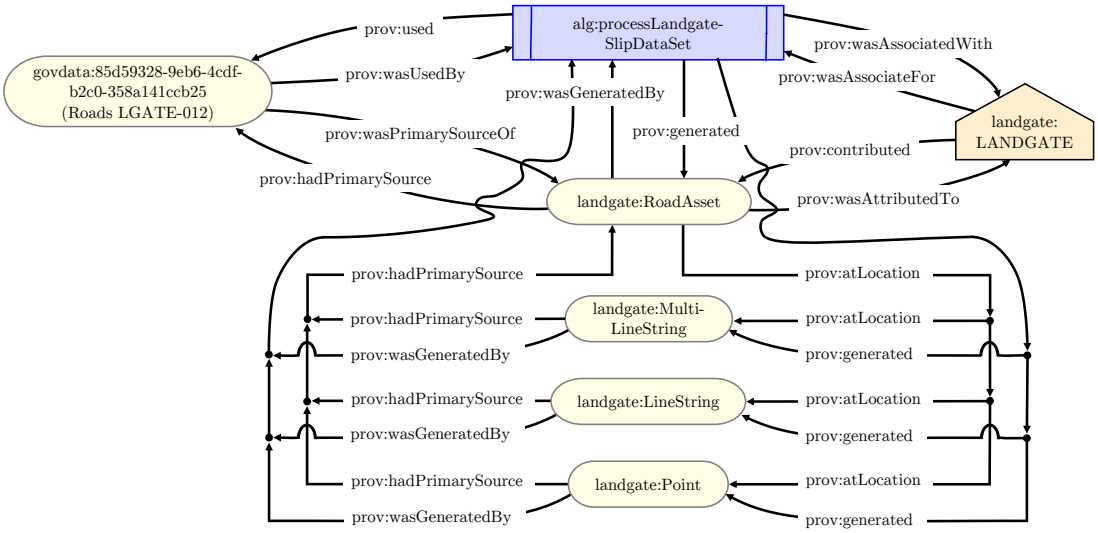


Figure 4.12: Provenance model of the Landgate road asset that will be used for the road network conflation and road network translation approaches.

dataset. The inverse provenance properties 'prov:wasAttributedTo', 'prov:wasAssociated-With' and 'prov:wasUsedBy' can be retrieved interactively from the graph.

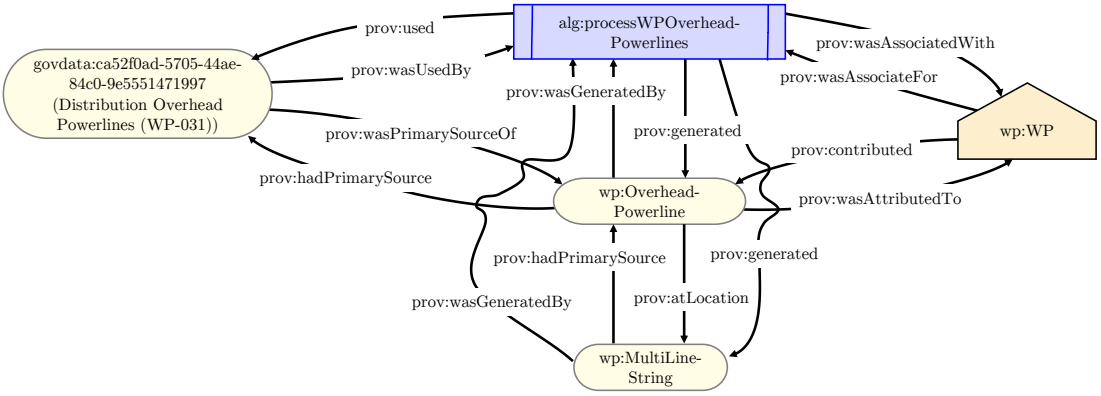


Figure 4.13: Provenance model of line strings from the Western Power overhead power lines dataset.

4.4.4 OSM Provenance

The newly created provenance of the OSM map lines dataset is indicated in Figure 4.14 and shows the 'osm:RoadAsset' entity, which represents the OSM roads and

roundabouts. Each 'osm:LineString' is a 'prov:atLocation' of an 'osm:RoadAsset'. The line strings are sub-divided into vertex points ('osm:Point'), which are a 'prov:atLocation' of 'osm:RoadAsset'. The road assets and locations are generated by the process 'alg:processOsmMapLines'.

The process 'prov:wasAssociatedWith' with the agent 'osm:OSM', and the agent 'prov:contributed' to the road asset entities. The data source 'geofabrik:australialatest.osm.pbf'[4] 'prov:used' the process 'alg:processOsmMapLines' and was the 'prov:wasPrimarySourceOf' of the road asset.

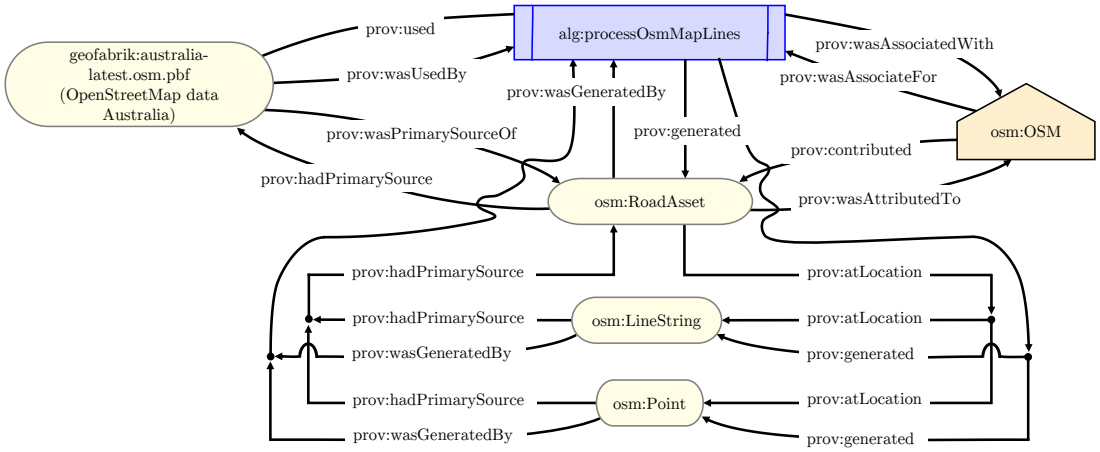


Figure 4.14: Provenance model of line strings from the OSM dataset.

4.4.5 Trust Provenance

The provenance model that is used to describe the trust score of the OSM, Landgate and MRWA datasets is indicated in Figure 4.15. The model enables a trust score allocation from one to three, with a higher value indicating a more trusted data source. The OSM data was collected by community effort, and OSM's road network datasets will receive a trust score of one. The Landgate datasets are approved by a governmental authority and will receive a trust score of two. As, MRWA specialises in road networks and its data are approved by a

[4] The prefix 'geofabrik' is used for 'https://download.geofabrik.de/australia-oceania/', which is a server that hosts OSM data.

governmental authority, the MRWA datasets will receive the highest trust score of three.

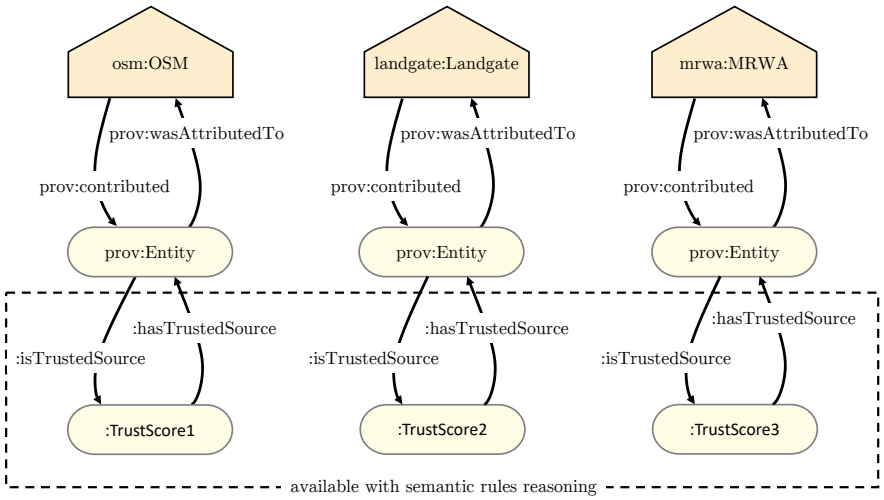


Figure 4.15: Provenance model that applies a trust score between one and three for the OSM, Landgate and MRWA datasets.

The allocation of the previously described trust scores will be performed after the application of semantic rules, as seen in the next section with rules 19–21. The newly created semantic rules identify a 'prov:agent of a 'prov:Entity' (e.g. road section, roundabout, intersection and regulatory sign) and set the corresponding trust score for each entity. For instance, a 'prov:Entity' of type 'MRWA:RoadSection' was 'prov:AttributedTo' the agent 'mrwa:MRWA'. Therefore, the related semantic rule will set ':isTrustedSource' to 'TrustScore3'.

Trust is a complex research branch in the Semantic Web. Several works that use custom trust-based Semantic Web models exist, such as those of Wang et al. (2015); Richardson et al. (2003); Golbeck et al. (2003); Yu et al. (2018); Artz & Gil (2007). As provenance of and trust in data on the web are an important research topic, a significant and novel contribution has been provided in this section, which presented a trust approach for road asset data. The investigation into a complex Semantic Web trust model for road networks is out of the scope of this thesis, as this thesis focuses on the representation of road network data

with the use of Semantic Web technologies, which leads to a case study of a route planner driven by these technologies.

4.5 Chapter Summary

This chapter delivered a new approach for the data management of road network data using Semantic Web technologies for the identification of road assets. A newly created conflated data ontology was introduced for the road network conflation approach as a main ontology that is capable of compounding the newly created OSM, MRWA and Landgate ontologies.

The newly created ontologies for the route planner were based on datasets from MRWA and Western Power. The two ontologies were designed to deliver road network information for the ontology-based route planning approach. In addition, a newly created route ontology was indicated and will be used in later parts of this thesis for the machine-understandable saving and reading of planned routes.

The newly created provenance models supported a trust-based approach for the data quality of road network assets based on their source. The development of a trust approach for the quality of road network data sources has not been accomplished by other researchers before and is a significant contribution of the author.

Chapter 5

Method

5.1 Chapter Introduction

This study has various stages, including the identification of Semantic Web technologies that can be applied to road network data, the GeoJSON data migration into the RDF format, the comparison of datasets with data that mean the same thing, and data provenance tracking. These stages all lead to a route planning approach for heavy vehicles driven by Semantic Web technologies.

To include Semantic Web rules within the context of this dissertation is a significant contribution, as it will enable machines to understand heterogeneous metadata that mean the same thing. Thus, this chapter will highlight the newly created Semantic Web rules, which are a key development towards road asset data harmonisation and the trust aspect of road asset data sources.

As this study uses MRWA and Landgate road network data, an investigation into the differences of each data source is significant, as both MRWA and Landgate are Western Australian authorities that provide access to their road network data through the Western Australian Government data portal.[1] Therefore, to activate the data comparison of MRWA and Landgate road network data, translation methods were newly designed by the author to translate the MRWA road networks

[1] http://data.wa.gov.au

data to the location of Landgate data. The translation methods will be further explained in this chapter.

This chapter will also be used to explain the key features of the newly created route planner so that the reader will be informed of its functionalities. In addition, the OWL Java API and the GeoTools GIS Toolkit are external libraries for the use of Semantic Technologies within Java software developments. The use of these two APIs was integrated by the author to simplify the use of Semantic Web technologies with a common programming language, such as Java.

5.2 Semantic Rules for Road Asset Conflation and Trust

The road network conflation approach enables the identification of road asset data that mean the same thing throughout the MRWA, Landgate and OSM datasets. For a working data conflation, SWRL rules need to be defined so that the ontology reasoner can reason over the data. In this section, Objective 4 of this dissertation will be addressed with significant and novel SWRL rules for the road network that are critical for the identification of road asset data that mean the same thing in the context of this thesis. In some of the following semantic rules, offset values for longitude and latitude that were selected based on experience values that worked in the tested road network selections will be defined; the offset values will be used to identify the road assets in a defined offset and are not related to possible survey errors. In addition, this section will contain newly defined SWRL rules for the novel and significant road network data trust models. A description of the contributed semantic rules is given next:

Rules 1, 2: Determine the same road within the MRWA and Landgate data, and Landgate and MRWA data.

```
landgate:Road(?b) ^ swrlb:notEqual(?bValue, "") ^
mrwa:ROAD(?a, ?aValue) ^ mrwa:RoadNode(?a) ^
swrlb:notEqual(?aValue, "") ^ swrlb:contains(?aValue, ?bValue)^
landgate:mrwaroadnumber(?b, ?bValue) -> isSameRoadAs(?a, ?b)
```

```
landgate:Road(?b) ^ swrlb:notEqual(?bValue, "") ^
mrwa:ROAD(?a, ?aValue) ^ mrwa:RoadNode(?a) ^
swrlb:notEqual(?aValue, "") ^ swrlb:contains(?aValue, ?bValue)^
landgate:mrwaroadnumber(?b, ?bValue) -> isSameRoadAs(?b, ?a)
```

Rules 3, 4: Determine the same road within the Landgate and OSM data, and OSM and Landgate data. The property 'landgate:rd_name_decoded' uses custom-created road name metadata that do not use road-type abbreviations (e.g. Street instead of St, Close instead of Cl, Avenue instead of Av and so on) to match with the OSM metadata road name notations. The unabbreviated values are extracted from Landgate (2018) and will be processed quickly with a hash table when creating the data individuals.

```
swrlb:containsIgnoreCase(?aValue, ?bValue) ^
landgate:rd_name_decoded(?a, ?aValue) ^ osm:name(?b, ?bValue)
-> isSameRoadAs(?a, ?b)
```

```
swrlb:containsIgnoreCase(?aValue, ?bValue) ^
landgate:rd_name_decoded(?a, ?aValue) ^ osm:name(?b, ?bValue)
-> isSameRoadAs(?b, ?a)
```

Rules 5, 6: Determine the same road within the MRWA and OSM data and OSM and MRWA data. The property 'mrwa:ROAD_NAME_DECODE' uses

custom-created road name metadata that do not use road-type abbreviations (e.g. Street instead of St, Close instead of Cl, Avenue instead of Av and so on) to match with the OSM metadata road name notations.

```
swrlb:containsIgnoreCase(?aValue, ?bValue) ^
mrwa:ROAD_NAME_DECODE(?a, ?aValue) ^ mrwa:RoadNode(?a) ^
osm:name(?b, ?bValue) -> isSameRoadAs(?a, ?b)
```

```
swrlb:containsIgnoreCase(?aValue, ?bValue) ^
mrwa:ROAD_NAME_DECODE(?a, ?aValue) ^ mrwa:RoadNode(?a) ^
osm:name(?b, ?bValue) -> isSameRoadAs(?b, ?a)
```

Rule 7: If an MRWA intersection is within a radius of 0.0001° longitude (≈9.45 m) and 0.0001° latitude (≈11.09 m) of an MRWA road section, set the intersection as part of the road section.[2]

```
mrwa:LONGITUDE(?b1, ?b1Long) ^
swrlb:subtract(?rangeLong, ?cLong, ?b1Long) ^
mrwa:LATITUDE(?b1, ?b1Lat) ^
swrlb:lessThanOrEqual(?absrangeLat, "0.0001"^^xsd:decimal) ^
mrwa:LONGITUDE(?c, ?cLong) ^ mrwa:LineString(?b) ^
mrwa:Point(?c) ^ mrwa:hasPointCoordinates(?a1, ?b1) ^
swrlb:abs(?absrangeLong, ?rangeLong) ^
swrlb:subtract(?rangeLat, ?cLat, ?b1Lat) ^
```

[2] The transformation from longitude and latitude in distances in metres was processed with Vincenty's inverse formula and applied for the coordinates in Perth, Western Australia. An online applet for the transformation is provided by Geoscience Australia at `https://geodesyapps.ga.gov.au/vincenty-inverse`.

```
mrwa:LATITUDE(?c, ?cLat) ^ mrwa:Intersection(?a1) ^
swrlb:lessThanOrEqual(?absrangeLong, "0.0001"^^xsd:decimal) ^
mrwa:hasLineCoordinates(?a, ?b) ^ mrwa:RoadNode(?a) ^
mrwa:hasPointCoordinates(?b, ?c) ^ mrwa:Point(?b1) ^
swrlb:abs(?absrangeLat, ?rangeLat) -> hasIntersectionPart(?a1, ?a)
```

Rule 8: If an MRWA intersection is within a radius of 0.0001° longitude (≈9.45 m) and 0.0001° latitude (≈11.09 m) of a Landgate roundabout connector, set the intersection as part of the connector.

```
mrwa:LONGITUDE(?b1, ?b1Long) ^ mrwa:LATITUDE(?b1, ?b1Lat) ^
swrlb:lessThanOrEqual(?absrangeLat, "0.0001"^^xsd:decimal) ^
swrlb:subtract(?rangeLong, ?dLong, ?b1Long) ^ landgate:Point(?d) ^
landgate:LineString(?c) ^ landgate:MultilineString(?b) ^
mrwa:hasPointCoordinates(?a1, ?b1) ^
swrlb:abs(?absrangeLong, ?rangeLong) ^
landgate:hasMultilineCoordinates(?a, ?b) ^ mrwa:Intersection(?a1) ^
swrlb:lessThanOrEqual(?absrangeLong, "0.0001"^^xsd:decimal) ^
landgate:LATITUDE(?d, ?dLat) ^ landgate:Connector(?a) ^
landgate:hasLineCoordinates(?b, ?c) ^
landgate:LONGITUDE(?d, ?dLong) ^
landgate:hasPointCoordinates(?c, ?d) ^
swrlb:subtract(?rangeLat, ?dLat, ?b1Lat) ^
mrwa:Point(?b1) ^ swrlb:abs(?absrangeLat, ?rangeLat)
-> hasIntersectionPart(?a1, ?a)
```

Rule 9: If an MRWA intersection is within a radius of 0.0001° longitude (≈9.45 m) and 0.0001° latitude (≈11.09 m) of a Landgate road section, set the intersection as part of the road section.

```
mrwa:LONGITUDE(?b1, ?b1Long) ^ mrwa:LATITUDE(?b1, ?b1Lat) ^
swrlb:subtract(?rangeLong, ?dLong, ?b1Long) ^ landgate:Point(?d) ^
landgate:LineString(?c) ^ landgate:MultilineString(?b) ^
swrlb:lessThanOrEqual(?absrangeLong, "0.0001"^^xsd:decimal) ^
mrwa:hasPointCoordinates(?a1, ?b1) ^
swrlb:abs(?absrangeLong, ?rangeLong) ^
landgate:hasMultilineCoordinates(?a, ?b) ^ mrwa:Intersection(?a1) ^
landgate:LATITUDE(?d, ?dLat) ^ landgate:Road(?a) ^
landgate:hasLineCoordinates(?b, ?c) ^
landgate:LONGITUDE(?d, ?dLong) ^
swrlb:lessThanOrEqual(?absrangeLat, "0.0001"^^xsd:decimal) ^
landgate:hasPointCoordinates(?c, ?d) ^
swrlb:subtract(?rangeLat, ?dLat, ?b1Lat) ^
mrwa:Point(?b1) ^ swrlb:abs(?absrangeLat, ?rangeLat)
-> hasIntersectionPart(?a1, ?a)
```

Rule 10: If an MRWA intersection is within a radius of 0.0002° longitude (≈18.90 m) and 0.0002° latitude (≈22.18 m) of an OSM road section, set the intersection as part of the road section.

```
mrwa:LONGITUDE(?b1, ?b1Long) ^
osm:hasLineCoordinates(?a, ?b) ^
```

```
swrlb:subtract(?rangeLong, ?cLong, ?b1Long) ^ osm:Point(?c) ^
mrwa:LATITUDE(?b1, ?b1Lat) ^ osmap:RoadNetwork(?a) ^
osm:LONGITUDE(?c, ?cLong) ^
mrwa:hasPointCoordinates(?a1, ?b1) ^
swrlb:abs(?absrangeLong, ?rangeLong) ^
swrlb:lessThanOrEqual(?absrangeLong, "0.0002"^^xsd:decimal) ^
swrlb:subtract(?rangeLat, ?cLat, ?b1Lat) ^
osmap:hasPointCoordinates(?b, ?c) ^ mrwa:Intersection(?a1) ^
swrlb:lessThanOrEqual(?absrangeLat, "0.0002"^^xsd:decimal) ^
osm:LATITUDE(?c, ?cLat) ^ osmap:LineString(?b) ^
mrwa:Point(?b1) ^ swrlb:abs(?absrangeLat, ?rangeLat)
-> hasIntersectionPart(?a1, ?a)
```

Rule 11: If an MRWA sign is within a radius of 0.00025° longitude (≈26.62 m) and 0.00025° latitude (≈27.72 m) of an MRWA intersection, set the signs as part of the intersection.

```
mrwa:LONGITUDE(?b1, ?b1Long) ^
swrlb:lessThanOrEqual(?absrangeLat, "0.00025"^xsd:decimal) ^
mrwa:AllSigns(?a) ^ mrwa:LATITUDE(?b1, ?b1Lat) ^
swrlb:subtract(?rangeLong, ?bLong, ?b1Long) ^
mrwa:LATITUDE(?b, ?bLat) ^ mrwa:Point(?b) ^
mrwa:hasPointCoordinates(?a1, ?b1) ^
swrlb:abs(?absrangeLong, ?rangeLong) ^
swrlb:lessThanOrEqual(?absrangeLong, "0.00025"^^xsd:decimal) ^
```

```
mrwa:LONGITUDE(?b, ?bLong) ^ mrwa:Intersection(?a1) ^
swrlb:subtract(?rangeLat, ?bLat, ?b1Lat) ^
mrwa:hasPointCoordinates(?a, ?b) ^
mrwa:Point(?b1) ^ swrlb:abs(?absrangeLat, ?rangeLat)
-> hasIntersectionPart(?a1, ?a)
```

Rule 12: Determine the same line string within the MRWA and Landgate data.

```
mrwa:LineString(?a) ^ mrwa:coordinates(?a, ?aValue) ^
landgate:LineString(?b) ^ landgate:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSameLineCoordinates(?a, ?b)
```

Rule 13: Determine the same line string within the MRWA and OSM data.

```
mrwa:LineString(?a) ^ mrwa:coordinates(?a, ?aValue) ^
osm:LineString(?b) ^ osm:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSameLineCoordinates(?a, ?b)
```

Rule 14: Determine the same line string within the Landgate and OSM data.

```
landgate:LineString(?a) ^ landgate:coordinates(?a, ?aValue) ^
osm:LineString(?b) ^ osm:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSameLineCoordinates(?a, ?b)
```

Rule 15: Determine the same point within the MRWA and Landgate data.

```
mrwa:Point(?a) ^ mrwa:coordinates(?a, ?aValue) ^
landgate:Point(?b) ^ landgate:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSamePointCoordinates(?a, ?b)
```

Rule 16: Determine the same point within the MRWA and OSM data.

```
mrwa:Point(?a) ^ mrwa:coordinates(?a, ?aValue) ^
osm:Point(?b) ^ osm:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSamePointCoordinates(?a, ?b)
```

Rule 17: Determine the same point within the Landgate and OSM data.

```
landgate:Point(?a) ^ landgate:coordinates(?a, ?aValue) ^
osm:Point(?b) ^ osm:coordinates(?b, ?bValue) ^
swrlb:equal(?aValue, ?bValue) -> hasSamePointCoordinates(?a, ?b)
```

Rule 18: Identify if an MRWA intersection is a roundabout.

```
RoadNetwork(?a) ^ hasIntersectionPart(?a, ?b) ^
landgate:Connector(?b) -> Roundabout(?a)
```

Rule 19: Set an OSM road network asset entity to a trust score of one.

```
prov:Entity(?a) ^ osm:OSM(?a) -> hasTrustedSource(?a, TrustScore1)
```

Rule 20: Set a Landgate road network asset entity to a trust score of two.

```
prov:Entity(?a) ^ landgate:Landgate(?a)
-> hasTrustedSource(?a, TrustScore2)
```

Rule 21: Set an MRWA road network asset entity to a trust score of three.

```
prov:Entity(?a) ^ mrwa:MRWA(?a)
-> hasTrustedSource(?a, TrustScore3)
```

5.3 Road Network Translation Features

The road network translation approach uses Landgate road section, roundabout and connector datasets as a reference for the MRWA road asset translation. In the scope of Objective 5, the vertices of the MRWA intersections and road sections are processed by an algorithm that considers a rules-based feature translation, that applies seven different methods in ascending order. Adjacent coordinates in offsets of 6.00 m, 6.20 m, 16.00 m and 25.00 m will be considered as valid neighbours depending on the applied translation method. The defined offsets are explicit based on the experience values that worked best during the algorithm design by empirical testing and are not associated with measurement uncertainties regarding the location accuracy of road assets. The reason for not including possible measurement uncertainties is related to the discrepancies of the observed road asset locations compared to the publicly available data taken from the MRWA open data portal, which is based on IRIS datasets (Karpinski, 2012). Table 5.1 summarises the translation methods with their configured offsets for the data processing in this thesis. The offsets may need adjustments for other datasets.

Methods 1 and 2 are used to translate MRWA intersection features to Landgate connectors and road sections, respectively. The Landgate features must be within a range of 6.00 m of an MRWA intersection. The same offset is applied in Method

Table 5.1: Road network translation methods and their offsets.

Method	Description	Offset
1	Translate an MRWA intersection to a Landgate connector point.	6.00 m
2	Translate an MRWA intersection to a Landgate road section point.	6.00 m
3	Translate an MRWA road section point to an MRWA intersection: - Method 3.1: intersection was translated by Method 1. - Method 3.2: intersection was translated by Method 2.	6.00 m
4	Translate an untranslated MRWA road section point to a Landgate road section: - Method 4.0: point remains between two Landgate road section line strings (exception case). - Method 4.1: translate to a road section point. - Method 4.2: translate to the nearest interpolated road section point.	25.00 m
5	Translate an MRWA intersection to the nearest translated MRWA road section point.	16.00 m
6	Translate an MRWA intersection to the centre of two Landgate road sections.	16.00 m
7	Translate an MRWA road section point to an MRWA intersection that previously untranslated or translated Method 5 or 6.	6.20 m

3 when translating MRWA road section features to translated MRWA intersections. If an MRWA road section feature is not translated before the processing of Method 4, then it will be translated to the nearest Landgate road section feature in a range of 25.00 m.

Method 5 translates an MRWA intersection to the nearest translated MRWA road section or in between an MRWA left and right carriageway so that the position of the intersection will be in between the two MRWA lanes.[3]

Method 6 translates an MRWA intersection in between two Landgate road sections in an offset of 16.00 m and is applied when MRWA provides for a two-lane road road node one piece of data, whereby the Landgate data represent the same road node with one datasets for each lane. If an intersection has been translated by Methods 5 or 6, then a related MRWA road section will be translated to this

[3] In the context of this thesis, the terms 'centreline', 'road section', 'edge' and 'lane' will always refer to a given centreline dataset of a road section.

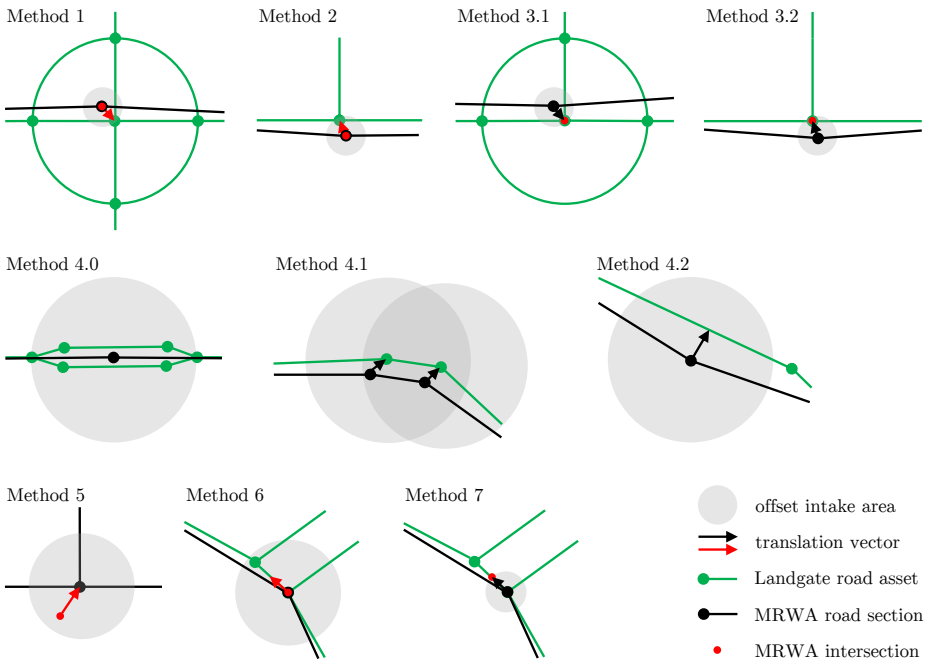


Figure 5.1: Graphical representation of the custom translation methods which are applied by the algorithm in ascending order.

intersection with Method 7 in a 6.20-m offset to enable a seamless road network representation.

For a better understanding of the translation methods and related offsets, a visualisation to indicate the translation principles is given in Figure 5.1.

5.4 Route Planner Features

The route planning approach implements Dijkstra's shortest path algorithm for the shortest route between two road nodes. An opportunity to select multiple points in order to determine a route through additional points of interest is given. Moreover, DBpedia will be queried with SPARQL statements to retrieve information about a town of a road section each time the town changes. The route planner can determine a road edge's weight by its distance or travel time. Configurable constraints can be taken into account to increase an edge weight, such as the following:

- Power lines: each time a road edge is crossed by an overhead power line, then a configurable value can be added to an edge weight.
- Rail crossings: If a rail crossing occurs at a road section, then a configurable value can be added to an edge weight.
- Turning circles: if a route from A to B requires left and/or right turns, then an additional edge weight can be configured per turn.
- Traffic signals: traffic signals are set to a green light by default. However, the signal state can be influenced by a random number generator that switches states between green and red. If a route includes passing through a red traffic signal, then a custom weight can be added to an edge weight.
- Regulatory signs: stop and give way signs are taken into account in the route planning, as the wait time at a sign can influence the travel time. Therefore, if a route includes regulatory signs, a configurable extra weight can be added to the related road edge.
- Road closures: road closures can be retrieved live from the Australian OGD portal to provide updated road closures information. If a road closure occurs, then its edge weight will be set to a very large value so that any other road edge will be considered as a better choice for the shortest route.

A further feature of the route planner is the consideration of road stopping places. This means that a user can configure to visit a road stopping place in a recurring range considering the air-line distance[4] from the start point to the road stopping place. A schema to plan a route with road stopping places is indicated in Figure 5.2. In Figure 5.2 a), a start node and a destination node are selected. Then, in Figure 5.2 b), a boundary polygon is created between the coordinates of the start node and the end node. After that, an algorithm iterates through the road stopping places dataset and selects the nearest road stopping place in

[4] A calculated distance between two points without considering the road network.

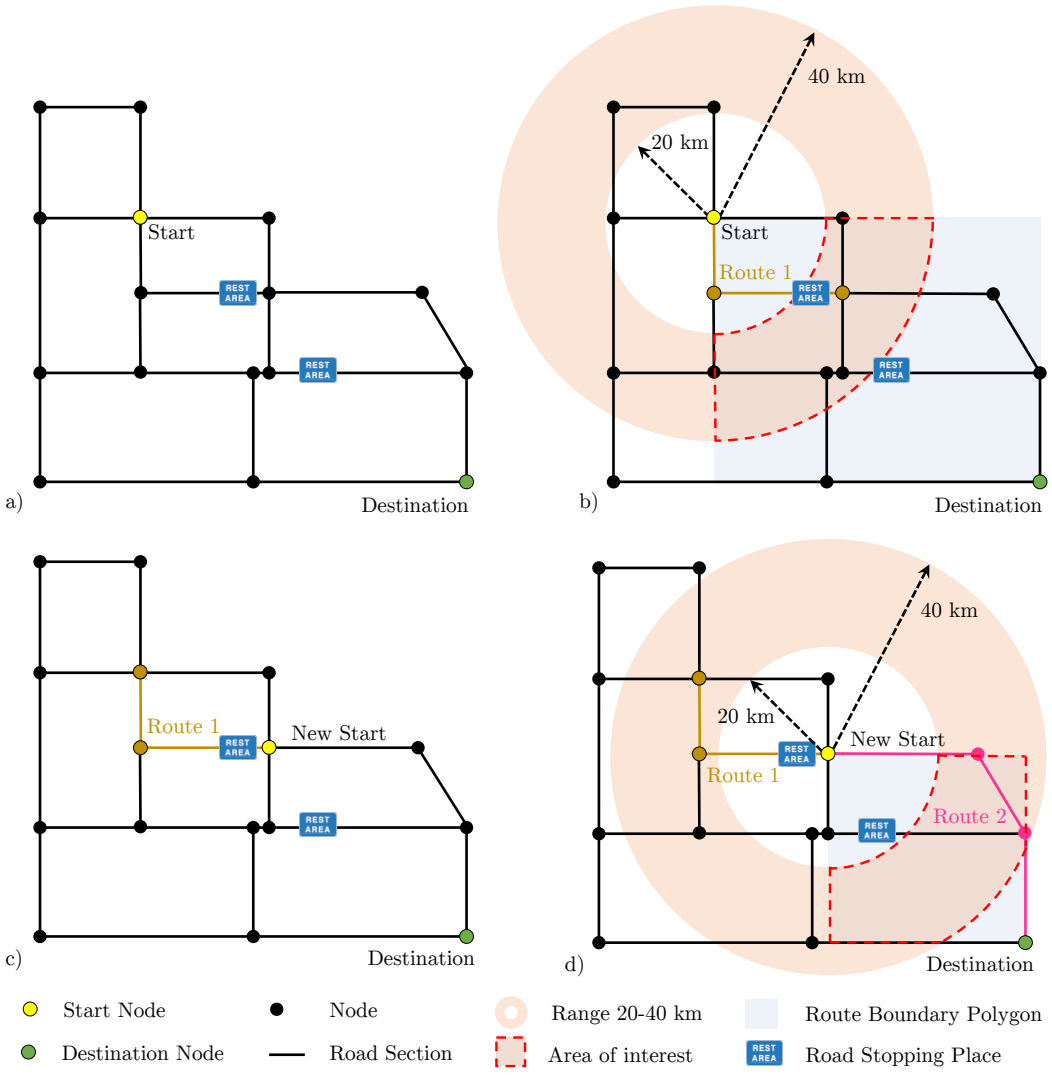


Figure 5.2: Schema to determine road stopping places between a start node and a destination node in a configured range.

the area of interest, which, in this example, is between 20 and 40 km within the highlighted boundary polygon. The identification of the road stopping place within the defined distance is done using Vincenty's[5] formula, which identifies a distance between two points. Next, a route can be calculated between the start node and nominated road stopping place. In Figure 5.2 c), a new start node is defined at the end of the calculated route, as the first temporary route to the road stopping place was processed. In Figure 5.2 d), the algorithm applies the same

[5] `http://www.movable-type.co.uk/scripts/latlong-vincenty.html`

process described before to identify a road stopping place in a configured range. As no road stopping place exists in the second area of interest in this example, the route planner will calculate a route from the newly defined start node to the destination node. Finally, Routes 1 and 2 will be merged into one compounded route (not indicated in Figure 5.2).

5.5 Route Planner Mock-Ups

The main frame and option frame functionalities of the route planner will be explained with mock-ups in this section. The mock-ups are used as a design guideline that contributes to thesis Objective 6. In later chapters of this thesis, the functionalities described here will be implemented, used with road network data and, finally, evaluated without further description. Thus, this section is fundamental for the understanding of the user interfaces. The mock-ups are generated with the Java Integrated Development Environment (IDE) IntelliJ IDEA from JetBrains.[6]

5.5.1 Main View

The mock-up of the main view of the route planning approach can be explained with a total of six panels, two left panels, i.e. three right panels and one middle panel (see Figure 5.3). The road network is displayed in the middle panel. The top-left panel shows the evaluation of the planned route, and the bottom-left panel will display the layer elements road nodes, road edges, regulatory signs, traffic signals, rail crossings, road stopping places, road closures and processed routes. The layers can be either shown, hidden or removed from the middle panel from the map using the buttons below the bottom-left panel.

The top-right area of the mock-up has two buttons, 'add routes' and 'reset map'. A click on the 'reset map' button will repaint the map to its initial state,

[6] `http://www.jetbrains.com/idea/`

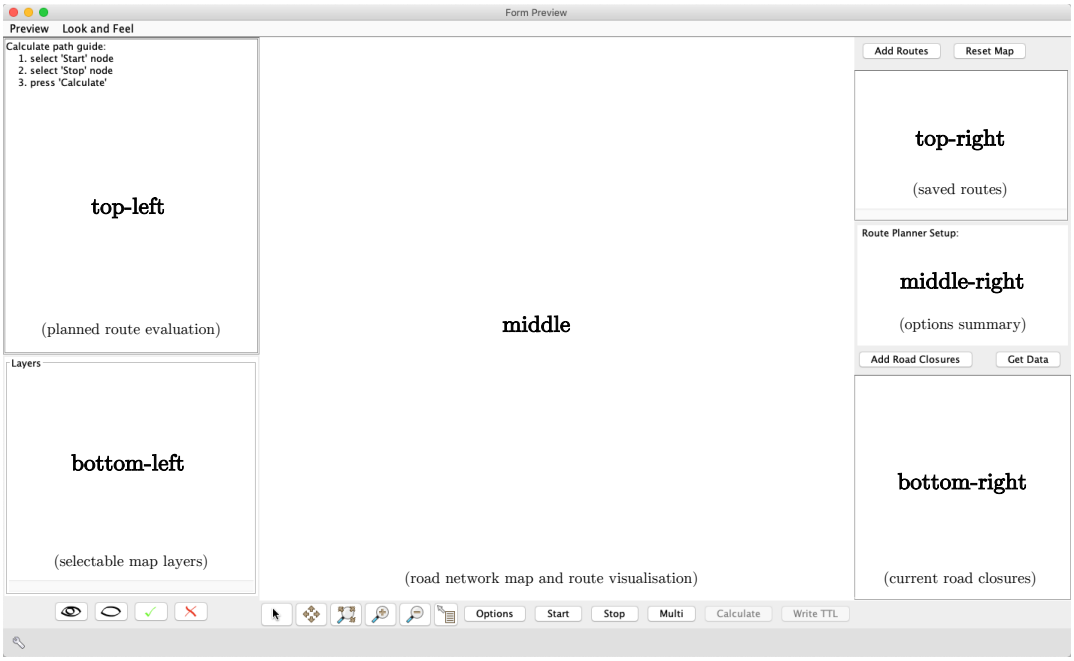


Figure 5.3: Mock-up of the route planner main view.

which is equal to the state after the programme start. Once the 'add routes' button is clicked, a previously calculated route will be listed in the panel below the buttons; a route can be loaded with a click on an entry.

The middle-right panel will summarise the current route planner configuration. Thus, the configured extra costs of power lines, rail crossings, left turns, right turns, traffic signals, stop signs and give way signs will be listed, as will the rest intervals of the configured road stopping places. In addition, the configured values for the edge weight types 'distance' and 'travel time', and the maximum number of processed routes will be shown.

Above the bottom-right panel are the buttons 'add road closures' and 'get data'. A click on the 'get data' button will download the current MRWA road closures data from the Australian OGD portal. After that, road closures can be added to the map with a click on the button 'add road closures'. The road closures data will be also listed as a text in the bottom-right panel.

Figure 5.4: Buttons related to the map content navigation and selection on the map panel in the order of road edge selection, map scrolling, fitting map to panel size, zoom in, zoom out and road asset information.

At the bottom of the main view panel, buttons are placed for map navigation and route planner interaction. The first six buttons are related to the navigation and the selection of the map content (see Figure 5.4). The buttons that are displayed next to the navigation buttons are explained in the next list:

- Options: view the options panel to configure the route planner.
- Start: after pressing the 'start' button, a road node can be selected and will be used as the start point for a to planned route.
- Stop: after pressing the 'stop' button, a road node can be selected and will be used as the end point for a to planned route.
- Multi: a click on the button 'multi' enables the selection of multiple travel points, which will be considered in ascending order by the route planning algorithm.
- Calculate: after the selection of either a start and end point or at least two multiple points, the 'calculate' button will be enabled, and a route can be calculated with a click on this button.
- Write TTL: the 'write TTL' button will be enabled after a route has been successfully calculated. A click on this button will save a processed route in the 'route' ontology in the RDF Turtle file format.

5.5.2 Options

Options are available, as shown in Figure 5.5. A user can choose to calculate an edge weight by its distance or its travel time. Next to the edge weight con-

figuration are options regarding the extra costs of power lines, turning circles, traffic signals, regulatory signs, and rail crossings. The options group allows the configuration of additional cost values in metres or seconds to be added to a corresponding edge weight. If processing multiple possible routes is required to find the best route, then a limit can be set for the number of route processings. The traffic signal configuration also contains a text box to randomise the traffic signal state. For example, if the value is set to '33%', then a random number generator will process random numbers between 0 and 100 for each traffic signal site. If the resulting number is below or equal to '33', then the traffic signal will be set to a green signal, and no additional edge weight will be added; otherwise, the traffic signal state will be set to a red signal state, and an extra weight will be added to a related road edge.

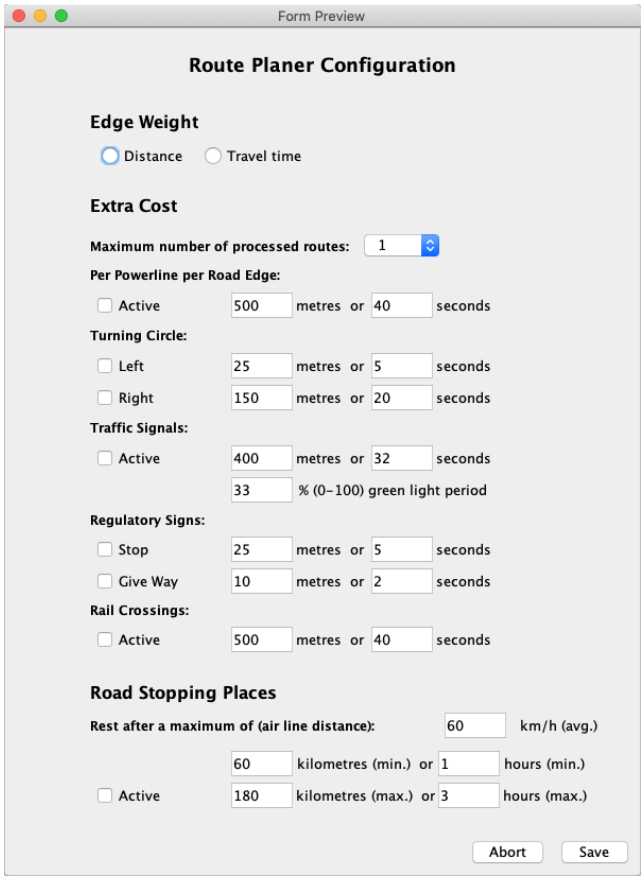


Figure 5.5: Mock-up of the route planner options panel.

The configuration for road stopping places can be used for an efficient identification of road stopping places. Therefore, the air-line distance between a start point and a road stopping place will be used, as it prevents the need to calculate a route between a start point and every road stopping place in a given offset. A user can enter an average speed in km/h that will be used if an edge weight evaluation will be conducted based on its travel time. Otherwise, if an edge weight is evaluated by its distance, the minimum and maximum distances in kilometres will be taken into account. For instance, if the edge weight configuration is set to 'travel time' and the average speed value is set to '60 km/h', then a minimum configured travel time of one hour to a road stopping place will be equal to a distance of 60 km. Furthermore, the configured maximum travel time to a road stopping place of three hours will be equal to a 180-km air-line distance.

The buttons at the very bottom of the options panel are used to abort or save a customised setup. Thus, the 'abort' button will decline a configuration so that the previous configuration remains active, and the 'save' button will store the current setup for use in the route planner.

5.6 Application Programming Interfaces

For the development of the route planner in this thesis, two Java APIs that enabled essential functionalities were used, namely OWL Java API[7] and GeoTools.[8] The OWL Java API provided a library for the data exchange with ontologies, and GeoTools made the following functionalities available: creating a graph of line strings and multi-line strings, applying Dijkstra's shortest path algorithm with a customisable edge weight function, plotting and selecting map features, and navigating through map content (e.g. scroll, zoom in and zoom out). The APIs are described here as part of the arrangements, as certain terminology is mandatory to follow up with the implementation in later parts of this thesis.

[7] `http://owlcs.github.io/owlapi/`
[8] `http://geotools.org`

5.6.1 OWL Java API

To read ontologies with OWL Java API, it is necessary to process data individuals and their properties and annotations separately. Once an ontology file is loaded, an iterator is required to process one data individual after the other. At every iteration, a data individual needs to be analysed in detail, meaning that data property assertion axioms, object property axioms and annotation assertion axioms will be extracted. The axioms can be interpreted as datasets that typically contain multiple data entries. Each axiom will be explained in the next list:

- *Data property assertion axioms* contain the metadata of an individual, such as the object identifier, LG number, LG name, start SLK, end SLK, road, carriageway, start node number, end node number, and common usage name for MRWA road sections.

- *Object property assertion axioms* relate to the properties and sub-properties of individuals. For instance, the object property 'geo:hasGeometry' retrieves the location individual of a road section individual.

- *Annotation assertion axioms* can contain the unknown information of an individual. This can occur if the data and object properties are not declared in an ontology, whereby they will remain as annotation assertions. The difference between data properties, object properties and annotation assertions is that object properties and data properties can be queried with semantic SWRL rules, while annotation assertions are not available with SWRL. The reason for this is that the SWRL composer in Protégé[9] verifies semantic rules for correctness, and annotation assertions are not part of a designed ontology.

An example of a line string and data property extraction from a road section individual is shown in Algorithm 5.1. The example mirrors the description

[9] `http://protege.stanford.edu`: a free and open source editor to design ontologies.

above, except the evaluation of annotation assertions is not implemented, as the required data properties 'geo:hasGeometry' and 'geo:asWKT' will be declared appropriately in the ontology.

Algorithm 5.1: Example of a line string and data property extraction from a road section individual.

```
1  O ⟵ load the ontology
2  foreach individual in O do
3      if individual equals RoadSection then
4          OP ⟵ get object properties
5          DP ⟵ get data properties
6          foreach property in OP do
7              if property equals geo : hasGeometry then
8                  I ⟵ get individual
9                  DPI ⟵ get data properties of I
10                 foreach dataproperty in DPI do
11                     if dataproperty equals geo : asWKT then
12                         LS ⟵ get line string
13                     end
14                 end
15             end
16         end
17         foreach DataProperty in DP do
18             P ⟵ get property (e.g. road name and object identifier)
19         end
20     end
21 end
```

5.6.2 GeoTools Java GIS Toolkit

The implementation of the GeoTools Java GIS Toolkit is an extensive process with real road network data. Thus, the explanation here in the arrangement will cover the elementary GeoTools procedures to communicate very basic API handling with simple examples.

Creating a graph with GeoTools is a straightforward process. Once a line string generator (variable: 'lsgg') has been initialised, line strings can be added to it. After that, the function call 'lsgg.getGraph()' will deliver a non-directed

graph object that contains nodes and edges. The following Source Code 5.1 shows the creation of a graph that consists of two line strings, as indicated in Figure 5.6, with a tolerance[10] of 0.00001 in longitude and latitude:

```
1 LineStringGraphGenerator lsgg = new LineStringGraphGenerator(0.00001);
2 GeometryFactory gf = JTSFactoryFinder.getGeometryFactory(null);
3 WKTReader reader = new WKTReader(gf);
4 LineString ls1 = reader.read("115.1 -31.6, 120.0 -31.0");
5 LineString ls2 = reader.read("115.1 -31.6, 110.0 -32.0");
6 lsgg.add(ls1);
7 lsgg.add(ls2);
8 Graph g = lsgg.getGraph();
```

Source Code 5.1: Example of creating a graph that contains two line strings.

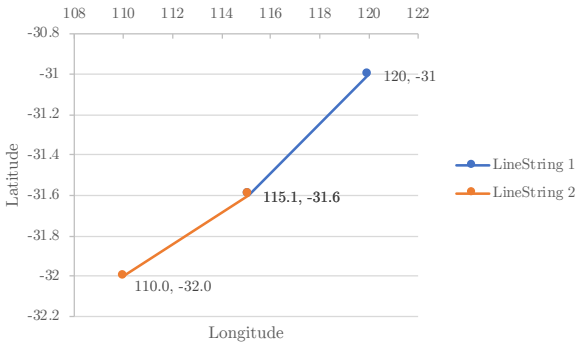


Figure 5.6: Sample graph that shows two connected line strings.

The implementation of Dijkstra's algorithm with GeoTools requires preparing a graph, a start/end node, and an edge weighter object. Source Code 5.2 shows a simple example to calculate the shortest route from a start node to an end node with a default edge distance weighter, assuming that both the start and end node have been selected. It can be seen that at line 8, the path will be calculated to every other node in the graph from the start node and, at line 9, the shortest route to a given end node will be retrieved. A more efficient approach could

[10] If the start or end of a line string is near another line string but not connected, then an activated tolerance will clean datasets so that a new vertex will be created and the line strings can be connected at a node.

include aborting a path calculation once the shortest path to a given end node has been found, but that functionality is not supported in GeoTools.

```
1 DijkstraIterator.EdgeWeighter weighter = new DijkstraIterator.EdgeWeighter() {
2   public double getWeight(Edge e) {
3     LineString ls = (LineString) e.getObject();
4     return ls.getLength(); } }
5 Node startNode = <selected start node from graph> // node selection hidden
6 Node endNode = <selected end node from graph> // node selection hidden
7 DijkstraShortestPathFinder pf = new DijkstraShortestPathFinder(graph, startNode,
      weighter);
8 pf.calculate();
9 Path route = pf.getPath(endNode);
```

Source Code 5.2: The example shows the processing of Dijkstra's shortest path algorithm by evaluating an edge length. The node selection has been hidden to simplify the view to focus on the path generation.

With GeoTools, plotting features on a map requires multiple procedures as indicated for the road section dataset in Algorithm 5.2. A feature type needs to be created with the function 'createFeatureType' to initialise the road edge features. Then, the road section line strings will be processed with a graph generator. After that, a simple feature builder will be initialised with the previously created road edge type. Now, a feature collection can be created with the function 'EdgeBuilder'. In the end, a map object will be created, and the feature collection will be added as a layer. The same principle can be applied to display the remaining datasets 'road nodes', 'power lines', 'traffic signal sites', 'road stopping places', 'rail crossings', 'give way signs' and 'stop signs'.

Algorithm 5.2: Principles to define feature types, create features, create layers and add content to a map with the Java OWL API.

```
1  Function createFeatureType(Type):
2      Builder ⟵ initialise simple feature type builder object
3      Builder ⟵ set attributes (e.g. name, namespace and CRS)
4      Builder ⟵ add GeoTools metadata (e.g. line string, name and
         number)
5      if Type equals RoadSection then
6          Builder ⟵ add type specific metadata, such as for MRWA road
             sections, namely: start SLK, end SLK, start node number, end
             node number, carriageway, common usage name, object
             identifier, town name, town number, road name, road, speed
             limit, road stopping place, well-known-text, has geometry,
             distance, and travel time
7      end
8      return feature type retrieved from Builder
9  End Function
10 Function EdgeBuilder(Graph, FeatureBuilder, EdgeType):
11     Result ⟵ new feature type collection
12     Edges ⟵ get edges from Graph
13     foreach Edge in Edges do
14         LineString ⟵ get line string of Edge
15         RoadEdge ⟵ get corresponding road edge object from Edge
16         FeatureBuilder ⟵ add LineString
17         FeatureBuilder ⟵ set GeoTools metadata 'name' and 'number'
18         if EdgeType equals RoadSection then
19             FeatureBuilder ⟵ set metadata retrieved from RoadEdge
20         end
21         Result ⟵ add feature created with FeatureBuilder
22     end
23     return Result
24 End Function
25 SimpleFeatureLineStringType ⟵ createFeatureType(RoadEdge)
26 LineStringGraphGenerator ⟵ process road section line strings
27 Graph ⟵ get Graph of LineStringGraphGenerator
28 SimpleFeaturesBuilder ⟵ setup line strings features
29 FeatureCollection ⟵ EdgeBuilder (Graph, SimpleFeaturesBuilder)
30 Map ⟵ create a new map object
31 RoadSectionLayer ⟵ create layer with FeatureCollection data
32 Map ⟵ add RoadSectionLayer to the map
```

5.7 Chapter Summary

In this chapter, key features were defined to describe the range of functionalities of the road network conflation, the road network translation and the route planning approaches. Regarding the road network conflation approach, significant newly created Semantic Web rules that can be used to enable machines to understand heterogeneous road asset datasets that mean the same thing were defined. These rules also included defined rules for the determination of the trust score of road asset data sources.

Considering the road network translation approach, the significant contributions of the author included the definitions of the newly designed translation methods. The translation methods can be activated for the comparison of road network centreline data that mean the same thing from different data sources. In the context of this thesis, the author will employ the translation methods for the comparison of the MRWA and Landgate road network datasets.

In the scope of the route planning approach, the user interfaces of the newly developed route planner were explained with mock-ups to define the functionalities of the route planner environment. Furthermore, the basic concepts of the employed OWL and GeoTools APIs were explained as part of the arrangements to convey the required background information before the APIs' use in the next chapter.

Chapter 6

Implementation

6.1 Chapter Introduction

In this chapter, the implementation of the methodology will be explained. The major parts of this chapter are about the data migration from GeoJson road asset datasets to the Turtle ontology format (Objective 3), the road network translation from MRWA road sections and intersections to the shape of the Landgate road network data (Objective 5) and the development of the route planning approach based on Semantic Web technologies (Objective 6). Flowcharts will be shown for a better understanding of each major part, source code will be indicated to follow up with important methods, and algorithms will be used to simplify complex logic approaches.

6.2 Data Creation

The approach of creating ontology individuals extracted from GeoJSON data files will be discussed in this section. Flowcharts will be used for the explanation of the data individuals creation. The creation of related geographic features will be explained with algorithms. At appropriate positions, the algorithms will refer to

elementary functions that will be explained with source code, e.g. functions to write points, line strings, multi-line strings, entities and metadata.

6.2.1 Flowchart Data Creation

In the data creation approach, datasets in the GeoJSON file format will be processed and written in the Turtle ontology format. This section will explain the data creation process with flowcharts for each of the nine datasets created, whereby the approach will be explained with two graphs,[1] as the remaining seven graphs will be matched with hash tables.

The difference between the two flowcharts in this section is that the first flowchart can be used for original (not translated) datasets, which include Landgate road sections, OSM map lines, Western Power overhead power lines and MRWA regulatory signs, speed limits, traffic signal sites, rail crossings and road stopping places. Meanwhile, the second flowchart can be used for translated datasets, such as the MRWA road sections and intersections.

6.2.1.1 Original Data

The flowchart for the original datasets is shown in Figure 6.1, and the corresponding hash table is shown in Table 6.1. The hash values '<process>', '<input>', '<activity>' and '<write>' can be placed with the hash table in the appropriate positions in the flowchart. The following example will clarify the flowchart for the first data row of the hash table.

After the start, the process 'LandgateSlipData' will be activated, and the Landgate LGATE-012 dataset will be loaded as input from a local GeoJSON file. Then, the loaded input will be read, and a feature collection of entities can be created with the data. As long as a feature exists in the features collection, a loop will iterate through each feature. This means the metadata of a current feature will be read, and the activities 'read multi-line string', 'extract line string', 'ex-

[1] For the completeness of this work, all flowcharts will be attached in Appendix A.3.

Table 6.1: Hash values for the placement in the flowchart in Figure 6.1.

<process>	<input>	<activity>	<write>
LandgateSlipData	Landgate LGATE-012	read multi-line string, extract line string and points, and get asset type	multi-line string, line string and points
OsmMapLines	OSM map lines	read line string, extract points and get asset type	line string and points
MrwaRegulatorySigns	MRWA regulatory signs	read point and get sign type	point
MrwaSpeedLimits	MRWA speed limits	read point	point
MrwaTrafficSignalSites	MRWA traffic signal sites	read point	point
MrwaRoadStoppingPlaces	MRWA road stopping places	read point	point
MrwaRailCrossings	MRWA rail crossings	read point	point
WpOverheadPowerlines	Western Power WP-031	read multi-line string	multi-line string

tract points' and 'get asset type' will be used to prepare the ontology individuals. A road asset individual will then be written in the Turtle file format, as will the related multi-line string, line string and point entities of the current road asset. If a next feature exists, then the procedure will be repeated. Otherwise, the loop can be left, and the processing of the flowchart will stop.

The process of creating individuals for the datasets of Landgate (LGATE-012), MRWA (regulatory signs, speed limits, traffic signal sites, rail crossings and road stopping places), Western Power (WP-031) and OSM (map lines) is indicated in Algorithm 6.1. The algorithm mirrors the above-mentioned flowchart and refers to the key functions 'writePoint', 'writeLineString', 'writeMultiLineString' and 'writeEntityContent' in order to create data and provenance in the Turtle ontology format.

The remaining input entries from the OSM map lines, Western Power WP-031, and MRWA regulatory signs, speed limits, traffic signal sites, rail crossings and road stopping places of the hash table can be interpreted in the same way described above for the Langate LGATE-012 dataset.

Algorithm 6.1: Create road asset individuals from GeoJSON datasets.

```
1  FeaturesCollection ⟵ load GeoJSON dataset
2  foreach Entity in FeaturesCollection do
3      RoadName ⟵ get road name of Entity
4      Identifier ⟵ get identifier of Entity
5      Individual ⟵ compound 'Road Asset' + RoadName + Identifier
6      Metadata ⟵ get metadata of Entity
7      if process equals LandgateSlipData then
8          MultiLineString ⟵ read multi-line string of Entity
9          LineString ⟵ extract line string of MultiLineString
10         Points ⟵ extract points of LineString
11         RoadAsset ⟵ get asset type of Entity
12         foreach Point in Points do
13             writePoint ⟶ see Section 6.2.2.1
14         end
15         writeLineString ⟶ see Section 6.2.2.2
16         writeMultiLineString ⟶ see Section 6.2.2.3
17     end
18     if process equals OsmMapLines then
19         LineString ⟵ read line string of Entity
20         Points ⟵ extract points of LineString
21         RoadAsset ⟵ get asset type of Entity
22         foreach Point in Points do
23             writePoint ⟶ see Section 6.2.2.1
24         end
25         writeLineString ⟶ see Section 6.2.2.2
26     end
27     if process equals MrwaRegulatorySigns then
28         RoadAsset ⟵ get sign type of Entity
29     end
30     if process equals MrwaRegulatorySigns or MrwaSpeedLimits or
         MrwaTrafficSignalSites or MrwaRailCrossings or
         MrwaRoadStoppingPlaces then
31         Point ⟵ read point of Entity
32         writePoint ⟶ see Section 6.2.2.1
33     end
34     if process equals WpOverheadPowerlines then
35         MultiLineString ⟵ read multi-line string of Entity
36         writeMultiLineString ⟶ see Section 6.2.2.3
37     end
38     writeEntityContent ⟶ see Section 6.2.2.4
39 end
```

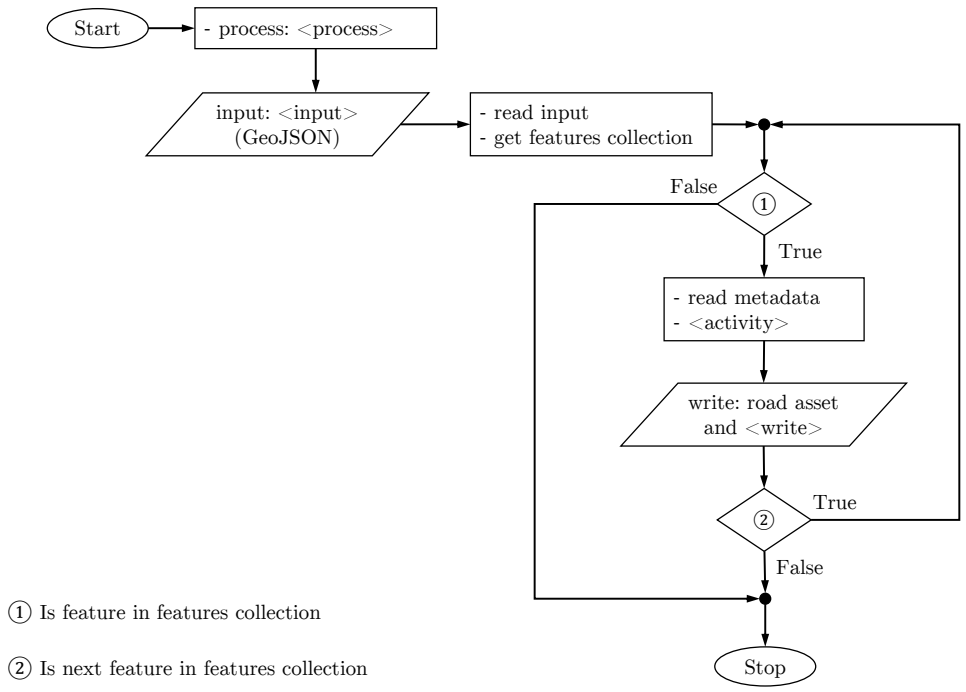


Figure 6.1: Flowchart to write road asset individuals and their related features in the RDF/Turtle format generated from GeoJSON datasets.

6.2.1.2 Translated Data

The flowchart for the MRWA intersections and MRWA road network datasets for both original features and translated features is indicated in Figure 6.2; the related hash table is shown in Table 6.2. With the hash table, the hash values '<process>', '<input>', '<activity>', '<write translation>', '<write original>' and '<write object>' are addressed in the graph. The following explanation will use MRWA intersections as an example to explain the processing of the translated data flowchart.

At the beginning of the flowchart, the process 'MrwaIntersection' will be used to trigger the flowchart activities after the start. The input file 'MRWA intersection' will be loaded in the GeoJSON file format. The input data will then be read and a feature collection will be created. After that, a loop will iterate through each feature in the collection. At the start of each iteration, each entity's metadata and point will be read.

Table 6.2: Hash values for the placement in the flowchart in Figure 6.2.

<process>	<input>	<activity>	<write translation>	<write original>	<write object>
MrwaIntersection	MRWA intersections	read point	point (original + translated)	point	intersection
MrwaRoadNetwork	MRWA road network	read line string and extract points	points (original + translated) and line string (original + translated)	line string and points	region, town and road section

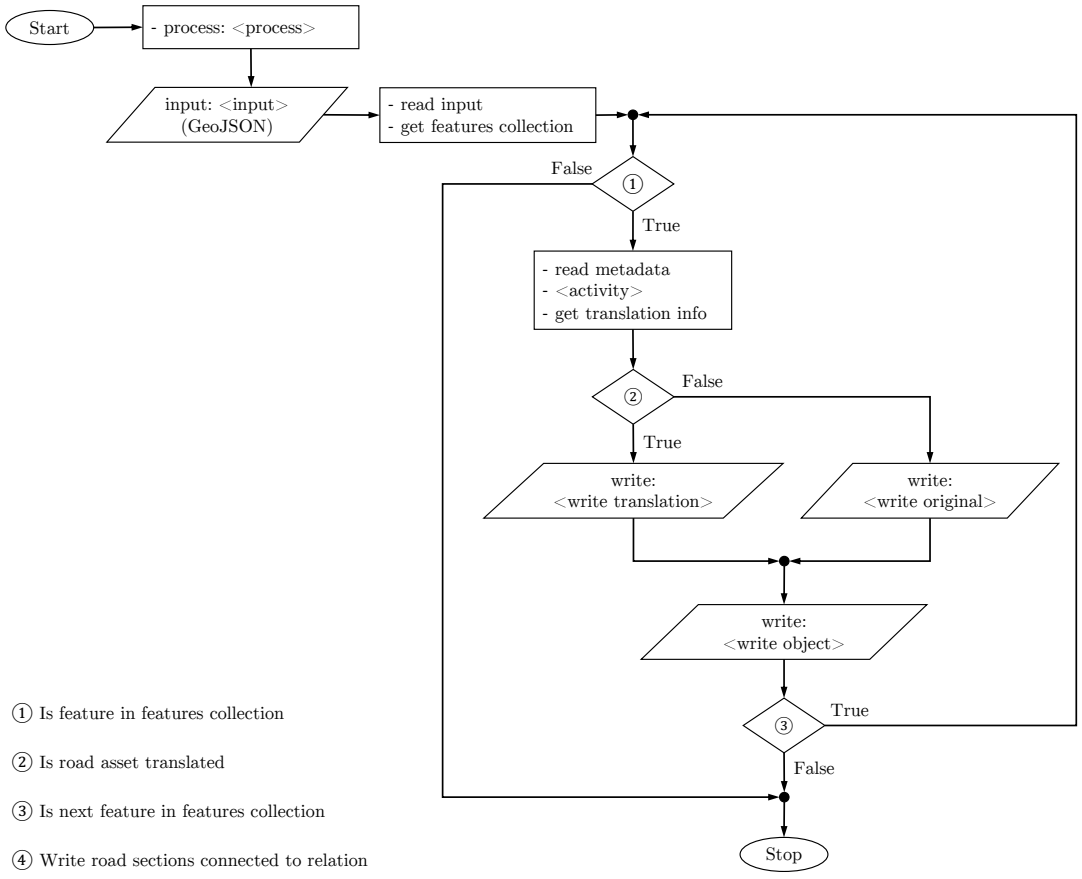


Figure 6.2: Flowchart to write original and translated MRWA ontology individuals generated from GeoJSON datasets.

After retrieving the translation info, it can be determined whether an entity location has been translated or not. If a translation has been processed, then the original point and the translated point will be written in the Turtle ontology format. Otherwise, it is sufficient to write the point into the ontology without adding the information 'original' or 'translated' for the identification of a translated road

asset, as the ontology model understands the original locations by default. The intersection individual and the relation to a connected road section will be written into the ontology after the end of condition statement 3. If the next feature exists, then the procedure will start again as indicated at the flowchart legend point 2. If there are no remaining features in the collection, then the flowchart finishes at the stop.

Algorithm 6.2 supports the explanation of the flowchart for the MRWA intersections and road network datasets. The algorithm mirrors the described flowchart functionality and refers to the functions that create data and provenance in the required Turtle ontology format, such as 'writePoint', 'writeLineString', 'writeMultiLineString' and 'writeEntityContent'.

Algorithm 6.2: Create road asset individuals from original and translated GeoJSON datasets.

```
1  FeaturesCollection ⟵ load GeoJSON dataset
2  foreach Entity in FeaturesCollection do
3      RoadName ⟵ get road name of Entity
4      Identifier ⟵ get identifier of Entity
5      Individual ⟵ compound 'Road Asset' + RoadName + Identifier
6      Metadata ⟵ get metadata of Entity
7      Translation ⟵ get translation info from Entity
8      if process equals MrwaRoadNetwork then
9          LineString ⟵ read line string of Entity
10         Points ⟵ extract points of LineString
11         if Entity is translated then
12             foreach Point in Points do
13                 writePoint (original) ⟶ see Section 6.2.2.1
14                 writePoint (translated) ⟶ see Section 6.2.2.1
15             end
16             writeLineString (original) ⟶ see Section 6.2.2.2
17             writeLineString (translated) ⟶ see Section 6.2.2.2
18         else
19             foreach Point in Points do
20                 writePoint ⟶ see Section 6.2.2.1
21             end
22             writeLineString ⟶ see Section 6.2.2.2
23         end
24         writeEntityContent ⟶ see Section 6.2.2.4
25     end
26     if process equals MrwaIntersection then
27         Point ⟵ read points of Entity
28         if Entity is translated then
29             writePoint (original) ⟶ see Section 6.2.2.1
30             writePoint (translated) ⟶ see Section 6.2.2.1
31         else
32             writePoint ⟶ see Section 6.2.2.1
33         end
34         writeEntityContent ⟶ see Section 6.2.2.4
35     end
36 end
```

6.2.2 GeoJSON to Turtle Functions

A Python script that reads GeoJSON files and writes the contained data with provenance in the Turtle file format has been developed. The script enables the automatic creation of ontology individuals extracted from datasets in the GeoJSON format. This section will explain the key functionalities that are used to automate the ontology creation process.

6.2.2.1 Write Point

The function 'writePoint' will be used to write a point with its features into the ontology, as shown in Source Code 6.1. The parameters 'prefix', 'longitude', 'latitude', 'crs' and 'entity' are the minimum requirements to successfully compound a point individual in the RDF format. The parameter 'prefix' will be used to differ between the MRWA, Landgate, OSM and Western Power ontologies. The parameters 'longitude', 'latitude' and 'crs' will be used for the localisation of an entity with its coordinate reference system. The parameter 'entity' will be used for the identification of a point individual; an example of a point individual is 'mrwa:PointCoordinates_Translated_115.71235101_-31.68378145'. Optional parameters can be set for data provenance in Source Code 6.1, such as 'generatedBy', 'primarySource', 'translation' and 'usedPoint', as explained next.

At the 'writePoint' function start, metadata and properties will be written into the ontology. If the variables 'generatedBy' and 'primarySource' are not set to a default value, then the 'prov:wasGeneratedBy' and 'prov:hadPrimarySource' attributes will be written into the ontology before the individual definition at line 16 is completed. If the variables 'usedPoint', 'generatedBy' and 'translation' are set to a value that is not equal to 'default', then the provenance information 'prov:used', 'prov:generated' and 'prov:wasUsedBy' can be written into the ontology. This means that without changing the default parameters of the variables 'generatedBy', 'primarySource', 'usedPoint' and 'translation', the function can be used without writing provenance into an ontology.

```
1  def writePoint(prefix, longitude, latitude, entity, crs, generatedBy='default',
       primarySource='default', usedPoint='default', translation='default'):
2    # write properties
3    Ontology.write(entity + '␣rdf:type␣owl:NamedIndividual;\n')
4    Ontology.write('␣a␣sf:Point,␣geo:Point,␣prov:Entity,␣prov:Location;\n')
5    Ontology.write('␣geo:lat␣' + str(latitude) + ';\n')
6    Ontology.write('␣geo:long␣' + str(longitude) + ';\n')
7    Ontology.write('␣geosparql:asWKT␣"Point(' + str(longitude) + '␣' + str(latitude)
         + ')"^^geosparql:wktLiteral;\n')
8    Ontology.write('␣' + prefix + 'LATITUDE␣' + str(latitude) + ';\n')
9    Ontology.write('␣' + prefix + 'LONGITUDE␣' + str(longitude) + ';\n')
10   Ontology.write('␣' + prefix + 'CRS␣' + str(crs) + ';\n')
11   # write provenance
12   if generatedBy != 'default':
13     Ontology.write('␣prov:wasGeneratedBy␣alg:' + generatedBy + ';\n')
14   if primarySource != 'default':
15     Ontology.write('␣prov:hadPrimarySource␣' + primarySource + ';\n')
16   Ontology.write('␣' + prefix + 'coordinates␣' + '"[' + str(longitude) + ',␣' +
       str(latitude) + ']".\n')
17   # add point related provenance individuals
18   if usedPoint != 'default':
19     Ontology.write('alg:' + generatedBy + '␣prov:used␣mrwa:' + usedPoint + '.\n')
20   if generatedBy != 'default':
21     Ontology.write('alg:' + generatedBy + '␣prov:generated␣' + entity + '.\n')
22   if translation != 'default':
23     Ontology.write(entity + '␣prov:wasUsedBy␣alg:' + translation + '.\n')
```

Source Code 6.1: Function to write a point in the Turtle format.

6.2.2.2 Write Line String

The function to write a line string in the Turtle format is indicated in Source Code 6.2. The approach requires the parameters ‘prefix’, ‘lineString’, ‘crs’ and ‘Ontology’. The parameters ‘generatedBy’ and ‘primarySource’ can be used to add provenance to a line string individual.

Brackets, whitespaces and commas will be removed from the line string at the function start and the results will be saved in the variable ‘identifier’. The entity will then be compounded with the variables ‘prefix’ and ‘identifier’. After that, the individual attributes ‘rdf:type’, ‘owl:NamedIndividual’, ‘sf:LineString’, ‘prov:Entity’ and ‘prov:Location’ will be written into the ontology, followed by

```
def writeLineString(prefix, lineString, crs, Ontology, generatedBy = 'default',
    primarySource = 'default'):
  identifier = str(re.sub('[(,)]', '', lineString)).replace(' ', '_')
  identifier = identifier.translate(table)
  entity = prefix + identifier
  firstLine = entity + ' rdf:type owl:NamedIndividual;\n'
  Ontology.write(firstLine)
  # each individual is a line string, an entity and a location
  Ontology.write(' a sf:LineString, prov:Entity, prov:Location;\n')
  # each individual is a GeoSPARQL well-known text line string
  Ontology.write(' geosparql:asWKT "' + lineString + '"^^geosparql:wktLiteral;\n')
  # write the 'prov:wasGeneratedBy' provenance
  if generatedBy != 'default':
    Ontology.write(' prov:wasGeneratedBy alg:' + generatedBy + ';\n')
  # write the 'prov:hadPrimarySource' provenance
  if primarySource != 'default':
    Ontology.write(' prov:hadPrimarySource ' + primarySource + ';\n')
  Ontology.write(' ' + prefix + 'CRS ' + str(crs) + '.\n')
  # write that the process 'prov:generated' the entity
  if generatedBy != 'default':
    Ontology.write('alg:' + generatedBy + ' prov:generated ' + entity + '.\n')
  return identifier
```

Source Code 6.2: Function to write a line string in the Turtle format.

the ‘geosparql:asWKT’ line string definition. If the variables ‘generatedBy’ and ‘primarySource’ were set as parameters, then the corresponding provenance can be set. After that, the coordinate reference system will be added, and the line string individual definition can be completed.

Furthermore, if the parameter ‘generatedBy’ has not been set to a default value, then the information will be written into the ontology file that the current process created for the ontology entity.

6.2.2.3 Write Multi-Line String

The function ‘writeMultiLineString’ will be used to write a multi-line string into the ontology (see Source Code 6.3). The function uses the parameters ‘prefix’, ‘multiLineString’, ‘crs’ and ‘Ontology’ as the requirements for processing a multi-

line string individual. The parameters 'generatedBy' and 'primarySource' can be used to add provenance.

```
def writeMultiLineString(prefix, multiLineString, crs, Ontology, generatedBy =
    'default', primarySource = 'default'):
  identifier = str(re.sub(' [(,) ]', '', multiLineString)).replace('␣', '_')
  identifier = identifier.translate(table)
  entity = prefix + identifier
  firstLine = entity + '␣rdf:type␣owl:NamedIndividual;\n'
  Ontology.write(firstLine)
  # each individual is a multi-line string, an entity and a location
  Ontology.write('␣a␣sf:MultiLineString,␣prov:Entity,␣prov:Location;\n')
  # each individual is a GeoSPARQL well-known text multi-line string
  Ontology.write('␣geosparql:asWKT␣"' + multiLineString + '"^^geosparql:
      wktLiteral;\n')
  # write the 'prov:wasGeneratedBy' provenance
  if generatedBy != 'default':
    Ontology.write('␣prov:wasGeneratedBy␣alg:' + generatedBy + ';\n')
  # write the 'prov:hadPrimarySource' provenance
  if primarySource != 'default':
    Ontology.write('␣prov:hadPrimarySource␣' + primarySource + ';\n')
  Ontology.write('␣' + prefix + 'CRS␣' + str(crs) + '.\n')
  # write that the process 'prov:generated' the entity
  if generatedBy != 'default':
    Ontology.write('alg:' + generatedBy + '␣prov:generated␣' + entity + '.\n')
  return identifier
```

Source Code 6.3: Function to write a multi-line string in the Turtle format.

At the beginning of the function, brackets, commas and whitespaces will be removed from the multi-line string, and the results will be saved in the variable 'identifier'. Then, the entity will be compounded with the 'prefix' and 'identifier' variables. After that, the individual attributes 'rdf:type', 'owl:NamedIndividual', 'sf:MultiLineString', 'prov:Entity' and 'prov:Location' will be written into the ontology, followed by the 'geosparql:asWKT' definition for a multi-line string.

If the variables 'generatedBy' and 'primarySource' are set at the function call, then the provenance 'prov:wasGeneratedBy' and 'prov:hadPrimarySource' can be written. After that, the multi-line string individual definition can be completed by adding the information of the coordinate reference system. Furthermore, if

the parameter 'generatedBy' has been set, then the 'prov:generatedBy' attribute will used to write that a current process created an ontology entity.

6.2.2.4 Write Entity Content

The function 'writeEntityContent' will be used to write a data individual in the Turtle ontology format (see Source Code 6.4). The function uses the parameters 'prefix', 'entry', 'metadata', 'geometryID', 'generatedBy', 'Ontology', 'agent' and 'dataset'. The parameter 'thisClass' can also be used to write the class of an individual, such as 'mrwa:SpeedLimit'.

```
1  def writeEntityContent(prefix, entry, metadata, geometryID, Ontology, agent,
       dataset, generatedBy = 'default', thisClass = 'default'):
2    # write type
3    Ontology.write(prefix + entry +'␣rdf:type␣owl:NamedIndividual;\n')
4    Ontology.write('␣a␣prov:Entity,␣')
5    if thisClass != 'default':
6      Ontology.write(thisClass + ',␣')
7    Ontology.write('owl:Thing;\n')
8    # write provenance
9    Ontology.write('␣prov:wasGeneratedBy␣alg:' + generatedBy + ';\n')
10   Ontology.write('␣prov:atLocation␣' + prefix + geometryID + ';\n')
11   Ontology.write('␣prov:wasAttributedTo␣' + prefix + agent + ';\n')
12   # add the GeoSPARQL geometry attribute
13   Ontology.write('␣geosparql:hasGeometry␣' + prefix + geometryID + ';\n')
14   # the function 'writeMetadata' will be explained in Source Code 6.5
15   writeMetadata(Ontology=Ontology, metadata=metadata, prefix=prefix)
16   Ontology.write('␣prov:hadPrimarySource␣' + dataset + '.\n')
17   Ontology.write(prefix + agent + '␣prov:contributed␣' + prefix + entry + '.\n')
18   Ontology.write(dataset + '␣prov:wasPrimarySourceOf␣' + prefix + entry + '.\n')
19   Ontology.write('alg:' + generatedBy + '␣prov:generated␣' + prefix+entry+'.\n')
```

Source Code 6.4: Function to write an entity in the Turtle format.

The processing starts by writing the entity types 'owl:NamedIndividual', 'prov:Entity' and 'owl:Thing' into the ontology. If the variable 'thisClass' has been set as a parameter, then the entity class definition will be added to an individual. After that, the data provenance and the geometry features will be written. The function 'writeMetadata' will be processed to include related metadata in

the Turtle format. After that, the provenance 'prov:generated', 'prov:contributed' and 'prov:wasPrimarySourceOf' will be included in the entity definition.

6.2.2.5 Write Metadata

The function 'writeMetadata' in Source Code 6.5 is used to write the metadata of an individual. For the processing, the function uses the parameters 'prefix', 'metadata' and 'Ontology' and distinguishes between various data types, such as integers, floating point numbers, strings, lists and dictionaries.

```
1  def writeMetadata(prefix, metadata, Ontology):
2    writtenEntry = False
3    error = False
4    # iterate metadata entries
5    for entry in metadata.properties:
6      if writtenEntry and not error:
7        Ontology.write(';\n')
8      value = metadata.properties[entry]
9      # distinguish between int, float, list and dict (otherwise skip)
10     if (type(value) is int) or (type(value) is float):
11       Ontology.write('␣' + prefix + entry + '␣' + str(value))
12     elif type(value) is str:
13       value = value.replace('"', "'")
14       Ontology.write('␣' + prefix + entry + '␣"' + str(value) + '"')
15       if (prefix == "osm:") and (entry == 'other_tags'):
16         writeOtherTags(value)
17     elif type(value) is list:
18       jsonValue = json.dumps(value)
19       jsonValue = jsonValue.replace('"', "'")
20       Ontology.write('␣' + prefix + entry + '␣"' + str(jsonValue) + '"')
21     elif type(value) is dict:
22       jsonValue = json.dumps(value)
23       jsonValue = jsonValue.replace('"', "'")
24       Ontology.write('␣' + prefix + entry + '␣"[' + str(jsonValue) + ']"')
25     else:
26       error = True
27       continue
28     error = False
29     writtenEntry = True
```

Source Code 6.5: Function to write entity metadata in the Turtle format.

The differentiation is required, as the different data types require a specific format in the ontology. For instance, numbers are written as digits, strings and lists are enclosed in quotation marks, and dictionaries are enclosed in quotation marks and square brackets. A loop is used to iterate through each entry of the metadata collection. If an undefined metadata type will be processed, then the value will be skipped to prevent a syntax error in the ontology.

6.3 Road Network Translation

This section will explain the implementation of the translation methods described earlier in Section 5.3 and their offsets, which are used to translate MRWA road sections and MRWA intersections to the shape of the Landgate road network. Methods 1–7 will then be processed in ascending order. The executing order is mandatory, as each method is built on top of the previous results to achieve the best possible road network translation results. The flowchart indicated in Figure 6.3 summarises the processing of the newly created translation methods.[2]

The overall process of completing a translation is briefly indicated by the flowchart in Figure 6.4. At the beginning of the flowchart MRWA and Landgate datasets are selected in the GeoJSON data format and processed into ontologies. Then, the ontologies are merged and saved in the JavaScript Object Notation for Linked Data (JSON-LD) data format to simplify the data handling by working with one ontology file instead of multiple files. Unfortunately, the JSON-LD data format does not support prefixes. Thus, a created Python script will be triggered to replace the URIs with the same prefixes used during the introduction of the Turtle format ontologies. Next, a created script will be executed to extract only the individuals of the edited JSON-LD ontology file. The extraction will be saved in an output file, i.e. the actual data input file for the translation script. The algorithm of the translation script applies the seven custom-created translation

[2] The methods expressed with algorithms can be retrieved in Appendix A.5 for the interested reader.

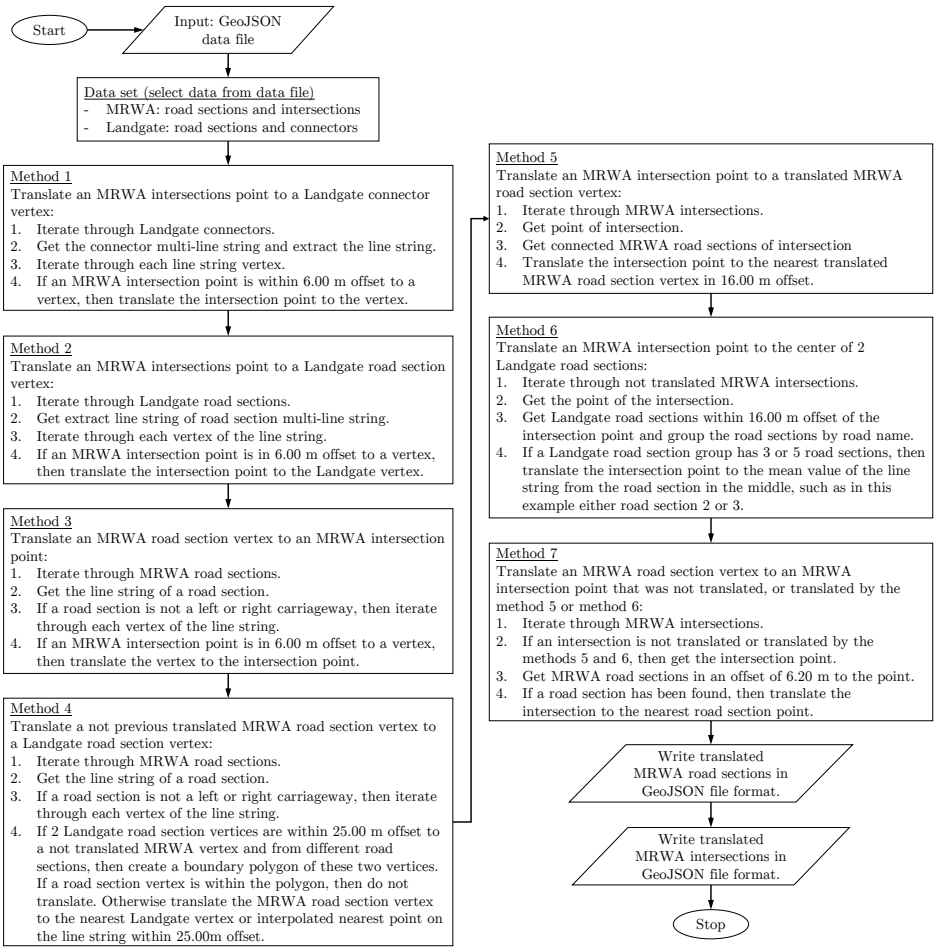


Figure 6.3: Flowchart that describes the translation of MRWA road sections and MRWA intersections to Landgate road sections and Landgate connectors.

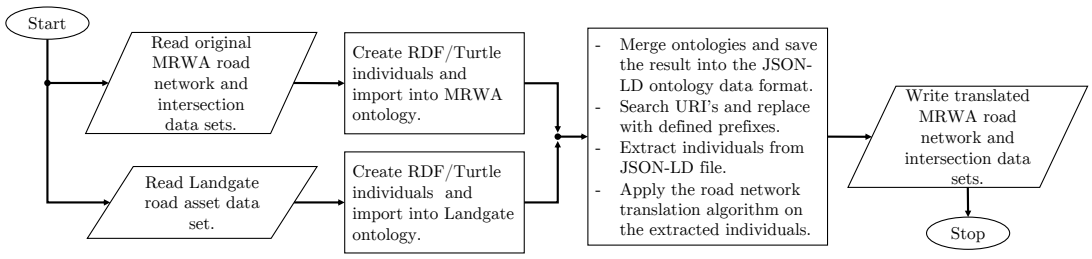


Figure 6.4: Flowchart that shows the processing cycle to translate MRWA road sections and intersections.

methods to the dataset and writes the translated MRWA intersections and road sections back in the original GeoJSON data format.

6.4 Route Planning

In this section, the newly developed route planner of this dissertation will be introduced with flowcharts and implemented. Several methods regarding the route planner will be presented in the route planner's functionalities. These methods include displaying data layers on the map, evaluating a planned route, writing a route into a route ontology, reading a route from a route ontology, handling currently closed roads and cleaning road network data by means of harmonising.

6.4.1 Flowchart Route Planning

The route planning approach to calculate a route between two or more nodes is explained with the flowchart in Figure 6.5. The consideration of specific constraints, such as power lines, turning circles, traffic signal sites, rail crossings and regulatory signs, is not part of the flowchart. The flowchart explains the following route planner setup:

- selected road stopping places
- a preloaded route
- a multi-node selection route
- a simple route from a start to a target

At the beginning of the flowchart and start of a new route plan, a list of current planned routes will be cleared so that one or more routes can be added depending on the route planner configuration. In the first condition statement, it will be determined whether road stopping places will be taken into account in the route planning. If that is true, then we will look further on the right-hand side of Condition Statement 1.

Before the processing of a route with road stopping places at Condition Statement 3 (for either a simple route between A and B or for a route selection that

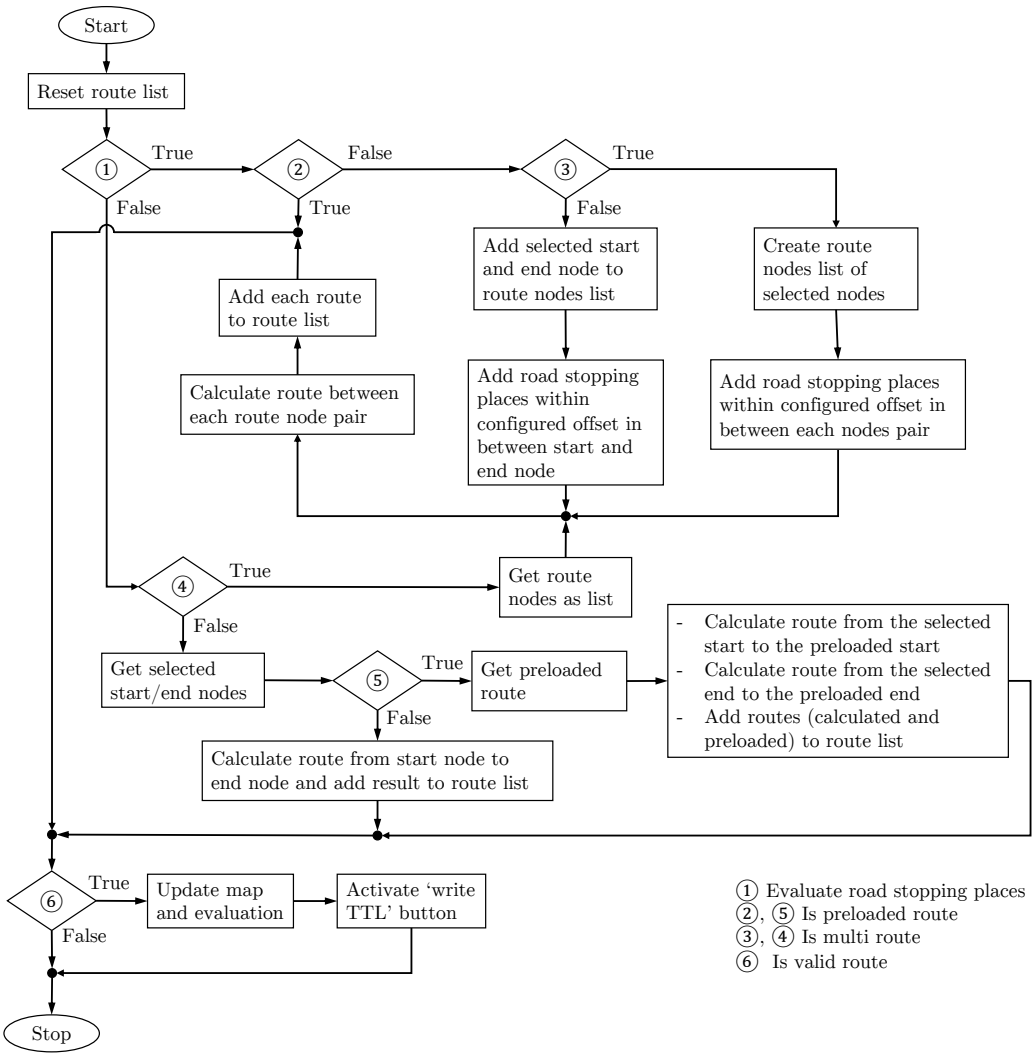


Figure 6.5: The flowchart describes the basic functionality of the route planner.

consists of more than two nodes), Condition Statement 2 will validate that no preloaded route is active. If a preloaded route is active, the calculation will be aborted, as the route is being calculated with a road network that contains only the road sections of that preloaded route; surrounding road stopping places cannot be considered since no additional data are available in the corresponding road network graph.

If the multi-node option is activated at Condition Statement 3, a list of selected nodes will be created. Then, road stopping places within a defined offset will be identified for each pair of nodes, e.g. between nodes 1 and 2, nodes 2

and 3 and so on. If the multi-node option is not activated, then road stopping places will be identified within a defined offset between a selected start node and an end node. For each identified road stopping place, two nodes will be added to the route nodes list which represents the start and end of a road section line string where a road stopping place is present (see Figure 6.6). After the nodes list has been updated, a route will be calculated for each node pair, and the routes will be added to the route list. If the route planning is successful, then a route evaluation will be processed, the information will be updated on the map, and the option to write the route into the ontology will be activated.

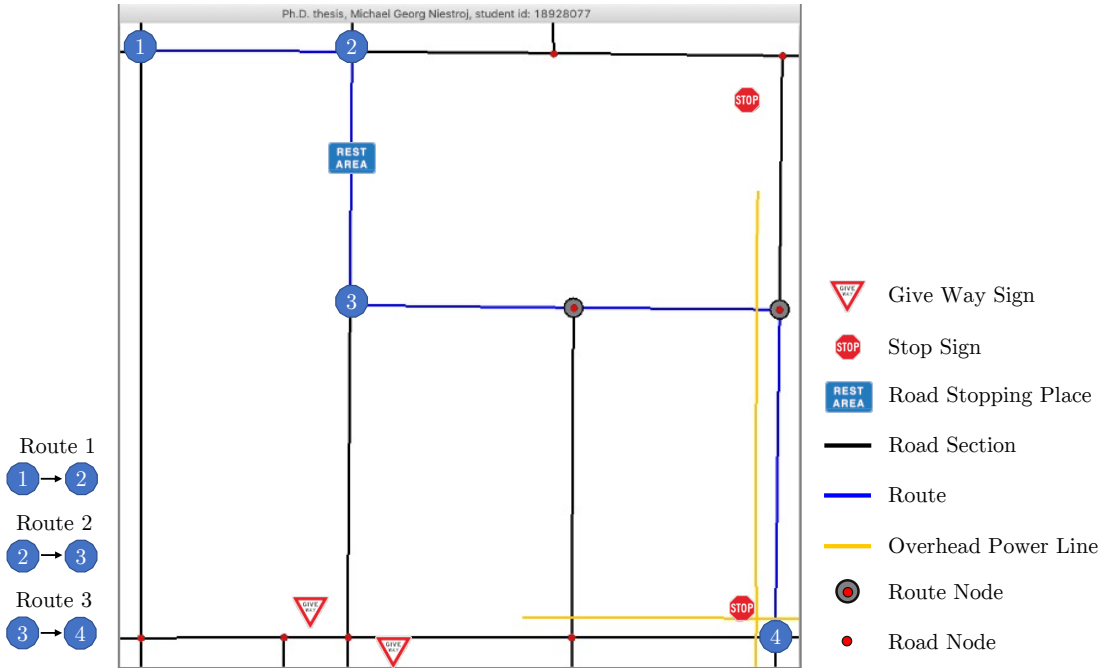


Figure 6.6: Processing of a route that considers road stopping places. The algorithm will merge sub-routes 1, 2 and 3 after the route planning has been completed for node pairs 1–2, 2–3 and 3–4.

If road stopping places are not considered in the route planner (see Condition Statement 1), then we will move on to Condition Statement 4 to determine whether a multi-node selection is active. If that is the case, then a route for each node pair will be calculated. If Condition Statement 4 is not true, meaning that

only a start and an end node have been selected for the route planning, then the two selected nodes will be retrieved.

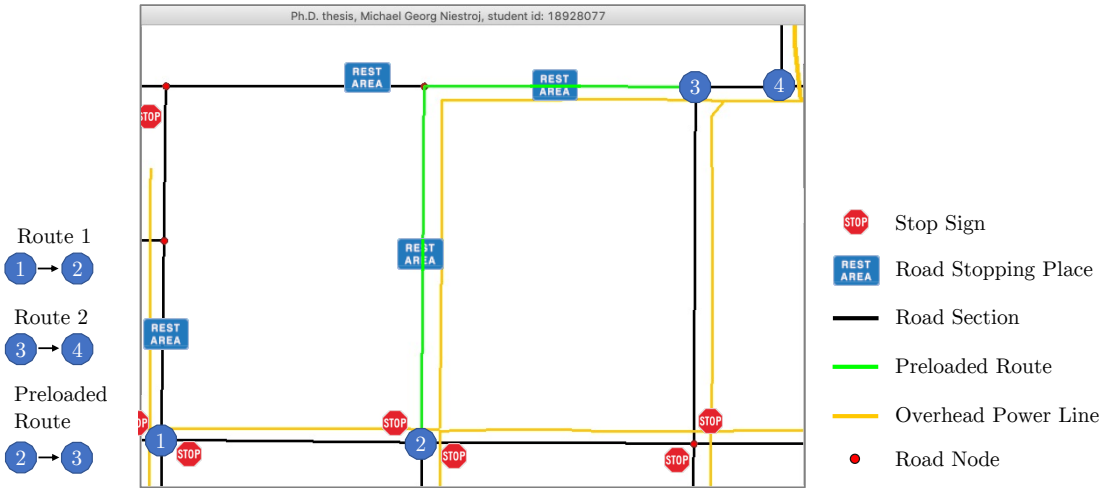


Figure 6.7: Processing of a route that includes a preloaded route. The algorithm will merge sub-routes 1 and 2 after the route planning has been completed for node pairs 1–2 and 3–4.

At Condition Statement 5, we will identified whether a preloaded route is active. If that is the case, then a route will be calculated from the selected start node to the beginning of a preloaded route as well as from the end of a preloaded route to the selected end node (see Figure 6.7). The route planning of a preloaded route will only consider a road network that contains preloaded road sections, as indicated with green line strings, to achieve the fastest possible preloaded route processing. After that, the calculated and preloaded routes will be added to the route list. Otherwise, if that is not the case, meaning that a preloaded route is not active, then a simple route will be calculated between a start and an end node, and the result will be added to the route list.

In any case, a valid route list will be evaluated, and the route planner will be updated with the results. Every time a valid route is processed, the ‘write TTL’ button will be activated so that a processed route can be written into the ontology. Finally, the flowchart stops after the processing.

6.4.2 Flowchart Edge Weight

This section explains the integration of the configurable route planner constraints with the help of a flowchart. 'Overhead power lines', 'turning circles', 'traffic signals', 'rail crossings' and 'regulatory signs' will be considered by the flowchart processing, as indicated in Figure 6.8, to simulate the identification of a road edge weight. At the beginning of the flowchart, a road edge individual and its metadata will be retrieved. A default edge weight unit will be defined; according to a user configuration, this is either a distance in metres or a travel time in seconds. At the first condition statement, it will be identified whether the configuration to evaluate crossing power lines is active. If that is the case and a road edge will be crossed by an overhead power line, the number of power lines multiplied by a configured factor will be added to the road edge weight.

For instance, if a road edge is crossed by two power lines and the configured factor is either 500 m or 40 s, according to the sample edge weight configuration, 2x500 m or 2x40 s will be added to the edge weight. After that, at Condition Statement 2, it will be identified whether a rail crossing occurs at a road section. If that is the case, then the number of rail crossings at the current road section will be multiplied by a configured factor, and the result will be added to the respective edge weight. The principle here works in the same manner as previously described for overhead power lines. Now, the retrieved 'distance', 'travel time', 'count of rail crossings' and 'count of power lines' will be saved to be available for the route evaluation.

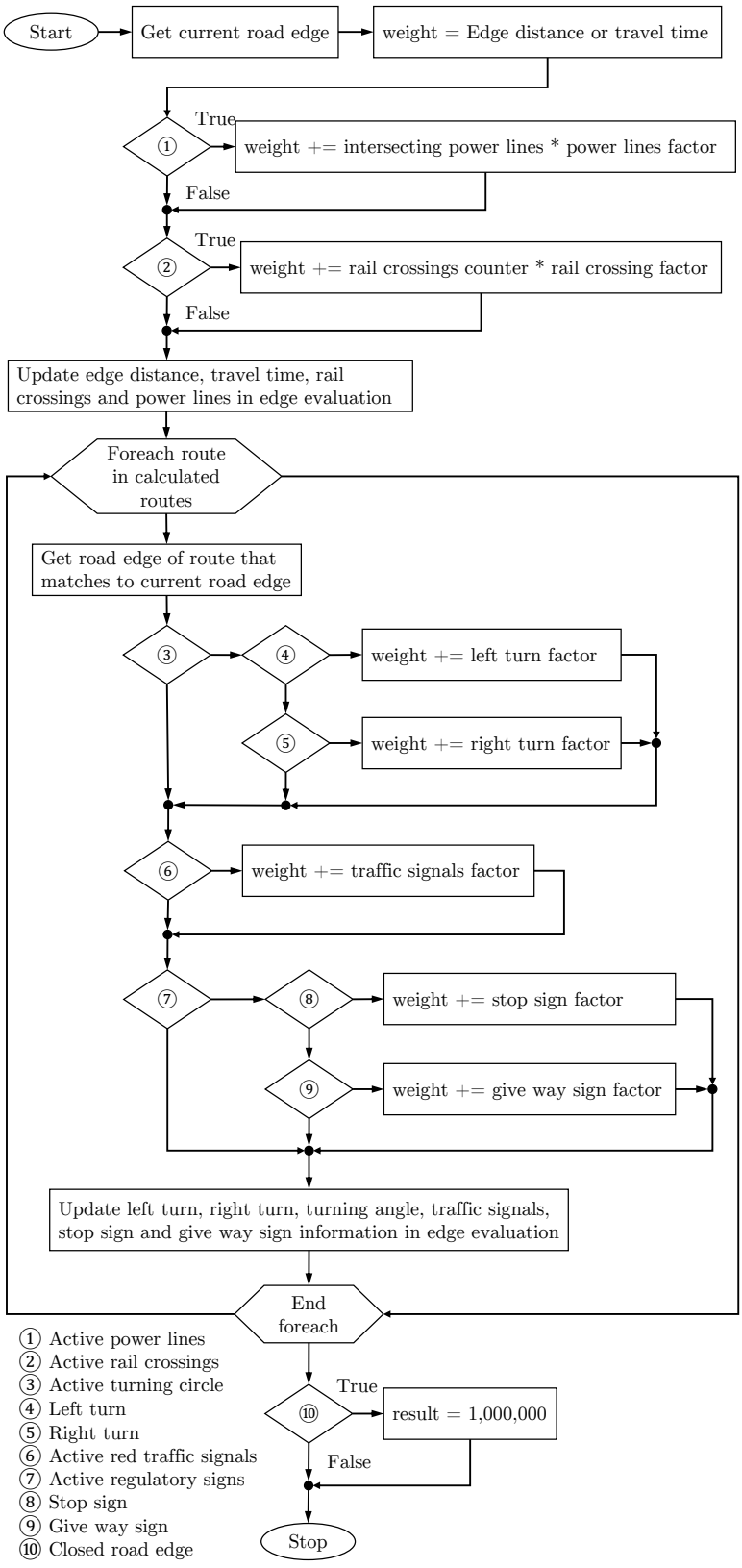


Figure 6.8: The flowchart describes the approach used to calculate a road edge weight.

The main part of the flowchart is processed by a foreach loop, which iterates through planned routes. The loop is a custom approach of the route planner and can be activated by users that configure it to consider turning circles, traffic signals and/or regulatory signs. For these constraints, a current and previous road edge of a planned route will be taken into account. For instance, in order to add weight at a left or right turn, a previously calculated route that defines the current road edge as part of a left or right turn must exist. Therefore, once a route has been calculated, the same route will be calculated again considering the road sections of a previously planned route to add additional costs at turning circles, traffic signals and regulatory signs. If the route planner identifies two identical consecutive routes, then the best possible route has been found.

At the beginning of the loop, a road edge that is the same as the current road edge will be retrieved from the top of the flowchart. At Condition Statement 3, it will be checked whether a turning circle has to be taken into account in the route planning. If that is true and if a current road edge is either a left turn (Condition Statement 4) or a right turn (Condition Statement 5) compared to the previous road edge of a planned route, then the weight of a left or right turn will be added to the edge weight, respectively.

At Condition Statement 6, it will be checked whether the configuration for traffic signals is active and whether a red light occurs between a current and previous road edge. If that is the case, then the factor of a red traffic signal will be added to the current edge weight.

Condition Statement 7 is used to identify if the configuration for regulatory signs is active. If that is the case and if a stop sign (Condition Statement 8) or a give way sign (Condition Statement 9) occurs between a current and previous road edge, then the respective cost factor will be added to the current edge weight. Before the next loop iteration, the retrieved information about left or right turns, the turning angle, traffic signals and regulatory signs will be saved for the route evaluation.

After the loop, it will be identified at Condition Statement 10 whether a current road edge is currently closed. If that is true, then the edge weight will be set to 1,000,000. This very large edge weight will prevent Dijkstra's algorithm from using a closed road section in any case as part of the shortest route. The flowchart processing finishes at the stop.

6.4.3 Data Layer Handling

The data layers visualisation of the route planner map can be compared to graphic software that allows a layer to be displayed on top of another layer. The visualisation of map content in this paper will be conducted by the GeoTools Java GIS Toolkit. Figure 6.9 shows a sample road network representation with multiple layers added to a map panel in the order of traffic signals (green circles), road edges (black line strings), road nodes (red circles), power lines (orange line strings) and road stopping places (blue signs) on top.

```
1  int layersIndexCount = mapPane.getMapContent().layers().size() - 1;
2  for (int i = layersIndexCount; i >= 0; i--) {
3    // remove a layer at index i from the map pane in descending order
4    mapPane.getMapContent().removeLayer(mapPane.getMapContent().layers().get(i));
5  }
6  // add a layer road edges to the map pane, and save the layer index
7  if (layerRoadEdge.getFeatureSource().getFeatures().size() > 0) {
8    mapPane.getMapContent().addLayer(layerRoadEdge);
9    MainView.roadEdgeLayerIndex = mapPane.getMapContent().layers().size() - 1;
10 }
```

Source Code 6.6: Principle of refreshing the map pane content while removing the current map pane layers and adding layers with features to the map.

Figure 6.10 shows a sample road network with a planned route in blue and a closed road in pink. The route also contains a start node shown as a yellow circle, a target node indicated as a green circle and highlighted nodes along the route indicated as large gray circles. The approach of this thesis to update the map content removes the current map layers and adds new layers to the map, as

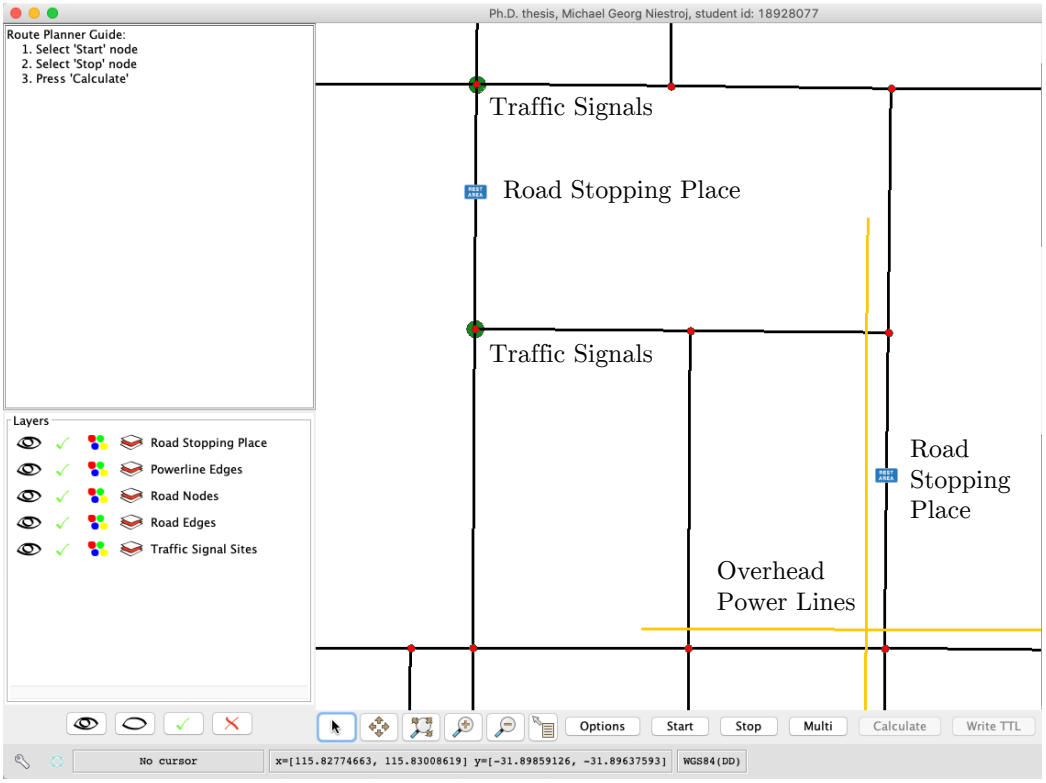


Figure 6.9: Map visualisation of a sample road network with the route planner.

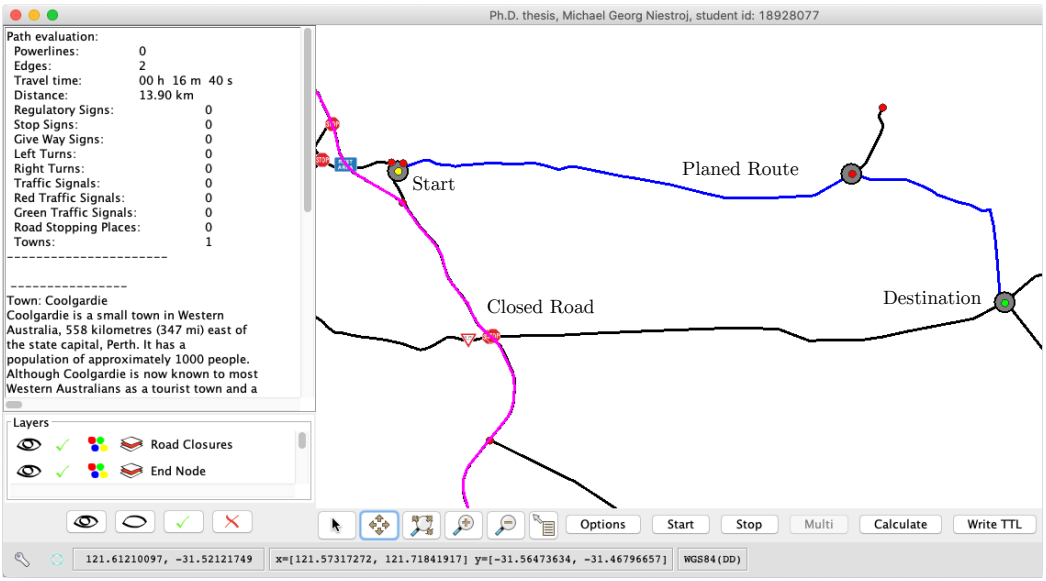


Figure 6.10: Map visualisation of a sample planned route in blue and a closed road in pink.

shown in the example regarding road edges in Source Code 6.6. The remaining layers 'route edges', 'traffic signal sites', 'road nodes', 'route nodes', 'power lines', 'road stopping places', 'regulatory signs', 'start node', 'end node', 'multi-node', 'rail crossings' and 'road closures layers' can be added via the same logic as the 'road edge layer' shown in the example of this section.

6.4.3.1 Selected Nodes

The highlighting of a selected start node, end node and multi-node will be done with a GeoTools filter function. If the 'start', 'end' or 'multi' button is pressed at the route planner main panel, then an 'on mouse clicked' event will be activated to record the coordinates of the mouse clicks on the map.

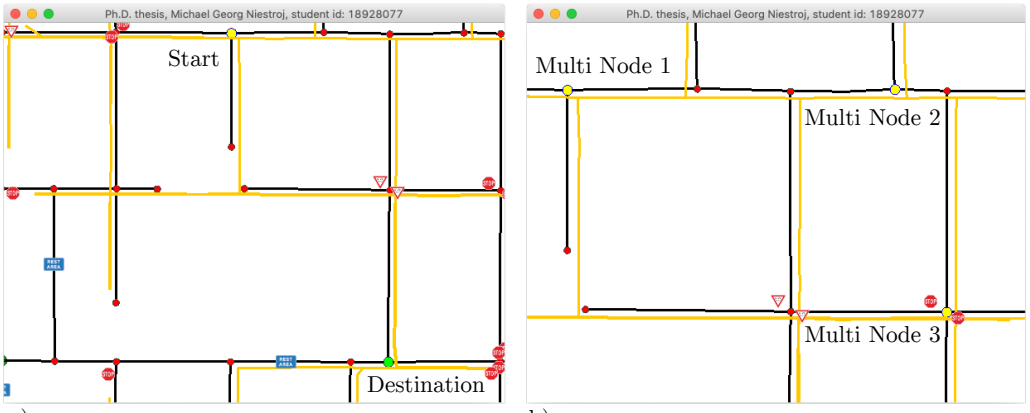


Figure 6.11: Selected nodes from a yellow start node to a green target node are indicated in a), whereby b) shows selected yellow multiple nodes for the processing of a multi-node route.

An example is given in Figure 6.11, whereby Figure 6.11 a) shows a simple route selection with a start and target node and Figure 6.11 b) shows multiple nodes selected for a route with multiple destinations. After the map is clicked, an area of 21x21 pixels will be filtered to identify the map features in the clicked area. If multiple features are present inside the filtered area, then the nearest element will be selected. For instance, Figure 6.11 a) shows a selected start node highlighted in yellow and a selected end node highlighted in green. The selected

multi-nodes are highlighted in yellow, as indicated in Figure 6.11 b), whereby the node selection order will be taken into account in the route planning.

6.4.4 Route Processing

This section will explain important technical details of the route processing, which includes the route calculation process and the evaluation process. After that, we will learn how to write a planned route into the ontology and how to load the data back into the route planner. Furthermore, an approach will be introduced to clean the road network data, which means that road section vertices that represent the same location will share the same coordinates after the processing.

6.4.4.1 Calculate Route

To consider left turns, right turns, traffic signal sites and regulatory signs, a new approach that evaluates and calculates a route multiple times until the best route is found has been designed (see Source Code 6.7).

At the beginning of the source code, a route list will be cleared, and a Dijkstra shortest pathfinder object will be initialised in the variable 'pf'. Then, a loop will be processed until two of the most current routes are equal or the configured route processing limit has been reached with the value of the variable 'MaxRouteCount', whereby the default value will be set to 10. That means even if the ninth and tenth calculated routes are different, the tenth will be taken as the best possible route. At each loop iteration, a new route will be calculated. After the calculation, the route direction will be validated for accuracy. If it is not correct, then the order will be reversed. In any case, a current processed route will be added to the routes list. After the loop processing, the best possible route will be saved in the variable 'result', which refers to the last processed route.

```
routeList.clear();
DijkstraShortestPathFinder pf = null;
for (int i = 0; i < MaxRouteCount; i++) {
    pf = new DijkstraShortestPathFinder(graph, routeStartNode, weighter);
    pf.calculate();
    Path route = pf.getPath(routeEndNode);
    ArrayList<BasicEdge> routeEdges = (ArrayList<BasicEdge>) route.getEdges();
    // If route is in wrong direction, then reverse it
    if (!routeEdges.get(0).getNodeA().equals(routeStartNode) && !routeEdges.get
        (0).getNodeB().equals(routeStartNode)) {
        Collections.reverse(routeEdges);
    }
    routeList.add(route);
    // If the last 2 routes are equal at index 'i' and index 'i-1', then
        the best possible route has been identified and the process ends
    if (i > 0 && routeList.get(i).equals(routeList.get(i - 1))) {
        i = MaxRouteCount;
    }
}
result = routeList.get(routeList.size() - 1);
```

Source Code 6.7: If left turns, right turns, traffic signals and/or regulatory signs (stop and give way) are taken into account in the route planning, then a route will be planned multiple times until it matches with a previously planned route or the configurable route processing limit has been reached.

6.4.4.2 Evaluate Route

An evaluation object is maintained continuously during the calculation of a route in the edge weighter function of the route planner. The evaluation object collects general route information, such as the travel time, distance, number of power lines, regulatory signs, rail crossings, road stopping places, towns and traffic signals. In addition, it contains information about a used road edge, which includes the distance, travel time, traffic signal state, left turns, right turns, turn angles, give way signs, stop signs, road stopping places, start/end SLK, town name and common usage name. The road edge evaluation object keeps track of possible extra travel time and extra distance. For instance, a route evaluation from the Federal Street in Nannup to the Main Street in Manjup of a synthetic dataset is indicated in the evaluation panel in Figure 6.12.

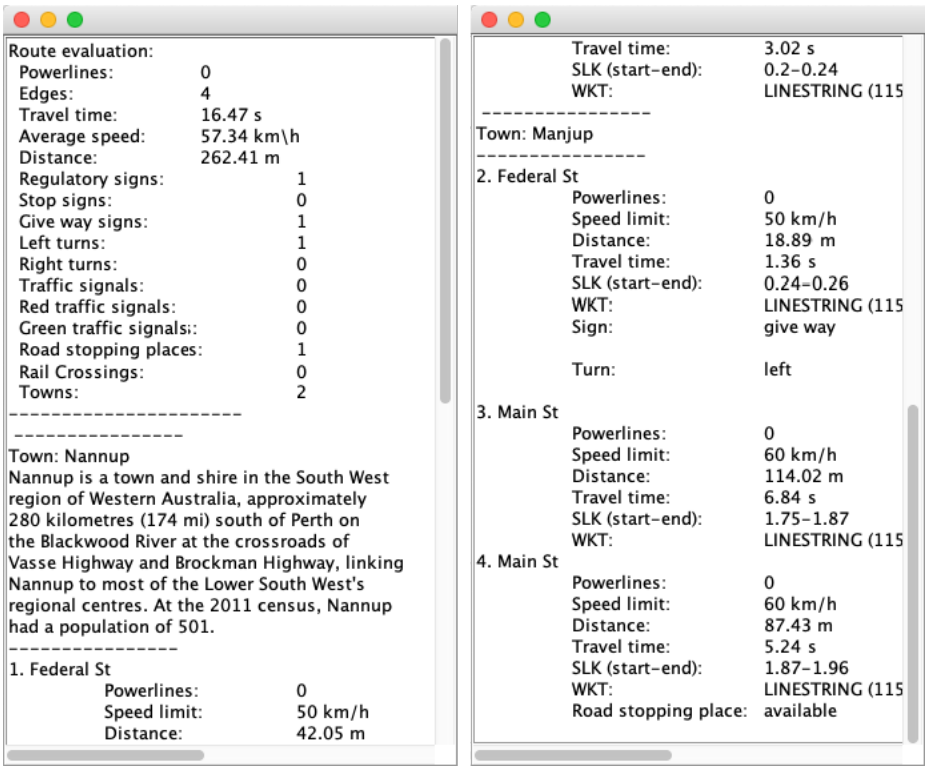


Figure 6.12: Sample evaluation of a route with a distance of about 262 m that contains one give way sign, one left turn, one road stopping place and four road edges and navigates through two towns.

In addition to the mentioned information, the figure shows a description of the town 'Nannup' that was retrieved from DBpedia[3] with the SPARQL query, as indicated next:

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbo: <http://dbpedia.org/ontology/>
select ?label where {
 <http://dbpedia.org/resource/ + < town > + ,_Western_Australia>
 dbo:abstract ?label .
 FILTER (lang(?label) = 'en')
}
```

[3] Source: ''http://dbpedia.org/resource/Nannup,_Western_Australia'

In the SPARQL query above, the hash value '<town>' will be replaced at each DBpedia request by a current town or suburb. Possible whitespaces will be replaced by an underscore as required by the DBpedia SPARQL endpoint. For instance, the suburb 'City Beach' can be parsed as 'City_Beach' by DBpedia.

6.4.4.3 Write Route into Ontology

After a route has been successfully calculated and evaluated, it can be written into the 'Route.owl' ontology file. The process that writes a route into the ontology is indicated in Algorithm 6.3. At the beginning of the algorithm, a route evaluation object is parsed to the algorithm, and the route ontology is defined as an output. After that, a foreach loop that iterates through each road section individual of the route evaluation object is processed. The first and last road section of a route will be marked as a key node.[4] Each road section will be written into the output file with the use of an ascending identifier.

Algorithm 6.3: Example of a line string and data property extraction from a road section individual.

```
1 RouteEvaluation ⟵ load the route evaluation object
2 File ⟵ 'Route.owl' ontology file as output file
3 foreach RoadSection in RouteEvaluation do
4 |   if RoadSection is first or last entry in RouteEvaluation then
5 |   |   RoadSection ⟵ set as key node and add first/last point as WKT
6 |   end
7 |   File ⟵ write RoadSection as part of route
8 end
```

For instance, let us consider a sample route from a start node to a target node, as indicated in Figure 6.13. As one can see, the route consists of three road sections. After a click on the 'write TTL' button at the main panel of the route planner, the algorithm will write route-related individuals into the ontology, as summarised in Table 6.3. Then the created individual 'alg:processRouteGenerator' will be used as provenance to inform about what algorithm created the data. The

[4] The key node marking simplifies the identification of a route; by definition, a complete route is always in between two key nodes.

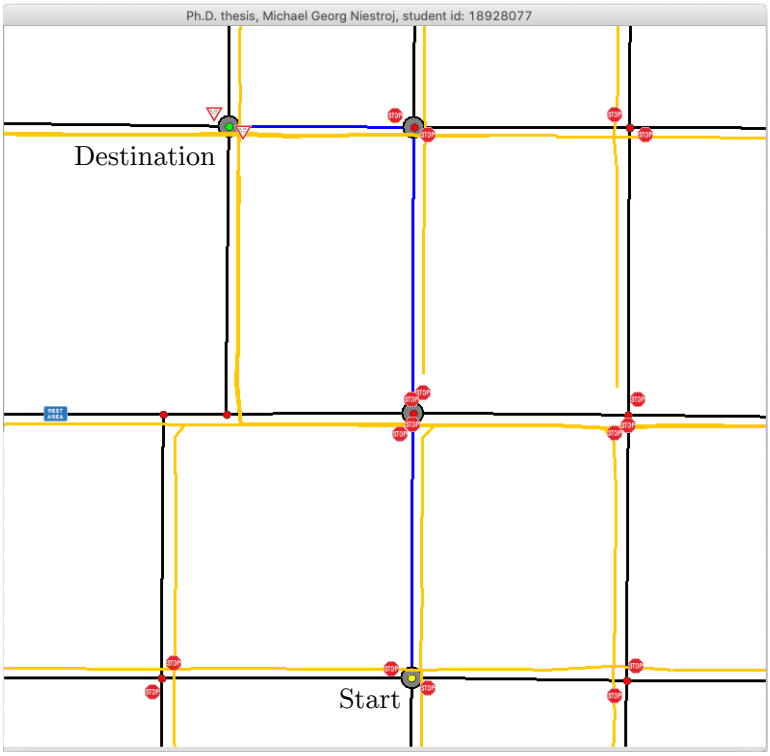


Figure 6.13: Arbitrary route from a yellow start node to a green target node.

individuals with the prefix 'route' contain the actual route information, while individuals with the prefix 'mrwa' refer to original entries of the MRWA ontology.

Table 6.3: List of route ontology individuals created for the route in Figure 6.13.

Individuals
alg:processRouteGenerator
route:MRWA
route:Route_FROM_YorkSt_10654371_TO_EldoradoSt_10654366_000001
route:Route_FROM_YorkSt_10654371_TO_EldoradoSt_10654366_000002
route:Route_FROM_YorkSt_10654371_TO_EldoradoSt_10654366_000003
mrwa:LineCoordinates_115.83275919_-31.89460181_115.83403647_-31.89460712
mrwa:LineCoordinates_115.83401817_-31.89840850_115.83402961_-31.89658192
mrwa:LineCoordinates_115.83402961_-31.89658192_115.83403647_-31.89460712
mrwa:MRWA_RoadSection_EldoradoSt_10654366
mrwa:MRWA_RoadSection_YorkSt_10654371
mrwa:MRWA_RoadSection_YorkSt_10654372

6.4.4.4 Read Route from Ontology

The function 'getOwlRoutes' will read an ontology with saved routes and make the content available for the route planner, as indicated in Algorithm 6.4. At the beginning of the algorithm, the route ontology will be loaded into the variable 'Ontology'. Then, an array list for the line strings of a route will be defined with the variable 'RouteLineStrings', and a list with the variable name 'Route' will be initialised as a container for the routes.

After that, the ontology will be iterated with a foreach loop, and each route individual that contains 'Route_FROM_' will be retrieved with its classes, properties and object properties. Then, a set of Boolean variables will be set to the initial value 'false'. The Boolean variables will be used to identify whether a current road section object is a key node, has a previous node, has a next node and is the start of a series of route nodes. Every time a route individual is identified, the following activities will be processed:

1. A foreach loop will be used to iterate through all classes of a route individual to identify whether a current individual is a key node (see lines 8–12). If a key node is found, then the variable 'KeyNode' will be set to the value 'true'.

2. Another foreach loop iterates through the object properties of an individual (see lines 13–21). If an individual has no 'route:hasPreviousRoute' or no 'route:hasNextRoute' object property, then it is either the start or end of a route, respectively. Each individual has a 'prov:hadPrimarySource' object property that establishes a connection between a route individual and an MRWA individual.

3. A further foreach loop will be used to iterate through individual properties to identify the start and target nodes of a route (see lines 22–29). A route is completed when a key node has been found and no next route individual is available. If that is the case, the collected route line strings, the start

node and the target node will be added to the list of routes. After that, the list 'RouteLineStrings' will be cleared, so that individuals of the next route can be added.

Once the algorithm stops iterating through the individuals, a list of routes will be added to the main panel as indicated in Figure 6.14. A so-called 'list selection listener' will then be activated to enable a route selection. If a user selects a route, then the map panel will be reset, the selected route will be displayed on the map, and a route evaluation will be processed.

Figure 6.14: The panel shows four selectable routes in the route selection panel.

6.4.4.5 Closed Roads

The functionality of displaying closed roads consists of two stages. The first stage requires downloading currently closed road data, and the second stage is related to the integration of the downloaded data in the route planner. The process downloading closed road data is indicated in Source Code 6.8. After the initialisation of an input stream, the GeoJSON file that contains the currently closed road data will be downloaded. The download process will be conducted within a try–catch block to prevent possible download errors. If the procedure is successful, then a try–catch statement will be used to transfer the road closure data from the stream to the hard drive. If the file saving approach is successful, then a message will inform the user about the successful download process, which will further enable access to the closed road data in the route planner.

Algorithm 6.4: Overall processing of the function 'getOwlRoutes' that reads a route ontology file back into the route planner.

```
1  Ontology ⟵ read the 'Route.owl' ontology and sort individuals by name
2  RouteLineStrings ⟵ create empty array list
3  Route ⟵ list of routes
4  foreach Individual in Ontology do
5      if Individual contains 'Route_FROM_' then
6          Classes, Properties, ObjectProperties ⟵ get from Individual
7          KeyNode, PreviosRoute, NextRoute, PrimarySource,
            StartSeries ⟵ set to false
8          foreach Class in Classes do
9              if Class 'KeyNode' in Classes then
10                 KeyNode ⟵ set true
11             end
12         end
13         foreach ObjectProperty in ObjectProperties do
14             if ObjectProperty is 'route:hasPreviousRoute' then
15                 PreviosRoute ⟵ set true
16             else if ObjectProperty is 'route:hasNextRoute' then
17                 NextRoute ⟵ set true
18             else if ObjectProperty is 'prov:hadPrimarySource' then
19                 RoadSection ⟵ get road section of MRWA ontology
20                 RouteLineStrings ⟵ add line string of RoadSection
21         end
22         foreach Property in Properties do
23             if Property is 'geosparql:asWKT' then
24                 if KeyNode and not PreviousRoute then
25                     Start ⟵ set point
26                 else if KeyNode and not NextRoute then
27                     Target ⟵ set point
28             end
29         end
30         if KeyNode not nextRoute then
31             Route ⟵ add RouteLineStrings, From and Target
32             RouteLineStrings ⟵ empty for next route
33         end
34     end
35 end
```

With a click on the button 'add road closures', the downloaded closed road file will be read from the hard drive and integrated into the route planner, as indicated

```
1  InputStream inputStream = null;
2  try {
3      // Source: https://catalogue.data.wa.gov.au/dataset/mrwa-road-
           closures-closed
4      String downloadUrl = "https://opendata.arcgis.com/datasets/8
           b47dad5f85948f6b2b8c5871b0a6987_5.geojson";
5      roadClosures.setText("Download:␣\n" + downloadUrl);
6      inputStream = new URL(downloadUrl).openStream();
7  } catch (IOException ex) {
8      ex.printStackTrace();
9      roadClosures.setText("Error:␣Download␣not␣possible"); }
10 // write the downloaded data from the stream onto the hard drive
11 try {
12     String fileDestination = "Road_Closures__Closed.geojson";
13     Files.copy(inputStream, Paths.get(fileDestination), StandardCopyOption.
           REPLACE_EXISTING);
14     roadClosures.setText(roadClosures.getText() + "\n" + "File␣Saved:␣" +
           fileDestination);
15 } catch (IOException ex) {
16     ex.printStackTrace();
17     roadClosures.setText(roadClosures.getText() +"\n"+"Error:␣File␣Not␣Saved");}
```

Source Code 6.8: Download current road closure data from the Western Australian OGD portal.

Algorithm 6.5: Approach to read closed roads into the route planner.

```
1 RoadClosures ⟵ read 'Road_Closures__Closed.geojson'
2 FeaturesCollection ⟵ get features of RoadClosures
3 if FeaturesCollection is not empty then
4 |   ClosedRoadsList ⟵ get line strings from FeaturesCollection
5 |   RoadClosuresLayer ⟵ create layer of FeaturesCollection
6 |   InformationPanel ⟵ update RoadClosures
7 |   Map ⟵ update RoadClosuresLayer
8 end
```

in Algorithm 6.5. In detail, the road closure file will be read, and a features collection will be created with the internal GeoTools function 'FeatureJSON'. If a feature collection with content exists, then a list of line strings that represent the closed roads and a layer that can be displayed on the map will be created. After that, the closed roads layer can be added to the map, and the information on the current road closures can be updated. For instance, Figure 6.15 shows

listed road closures, with the information of a border restriction and a closed road indicated on the map as pink road edges.

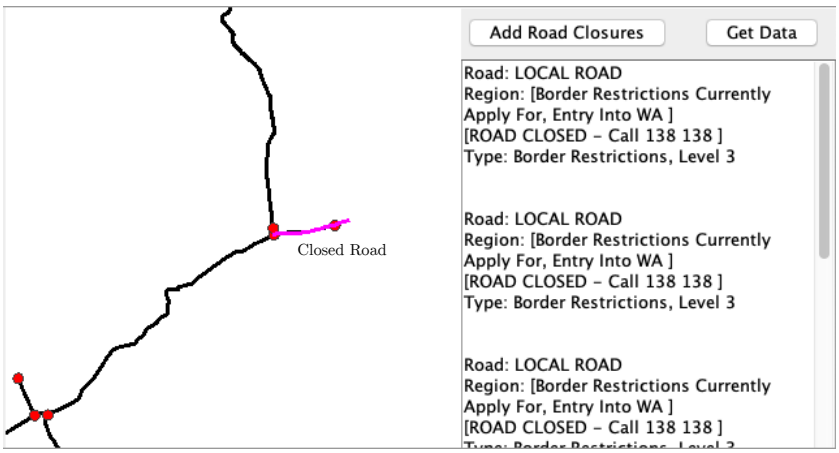


Figure 6.15: Closed road data indicated on the map with pink line strings and summarised in the text panel.

6.4.4.6 Clean Road Network

When working with real-world road network data, road nodes that describe the same intersection may not share the same coordinates. As a result, the GeoTools graph generator considers nodes with different coordinates as different assets. Although GeoTools provides a configurable tolerance for nearby nodes with its graph generator function, the functionality is limited and does not always the desired result.

For example, let us consider the sample road network in Figure 6.16. In the left image, the road nodes of the black road section and the highlighted blue road section are not connected, as the coordinates are not equal. In the middle image, a GeoTools graph builder tolerance has been configured and activated so that a line string point has been added to connect the road edge to a nearby road node. However, the disadvantage of the tolerance functionality is that the route planner can interfere with the information of a left or right turn in some cases as indicated in the example. To eliminate this kind of error, a Python script that

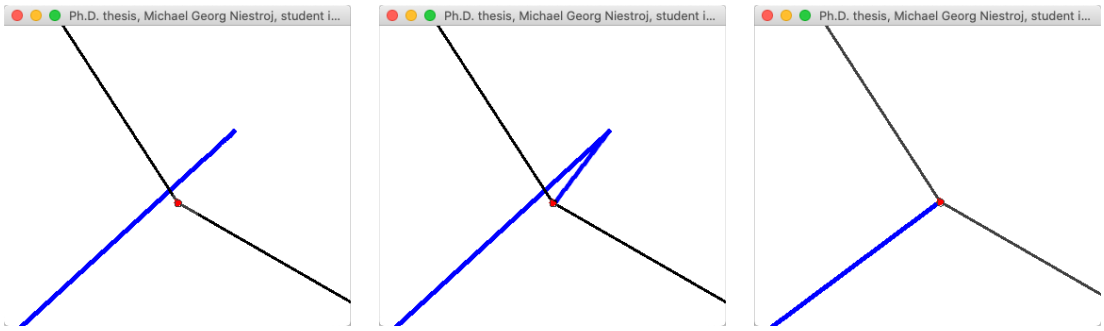


Figure 6.16: Road network that shows a graph with zero tolerance on the left, a road network with a configured graph tolerance in the middle, and a road network that was cleaned by a developed Python script on the right.

merges line string coordinates within an offset of 2 m into the same values was developed, as indicated in Algorithm 6.6.

Algorithm 6.6: Translate MRWA road network data to match data within a defined offset.

```
1  RoadNetwork ⟵ read 'Road_Network_MRWA.geojson'
2  PointsList ⟵ initialise empty list
3  Offset ⟵ set to 0.0002 longitude and 0.0002 latitude, about 2 m
4  foreach Entry in RoadNetwork do
5  |  LineString ⟵ get of Entry
6  |  FirstPoint ⟵ get first point of LineString
7  |  if FirstPoint within Offset of a point in PointsList then
8  |  |  FirstPoint ⟵ translate to found point
9  |  end
10 |  FirstPoint ⟵ add to PointsList
11 |  if LastPoint within Offset of a point in PointsList then
12 |  |  LastPoint ⟵ translate to found point
13 |  end
14 |  LastPoint ⟵ add to PointsList
15 |  LineString ⟵ update with FirstPoint and LastPoint
16 end
17 RoadNetwork ⟵ overwrite 'Road_Network_MRWA.geojson'
```

At the beginning of the algorithm, a file with MRWA road network data will be read, and a list of point coordinates that contains only the first and last coordinates of a road section will be loaded. Then, the offsets in longitude and latitude will be defined. After that, a foreach loop will iterate through the data

entries and retrieve the first and last points (vertex) of each road edge line string. For each of the retrieved points, a condition statement will verify whether a point that is within the defined offset of 0.0002 longitude and 0.0002 latitude exists in the list of points. If that is the case, then the first and/or last point will be translated to a nearby point of the same intersection. After that, both points will be added to the list of points, and the original line string will be updated. Once the processing of the foreach loop is completed, the road network file will be overwritten with the updated dataset. The result is a clean road network with matching road nodes, as indicated in the right image of Figure 6.16.

6.5 Chapter Summary

This chapter discussed the principles of new methods that have been designed, applied and developed during the research of this thesis. The methods were grouped into three sections, namely data creation, road network translation and route planning.

The first section on 'data creation' treated the automated creation of ontology data that can be retrieved in the GeoJSON data format from the employed MRWA, Landgate and OSM datasets. The data creation approach was explained with flowcharts and covered original road network data, as well as translated road network datasets. Important functionalities that are required for the automated data individuals data creation with the use of the Turtle syntax were indicated with source codes where possible and with algorithms for complex procedures.

The second section on 'road network translation' was about the implementation of a road network translation algorithm that was optimised to translate MRWA road sections and intersections to the shape of Landgate road sections, roundabouts and roundabout connectors. In the scope of that section, a flowchart was introduced that explained the different translation methods in detail.

The third section on 'route planning' covered the main functionalities of the route planner, which included route processing and edge weight determination.

The road network data were cleaned so that road assets that describe the same node shared the same coordinates. The handling of road map layers was explained, and various road map layers were introduced, such as road edges, road nodes and road stopping places. Planned routes were written into a route ontology and loaded back into the route planner as selectable preloaded routes.

Chapter 7

Evaluation

7.1 Chapter Introduction

In this chapter, the road network translation, road network ontology conflation and route planner will be evaluated with several experiments. The results of this chapter are a significant contribution of this dissertation, as the analysis of the results will lead to a conclusion about the suitability of the Semantic Web as an efficient processing method of heterogeneous road network data. In order to evaluate this research, the road network translation will indicate two small road network selections and one larger-scaled road network to validate the seven designed translation methods. The road network conflation will be evaluated at a roundabout and a road section, with datasets from MRWA, Landgate and OSM. The route planner will be evaluated with an inner city route, a large-scale route, a route with overhead power lines, a route with rail crossings and a route that includes a closed road on the track.

7.2 Road Network Translation

Early road network translation results were published by Niestroj et al. (2019). Compared to the published results, the road network translation approach of this

thesis enables the application on a larger-scale road network. In the background, the road network examples in this section will show a street layer that is based on an Esri dataset,[1] which is available for use in QGIS after the installation of the HCMGIS[2] plugin.

7.2.1 Road Network Selection 1

The road network indicated in Figure 7.1 is used as the first example to evaluate the road network translation approach of this thesis and shows 10 numbered MRWA intersections and their related road sections. The red line strings indicate Landgate road sections, roundabouts and roundabout connectors, and the black line strings indicate MRWA road sections. One can see that the visualisation is not harmonised, as the road network shows discrepancies in comparing the shape of the black and red line strings in some cases, such as at the roundabout.

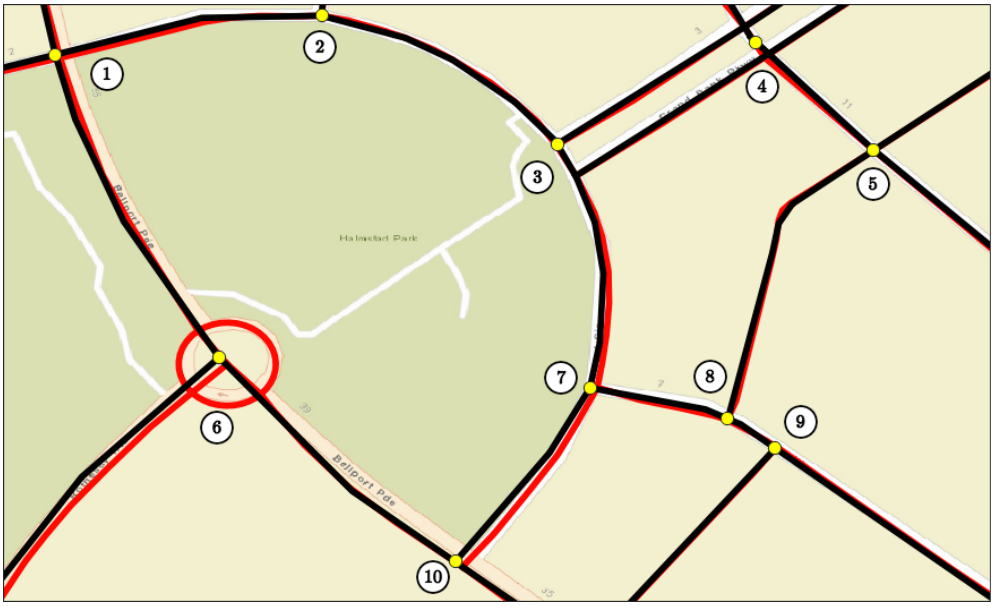


Figure 7.1: Road network selection 1 with road assets from Landgate and MRWA with original coordinates.

[1] https://www.arcgis.com/home/webmap/viewer.html
[2] https://plugins.qgis.org/plugins/HCMGIS/

In Figure 7.2, the same road network selection is indicated after the application of the translation algorithm. The figure shows that the red line strings are barely visible, which means that the road centreline representation of the Landgate and MRWA datasets is harmonised at the selected area of interest.

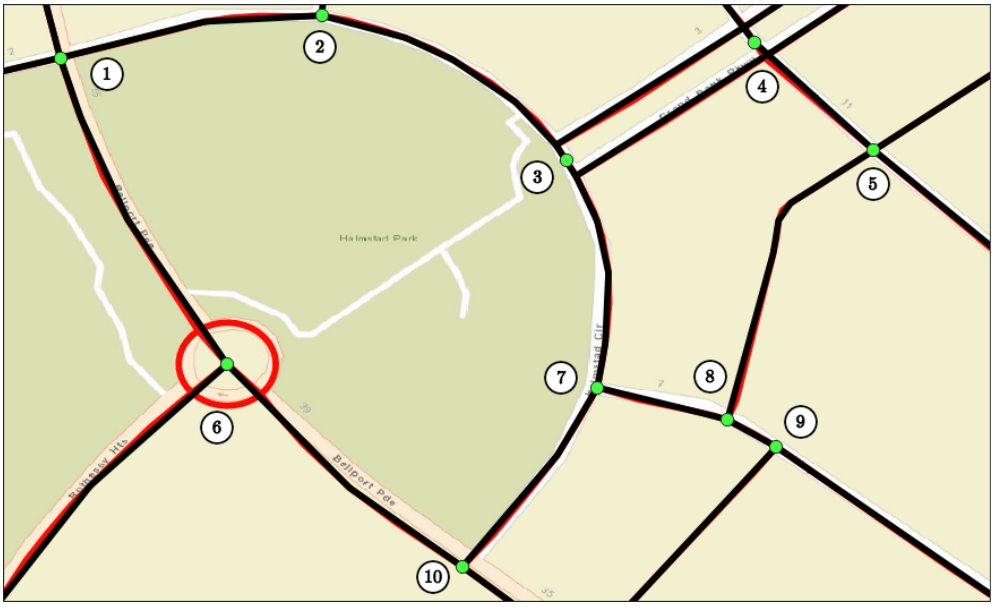


Figure 7.2: Road network selection 1 with road assets from Landgate and MRWA. The road assets of MRWA are indicated with translated coordinates.

Table 7.1: Distance and translation method evaluation of MRWA intersections from Figure 7.1 into Figure 7.2.

ID	Original		Translated		Translation	Distance / m
	Longitude / °	Latitude / °	Longitude / °	Latitude / °		
1	115.70904402	–31.68000999	115.70905800	–31.68001900	Method 2	1.65
2	115.70978803	–31.67990101	115.70978800	–31.67990100	Method 2	0.05
3	115.71044575	–31.68025979	115.71047100	–31.68030600	Method 6	5.61
4	115.71100117	–31.67997513	115.71099600	–31.67997500	Method 6	0.44
5	115.71133003	–31.68027599	115.71133000	–31.68027700	Method 2	0.14
6	115.70950098	–31.68085502	115.70952400	–31.68087400	Method 1	3.05
7	115.71053697	–31.68094002	115.71055800	–31.68093900	Method 2	1.98
8	115.71092100	–31.68102900	115.71092202	–31.68102604	Method 2	0.36
9	115.71105497	–31.68110903	115.71105800	–31.68110500	Method 2	0.58
10	115.71016099	–31.68142499	115.71018100	–31.68143900	Method 2	2.48

The evaluation of the translated MRWA intersections (green circles) is indicated in Table 7.1. The table shows that the algorithm translated five inter-

sections with a distance of less than 0.59 m, four intersections were translated with a distance of approximately 1.98 m to 3.05 m, and the intersection with ID 3 was translated with a distance of approximately 5.61 m after the application of the Translation Method 6. The reason for the larger translation distance at Intersection 3 is related to the position of the original intersection (see Figure 7.1). The intersection was previously connected to a road edge link, and after the application of the translation algorithm, the intersection was translated to the centre location of a road edge. Thus, in that particular case, a translation distance of above 5 m could be considered appropriate.

An evaluation of the road section vertices that occur between the labelled intersections is given in Table 7.2 and provides the translation information of 45 road section vertices. The evaluation of the road section from Intersection 6 to an indiscernible intersection has also been evaluated, as the Landgate and MRWA datasets indicated a larger visible discrepancy between the red Landgate and black MRWA road section data. Overall, the evaluation of Table 7.2 can be summarised as follows:

- Twenty-seven vertices were translated with less than 1 m, whereby two of them were not translated due to the application of Translation Method 4.0.
- Ten vertices were translated between 1.00 m and 2.00 m.
- One vertex was translated with about 2.19 m.
- Three vertices were translated with more than 3.00 m, with the largest translation being about 3.92 m.
- Four vertices were not translated, as they were part of a left or right carriageway, which was not supported by the translation algorithm.

After evaluating the translation values, it can be said that in most cases, a translation occured with less than 2.00 m. The values seem to be reasonable

Table 7.2: Distance and translation method evaluation of MRWA road sections from Figure 7.1 into Figure 7.2.

Between Intersections	Original		Translated			
	Longitude / °	Latitude / °	Longitude / °	Latitude / °	Translation	Distance / m
1 and 2	115.70904420	–31.68001027	115.70905761	–31.68001929	Method 3.2	0.04
1 and 2	115.70945322	–31.67990751	115.70946233	–31.67991836	Method 4.1	1.48
1 and 2	115.70978757	–31.67990147	115.70978844	–31.67990128	Method 3.2	0.07
1 and 6	115.70904420	–31.68001027	115.70905761	–31.68001929	Method 3.2	0.04
1 and 6	115.70910020	–31.68019098	115.70911501	–31.68018529	Method 4.2	1.54
1 and 6	115.70923526	–31.68047556	115.70924591	–31.68046957	Method 4.2	1.21
1 and 6	115.70950153	–31.68085502	115.70952448	–31.68087374	Method 3.1	0.05
2 and 3	115.70978757	–31.67990147	115.70978844	–31.67990128	Method 3.2	0.07
2 and 3	115.71002181	–31.67996351	115.71002193	–31.67996315	Method 4.2	0.04
2 and 3	115.71015330	–31.68002454	115.71015395	–31.68002442	Method 4.1	0.06
2 and 3	115.71032780	–31.68014267	115.71032860	–31.68014247	Method 4.1	0.08
2 and 3	115.71044026	–31.68025046	115.71044333	–31.68026265	Method 3.2	1.16
2 and 3	115.71049763	–31.68034855	115.71049763	–31.68034855	Method 4.0	0.00
3 and 4	115.71044600	–31.68026000	-	-	No Translation	-
3 and 4	115.71096900	–31.67993300	-	-	No Translation	-
3 and 7	115.71049763	–31.68034855	115.71049763	–31.68034855	Method 4.0	0.00
3 and 7	115.71055088	–31.68047803	115.71055759	–31.68047582	Method 4.2	0.68
3 and 7	115.71057467	–31.68061958	115.71058900	–31.68061929	Method 4.2	1.36
3 and 7	115.71056470	–31.68081439	115.71057961	–31.68081665	Method 4.2	1.43
3 and 7	115.71053716	–31.68094002	115.71055786	–31.68093902	Method 3.2	0.02
4 and 3	115.71103600	–31.68000800	-	-	No Translation	-
4 and 3	115.71049800	–31.68034900	-	-	No Translation	-
4 and 5	115.71098634	–31.67996150	115.71099648	–31.67997525	Method 7	1.80
4 and 5	115.71132142	–31.68026913	115.71132997	–31.68027722	Method 3.2	1.12
4 and 5	115.71133030	–31.68027654	115.71132997	–31.68027722	Method 3.2	0.07
5 and 8	115.7113303	–31.68027654	115.71133000	–31.68027722	Method 3.2	0.07
5 and 8	115.7111021	–31.68042679	115.71109770	–31.68042402	Method 4.1	0.51
5 and 8	115.7110644	–31.68048123	115.71106420	–31.68047259	Method 4.1	0.96
5 and 8	115.71104637	–31.68056431	115.71104873	–31.68056484	Method 4.2	0.23
5 and 8	115.71092238	–31.68102649	115.71092126	–31.68102917	Method 3.2	0.06
6 and n.d.	115.70950153	–31.68085502	115.70952448	–31.68087374	Method 3.1	0.05
6 and n.d.	115.70911759	–31.68118882	115.70914211	–31.68121313	Method 4.2	3.56
6 and n.d.	115.70883210	–31.68155144	115.70887323	–31.68155599	Method 4.1	3.92
6 and 10	115.70950153	–31.68085502	115.70952448	–31.68087374	Method 3.1	0.05
6 and 10	115.70987376	–31.68122826	115.70986582	–31.68121851	Method 4.1	1.32
6 and 10	115.71016081	–31.68142554	115.71018118	–31.68143920	Method 3.2	0.06
7 and 8	115.71053716	–31.68094002	115.71055786	–31.68093902	Method 3.2	0.02
7 and 8	115.71085970	–31.68099977	115.71085703	–31.68101156	Method 3.2	1.34
7 and 8	115.71092238	–31.68102649	115.71092126	–31.68102917	Method 3.2	0.06
7 and 10	115.71016081	–31.68142554	115.71018118	–31.68143920	Method 3.2	0.06
7 and 10	115.71042452	–31.68112092	115.71044595	–31.68112830	Method 4.1	2.19
7 and 10	115.71053716	–31.68094002	115.71055786	–31.68093902	Method 3.2	0.02
8 and 9	115.71092238	–31.68102649	115.71092126	–31.68102917	Method 3.2	0.06
8 and 9	115.71095834	–31.68104159	115.71092126	–31.68102917	Method 3.2	3.85
8 and 9	115.71105524	–31.68110875	115.71105831	–31.68110462	Method 3.2	0.04

considering that the MRWA and Landgate road network data are collected by independent authorities.

7.2.2 Road Network Selection 2

The road network selection indicated in Figure 7.3 shows a second sample road network to evaluate the translation algorithm of this thesis. The road network shows 10 numbered MRWA intersections with their connected road sections. The road asset colour notation is defined as in the previous example, meaning that the Landgate data are indicated with red line strings, the MRWA road sections are visualised with black line strings, and the original MRWA intersections are highlighted as yellow circles.

The road network selection in the example shows an Intersection 7 surrounded by two Landgate road sections without a connecting MRWA side road. The reason behind placing an intersection at a straight road here is related to the road name change from 'Seaham Way' to 'Anchorage Drive North' in the driving direction from Intersection 6 and through Intersection 7 into Intersection 8.

Table 7.3: Distance and translation method evaluation of MRWA intersections from Figure 7.3 into Figure 7.4.

ID	Original		Translated		Translation	Distance / m
	Longitude / °	Latitude / °	Longitude / °	Latitude / °		
1	115.70253197	–31.67799996	115.70253100	–31.67803000	Method 2	3.37
2	115.70335174	–31.67758646	115.70333800	–31.67760000	Method 2	1.97
3	115.70257598	–31.67847101	115.70255200	–31.67846400	Method 2	2.36
4	115.70374502	–31.67821801	115.70374500	–31.67822700	Method 1	1.03
5	115.70503236	–31.67753101	115.70502900	–31.67753300	Method 2	0.37
6	115.70266098	–31.67925702	115.70266100	–31.67925700	Method 2	0.05
7	115.70364800	–31.67948300	-	-	No Translation	-
8	115.70429696	–31.67885203	115.70428800	–31.67886000	Method 6	1.20
9	115.70530596	–31.67827904	115.70532900	–31.67828500	Method 1	2.24
10	115.70563097	–31.67901298	115.70564500	–31.67902100	Method 2	1.58

The same road network selection after the application of the translation algorithm is displayed in Figure 7.4. One can see that the black line strings dominate the road network representation, which is an indication of good road network

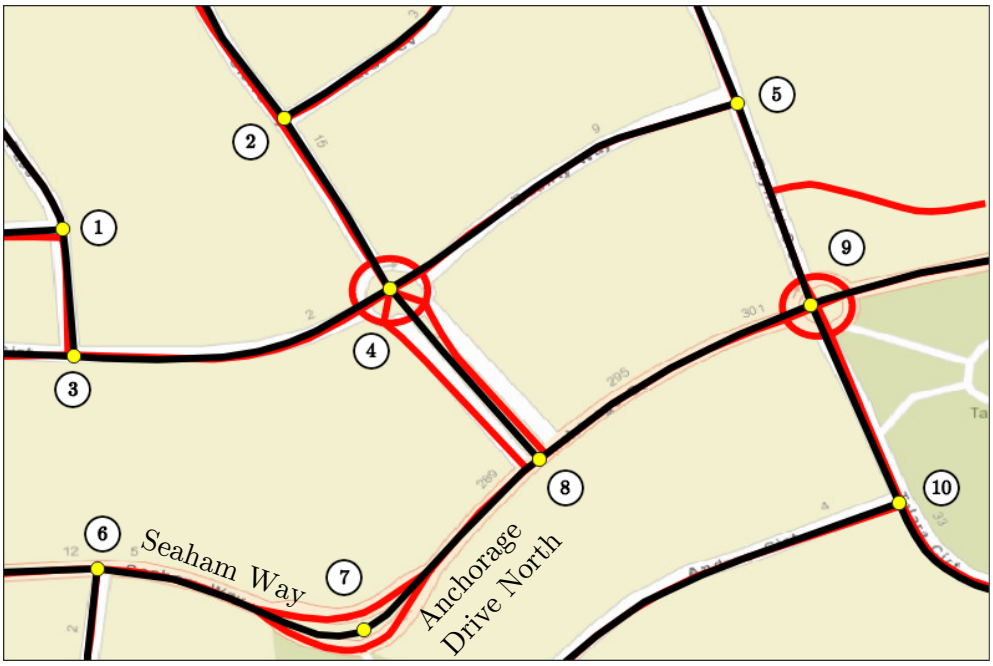


Figure 7.3: Road network selection 2 with road assets from Landgate and MRWA.

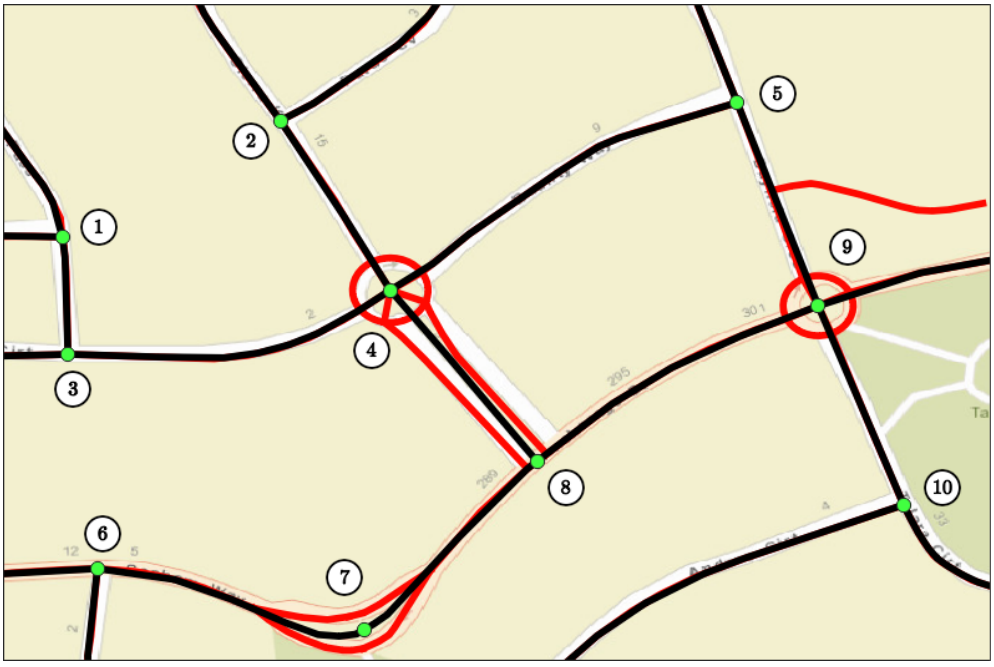


Figure 7.4: Road network selection 2 with road assets from Landgate and MRWA. The road assets of MRWA are indicated with translated coordinates.

data harmonisation between the MRWA and Landgate datasets. The evaluation of the translated green intersections is indicated in Table 7.3. The table indicates that the algorithm translated the intersections with Methods 1, 2 and 6. The translated intersections include two translations of less than 0.38 m, four translations between 1.00 m and less than 2.00 m and three translations of more than 2.00 m. The largest translated distance is about 3.37 m. Intersection 7 has not been translated, as it was identified by trial and error that the best possible approach to handle the case of an MRWA road section between two Landgate road sections is to not translate it. It has also been identified, that MRWA and Landgate use their own guidelines for the description of the road networks.

An evaluation of nine road sections occuring between the numbered intersections is indicated in Table 7.4, which shows the translation of 51 road section vertices. Overall, the translation table can be summarised as follows:

- Twenty-three vertices were translated with less than 1 m, while six of the vertices were not translated due to the application of Translation Method 4.0.
- Seventeen vertices were translated between 1 m and 2 m.
- Five vertices were translated between 2 m and 3 m.
- Six vertices were translated with more than 3 m, that includes the two largest translations with distances of about 12.45 m and 10.30 m.

After evaluating the translation values, in most cases, it can be said in most cases that a translation occured with less than 2.00 m, meaning that both authorities (MRWA and Landgate) are capturing road network data features at similar locations.

Table 7.4: MRWA road section evaluation from Figure 7.3 into Figure 7.4.

Between Intersections	Original		Translated			
	Longitude / °	Latitude / °	Longitude / °	Latitude / °	Translation	Distance / m
1 to 3	115.70257598	–31.67847101	115.70255237	–31.67846428	Method 3.2	2.36
1 to 3	115.70254533	–31.67810271	115.70254088	–31.67810308	Method 4.2	0.42
1 to 3	115.70253508	–31.67801341	115.70253064	–31.67803027	Method 3.2	1.92
1 to 3	115.70253242	–31.67800051	115.70253064	–31.67803027	Method 3.2	3.31
2 to 4	115.70374511	–31.67821773	115.70374516	–31.67822729	Method 3.1	1.06
2 to 4	115.70372950	–31.67819888	115.70374516	–31.67822729	Method 3.1	3.49
2 to 4	115.70358736	–31.67794926	115.70357493	–31.67795644	Method 4.2	1.42
2 to 4	115.70335174	–31.67758646	115.70333823	–31.67759994	Method 3.2	1.97
3 to 4	115.70257598	–31.67847101	115.70255237	–31.67846428	Method 3.2	2.36
3 to 4	115.70268304	–31.67847742	115.70255237	–31.67846428	Method 4.1	12.45
3 to 4	115.70288480	–31.67848263	115.70288505	–31.67847449	Method 4.2	0.91
3 to 4	115.70312911	–31.67847266	115.70312956	–31.67847707	Method 4.2	0.49
3 to 4	115.70325648	–31.67845216	115.70325804	–31.67845921	Method 4.2	0.80
3 to 4	115.70336729	–31.67842014	115.70337039	–31.67842909	Method 4.2	1.04
3 to 4	115.70347069	–31.67837676	115.70347596	–31.67838692	Method 4.2	1.23
3 to 4	115.70361398	–31.67829450	115.70362150	–31.67830677	Method 4.2	1.54
3 to 4	115.70374511	–31.67821773	115.70374516	–31.67822729	Method 3.1	1.06
4 to 5	115.70374511	–31.67821773	115.70374510	–31.67822729	Method 3.1	1.06
4 to 5	115.70378116	–31.67819687	115.70374516	–31.67822729	Method 3.1	4.80
4 to 5	115.70390048	–31.67812348	115.70390379	–31.67812760	Method 4.2	0.55
4 to 5	115.70406866	–31.67800334	115.70403587	–31.67802024	Method 4.1	3.63
4 to 5	115.70428680	–31.67784166	115.70437401	–31.67778608	Method 4.1	10.31
4 to 5	115.70451675	–31.67769278	115.70451771	–31.67769500	Method 4.2	0.26
4 to 5	115.70459361	–31.67766085	115.70459417	–31.67766261	Method 4.2	0.20
4 to 5	115.70503236	–31.67753101	115.70502937	–31.67753322	Method 3.2	0.37
4 to 8	115.70374511	–31.67821773	115.70374516	–31.67822729	Method 3.1	1.06
4 to 8	115.70393543	–31.67844548	115.70393543	–31.67844548	Method 4.0	-
4 to 8	115.70429723	–31.67885166	115.70428824	–31.67885980	Method 7	1.24
5 to 9	115.70503236	–31.67753101	115.70502937	–31.67753322	Method 3.2	0.37
5 to 9	115.70530651	–31.67827858	115.70532872	–31.67828468	Method 3.1	2.21
6 to 7	115.70266053	–31.67925693	115.70266130	–31.67925667	Method 3.2	0.08
6 to 7	115.70294483	–31.67929581	115.70294455	–31.67929695	Method 4.2	0.13
6 to 7	115.70314897	–31.67936591	115.70315836	–31.67937131	Method 4.1	1.07
6 to 7	115.70336171	–31.67945942	115.70336171	–31.67945942	Method 4.0	-
6 to 7	115.70347792	–31.67950106	115.70347792	–31.67950106	Method 4.0	-
6 to 7	115.70355570	–31.67950655	115.70355570	–31.67950655	Method 4.0	-
6 to 7	115.70363100	–31.67949364	115.70361965	–31.67949883	Method 4.2	1.22
6 to 7	115.70364811	–31.67948340	115.70364802	–31.67948303	Method 7	0.04
7 to 8	115.70364811	–31.67948340	115.70364802	–31.67948303	Method 7	0.04
7 to 8	115.70368206	–31.67946262	115.70368206	–31.67946262	Method 4.0	-
7 to 8	115.70373367	–31.67942712	115.70373367	–31.67942712	Method 4.0	-
7 to 8	115.70399912	–31.67914108	115.70399856	–31.67914060	Method 4.2	0.08
7 to 8	115.70423968	–31.67889549	115.70425271	–31.67888650	Method 4.1	1.59
7 to 8	115.70429723	–31.67885166	115.70428824	–31.67885980	Method 7	1.24
8 to 9	115.70429723	–31.67885166	115.70428824	–31.67885980	Method 7	1.24
8 to 9	115.70456744	–31.67864624	115.70456738	–31.67864663	Method 4.1	0.04
8 to 9	115.70478549	–31.67851310	115.70478602	–31.67851417	Method 4.2	0.13
8 to 9	115.70503749	–31.67838939	115.70504051	–31.67839666	Method 4.2	0.86
8 to 9	115.70530651	–31.67827858	115.70532872	–31.67828468	Method 3.1	2.21
9 to 10	115.70530651	–31.67827858	115.70532872	–31.67828468	Method 3.1	2.21
9 to 10	115.70563116	–31.67901353	115.70564454	–31.67902130	Method 3.2	1.53

7.2.3 Large-Scaled Road Network Selection

The largest evaluated road network translation processed an area of about 1,600 km^2 and included 3,788 MRWA road sections with 23,124 vertices, 2,986 MRWA intersections and 7,533 Landgate features comprising road sections, connectors and roundabouts, as indicated in Figure 7.5.

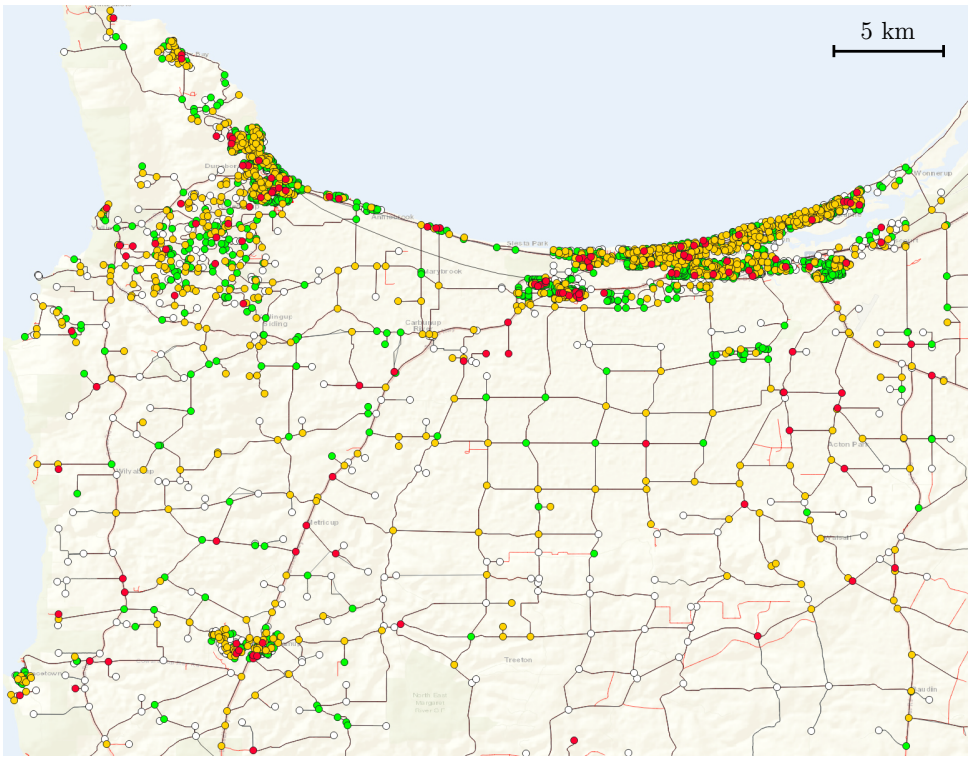


Figure 7.5: The largest evaluated road network selection with red Landgate road sections and in dark colour indicated translated MRWA road sections.

The representation of the translated intersections is categorised by the translated distances and indicated with the colours white, green, orange and red. A white circle represents no translation, a green circle is used for a translation larger than 0 m and less than 2.0 m, an orange circle indicates a translation between 2.0 m and less than 5.0 m, and a red circle shows a translation of 5.0 m or more.

The numeral evaluation of the translated road sections and intersections is given in Table 7.5. The table shows that 23,124 road section features and 2,986 in-

tersection features were processed by the translation algorithm. Of the processed features, 162 road section vertices (≈0.7%) and 570 intersection points (≈19.1%) were not translated. About 64.8% of all road section vertices and 44.6% of all intersection points were translated with a distance between 0 and 2.0 m; approximately 26.2% and 31.8% were translated between 2.00 m and 5.00 m, and about 8.3% and 4.5% of all features were translated with a distance larger or equal to 5.00 m. Translation Method 7 considers untranslated features or features that were translated by Methods 5 and 6.

Table 7.5: Evaluation of the road network translation in Figure 7.5.

Translation Info	Road Sections	Intersections
Total features	23,124	2,986
Not translated	162 (see Method 4.0)	570
Translation > 0 and < 2.0m	14,982	1,331
Translation >= 2.0m and < 5.0m	6,070	951
Translation >= 5.0m	2,072	134
Method 1	-	92
Method 2	-	2,229
Method 3.1	306	-
Method 3.2	5,836	-
Method 4.0	162	-
Method 4.1	4,844	-
Method 4.2	1,173	-
Method 5	-	83
Method 6	-	12
Method 7	244	-

Overall, the translation algorithm is capable of being applied on a larger scale. It can be concluded that the distribution of translation distances is similar compared to the two previous examples of road network selections.

7.2.4 Section Discussion

The road network translation approach was an investigation into the differences of the government-provided road network data within Western Australia, especially considering the comparison of the road centrelines from MRWA and Landgate. The investigation proceeded to analyse the given datasets so that an overall un-

derstanding of the datasets could be conducted. The author expected that the datasets would have common features in the most cases and that the heterogeneous road centreline data would overlap at some locations. The road section evaluation tables of road network selections 1 and 2 often retrieved a road section vertex translation of approximately 1 m or less. The result underlined the initial expectation that the locations of the road network datasets would be very similar in the given datasets. The author discovered the fact that some road sections from MRWA were described with one line string, whereas Landgate had chosen to represent the same road section with two line strings. The evaluation of the large road network selection was only possible with a summarised table of the applied translation methods, as the amount of processed data was too large for a detailed view. This proved that the translation is capable of being applied on larger datasets. Furthermore, the evaluation of the translated MRWA intersections is only relevant for MRWA datasets, as Landgate does not provide an intersection dataset.

7.3 Road Network Conflation

To evaluate the road network conflation approach of this thesis, selected road network assets will be introduced on a map, and then an ontology reasoning will be processed with Pellet. The road network conflation approach is based on Niestroj et al. (2019) and is introduced in this thesis.

7.3.1 Ontology and Semantic Rules Integration

In this section, the road network conflation approach will be evaluated with two examples to prove the functionality of the designed SWRL rules. The first example shows an arbitrarily chosen roundabout intersection with data available from MRWA, Landgate and OSM that describe the same road asset, as indicated in Figure 7.6. The intersection contains three MRWA give way signs to regulate the

driving behaviour, but the MRWA road section dataset contains no information about the roundabout in place.

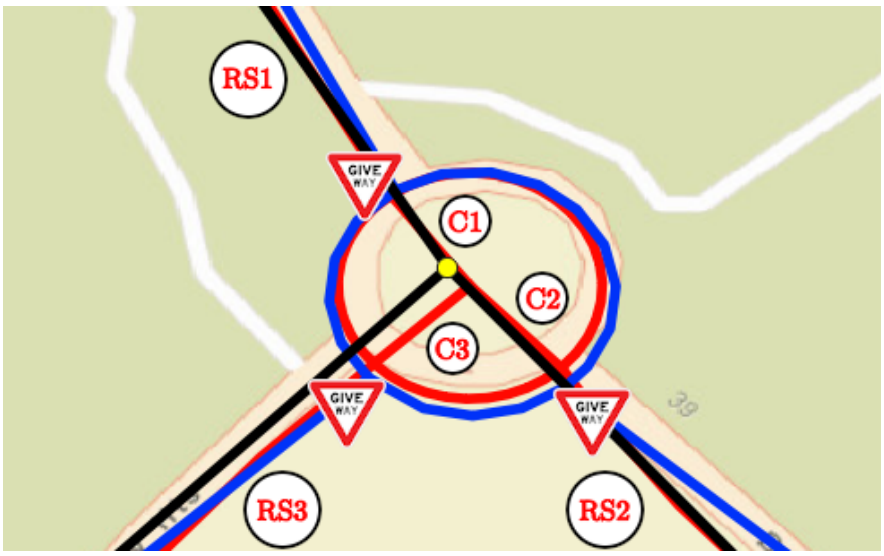


Figure 7.6: The roundabout shows Landgate data road sections (RS), connectors (C) and roundabouts elements with red lines, MRWA road sections with black lines, an MRWA intersection with a yellow circle, MRWA give way signs (red triangles) and OSM road asset data with blue lines.

The data conflation evaluation can be conducted with the property assertion view of Protégé, as shown in Figure 7.7. The object property assertion entries with a white background are part of the ontology and were created during the data creation process. The given information includes the three MRWA road sections that are connected to the intersection, the related point coordinate individuals that describe the geometrical location of the intersection and the intersection individual that was created by the process 'alg:processMrwaIntersections'. The entries with the yellow background can be used to prove that data conflation among the different datasets and data sources is available after reasoning the ontology with Pellet. For example, we can see that two Landgate road sections are connected to the selected intersection of the property assertion view although the intersection consists of three road sections. The reason for this is that the road section to the right of the intersection location is too far to be captured by

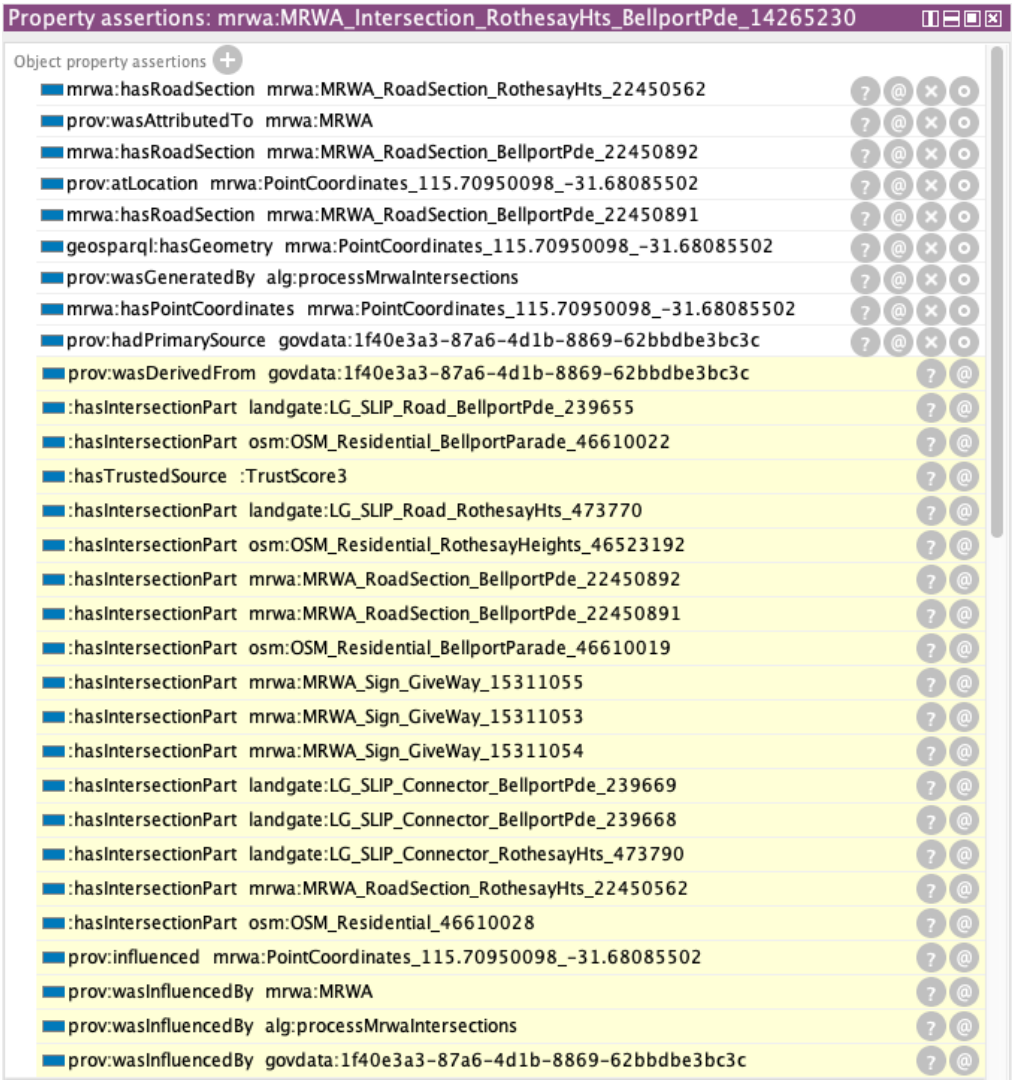


Figure 7.7: Object property assertion view in the software Protégé of the intersection that was indicated with a yellow circle in Figure 7.6. The information with a yellow background was available after reasoning the ontology with Pellet.

the SWRL rule. That behaviour is not wrong, as the intersection is connected with a roundabout to a road section through connectors. It can also be seen that three MRWA give way signs, three MRWA road sections and three OSM road sections are identified as part of the intersection. Another OSM individual ('osm:OSM_Residential_46610028') has been determined at the intersection and refers to a roundabout circle element. The intersection entity receives a trusted source score of three, as defined for the MRWA road asset datasets.

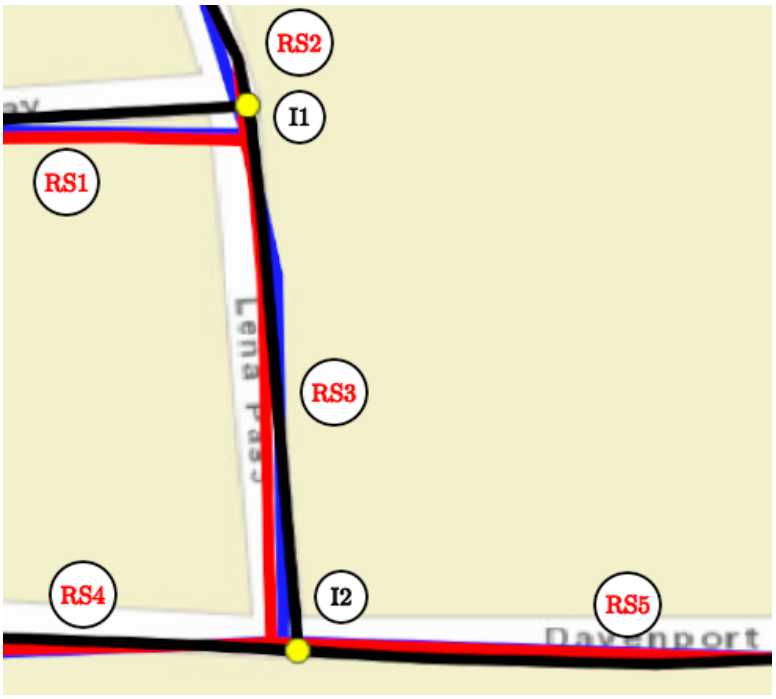


Figure 7.8: Arbitrary road section that shows red Landgate road sections (RS), MRWA road sections with black lines, MRWA intersections (I) with yellow circles and OSM road asset data with blue lines.

The second example to evaluate the road network conflation approach is indicated in Figure 7.8 and shows five road sections and two intersections. An object property assertion view of the Landgate road section 'RS3' is shown in Figure 7.9. By default, the selected individual includes data relations, e.g. that the road section is connected to four Landgate road sections, that it has a 'geosparql:hasGeometry' relation, that the agent is 'landgate:Landgate' and that the data were taken from the dataset 'govdata:85d59328-9eb6-4cdf-b2c0-358a141ccb25'. After the ontology reasoning, the data individual includes conflated information retrieved from the MRWA and OSM datasets. For instance, the selected Landgate individual is on the same road as two MRWA individuals and one OSM individual, and the road section 'RS3' is also connected to two MRWA intersections. The Landgate road section receives a trusted source score of two.

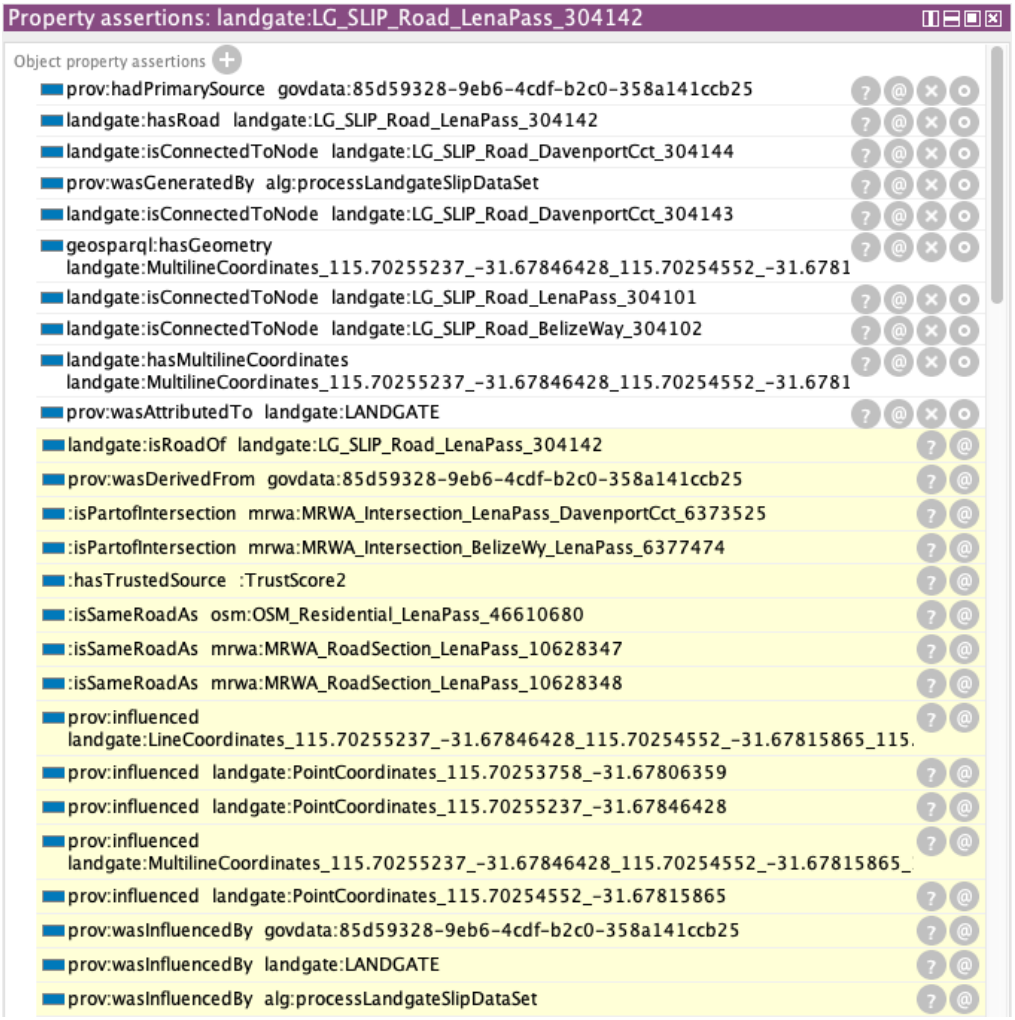


Figure 7.9: Object property assertion view in the software Protégé of the road section 'RS3' that was indicated in Figure 7.8. The information with a yellow background was available after reasoning the ontology with Pellet.

The two examples in this section show that Semantic Web technologies can be used to conflate road asset data from different data sources. The conflated information can be made available after the successful execution of an ontology reasoning process. The application of semantic rules is not limited for use with roundabouts and road sections, as seen in the previous examples. Other road network assets, such as traffic signal sites, road stopping places, pedestrian crossings, bicycle lanes, bridges and tunnels, are outside the scope of this thesis but could be added in future; rules for the other road assets mentioned do not exist

yet, as the conflation approach was conducted as a case study. Once appropriate SWRL rules have been designed, it will be sufficient to reason an ontology once, as the retrieved results can be saved into the ontology.

7.3.2 Reasoning Time and Efficiency

A disadvantage of ontology processing with Protégé and reasoning with Pellet is that they require an enormous amount of computing power. The reasoning time of two road network selections with areas of 0.2 km^2 and 1.3 km^2 was measured and evaluated, as indicated in Table 7.6. The smaller road network was reasoned with a common MacBook Pro computer with 16 GB RAM,[3] and the larger road network was outsourced to a Google Cloud virtual machine with 624 GB RAM. The outsourcing was required, as Pellet reasoning is memory intensive and reasoning with larger datasets is often aborted with an 'out of memory' exception. Pellet has been criticised as requiring huge amounts of memory (Steller & Krishnaswamy, 2008).

The smaller road network was reasoned in approximately 29 s, the larger road network with original coordinates was reasoned in approximately 20:24 min, and the larger road network with translated coordinates was reasoned in 41:30 min.

Table 7.6: Run-time of road network data reasoned with Pellet.

Dataset / Reasoner	**Pellet**
Road network of 0.2 km^2 (original)	29.78 s
Road network of 0.2 km^2 (translated)	29.18 s
Road network of 1.3 km^2 (original)	1,224.89 s (≈20:24 min)
Road network of 1.3 km^2 (translated)	2,490.82 s (≈41:30 min)

Two further Pellet reasoning trials were conducted in relation to a road network selection of approximately 1,279 km,[2] as shown in Figure 7.10 with blue OSM road sections, black MRWA road sections, red Landgate road sections, yellow MRWA intersections and red MRWA regulatory signs. The first ontology

[3] Protégé reasons an ontology with a single thread, meaning that only one CPU core is being employed. Thus, the CPU performance of each machine is not relevant enough to be mentioned.

reasoning trial was processed with a Google Cloud 'm1-ultramem-80' computer with 80 CPU cores and 1,922 GB of memory. The test included the processing of the newly designed SWRL rules introduced in Figure 5.2 and was aborted after 22 hours.

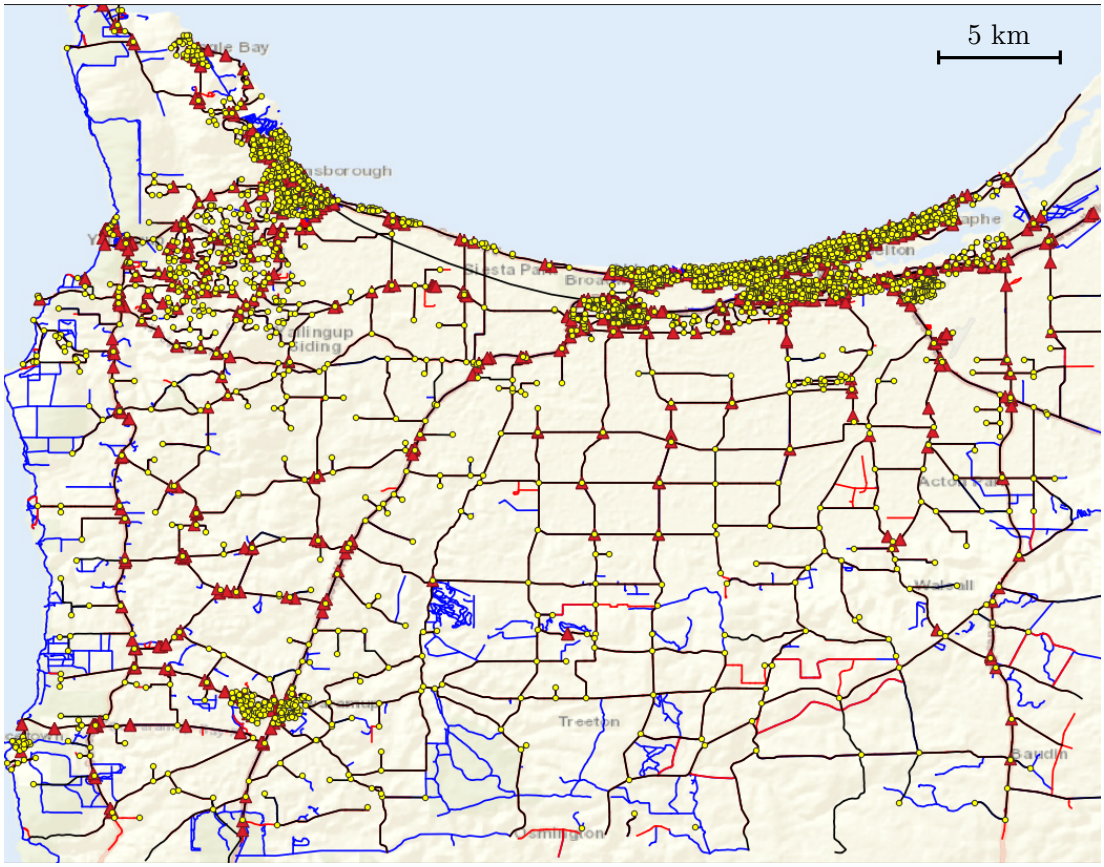


Figure 7.10: Largest road network selection approached with the Pellet reasoner.

The second ontology reasoning trial of the road network selection in Figure 7.10 was conducted with an Amazon AWS 'x1e.8xlarge' computer with 32 CPU cores and 976 GB of memory. The reasoning excluded rules 7–11, introduced earlier, for the comparison of geographic coordinates to mitigate the computing power requirements, as with these SWRL rules, each coordinate would be compared with another. Although the reasoning excluded a huge amount of the workload, the ontology reasoning crashed with an out-of-memory exception after about 26 hours. In both cases, the reasoner was able to process the seman-

tic rules for class hierarchy, object/data property hierarchy, class assertions and object property assertions but was not able to process through the same individuals' rules. It has to be mentioned that in a separate test, after deleting all the SWRL rules that contained the 'same as' attribute, the reasoner was still unable to complete the processing.

7.3.3 Section Discussion

The road network conflation approach tried to conflate road network data that mean the same thing from MRWA, Landgate and OSM. The author's expectation was to underline the efficiency of Semantic Web technologies as a first step towards data harmonisation of Western Australia's road network data. The author's expectations were fulfilled, as by selecting a certain entity in Protégé after an ontology reasoning process, all road assets with the same meaning were identified. In addition, it was even possible to determine neighbouring assets, such as the connected road sections of an intersection and the road signs at an intersection. At an early stage of this research, the author was motivated to verify the position of road signs in relation to the guidelines from MRWA about the location of road assets, as publishedin Niestroj et al. (2018). However, after sharing the research motivation with MRWA, MRWA clarified that the surveyed locations of operated road signs were not accurate, as MRWA only keeps track of where a road sign is placed (e.g. road sign x at intersection y) but not its exact geographic location. Thus, the investigation to verify the location of road assets against the official positioning guidelines was not further processed. With regard to the reasoning time the author assumed that it would be intensive, as the ontology reasoner compared each data entry against every other entry. Although the author outsourced the computing power to high-end cloud computers, the reasoning process was still not successful on a very large scale.

7.4 Route Planning

A visualisation of the route planner with the road network selection used for the evaluation in this thesis is shown in Figure 7.11. The map indicates the layers 'traffic signals sites', 'road edges', 'road nodes', 'power line edges', 'rail crossings' and 'road stopping places' in the initial view after the route planner has been opened. Stop and give way signs are not loaded in the initial view to prevent an overfilled map representation.

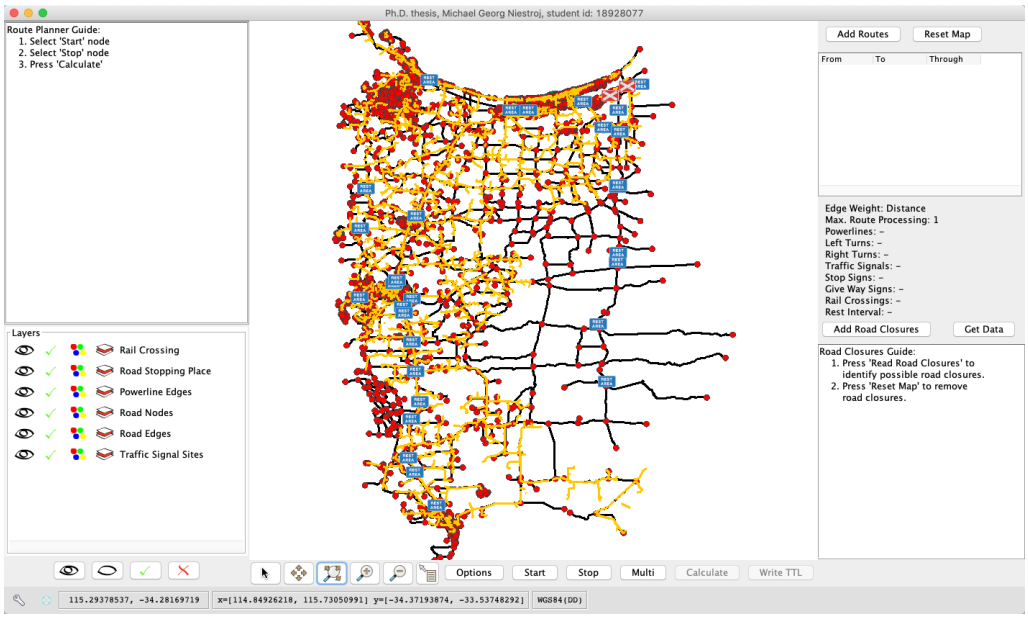


Figure 7.11: Road network selection used for the evaluation of the route planner.

7.4.1 Inner City Route

The first route evaluated was an inner city route with a length of approximately 3.0 km, as shown in Figure 7.12. The route was processed with Google Maps[4] as a reference and shows a track from the top right of the map to the bottom left of the map through six different roads, namely Dorset St, Bay View St, Bussel Hwy, Fairway Dr, Settlers Gate and Frances Louisa St. In comparison, the same route

[4] Source: `https://www.google.com/maps/dir/-33.653924,115.3241257/-33.6636759,115.3132378/@-33.6592249,115.3243101,16z/am=t/data=!3m1!4b1!4m2!4m1!3e0`

was planned with the route planner of this thesis considering the shortest route by distance in Figure 7.13 and considering the shortest route by travel time in Figure 7.14. At the beginning of the route, both routes showed a different path compared to the route from Google. The approach of this thesis navigated first to the west and then to the east, while the path planned by Google navigated first to the east and then to the west.

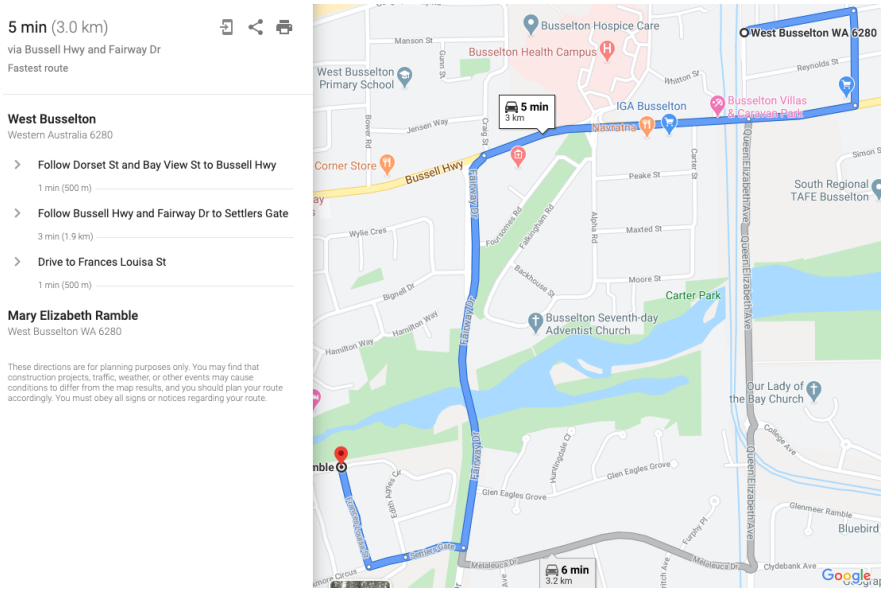


Figure 7.12: Route from West Busselton to Mary Elizabeth Ramble planned with Google Maps.

The route remaining in Figure 7.14 was the same as the Google route, and in Figure 7.13, the route planner processed a slightly different route in the middle of the route resulting in a shorter distance. Overall, the resulting distances were very similar. Google estimated a distance of 3.00 km, and the route planner of this thesis calculated distances of 2.93 km for the shortest route and 2.96 km for the fastest route. It has to be noted that Google predicted a travel time of approximately 5 min, whereas the designed route planner of this thesis calculated 3:19 min and 3:21 min without considering left turns, right turns, traffic signals or regulatory signs.

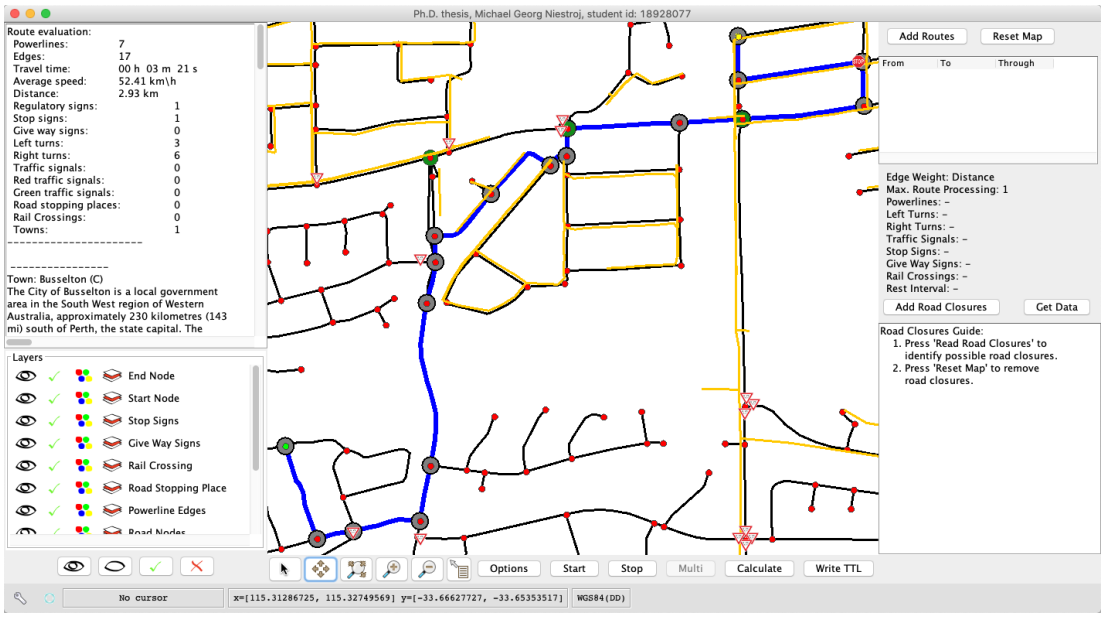


Figure 7.13: Route from West Busselton to Mary Elizabeth Ramble planned with the route planner of this thesis considering the shortest route by distance.

Another possible best route was identified by the route planner of this thesis with two activated red traffic signals on the track, as shown in Figure 7.15; the alternate track was also highlighted by Google as a possible route. The route took a left turn at a red traffic signal site and followed the track to the west with a total length of about 3.19 km. The estimated travel time was about 4:17 min and included a waiting period of 32 s at the red traffic lights.

The route was also processed with cost factors that were taken into account, as indicated in Table 7.7. Each of the tested configurations resulted in one of the routes indicated in Figure 7.13, Figure 7.14 and Figure 7.15. Road stopping places and rail crossings were not considered in this example, as neither constraint was present within the borders of the start point and the destination. A reversed route was tested and showed the same reversed results as the original route. Sometimes, reversed routes will not share the same road sections due to the placement of regulatory signs, as regulatory signs are evaluated by the route planner in the driving direction.

Figure 7.14: Route from West Busselton to Mary Elizabeth Ramble planned with the route planner of this thesis considering the shortest route by travel time.

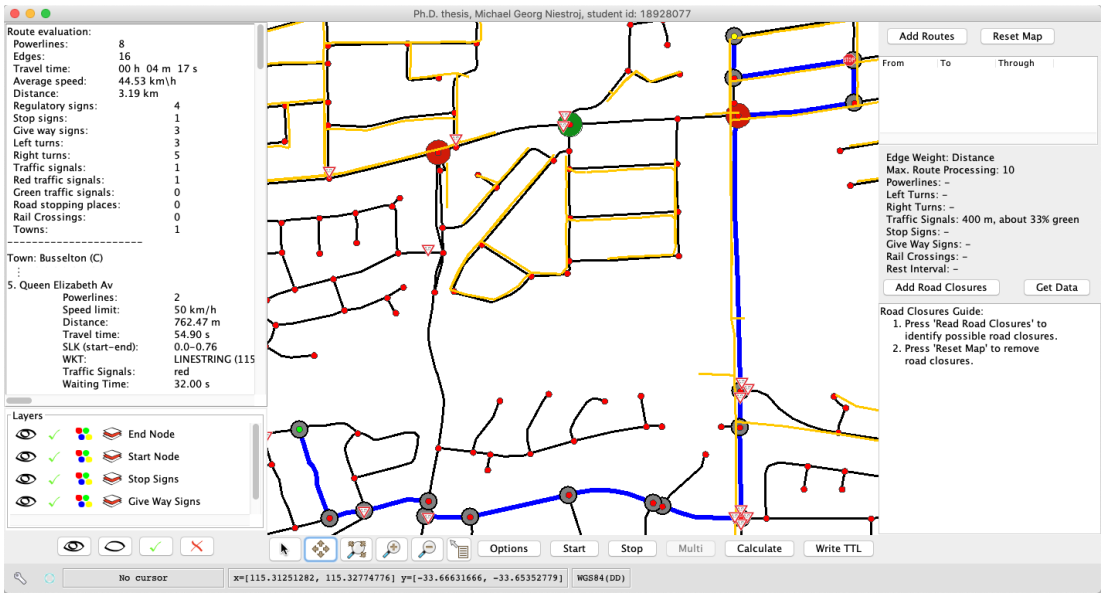


Figure 7.15: Route from West Busselton to Mary Elizabeth Ramble planned with the route planner of this thesis considering the shortest route by distance with an activated extra red traffic signal cost of 400 m.

Table 7.7: Route evaluation with configurable constraints from West Busselton to Mary Elizabeth Ramble.

Edge Weight	Max. RP	Power Lines	Left Turns	Right Turns	Traffic Signals	Stop Signs	Give Way Signs	Figure
Distance	1	-	-	-	-	-	-	5.14
Travel Time	1	-	-	-	-	-	-	5.15
Distance	1	500 m	-	-	-	-	-	5.15
Travel Time	1	40 s	-	-	-	-	-	5.15
Distance	10	-	25 m	-	-	-	-	5.15
Travel Time	10	-	5 s	-	-	-	-	5.15
Distance	10	-	-	150 m	-	-	-	5.15
Travel Time	10	-	-	20 s	-	-	-	5.15
Distance	10	-	-	-	400 m	-	-	5.16
Travel Time	10	-	-	-	32 s	-	-	5.16
Distance	10	-	-	-	-	25 m	-	5.14
Travel Time	10	-	-	-	-	5 s	-	5.15
Distance	10	-	-	-	-	-	10m	5.14
Travel Time	10	-	-	-	-	-	2 s	5.15
Distance	10	500 m	25 m	150 m	-	25 m	10m	5.15
Travel Time	10	40 s	5 s	20 s	-	5 s	2 s	5.15
Distance	10	500 m	25 m	150 m	400 m	25 m	10m	5.16
Travel Time	10	40 s	5 s	20 s	32 s	5 s	2 s	5.16

7.4.2 Large-Scaled Route

The second route that was evaluated was an approximately 111–129 km drive from Pericles Street in East Augusta to Quedjinup Drive in Quedjinup. Two reference routes are indicated in Figure 7.16, with the left route planned by Google Maps[5] and the right route planned by Bing Maps.[6]

The same route was planned with the route planner of this thesis considering the shortest route by distance in Figure 7.18. As shown, the route planner identified the shortest route as 105.7 km, which was approximately 6 km shorter than the shortest reference route. With the approach of this thesis, although the identified route was shorter compared to the suggested routes from Google and Bing, the travel time was 1:59 h, about 23–38 min slower. A summary of the large-scale route is indicated in Table 7.17.

The reason for the longer travel time is related to the MRWA speed limit dataset, as visualised with black, light brown and red line strings in Figure 7.19. The black line strings are visible road sections with no available speed limit

[5] Source: `https://www.google.com/maps/dir/Pericles+St,+East+Augusta+WA+6290/Quedjinup+Dr,+Quedjinup+WA+6281/`

[6] Source: `https://www.bing.com/maps?osid=6a40f7a7-546e-4f60-83c7-41e656cdba5e`

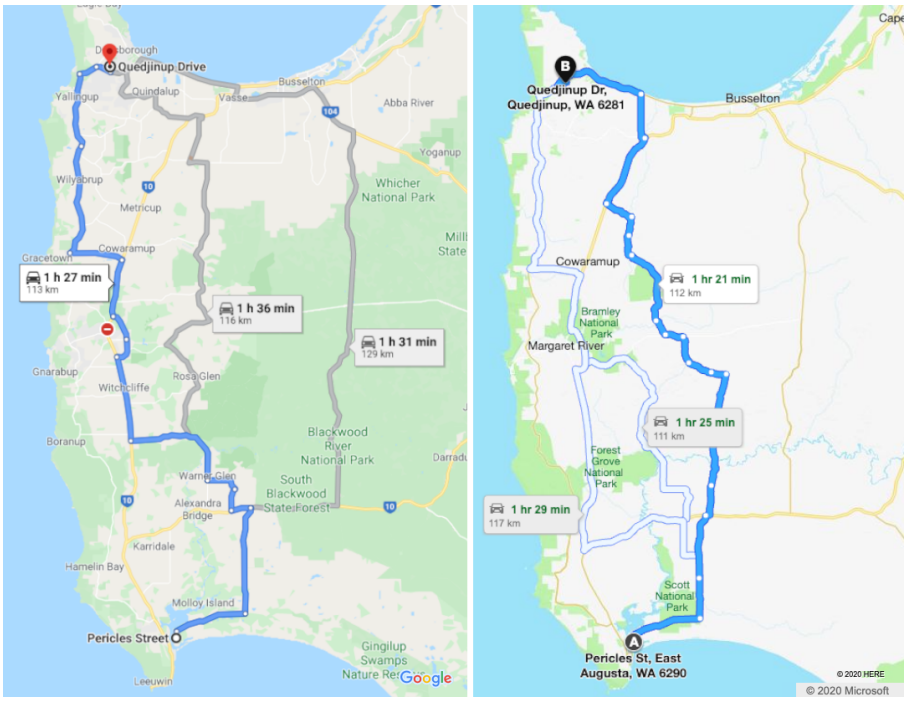


Figure 7.16: An indicated route from Pericles Street in East Augusta to Quedjinup Drive in Quedjinup planned with Google Maps (left) and Bing Maps (right).

Route	Route Planner		Google Maps		Bing Maps	
	Distance	Travel Time	Distance	Travel Time	Distance	Travel Time
Shortest	105.7 km	1:59 h	113.0 km	1:27 h	111.0 km	1:25 h
Fastest	131.16 km	1:36 h	113.0 km	1:27 h	112.0 km	1:21 h

Figure 7.17: Summary of the route planner evaluation of the large-scale route.

information; the default travel speed was set by the route planner to 50 km/h. The red line strings are common speed limit values from 10 km/h to 110 km/h in steps of 10 km/h. The light brown line strings contain the speed limit information '50km/h applies in built-up areas or 110 km/h outside built-up areas', which was set to a value of 50 km/h as the route planner did not have the data required to distinguish between built-up areas and outside built-up areas. For this particular case, the test showed that changing the default value of the light brown line strings from 50 km/h to 110 km/h changed the travel time from about 1:59 h to about 1:20 h, which was much closer to the results of Google and Bing. The limitation

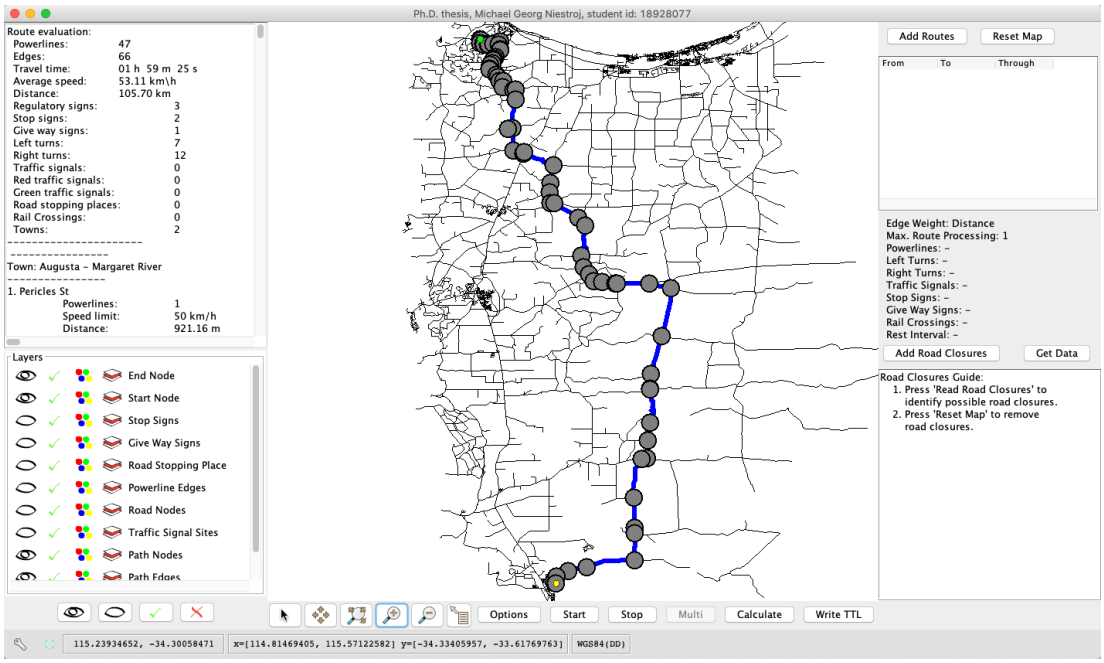


Figure 7.18: An indicated route from Pericles Street in East Augusta to Quedjinup Drive in Quedjinup planned with the route planner of this thesis considering the shortest route by distance.

of a lack of accurate information in the speed limit dataset was identified at the very end of this thesis. In future research activities, additional datasets, such as building locations, population densities and forest regions, could be considered for an investigation into the identification of built-up areas based on semantic rules.

Table 7.8: Route evaluation with configurable constraints from Pericles Street in East Augusta to Quedjinup Drive in Quedjinup planned with the shortest distance for the activation of constraints.

Edges	Max. RP	Power Lines		Left Turns		Right Turns		Traffic Signals		Stop Signs		Give Way Signs		Rail Crossings		Road Stopping Places		Distance	Figure
66	1	-	47	-	7	-	12	-	0	-	2	-	3	-	0	-	0	105.70 km	5.18
69	1	500 m	30	-	10	-	10	-	0	-	3	-	6	-	0	-	1	108.76 km	5.20 a
66	1	-	47	25 m	7	-	12	-	0	-	2	-	1	-	0	-	0	105.87 km	5.20 b
66	10	-	47	-	7	150 m	12	-	0	-	2	-	1	-	0	-	0	107.50 km	5.20 c
66	10	-	47	-	7	-	12	400 m	0	-	2	-	3	-	0	-	0	105.70 km	5.18
66	10	-	47	-	7	-	12	-	0	25 m	2	-	3	-	0	-	0	105.70 km	5.18
66	10	-	47	-	7	-	12	-	0	-	2	10 m	3	-	0	-	0	105.70 km	5.18
66	1	-	47	-	7	-	12	-	0	-	2	-	3	500 m	0	-	0	105.70 km	5.18
69	10	500 m	30	25 m	10	150 m	9	400 m	0	25 m	3	10 m	3	500 m	0	-	1	110.36 km	5.20 d
108	1	-	73	-	12	-	15	-	0	-	1	-	2	-	0	20-60 km	3	106.49 km	5.20 e
80	10	500 m	42	25 m	7	150 m	10	400 m	0	25 m	0	10 m	3	500 m	0	20-60 km	1	110.33 km	5.20 f

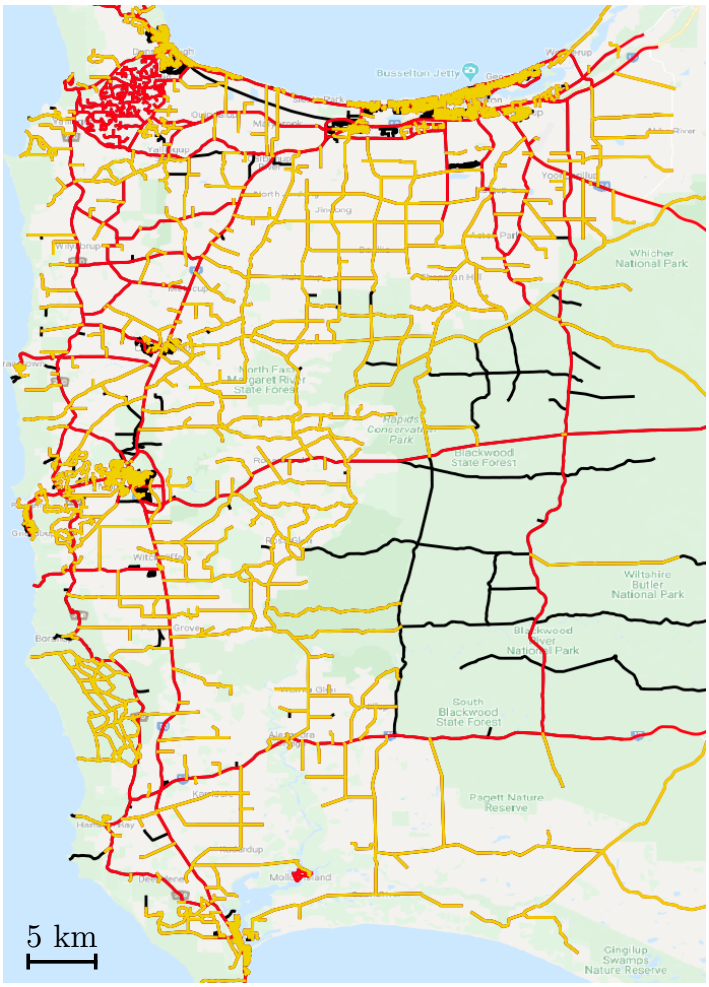


Figure 7.19: The map indicates MRWA road sections without speed limits in black, road sections with speed limits between 10 km/h and 110 km/h in red, and speed limits that are either 50 km/h or 110 km/h in light brown.

As the exact speed limit could not be identified with the given datasets, the route evaluation only considered the shortest route by distance with the activation of the constraints 'overhead power lines', 'left turns', 'right turns', 'traffic signal sites', 'stop signs', 'give way signs', 'rail crossings' and 'road stopping places', as indicated in Table 7.8.

For the evaluation of the last row of the table, all previously mentioned constraints were activated. It can be seen that each constraint column was subdivided into a left column and a right column. The left column showed either

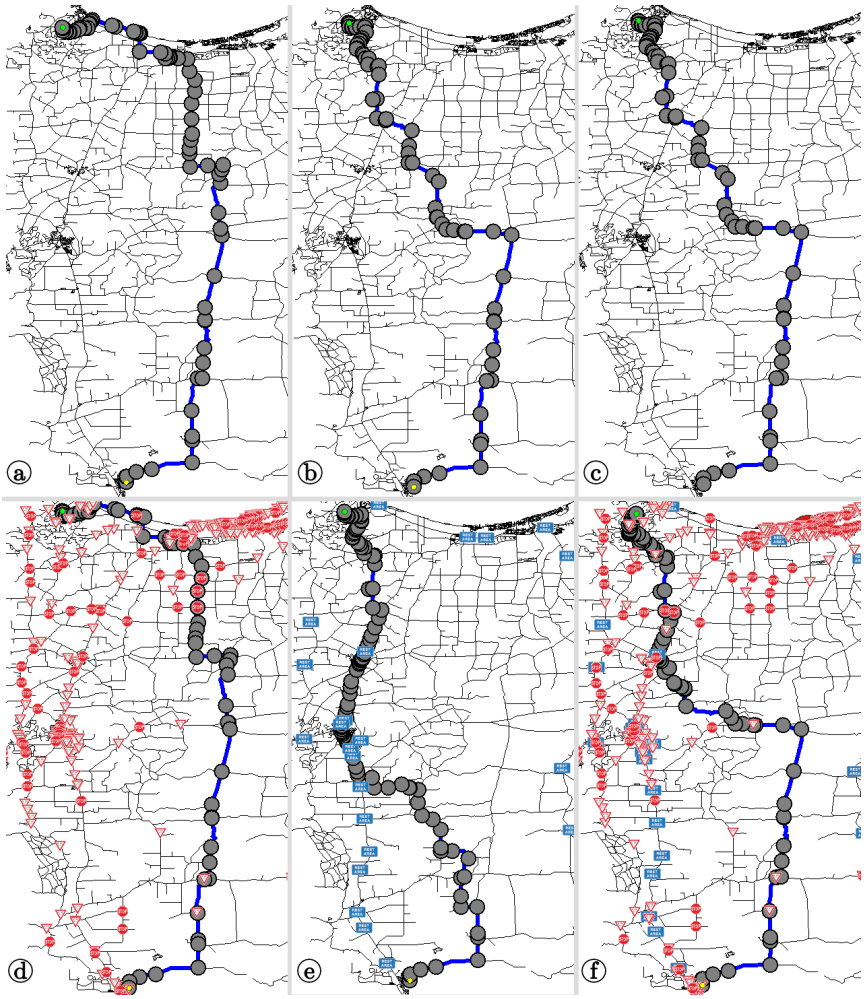


Figure 7.20: Results from the route planner with the configurations as indicated in Table 7.8. Figures a–e are used to evaluate the configurable constraints, and Figure f is related to a route with road stopping places.

a value that would be added to an edge weight or a dash sign ('-') if not used, and the right column indicated the number of constraint occurrences. For instance, the second row processed a route planned with an activated power lines consideration, whereby each crossing overhead power line added 500 m to an edge weight. The results of the planned route are indicated in Figure 7.20 a) and can be summarised with 69 edges, 30 overhead power lines, 10 left turns, 10 right turns, zero traffic signals, three stop signs, six give way signs, zero rail crossings and one road stopping place. The route was only processed once and resulted

in a distance of about 108.7 km. For the route of Figure 7.20 f) all configurable edge weight costs and road stopping places in an interval of 20.0–60.0 km were activated. The results can be summarised as a route with 73 edges, 108 crossing overhead power lines, one stop sign, two give way signs, 12 left turns, 15 right turns, zero rail crossings and three road stopping places. The travel distance with activated road stopping places was about 106.5 km.

7.4.3 Route with Overhead Power Lines

In this section, the evaluation of overhead power lines was conducted with a processed route from a yellow start node to a green destination node, as indicated in Figure 7.21. After setting the power lines extra weight to a value of 3,100 m per crossing overhead power line in a road section, the route planer calculated the best route between the two nodes as a 15.57-km distance with the occurrence of four overhead power lines. The evaluation also showed three left turns, one right turn and one stop sign. Though not visualised here, an independent route planner configuration with overhead power line extra costs of less than 3,100 m resulted in a straight route between the two nodes with a distance of 6.38 km, seven crossing overhead power lines and one stop sign.

It has to be noted that completely avoiding crossing overhead power lines is often not possible due to their frequent presence across the road network.

7.4.4 Route with Rail Crossings

The avoidance of rail crossings with the route planer was evaluated in this section. Figure 7.22 and Figure 7.23 show the same road network selection. In Figure 7.22, the route planning was conducted with no extra costs, which resulted in a route of 3.52 km between a start and a destination with one rail crossing on the route. In Figure 7.23, an extra weight of 1,500 m per rail crossing was considered by the route planner. The result indicated a route with a distance of 4.97 km without a rail crossing in place; as highlighted in Figure 7.23 and the rail crossing belonged

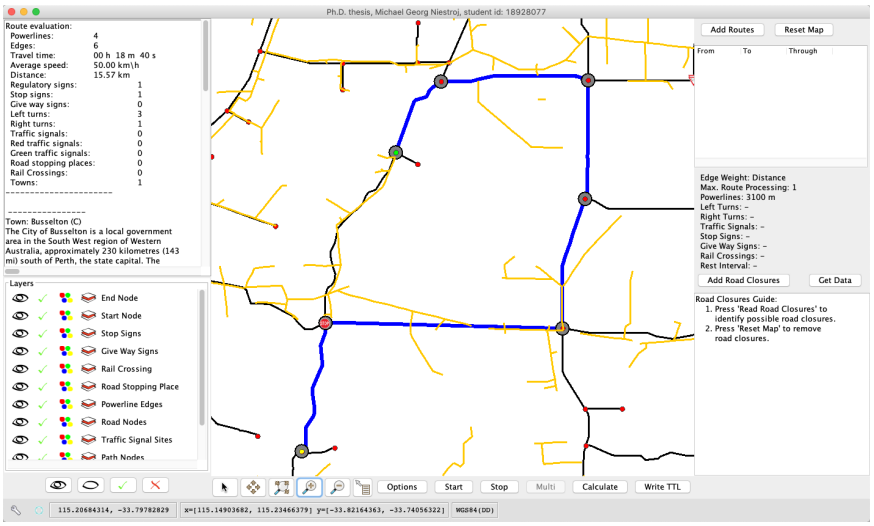


Figure 7.21: Route planning evaluation to avoid passing under overhead power lines.

to a side road that was not part of the route. In conclusion, once again, the route planner indicated the capability to adopt the constraints of external datasets for seamless adoption in the route planner.

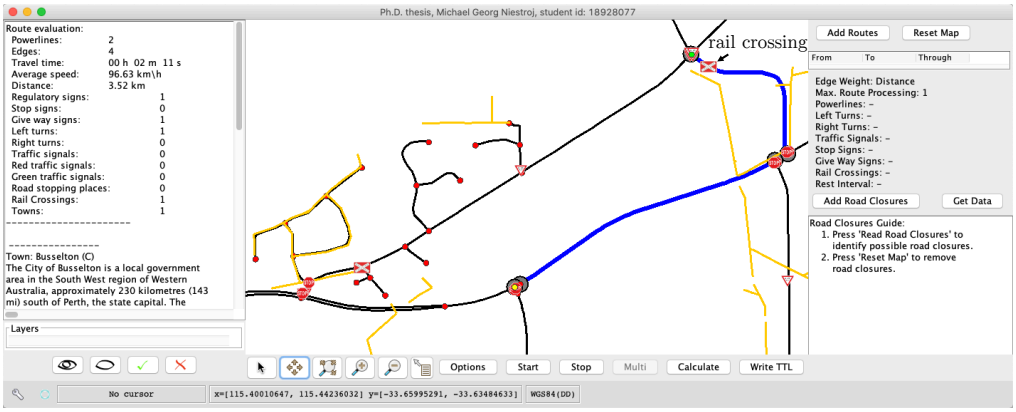


Figure 7.22: Route planning evaluation with driving over rail crossings.

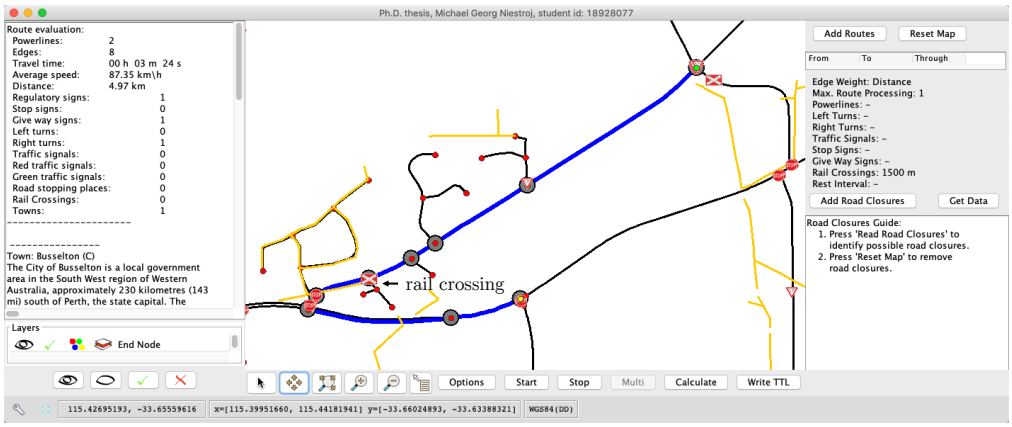


Figure 7.23: Route planning evaluation to avoid driving over rail crossings.

7.4.5 Route with Closed Roads

This section evaluated the functionality of considering closed roads with the route planner, as indicated in Figure 7.24. A closed road dataset was quickly downloaded with a click on the button 'get data' from the Western Australian data portal.[7] Then, the data was loaded into the route planner, as visualised with the pink line strings, and a textual summary of the closed roads was loaded into the panel at the bottom-right of the route planner. Therefore, a start node and an end node were selected in such a manner that the shortest possible route was affected by a closed road. As we can see, after the interactive map inspection, the route planner identified the shortest possible detour to reach the target. If a target could not be reached, then the route planner showed a message to the user that a route could not be processed. To conclude the evaluation of closed roads, the adoption of external datasets that were not available as ontologies was sucessfully integrated in this section. That supported the aim of data harmonisation in a positive way, as it proved that the data migration of ontologies with simple datasets is possible.

[7] `https://data.wa.gov.au`

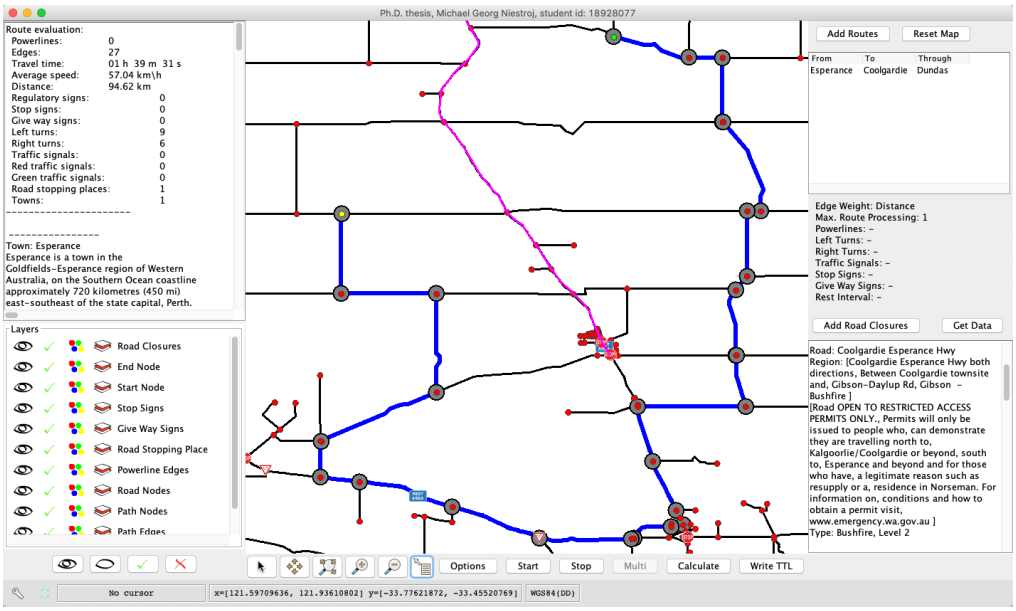


Figure 7.24: Route planning with a closed road that prevents the route planner from taking a shorter route to reach the target.

7.4.6 Write and Read a Planned Route

This section evaluated the route planner functionality of writing a route into a route ontology and reading the route back from the ontology into the route planner. This part of the research was conducted to analyse the data handling efficiency of Semantic Web data. The route of the previous section was used, as seen in Figure 7.24. With the button 'write TTL', the route was written in the RDF format into the route ontology. A screenshot was taken of the first two route individuals written (see Figure 7.25). The button 'add routes' migrated the read route ontology back into the route planner. After the route selection, the selected route was successfully displayed on the map, as indicated in Figure 7.26. Then, the route evaluation was processed and summarised with the following information: number of overhead power lines, number of edges, travel time, average travel speed, route distance, and number of regulatory signs, stop signs, give way signs, left turns, right turns, traffic signals, red traffic signals, green traffic signals, road stopping places and towns. Below that, we can see that the

route started in the town Esperance. The description of the town was retrieved on the fly by the route planner with a SPARQL query from the DBpedia SPARQL endpoint. With the evaluation of a reading and writing cycle of ontologies for the road network, the author proved that heterogenous road asset ontologies can be adopted in common data handling processes. Overall, that enabled seamless data sharing among various stakeholders, which could be a step towards road asset data harmonisation for Australia's road networks.

```
4828 route:Route_FROM_DwyerRd_10732184_TO_ScaddanRd_10731345_000001 rdf:type owl:NamedIndividual ;
4829  a route:Edge, prov:Entity, route:KeyNode, owl:Thing ;
4830  geosparql:asWKT "POINT (121.65486857 -33.57153091)"^^geosparql:wktLiteral ;
4831  route:hasNextRoadEdge mrwa:MRWA_RoadSection_BrowningRd_10731310 ;
4832  route:hasNextRoute route:Route_FROM_DwyerRd_10732184_TO_ScaddanRd_10731345_000002 ;
4833  route:powerlines 0 ;
4834  geosparql:hasGeometry mrwa:LineCoordinates_121.65453871_-33.61961503_121.65454713_-33.57296503_12
4835  prov:hadPrimarySource mrwa:MRWA_RoadSection_DwyerRd_10732184 ;
4836  prov:wasGeneratedBy alg:processRouteGenerator ;
4837  prov:wasAttributedTo mrwa:MRWA .
4838 route:Route_FROM_DwyerRd_10732184_TO_ScaddanRd_10731345_000002 rdf:type owl:NamedIndividual ;
4839  a route:Edge, prov:Entity, owl:Thing ;
4840  route:hasPreviousRoadEdge mrwa:MRWA_RoadSection_DwyerRd_10732184 ;
4841  route:hasPreviousRoute route:Route_FROM_DwyerRd_10732184_TO_ScaddanRd_10731345_000001 ;
4842  route:hasNextRoadEdge mrwa:MRWA_RoadSection_HockeysRd_10731501 ;
4843  route:hasNextRoute route:Route_FROM_DwyerRd_10732184_TO_ScaddanRd_10731345_000003 ;
4844  route:powerlines 0 ;
4845  geosparql:hasGeometry mrwa:LineCoordinates_121.71214339_-33.61962958_121.65453871_-33.61961503 ;
4846  prov:hadPrimarySource mrwa:MRWA_RoadSection_BrowningRd_10731310 ;
4847  prov:wasGeneratedBy alg:processRouteGenerator ;
4848  prov:wasAttributedTo mrwa:MRWA .
```

Figure 7.25: The screenshot was taken from the first two written route individuals of the route in Figure 7.24.

7.4.7 Section Discussion

An investigation into a route planner driven by Semantic Web technologies was conducted by the author to integrate the research findings into a real-world application. The author's expectation was to provide a route planner that delivers the same results as commonly used route planners provided by well-known organisations, such as Google and Microsoft Bing. A further expectation was that the use of Semantic Web technologies would activate a simple interface for the integration of new datasets and constraints. The route planner showed (for planned routes by distance) good results compared to the routes planned by Google and Microsoft Bing. Route planning considering the allowed travel speed was not possible to

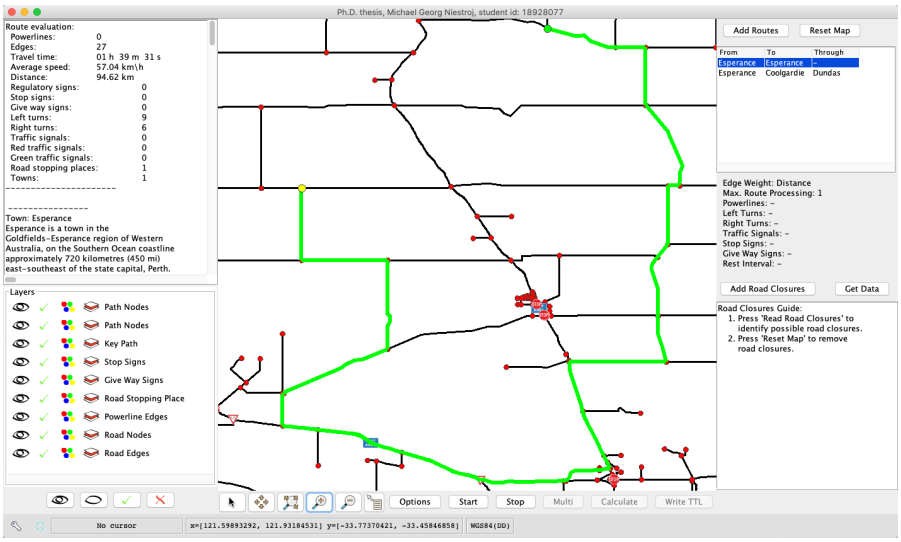


Figure 7.26: Route loaded from the ontology after the selection of a saved route in the top right of the panel.

conduct, as gaps in the MRWA speed limit datasets were identified. It would be interesting to discover how Google and Microsoft Bing are dealing with the data gaps. A possible approach could include collecting and evaluating live travel speed data from users to the update speed limit information in poor datasets. It is not clear for the author whether a specific data manipulation approach is taken into account by Google and Microsoft Bing, as such information cannot be located on the web. At the end of this research, the rail crossings dataset was used in the route planner as a constraint, as rail crossings can also contain overhead power lines. The rail crossings were integrated into the developed framework within a few hours, meaning that a new ontology for rail crossings was created, the rail crossings GeoJSON dataset was migrated into the rail crossings ontology, and the route planner was updated appropriately. This showed that the addition of new constraints into the newly developed route planner framework could be processed efficiently.

7.5 Chapter Summary

The road network translation approach was evaluated with three different road network selections. The first two road networks were used to compare the interactive road network data from Landgate and MRWA on a map. The translation results indicated that it is possible to translate a road asset dataset to the shape of another dataset so that the overall data projection will share the same shape. The third road network selection covered an area of about 1,600 km^2. The translation analysis showed that in most cases, the road section vertices and intersection points were translated with a distance of less than 2 m. From a scientific point of view, the analysis of the Western Australian road network datasets revealed that the publicly available datasets provided by local governments can be used for the overall description of road network assets. However, if the research requires the exact locations of road network assets, then the public road asset datasets cannot be taken into account, as uncertainties were identified, e.g. road signs and overhead power lines were allocated on the wrong road side and, in some cases, a dataset indicated road assets had not been placed.

The evaluation of the road network conflation approach successfully demonstrated that the use of semantic rules is a possible technology for road asset data conflation. After the semantic rules, an ontology reasoner can be used to retrieve reasoned ontology information based on these newly created semantic rules. It is further possible to save the reasoned information in an ontology, which is very effective, as a reasoning process can be processing intensive. Overall, the results were very good, as the data analysis showed a clear connection of the heterogeneous datasets. The scientific achievement of processing road asset data harmonisation using Semantic Web technologies was fulfilled and could be expanded in further research.

The route planner was tested with a small and large scaled road network dataset that was read from an ontology. Several tests to plan a shortest route showed that the approach of this thesis is comparable to route planners by Google

and Bing. The advantage of the route planner, is that due the application of Semantic Web technologies various constraints, e.g. overhead power lines, traffic signals, regulatory sings, rail crossings and road stopping places, can be taken into account. Other not-yet-defined constraints can be integrated with minimal programming effort due to the advantage of Semantic Web technologies.

The route planning was applicable for an inner city route and failed on a long-range route of about 120 km due to a dataset without a well-defined speed limit. The route planner was able to find the shortest possible detour for routes that were affected by overhead power lines, rail crossings and closed roads. Current closed road data was downloaded on the fly from the Western Australian data portal, and town information was retrieved live from DBpedia. From a research perspective in this thesis, the route planner demonstrated very well that integrating newly created ontologies into real-world applications can be done seamlessly in a timely manner.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In conclusion, this thesis dealt with the identification, analysis and utilisation of suitable Semantic Web technologies for road networks. Literature that considered ontology-based road asset data, road-related applications and route planning driven by Semantic Web technologies was reviewed. So-called ITSs are often used in combination with Semantic Web technologies, which use sensors installed at road environments (e.g. bridges, traffic signal sites and kilometre points) that communicate through wireless technologies with transport authorities to improve road safety. A few ontology-based road network applications that use semantic rules to reason about various traffic and travel situations, such as public transportation and route planning for vehicles, exist in the literature. Although the issue of accidents due to contact with heavy vehicles and overhead power lines is well known, no information was found about a route planner that considers overhead power lines as a constraint. To conclude with the research objectives, the next list will address each research objective separately:

- Objective 1: the Semantic Web is based on technologies that can be simplified and represented in a layer-based stack, such as OWL, RDF-S, SWRL,

PROV-O and SPARQL. The application of technologies from the Semantic Web stack enabled the design of the road network and route ontologies.

- Objective 2: a road network data model was conceptualised for each of the employed datasets 'road sections', 'overhead power lines', 'intersections', 'regulatory signs', 'traffic signal sites', 'speed limits', 'rail crossings' and 'road stopping places'. The retrieved information was then used to design MRWA, Landgate, Western Power and OSM ontologies with the support of PROV-O, which enabled us to keep track of a data life cycle.

- Objective 3: the data integration was performed with a Python script that read the various datasets in the GeoJSON format, processed the information for each dataset entry, and wrote the ontology individuals in the RDF/Turtle format. It was also possible to process either all heterogeneous MRWA, all Landgate or all OSM dataset selections at once, which was very efficient.

- Objective 4: a set of SWRL rules was defined to conflate road network asset data, to assign a trust score for road asset data sources and to find information that means the same thing within the data from different data sources. The rules were successfully evaluated with sample road assets. A disadvantage of the semantic rules reasoning was the limitation of the road network selection's size due to the vast computing power requirements. However, an advantage of the semantic rules reasoning was the efficient integration and adoption of new datasets from heterogeneous data sources. A suggestion for further research activities includes avoiding Protégé as an ontology management tools for big data ontologies and processing the ontologies directly in a custom-designed environment developed with Python, Java or C. This may enable scaling the amount of processed ontology data, as custom development commonly allows for full control of data processing.

- Objective 5: road network selections were processed with a Python script that applied a designed translation algorithm to MRWA road sections and intersections so that the overall road centreline representation was harmonised with a reference dataset. The results were evaluated and showed that, in most cases, a translation occurred within a distance of less than 2.0 m, though translations of more than 5.0 m were also present.

- Objective 6: the route planner was able to plan routes with the optional consideration of overhead power lines, left/right turns, regulatory signs, traffic signal sites, rail crossings and road stopping places as constraints. The planned routes delivered similar results compared to the route planners from Google and Bing considering the shortest distance. Route planning by the shortest travel time was only possible in a built-up area, as the speed limit data contained speed limit information that was not directly processable, e.g. '50 km/h or 110 km/h' for roads outside of built-up areas.

To conclude with the thesis statement, the results above prove that Semantic Web technologies can be used to bridge the gap of the lack of unified data standards, tools and data formats of the Australian road and transport authorities. With ontologies and semantic rules, data with the same meaning can be seamlessly identified and processed with external applications, as was shown by the case study of a route planner. A Semantic-Web-based data conflation approach could be used as a first step towards data harmonisation. However, it has to be considered that the application of Semantic Web technologies with big datasets requires a huge amount of computing power, especially with the consideration of ontology reasoning. Thus, an implementation to manage the Australian road network asset data is not suggested, as further research is required to scale the findings for big data.

8.2 Future Work

Further research could include an investigation into big data ontology processing, using these research findings as a solid base. A successful extension for big data could enable the extension of the road asset ontology processing for the entire Australian road network. For instance, a cluster-based reasoning approach could enable the processing of data in data packages instead of the processing of a dataset immediately. That could result in an effective processing approach, as irrelevant information of certain areas could be excluded.

This thesis could also be used as a foundation for various research activities that include road network data and road asset management driven by Semantic Web technologies. That means new road asset datasets outside of Western Australia could be processed into ontologies with minor changes to the current metadata definition. For instance, once the whole Australian road network has been migrated into ontologies, the route planner could be upgraded with much less effort in order to handle road network graph creation based on multiple data sources. In the future, other live data, e.g. weather and traffic data, could be integrated in addition to the currently closed road and DBpedia suburb information datasets. Another possible outlook includes the migration of the route planner into a Protégé plugin so that other researchers can experiment with their local road network data.

Bibliography

P. Agarwal, et al. (2013). 'Approximate Incremental Big-Data Harmonization'. In *2013 IEEE International Congress on Big Data (BigData Congress)*, pp. 118–125. IEEE.

K. Alexander (2008). 'RDF in JSON: a specification for serialising RDF in JSON'. *Scripting for the Semantic Web* .

D. Allemang & J. A. Hendler (2011). *Semantic Web for the working ontologist. Effective Modeling in RDFS and OWL*. Morgan Kaufmann, Waltham, MA, USA.

D. Artz & Y. Gil (2007). 'A survey of trust in computer science and the semantic web'. *Journal of Web Semantics* **5**(2):58–71.

Austroads (2016). *Austroads Annual Report 2015-16*. No. AP-C20-16. Austroads.

D. F. Barbieri, et al. (2010). 'Querying RDF Streams with C-SPARQL'. *SIGMOD Record* **39**(1):20–26.

H. Bast, et al. (2016). 'Route Planning in Transportation Networks'. In *Algorithm Engineering*.

R. Battle & D. Kolas (2012). 'Enabling the geospatial semantic web with parliament and geosparql'. *Semantic Web* **3**(4):355–370.

M. Baum, et al. (2014). 'Speed-consumption tradeoff for electric vehicle route planning'. In *14Th workshop on algorithmic approaches for transportation mod-*

elling, optimization, and systems. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

D. Beckett, et al. (2014). 'RDF 1.1 Turtle. Terse RDF Triple Language. W3C Recommendation 25 February 2014'. http://www.w3.org/TR/2014/REC-turtle-20140225/.

M. Belshe, et al. (2015). 'Hypertext transfer protocol version 2 (http/2)'. *Internet Engineering Task Force* .

T. Berners-Lee (1991). 'The Original HTTP as defined in 1991'. http://www.w3.org/Protocols/HTTP/AsImplemented.html.

T. Berners-Lee (2006). 'Artificial Intelligence and the Semantic Web: AAAI 2006 Keynote'. http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html.

T. Berners-Lee (2009). 'Linked Data. Design Issues'. https://www.w3.org/DesignIssues/LinkedData.html.

T. Berners-Lee, et al. (2001). 'The semantic web'. *Scientific American* **284**(5):28–37.

C. Bizer, et al. (2009). 'Linked Data - The Story So Far'. *International Journal on Semantic Web and Information Systems* **5**(3):1–22.

M. Borrego, et al. (2009). 'Quantitative, qualitative, and mixed research methods in engineering education'. *Journal of Engineering Education* **98**(1):53–66.

W. N. Borst (1997). *Construction of engineering ontologies for knowledge sharing and reuse*. Centre for Telematics and Information Technology.

D. Brickley & R. Guha (2014). 'RDF Schema 1.1. W3C Recommendation 25 February 2014'. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

J. C. Cawley & G. T. Homce (2003). 'Occupational electrical injuries in the United States, 1992–1998, and recommendations for safety research'. *Journal of Safety Research* **34**(3):241–248.

B. V. Cherkassky, et al. (1996). 'Shortest paths algorithms: Theory and experimental evaluation'. *Mathematical Programming* **73**(2):129–174.

P. Chhetri, et al. (2012). 'Bushfire, heat wave and flooding case studies from Australia'. *Report from the International Panel of the Weather project funded by the European Commission's 7th framework programme. Melbourne* p. 39.

D. Corsar, et al. (2015). 'The transport disruption ontology'. In *International Semantic Web Conference*, pp. 329–336. Springer.

D. R. Crow (2009). 'Contact with overhead power lines how can we prevent this?'. In *2009 IEEE IAS Electrical Safety Workshop*, pp. 1–3. IEEE.

O. Curé & G. Blin (2014). *RDF Database Systems: Triples Storage and SPARQL Query Processing*. Elsevier Science & Technology.

K. M. De Oliveira, et al. (2013). 'Transportation ontology definition and application for the content personalization of user interfaces'. *Expert Systems with Applications* **40**(8):3145–3159.

B. C. Dean (2004). 'Shortest paths in FIFO time-dependent networks: Theory and algorithms'. *Rapport technique, Massachusetts Institute of Technology* .

D. Delling, et al. (2011). 'Faster batched shortest paths in road networks'. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

D. Delling, et al. (2009). 'Engineering route planning algorithms'. In *Algorithmics of large and complex networks*, pp. 117–139. Springer.

Department of Mines (2019). 'Overhead Powerlines'. *Department of Minerals and Energy Western Australia Mines Safety Bulletin* **165**:1–2.

M. Desertot, et al. (2012). 'Intelligent Transportation Systems. In Gaëlle Calvary, Thierry Delot, Florence Sedes, Jean-yves Tigli (Ed.), Computer Science and Ambient Intelligence'. In *Intelligent Transportation Systems. In Gaëlle Calvary, Thierry Delot, Florence Sedes, Jean-yves Tigli (Ed.), Computer Science and Ambient Intelligence*, p. 21p. John Wiley.

J. Dibbelt, et al. (2015). 'User-constrained multimodal route planning'. *Journal of Experimental Algorithmics* **19**:2–3.

E. W. Dijkstra (1959). 'A note on two problems in connexion with graphs'. *Numerische Mathematik* **1**(1):269–271.

L. Ding, et al. (2005). 'Tracking rdf graph provenance using rdf molecules'. *Proceedings of the 4th International Semantic Web Conference* .

Z. Dörnyei (2007). *Research methods in applied linguistics*. Oxford university press.

N. Edmonds, et al. (2006). 'Single-Source Shortest Paths with the Parallel Boost Graph Library.'. In *The Shortest Path Problem*, pp. 219–248.

T. Faaß (2015). 'Ontology-Based Route Queries with Time Windows'. In *Master thesis, University of Freiburg*.

S. Fernandez, et al. (2016). 'Ontology-based architecture for intelligent transportation systems using a traffic sensor network'. *Sensors* **16**(8):1287.

S. Fernandez, et al. (2015). 'Architecture for Intelligent Transportation System Based in a General Traffic Ontology'. In *Computer and Information Science 2015*, pp. 43–55. Springer International Publishing.

A. Fichtinger, et al. (2011). 'Data Harmonisation Put into Practice by the HUMBOLDT Project.'. *International Journal of Spatial Data Infrastructures Research* **6**:234–260.

G. Gallo & S. Pallottino (1988). 'Shortest path algorithms'. *Annals of operations Research* **13**(1):1–79.

E. Gardner (2015). 'Austroads Annual Report 2014-15'. Tech. rep., Austroads.

R. Geisberger, et al. (2008). 'Contraction hierarchies: Faster and simpler hierarchical routing in road networks'. In *International Workshop on Experimental and Efficient Algorithms*, pp. 319–333. Springer.

M. R. Genesereth & N. J. Nilsson (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc.

A. Gerber, et al. (2008). 'A functional semantic web architecture'. In *European Semantic Web Conference*, pp. 273–287. Springer.

P. Giaretta & N. Guarino (1995). 'Ontologies and knowledge bases towards a terminological clarification'. *Towards very Large Knowledge Bases: Knowledge Building & Knowledge Sharing* **25**(32):307–317.

J. Golbeck, et al. (2003). 'Trust networks on the semantic web'. In *International workshop on cooperative information agents*, pp. 238–249. Springer.

C. Golbreich & E. K. Wallace (2012). 'OWL 2 Web Ontology Language. New Features and Rationale (Second Edition). W3C Recommendation 11 December 2012'. `https://www.w3.org/TR/owl2-new-features/`.

A. V. Goldberg, et al. (2007). 'Better landmarks within reach'. In *International Workshop on Experimental and Efficient Algorithms*, pp. 38–51. Springer.

M. Goodlet (2020). *Council Committee Meeting. 14 April 2019*. City of Nedlands.

T. R. Gruber (1995). 'Toward principles for the design of ontologies used for knowledge sharing?'. *International Journal of Human-Computer Studies* **43**(5-6):907–928.

N. Guarino (1998). *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy.* IOS Press.

N. Guarino, et al. (2009). 'What is an ontology?'. In *Handbook on ontologies*, pp. 1–17. Springer.

J. Hendler, et al. (2012). 'US Government Linked Open Data: Semantic.data.gov'. *IEEE Intelligent Systems* **27**(3):25–31.

J. Herring et al. (2011). 'OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture'.

P. Hitzler, et al. (2012). 'OWL 2 Web Ontology Language: Primer (Second Edition). W3C Recommendation 11 December 2012'. `https://www.w3.org/TR/2012/REC-owl2-primer-20121211/`.

I. Horrocks, et al. (2019). 'SWRL: A Semantic Web Rule Language. Combining OWL and RuleML. W3C Member Submission 21 May 2004'. `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.

M. Houda, et al. (2010). 'A public transportation ontology to support user travel planning'. *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)* pp. 127–136.

M. Hülsen, et al. (2011). 'Traffic intersection situation description ontology for advanced driver assistance'. *2011 IEEE Intelligent Vehicles Symposium (IV)* pp. 993–999.

International Organization for Standardization (2011). *ISO 14825:2011, Intelligent transport systems – Geographic Data Files (GDF) – GDF5.0.* ISO/TC 204, Intelligent transport systems.

J. Johnston (2010). 'Qualitative research methods'. *Radiologic Technology* **82**(2):188–189.

J. Karpinski (2012). 'Main Roads roadside hazard rating'. In *ARRB Conference, 25th, 2012, Perth, Western Australia, Australia.*

R. Kenley, et al. (2014). 'Road asset management: the role of location in mitigating extreme flood maintenance'. *Procedia Economics and Finance* **18**:198–205.

J. Koustellis, et al. (2011). 'Contact of heavy vehicles with overhead power lines'. *Safety Science* **49**(6):951–955.

Landgate (2018). 'Property Street Address Data Dictionary'. `https://www0.landgate.wa.gov.au/__data/assets/pdf_file/0018/52380/Property-Street-address-Data-Dictionary.pdf`.

U. Lauther (2006). 'An Experimental Evaluation of Point-To-Point Shortest Path Calculation on Road Networks with Precalculated Edge-Flags.'. In *The Shortest Path Problem*, pp. 19–40.

T. Lebo, et al. (2013). 'PROV-O: The PROV Ontology. W3C Recommendation 30 April 2013'. `http://www.w3.org/TR/2013/REC-prov-o-20130430/`.

Q. Li & A. Kumar (2003). *National & International Practices in Decision Support Tools in Road Asset Management.* CRC for Construction Innovation, Brisbane.

D. Linders & S. Wilson (2011). 'What is open government? One year after the directive'. *ACM International Conference Proceeding Series* pp. 262–271.

S. Löbner (2013). *Understanding semantics.* Routledge.

B. Lorenz, et al. (2005). 'Ontology of transportation networks'. *Reasoning on the Web with Rules and Semantics* .

K. Madduri, et al. (2006). 'Parallel Shortest Path Algorithms for Solving Large-Scale Instances'. In *The Shortest Path Problem.*

T. Marasovic & R. Marasovic (2010). 'Ontology based road service task manager and route planning system'. *SoftCOM 2010, 18th International Conference on Software, Telecommunications and Computer Networks* pp. 331–335.

D. T. Martin, et al. (2019). 'Data Standard for Road Management and Investment in Australia and New Zealand Version 3.0' pp. 1–254.

F. Mihai & J. Robertson (2003). 'Applying knowledge management principles for implementing a road information System at Main Roads Western Australia'.

E. J. Miller (2001). 'An Introduction to the Resource Description Framework'. *Journal of Library Administration* **34**(3–4):245–255.

L. Moreau & P. Missier (2013). 'PROV-DM: The PROV Data Model. W3C Recommendation 30 April 2013'. `http://www.w3.org/TR/2013/REC-prov-dm-20130430/`.

M. A. Musen (2015). 'The protégé project: A look back and a look forward'. *AI matters* **1**(4):4–12.

F. R. A. Nascimento, et al. (2018). 'Automated Evaluation of Open Government Data Portals'. *International Journal of Electronic Government Research* **14**(3):57–72.

A. S. Niaraki & K. Kim (2009). 'Ontology based personalized route planning system using a multi-criteria decision making approach'. *Expert Systems With Applications* **36**(P1):2250–2259.

M. Niestroj, et al. (2019). 'Introducing a Framework for Conflating Road Network Data with Semantic Web Technologies'. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* pp. 231–238.

M. G. Niestroj, et al. (2018). 'A proposal to use semantic web technologies for improved road network information exchange'. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* **4**(4).

M. Niknam & S. Karshenas (2017). 'A shared ontology approach to semantic representation of BIM data'. *Automation in Construction* **80**:22–36.

N. J. Nilsson (1991). 'Logic and artificial intelligence'. *Artificial Intelligence* **47**(1–3):31–56.

OECD (2017). 'Recommendation of the Council on Open Government. 14 December 2017 - C(2017)140 - C/M(2017)22'. `https://www.oecd.org/gov/Recommendation-Open-Government-Approved-Council-141217.pdf`.

B. Parsia, et al. (2017). 'The OWL Reasoner Evaluation (ORE) 2015 Competition Report'. *Journal of Automated Reasoning* **59**(4):455–482.

P. Parycek & M. Sachs (2010). 'Open Government – Information Flow in Web 2.0'. *Journal of ePractice* **9**:57–68.

J. A. Patel & P. Sharma (2017). 'Big Data Harmonization – Challenges and Applications'. *International Journal on Recent and Innovation Trends in Computing and Communication* **5**(6):206–208.

M. Perry & J. Herring (2012). 'OGC GeoSPARQL-A geographic query language for RDF data'. *OGC implementation standard* **40**(OGC 09-157r4).

R. Perumpilly, et al. (2019). 'CONie is not COBie: information exchange differences between network and building handover to asset management'. In *IOP Conference Series: Materials Science and Engineering*, vol. 512, p. 012041. IOP Publishing.

R. Provine, et al. (2004a). 'Ontology-based methods for enhancing autonomous vehicle path planning'. *Robotics and Autonomous Systems* **49**(1–2):123–133.

R. Provine, et al. (2004b). 'Observations on the use of ontologies for autonomous vehicle navigation planning'. In *Robotics and Autonomous Systems Journal: Special Issue on the 2004 AAAI Knowledge Representation and Ontologies for Autonomous Systems Spring Symposium*.

G. Qi, et al. (2013). *Linked Data and Knowledge Graph. 7th Chinese Semantic Web Symposium and 2nd Chinese Web Science Conference, CSWS 2013. Shanghai, China, August 2013. Revised Selected Papers. Preface.* Springer.

M. Richardson, et al. (2003). 'Trust management for the semantic web'. In *International semantic Web conference*, pp. 351–368. Springer.

S. Ridge (2008). 'Overhead Powerlines'. *Department of Minerals and Energy Western Australia Safety Bulletin* **85**:1–4.

C. Schlenoff, et al. (2004). 'Using ontologies to aid navigation planning in autonomous vehicles'. *The Knowledge Engineering Review* **18**(3):243–255.

G. Schreiber & Y. Raimond (2014). 'RDF 1.1 Primer. W3C Working Group Note 24 June 2014'. `https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/`.

D. Schultes & P. Sanders (2007). 'Dynamic highway-node routing'. In *International Workshop on Experimental and Efficient Algorithms*, pp. 66–79. Springer.

A. Seneviratne (2015). 'Distribution Design Rules. Standard Number: HPC-9DJ-01-0002-2015'. *Horizon Power Corporation* pp. 1–100.

E. Sirin, et al. (2007). 'Pellet: A practical OWL-DL reasoner'. *Journal of Web Semantics* **5**(2):51 – 53.

L. Steller & S. Krishnaswamy (2008). 'Pervasive service discovery: mTableaux mobile reasoning'. *International Conference on Semantic Systems* .

D. R. Stinson (2005). *Cryptography: theory and practice*. Chapman and Hall/CRC.

R. Studer, et al. (1998). 'Knowledge engineering: principles and methods'. *Data & Knowledge Engineering* **25**(1–2):161–197.

G. Sutcliffe (2017). 'The TPTP Problem Library and Associated Infrastructure'. *Journal of Automated Reasoning* **59**(4):483–502.

T. Sutton, et al. (2017). 'Documentation QGIS 3.4. QGIS User Guide. Features'. `http://docs.qgis.org/3.4/en/docs/user_manual/preamble/features.html`.

P. Szwed, et al. (2012). 'Ontology based integration and decision support in the INSIGMA route planning subsystem'. In *2012 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 141–148.

J. Tauberer (2014). 'Open Government Data: The Book. By Joshua Tauberer. Second Edition: 2014. Open Government Data Definition: The 8 Principles of Open Government Data.'. `https://opengovdata.io/2014/8-principles/`.

J. M. Torlach (2001). 'Overhead Powerlines'. *Department of Minerals and Energy Western Australia Safety Bulletin* **51**:1–2.

TTI (2015). 'TTI Core v0.03: Core Ontologies for Safe Autonomous Driving'. `https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/Ontology/TTICore-0.03/`.

D. Wang, et al. (2015). 'A novel trust-aware composite semantic web service selection approach'. *Mathematical Problems in Engineering* **2015**.

J. Wang, et al. (2005). 'An Ontology-based Public Transport Query System'. In *2005 First International Conference on Semantics, Knowledge and Grid*, p. 62.

S. Winter (2002). 'Modeling costs of turns in route planning'. *GeoInformatica* **6**(4):345–361.

F. Yu, et al. (2018). 'Semantic web technologies automate geospatial data conflation: conflating points of interest data for emergency response services'. In *LBS 2018: 14th International Conference on Location Based Services*, pp. 111–131. Springer.

L. Zhao, et al. (2015). 'Ontologies for Advanced Driver Assistance Systems'. *Journal of the Japanese Society for Artificial Intelligence (JSAI)* .

L. Zhao, et al. (2017). 'Ontology-Based Driving Decision Making: A Feasibility Study at Uncontrolled Intersections'. *IEICE Transactions on Information and Systems* **E100.D**(7):1425–1439.

Appendices

A.1 Software

This section introduces the software that was used in this thesis, including the ontology editor Protégé, Java IDE IntelliJ IDEA, Python IDE PyCharm, and the GIS software QGIS.

A.1.1 Protégé

Protégé[1] is a popular OWL 2 ontology editor with a user-friendly Graphical User Interface (GUI) developed by the Standford Center for Biomedical Informatics Research at the Standford University School of Medicine (Musen, 2015). With Protégé, it is possible to design and maintain different ontology formats (e.g. RDF/XML, Turtle, OWL/XML, OWL Functional Syntax, Manchester OWL, OBO Format, LaTeX and JSON-LD). The program view of Protégé after loading the program is indicated in Figure A.1. The Protégé ontology editor enables by default a variety of different features, such as the creation of classes, properties and individuals with their relations. Protégé version 5.5 was used in this thesis to design road network ontologies (e.g. OSM, Landgate and MRWA), formulate SWRL rules (e.g. determine the road node a road sign belongs to or identify if an intersection is a roundabout) and to reason ontology knowledge.

Protégé has an active community that develops plugins to extend the program's functionality. The plugins used in this thesis include ROWLTab to design

[1] `http://protege.stanford.edu`

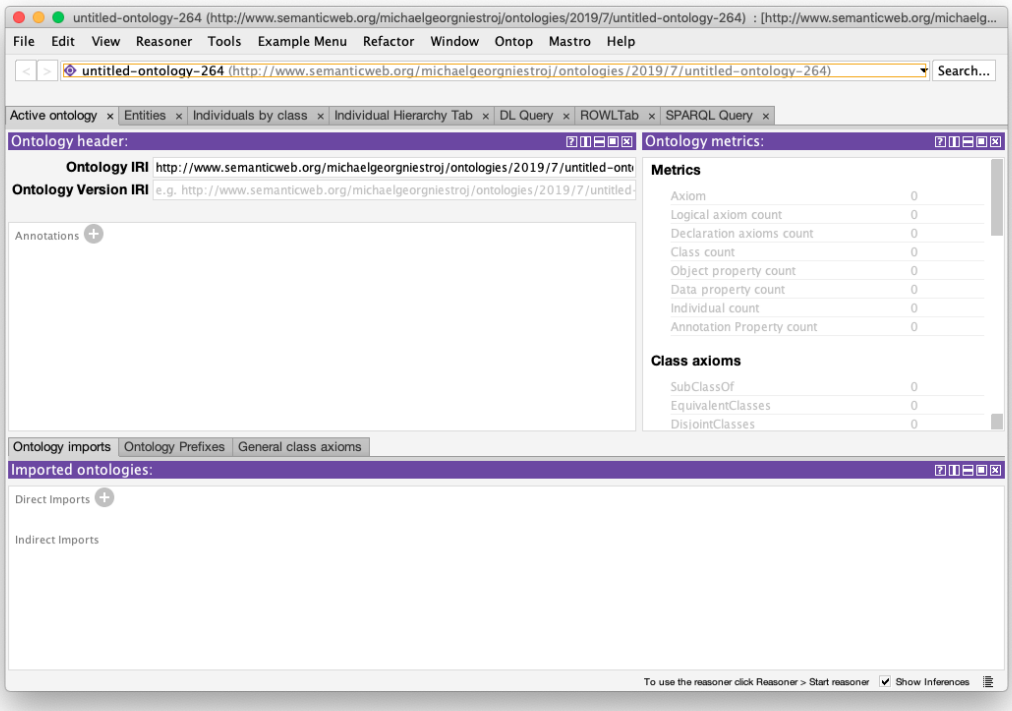


Figure A.1: At program start Protégé loads an empty ontology by default.

the SWRL rules, OWLViz for the ontology graph visualisation and Pellet as the ontology reasoner.

A.1.2 Programming Environments

The IDEs used int his thesis were IntelliJ IDEA[2] and PyCharm[3]; both were developed by the company JetBrains s.r.o.:

- The IntelliJ IDEA IDE is a powerful environment for Java programming. It supports code completion, coding assistance, debugging, version history and various programming languages (e.g. Java, Java EE, Kotlin, Android, JavaScript and Scala). The IntelliJ IDE version 2019.1 (Ultimate Edition) was used in this thesis together with the Java OWL API[4] for the devel-

[2] `http://www.jetbrains.com/idea/`
[3] `http://www.jetbrains.com/pycharm/`
[4] `http://owlcs.github.io/owlapi/`

opment of a Semantic Web-based road network planning approach. The program view of IntelliJ after loading a project is indicated in Figure A.2.

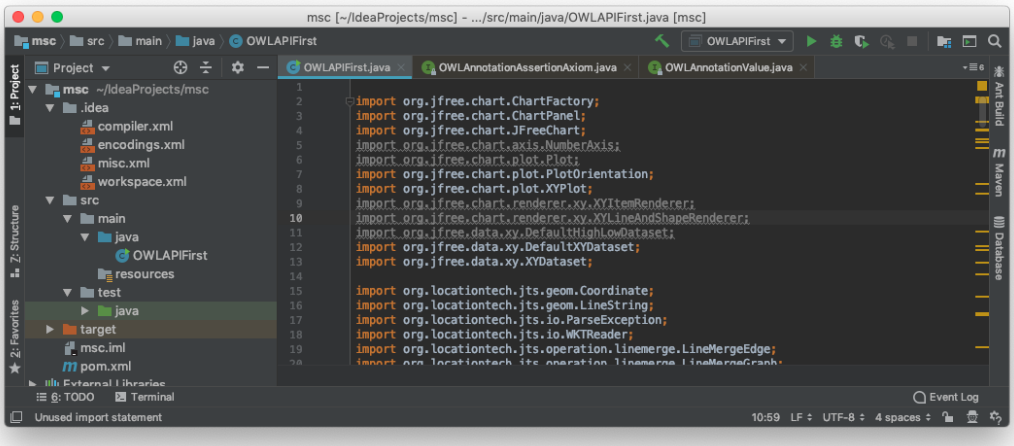


Figure A.2: JetBrains IntelliJ IDEA program view.

- PyCharm is a professional IDE for Python developers. It has similar IDE features as IntelliJ IDEA (e.g. code completion, coding assistance, debugging and version history) and supports Python-specific tools, such as Anaconda[5] and Jupyter Notebook.[6] The PyCharm IDE version 2019.2 (Professional Edition) was used in this thesis for the development of four different Python scripts for data processing.

  The first script, 'GeoJSON to TTL', reads GeoJSON data from different data sources (Landgate, MRWA and OSM) and creates Turtle syntax of for each individual so that the results can be inserted into the developed ontologies. Before the processing of the second script, 'create individuals', the onologies (MRWA, OSM and Landgate) must be merged and saved in JSON-LD format for simpler data handling.

[5] GUI-based package manager for Python libraries, URL: `http://www.anaconda.com`.
[6] Web-based application to create shareable source code files with visualisation support (e.g. graphs, interactive navigation elements and figures), URL: `http://www.jupyter.org`.

The second script then extracts all JSON-LD individuals from the merged ontology and saves them as a JSON-LD data file. The result is used for the third script, 'translate road network', which translates MRWA road network coordinates to Landgate Shared Location Information Platform data. The program view of PyCharm after loading a project is indicated in Figure A.3.

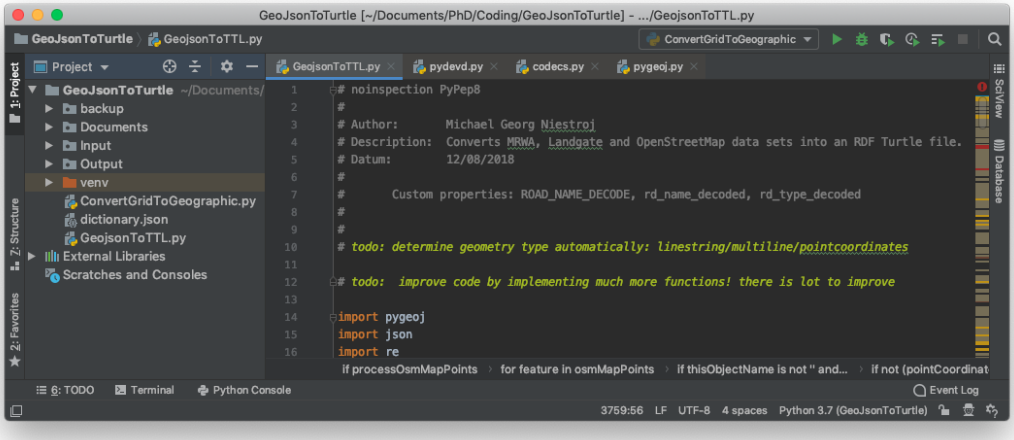


Figure A.3: JetBrains PyCharm program view.

A.1.3 QGIS

The software QGIS[7] is an open-source GIS to view, create, edit and export vector and raster data in different data formats (e.g. GeoJSON, PostGIS, ESRI Shapefile, GML, GeoTIFF, JPEG, PNG and OGC Web Services) (Sutton et al., 2017). The QGIS version 3.4.3 (Madeira) software was used in this thesis to load road assets in different data formats (e.g. ESRI Shapefile, XML and GeoJSON), view and select road assets, as indicated in Figure A.4, and export data selections into the GeoJSON format for post-processing.

[7] https://qgis.org/

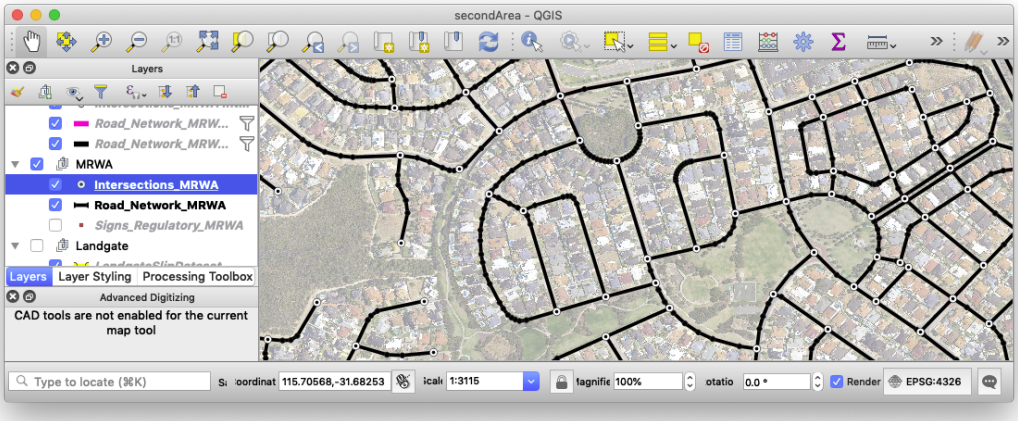
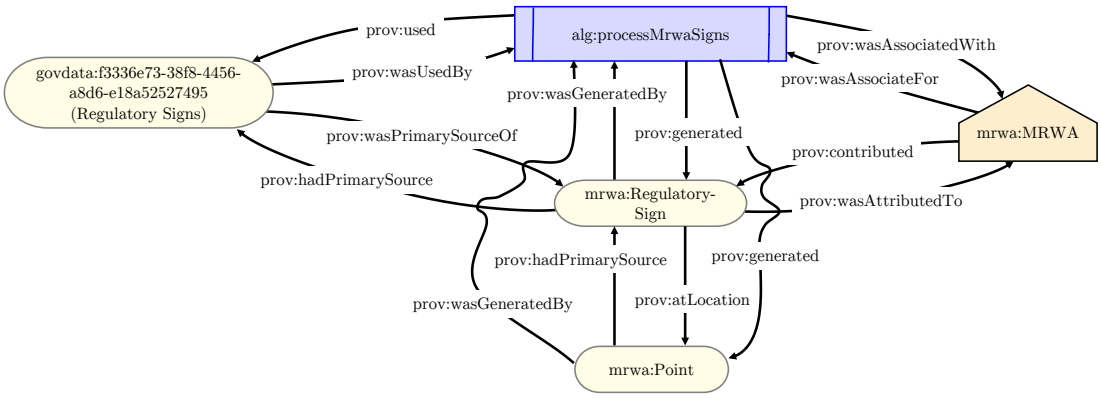


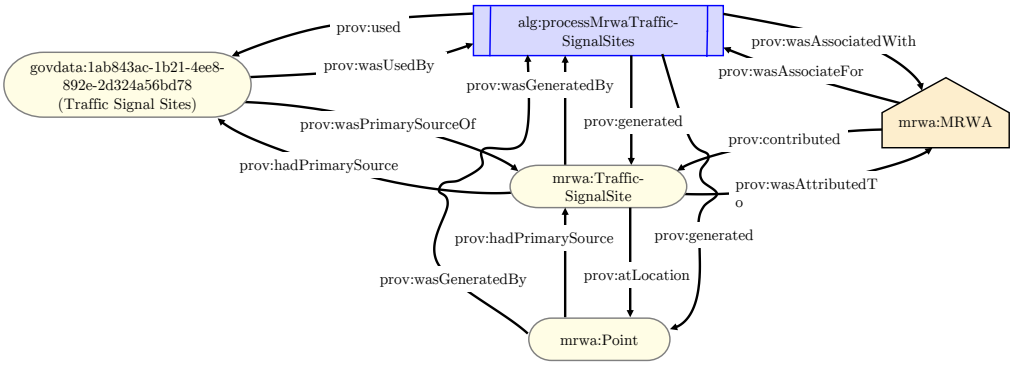
Figure A.4: QGIS program view after loading a project.

A.2 MRWA Provenance Graphs

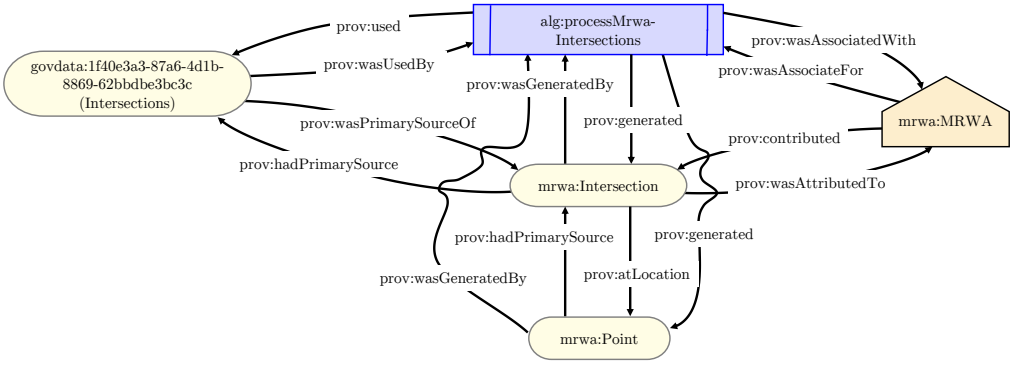
A.2.1 Provenance Model of MRWA Regulatory Signs



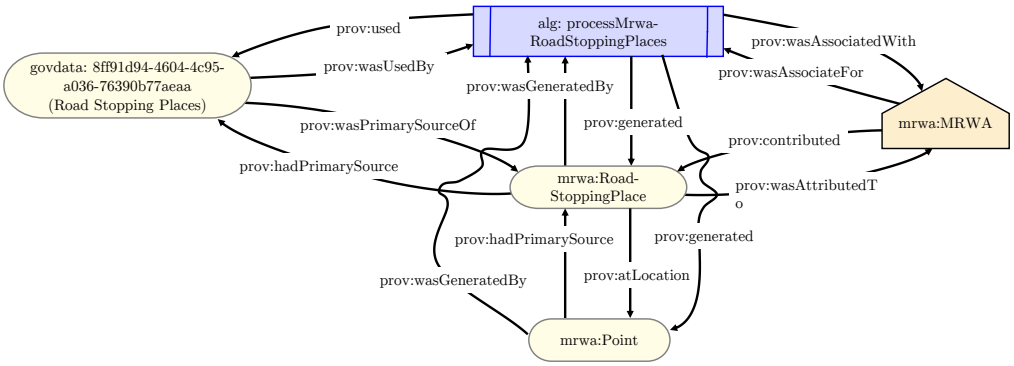
A.2.2 Provenance Model of MRWA Traffic Signal Sites



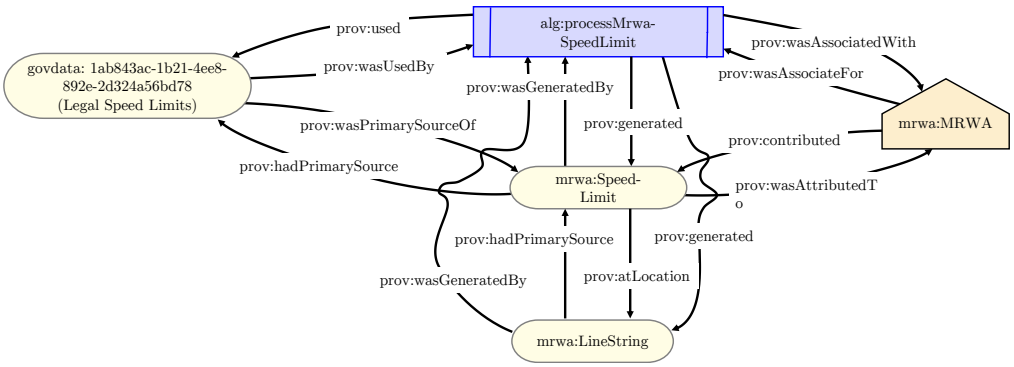
A.2.3 Provenance Model of MRWA Intersections



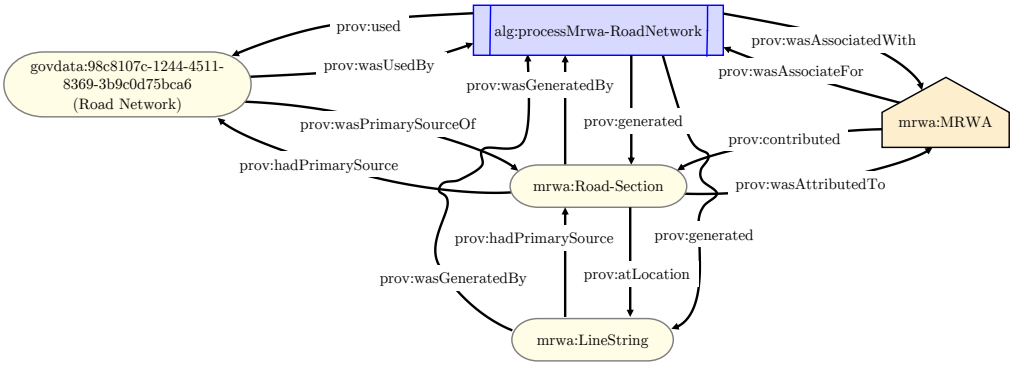
A.2.4 Provenance Model of MRWA Road Stopping Places



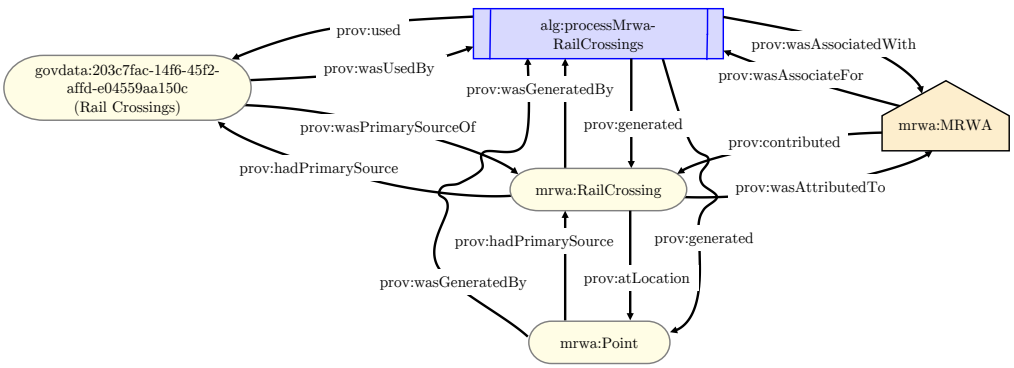
A.2.5 Provenance Model of MRWA Legal Speed Limits



A.2.6 Provenance Model of MRWA Road Sections

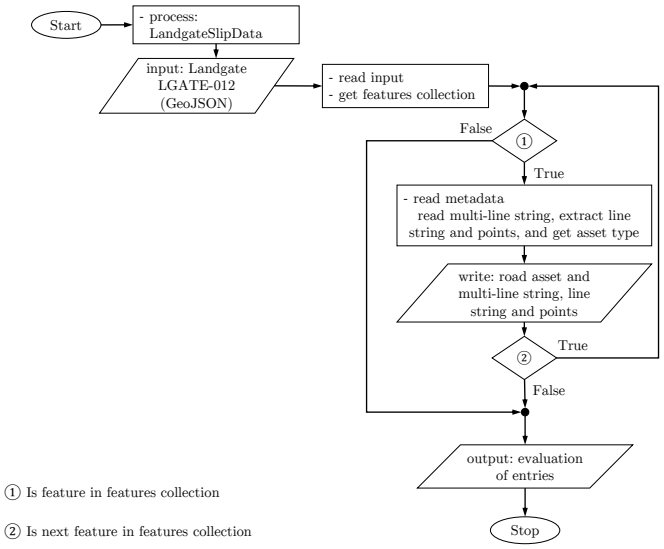


A.2.7 Provenance Model of MRWA Rail Crossings

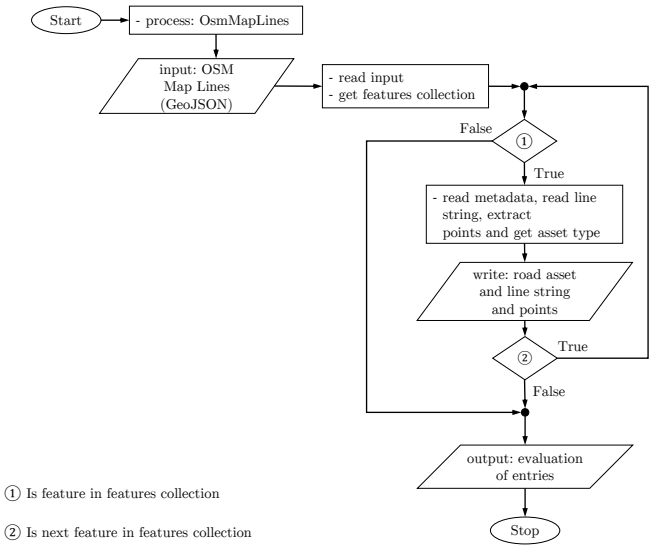


A.3 Flowchart Data Creation

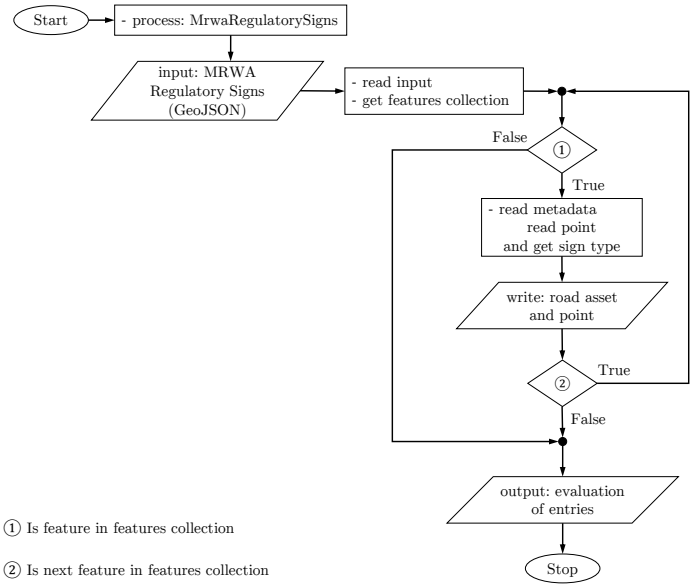
A.3.1 Write Landgate Road Asset Individuals Generated from GeoJSON Datasets



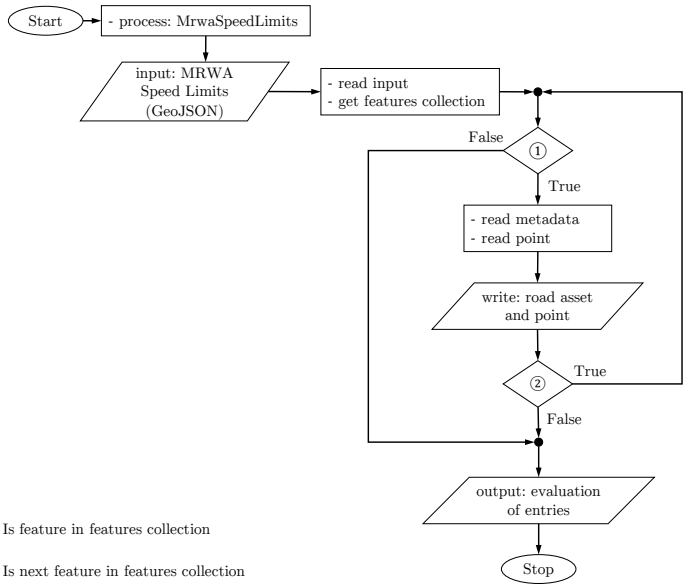
A.3.2 Write OSM Map Line Individuals Generated from GeoJSON Datasets



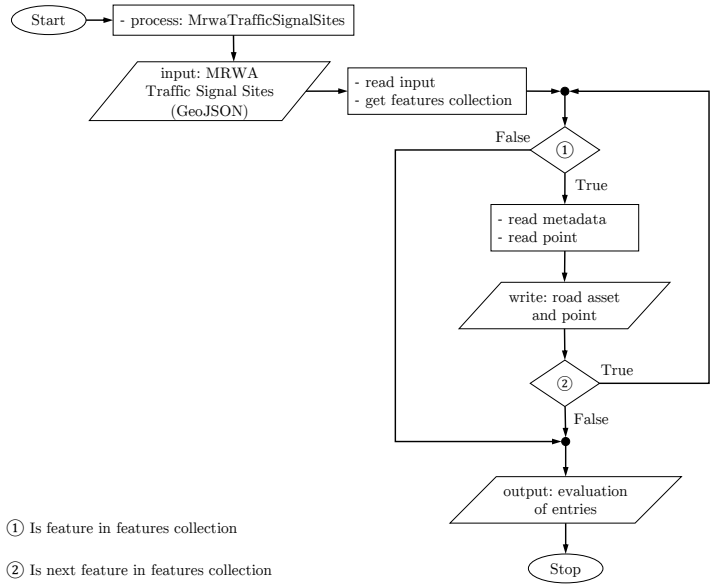
A.3.3 Write MRWA Regulatory Sign Individuals Generated from GeoJSON Datasets



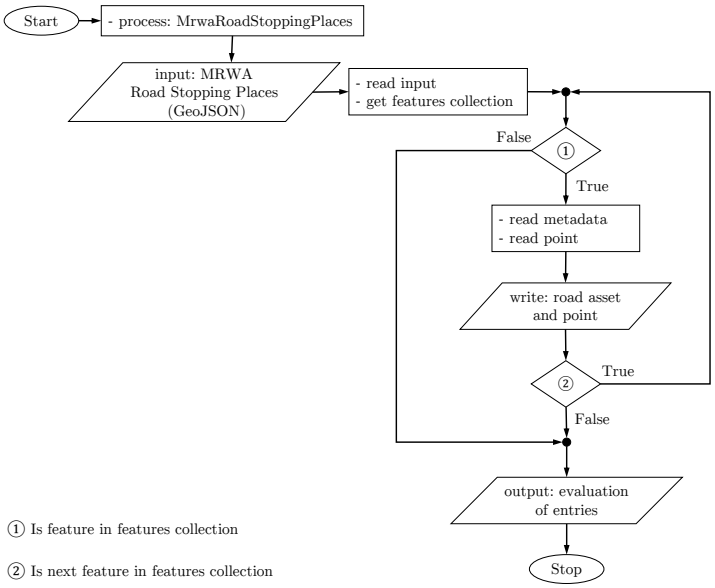
A.3.4 Write MRWA Speed Limit Individuals Generated from GeoJSON Datasets



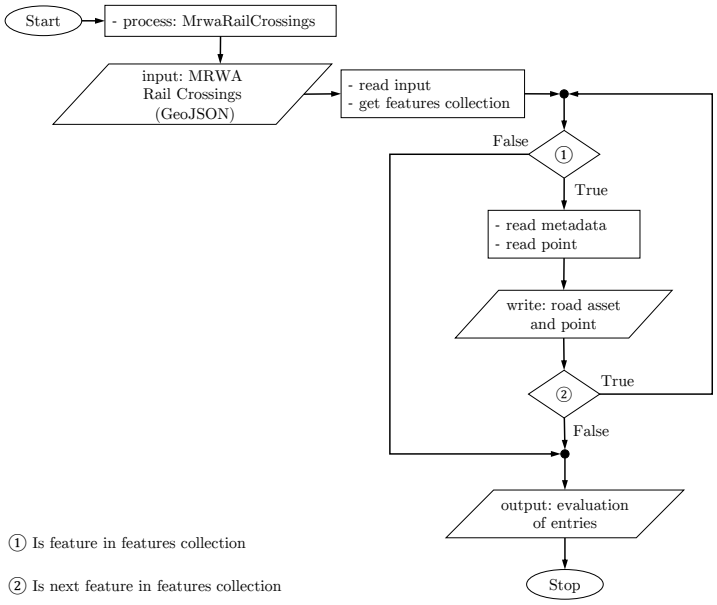
A.3.5 Write MRWA Traffic Signal Site Individuals Generated from GeoJSON Datasets



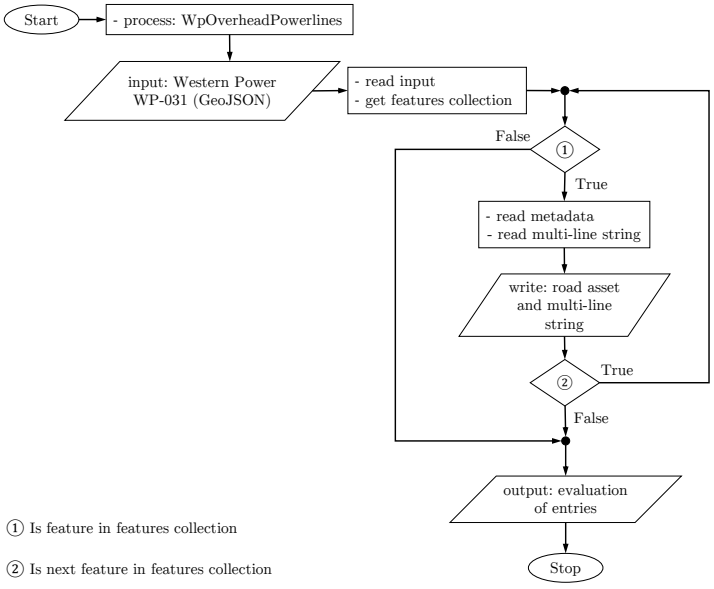
A.3.6 Write MRWA Road Stopping Place Individuals Generated from GeoJSON Datasets



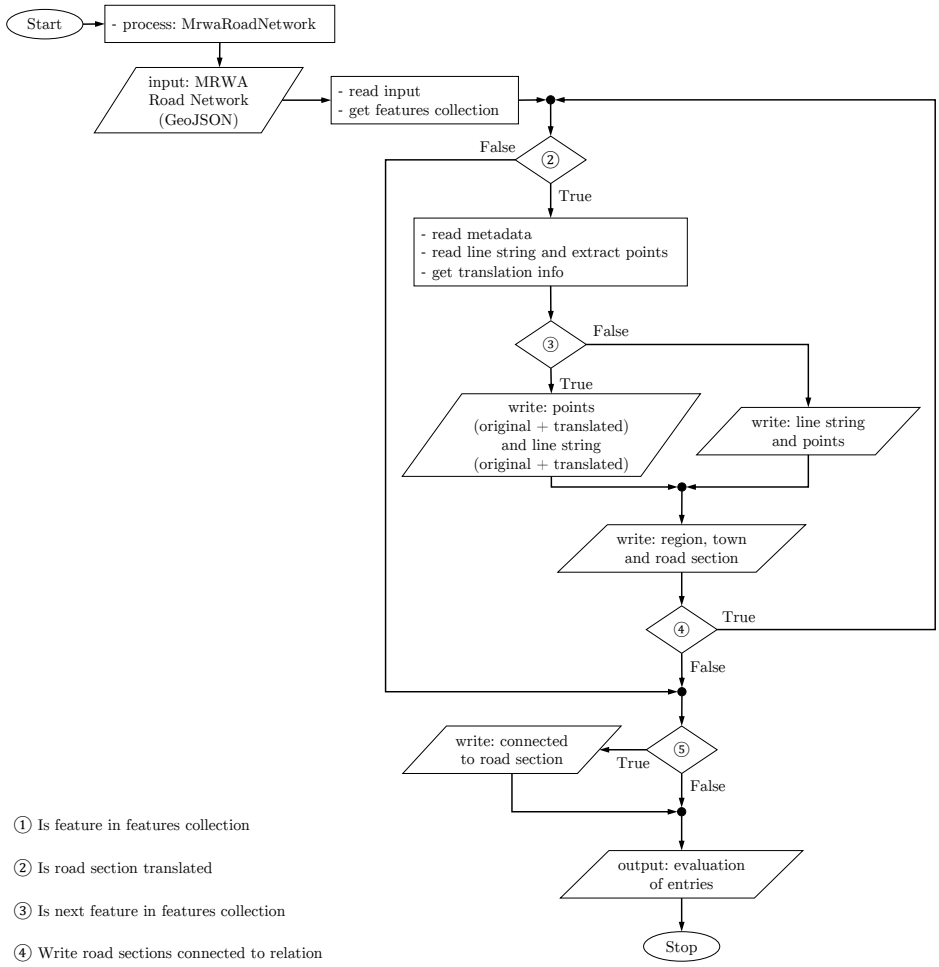
A.3.7 Write MRWA Rail Crossing Individuals Generated from GeoJSON Datasets



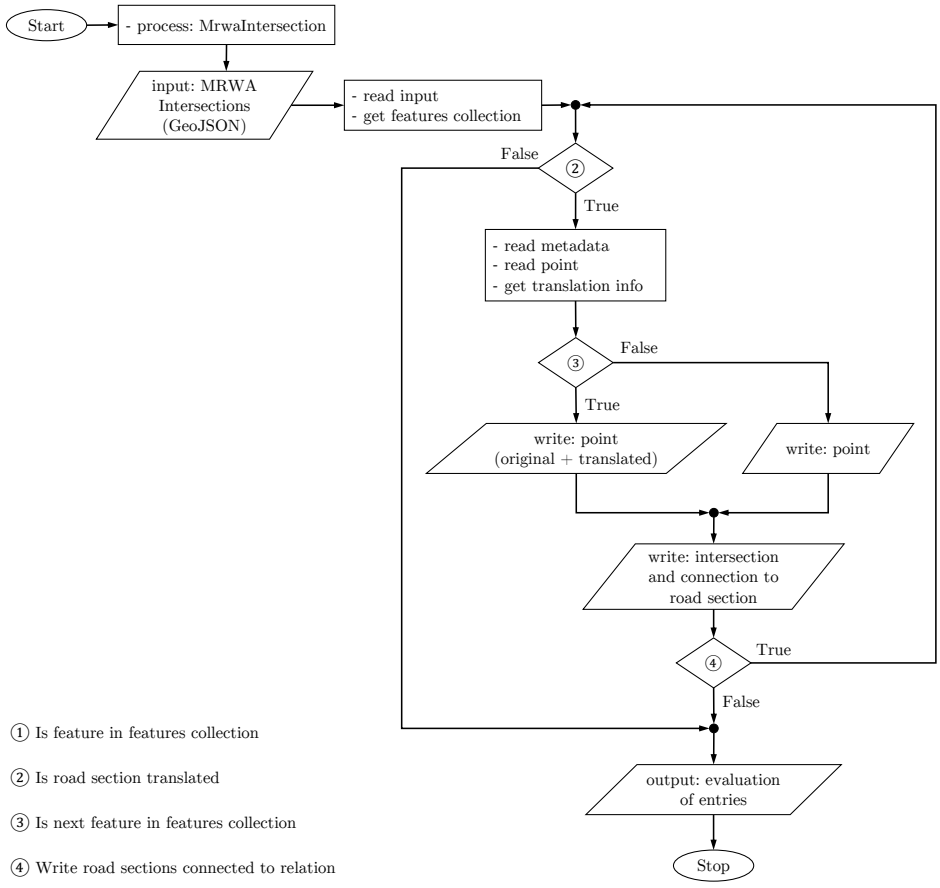
A.3.8 Write Western Power Overhead Power Line Individuals Generated from GeoJSON Datasets



A.3.9 Write Translated MRWA Road Network Individuals Generated from GeoJSON Datasets



A.3.10 Write Translated MRWA Intersection Individuals Generated from GeoJSON Datasets



A.4 Ontology Data Creation

A.4.1 Write MRWA Speed Limit Individuals

```
if processMrwaSpeedLimits:
  file = pygeoj.load("./Input/" + InputFileMrwaSpeedLimits)
  crs = file.crs
  crs = '"' + crs['properties']['name'] + '"'
  # source: https://catalogue.data.wa.gov.au/dataset/legal-speed-limits
  dataset = dataset_MrwaSpeedLimit
  agent = "MRWA"
  for entry in file:
    object_id = entry.properties['OBJECTID']
    entryName = "SpeedLimit"
    common_usuage_name = entry.properties['COMMON_USAGE_NAME']
    geometryCoordinates = None
    geometryID = None
    if common_usuage_name is not None:
      entryName = entryName + "_" + cleanString(common_usuage_name)
    if object_id is not None:
      entryName = entryName + "_" + str(object_id)
    objectName = prefixMrwa + entryName
    if entry.geometry.coordinates is not None:
      geometryCoordinates = str(entry.geometry.coordinates)
    GeoJSONcoordinates = {}
    GeoJSONcoordinates['coordinates'] = (entry.geometry.coordinates)
    GeoJSONcoordinates['type'] = (entry.geometry.type)
    entryLineString = wkt.dumps(GeoJSONcoordinates, decimals=8)
    thisLineString = prefixMrwa + entryLineString
    if geometryCoordinates is not None and not (lineCoordinateDict.get(
        thisLineString)):
      geometryID = functions.writeLineString(prefix=prefixMrwa, lineString=
          entryLineString, crs=crs, Ontology=Ontology, generatedBy=
          processMrwaSpeedLimitLabel, primarySource=objectName)
      lineCoordinateDict[thisLineString] = objectName
    if not (speedlimitDictMrwa.get(entryName)):
      functions.writeEntityContent(prefix=prefixMrwa, entry=entryName, metadata
          =entry.properties, geometryID=geometryID, Ontology=Ontology, agent=
          agent, dataset=dataset, generatedBy=processMrwaSpeedLimitLabel,
          thisClass="mrwa:SpeedLimit")
      speedlimitDictMrwa[entryName] = objectName
```

A.4.2 Write MRWA Traffic Signal Site Individuals

```
if processMrwaTrafficSignals:
  file = pygeoj.load("./Input/" + InputFileMrwaTrafficSignalSites)
  crs = file.crs
  crs = '"' + crs['properties']['name'] + '"'
  # source: https://catalogue.data.wa.gov.au/dataset/traffic-signal-sites
  dataset = dataset_MRWA_Traffic_Signal_Sites
  agent = "MRWA"
  for entry in file:
    object_id = entry.properties['OBJECTID']
    entryName = "TrafficSignalSite"
    node_name = entry.properties['NODE_NAME']
    node_description = entry.properties['NODE_DESCR']
    geometryCoordinates = None
    if node_description is not None:
    entryName = entryName + "_" + cleanString(node_description)
    if object_id is not None:
      entryName = entryName + "_" + str(object_id)
    objectName = prefixMrwa + entryName
    if entry.geometry.coordinates is not None:
      geometryCoordinates = str(entry.geometry.coordinates)
    GeoJSONcoordinates = {}
    GeoJSONcoordinates['coordinates'] = (entry.geometry.coordinates)
    GeoJSONcoordinates['type'] = (entry.geometry.type)
    entryPoint = wkt.dumps(GeoJSONcoordinates, decimals=8)
    thisPoint = prefixMrwa + entryPoint
    longitude = "{:.{}f}".format(entry.geometry.coordinates[0], decimalPoints)
    latitude = "{:.{}f}".format(entry.geometry.coordinates[1], decimalPoints)
    pointCoordinatesLabel = "PointCoordinates"
    thisPointCoordinates = pointCoordinatesLabel + "_" + str(longitude) + "_" +
        str(latitude)
    pointCoordinates = prefixMrwa + thisPointCoordinates
    geometryID = thisPointCoordinates
    if geometryCoordinates is not None and not (pointCoordinatesDictMrwa.get(
      pointCoordinates)):
      writePoint(prefix=prefixMrwa, longitude=longitude, latitude=latitude, entity=
        pointCoordinates, generatedBy=processMrwaTrafficSignalSitesLabel,
        primarySource=objectName)
      pointCoordinatesDictMrwa[pointCoordinates] = objectName
    if not (trafficsignalsDictMrwa.get(entryName)):
      functions.writeEntityContent(prefix=prefixMrwa, entry=entryName, metadata
        =entry.properties, geometryID=geometryID, Ontology=Ontology, agent=
        agent, dataset=dataset, generatedBy=processMrwaTrafficSignalSitesLabel,
        thisClass="mrwa:TrafficSignalSite")
      trafficsignalsDictMrwa[entryName] = objectName
```

A.4.3 Write MRWA Road Stopping Place Individuals

```
if processMrwaRoadStoppingPlaces:
  file = pygeoj.load("./Input/" + InputFileMrwaRoadStoppingPlaces)
  crs = '"' + file.crs['properties']['name'] + '"'
  # source: https://catalogue.data.wa.gov.au/dataset/road-stopping-places
  dataset = dataset_MRWA_Road_Stopping_Places
  agent = "MRWA"
  for entry in file:
    object_id = entry.properties['OBJECTID']
    entryName = "RoadStoppingPlace"
    commonUsageName = entry.properties['COMMON_USAGE_NAME']
    commonUsageNameClean = ""
    geometryCoordinates = None
    if commonUsageName is not None:
      entryName = entryName + "_" + cleanString(commonUsageName)
    if object_id is not None:
      entryName = entryName + "_" + str(object_id)
    objectName = prefixMrwa + entryName
    if entry.geometry.coordinates is not None:
      geometryCoordinates = str(entry.geometry.coordinates)
    GeoJSONcoordinates = {}
    GeoJSONcoordinates['coordinates'] = (entry.geometry.coordinates)
    GeoJSONcoordinates['type'] = (entry.geometry.type)
    entryPoint = wkt.dumps(GeoJSONcoordinates, decimals=8)
    thisPoint = prefixMrwa + entryPoint
    longitude = "{:.{}f}".format(entry.geometry.coordinates[0], decimalPoints)
    latitude = "{:.{}f}".format(entry.geometry.coordinates[1], decimalPoints)
    pointCoordinatesLabel = "PointCoordinates"
    thisPointCoordinates = pointCoordinatesLabel + "_" + str(longitude) + "_" +
        str(latitude)
    pointCoordinates = prefixMrwa + thisPointCoordinates
    geometryID = thisPointCoordinates
    if geometryCoordinates is not None and not (pointCoordinatesDictMrwa.get(
        pointCoordinates)):
      writePoint(prefix=prefixMrwa, longitude=longitude, latitude=latitude, entity=
          pointCoordinates, generatedBy=processMrwaRoadStoppingPlacesLabel,
          primarySource=objectName)
      pointCoordinatesDictMrwa[pointCoordinates] = objectName
    if not (roadstoppingplacesDictMrwa.get(entryName)):
      functions.writeEntityContent(prefix=prefixMrwa, entry=entryName, metadata
          =entry.properties, geometryID=geometryID, Ontology=Ontology, agent=
          agent, dataset=dataset, generatedBy=
          processMrwaRoadStoppingPlacesLabel, thisClass="mrwa:
          RoadStoppingPlace")
      roadstoppingplacesDictMrwa[entryName] = objectName
```

A.4.4 Write Western Power Overhead Power Line Individuals

```
1  def processWPOverheadPowerlines():
2    file  = pygeoj.load("./Input/" + inputFileWPOverheadPowerlines)
3    crs  = '"' + file.crs['properties']['name'] + '"'
4    # source: https://catalogue.data.wa.gov.au/dataset/road-network
5    dataset = dataset_WP_Overhead_Powerlines_031
6    agent = "WP"
7    for entry in file:
8      pick_id = entry.properties['pick_id']
9      kv = entry.properties['kv']
10     entryName = "OverheadPowerline_031"
11     if kv is not None:
12       entryName = entryName + "_" + kv
13     if pick_id is not None:
14       entryName = entryName + "_" + pick_id
15     objectName = prefixWP + entryName
16     geometryCoordinates = None
17     if entry.geometry.coordinates is not None:
18       geometryCoordinates = str(entry.geometry.coordinates)
19     GeoJSONcoordinates = {}
20     GeoJSONcoordinates['coordinates'] = (entry.geometry.coordinates)
21     GeoJSONcoordinates['type'] = (entry.geometry.type)
22     entryMultiLineString = wkt.dumps(GeoJSONcoordinates, decimals=8)
23     thisMultiLineString = prefixWP + entryMultiLineString
24     geometryID = None
25     if geometryCoordinates is not None and not (multiLineDictWp.get(
           thisMultiLineString)):
26       geometryID = functions.writeMultiLineString(prefix=prefixWP,
             multiLineString=entryMultiLineString, crs=crs, Ontology=Ontology,
             generatedBy=processWPOverheadPowerlinesLabel, primarySource=
             objectName)
27       multiLineDictWp[thisMultiLineString] = objectName
28     if not (overhead_Powerlines_031_Dict.get(entryName)) and geometryID:
29       functions.writeEntityContent(prefix=prefixWP, entry=entryName, metadata=
             entry.properties, geometryID=geometryID, Ontology=Ontology, agent=
             agent, dataset=dataset, generatedBy=processWPOverheadPowerlinesLabel,
              thisClass="wp:WP031")
30       Ontology.write(prefixWP + entryName + '␣' + prefixWP +'
             hasMultiLineCoordinates␣' + prefixWP + geometryID + '.\n')
31       overhead_Powerlines_031_Dict[entryName] = objectName
32   writeDatasetAndProcess("wp:", 'WP', processWPOverheadPowerlinesLabel,
       dataset_WP_Overhead_Powerlines_031, Ontology)
```

```
33  writeAgent("WP", "wp:", "Western␣Power", Ontology)
```

A.4.5 Write Agent Individual

```
1  def writeAgent(name, prefix, label, Ontology):
2  Ontology.write(prefix + name + '␣a␣prov:Organization;\n')
3  Ontology.write('rdfs:label' + '␣"' + label + '"' + '.\n')
```

A.4.6 Write Dataset and Process Individuals

```
1  def writeDataSetAndProcess(prefix, agent, process, dataset, Ontology):
2    # write agent was accosiate for process
3    Ontology.write(prefix + agent + '␣prov:wasAssociateFor␣alg:' + process + '.\n')
4    # write the dataset provenance
5    Ontology.write(dataset + '␣a␣prov:Entity;\n')
6    Ontology.write('␣prov:wasUsedBy␣alg:' + process + '.\n')
7    # write the process provenance
8    Ontology.write('alg:' + process + '␣a␣prov:Activity;\n')
9    Ontology.write('␣prov:wasAssociatedWith␣' + prefix + agent + ';\n')
10   Ontology.write('␣prov:used␣' + dataset + '.\n')
11   Ontology.write(prefix + agent + '␣a␣prov:Agent.\n')
```

A.4.7 Write Other Tags

```
1  def writeOtherTags(values):
2    dictValues = dict(item.split("=>") for item in values.split(","))
3    for key, value in dictValues.items():
4      key = key.replace("'", "")
5      value = value.replace("'", "")
6      Ontology.write(';\n' + prefixOsm + key + '␣"' + str(value) + '"')
```

A.4.8 Write Translation Method

```
1  def writeTranslationMethod(method, label):
2    Ontology.write(prefixAlg + method + '␣a␣prov:Activity;\n')
3    Ontology.write('␣' + 'rdfs:label␣"' + label + '".\n')
```

A.5 Translation Methods

A.5.1 Method 1 and Method 2

```
1  Function translateMrwaIntersectionToLgSlipData(DataSet):
2      Result ⟵ set to DataSet
3      foreach Road in DataSet do
4          foreach Individual1 in Road do
5              if Individual1 is Landgate RoadSection or Connector then
6                  MultiLineString ⟵ get multi-line string of RoadSection
7                  LineString ⟵ get line string of MultiLineString
8                  Points ⟵ get points of LineString
9                  foreach Point1 in Points do
10                     foreach Individual in Road do
11                         if Individual2 is MRWA Intersection then
12                             Point2 ⟵ get point of Individual2
13                             if Point2 within 6.00m offset to Point1 then
14                                 Point2 ⟵ update with Point1 coordinates
15                                 if Individual1 is Landgate RoadSection
                                    then
16                                     Result[Road][Individual1] ⟵ set
                                        Point2 and translation method 1
17                                 else
18                                     Result[Road][Individual1] ⟵ set
                                        Point2 and translation method 2
19     return Result
20 End Function
```

A.5.2 Method 3

```
1  Function prepareMrwaRoadNodes(DataSet):
2      Result ⟵ set to DataSet
3      Intersections ⟵ get MRWA intersections from DataSet with
         translated coordinates
4      foreach Road in DataSet do
5          foreach Individual in Road do
6              if Individual is MRWA RoadSection then
7                  LineString ⟵ get line string of Individual
8                  Points ⟵ get points of LineString
9                  if Individual is not left or right carriageway then
10                     foreach Point1 in Points do
11                         foreach Intersection in Intersections do
12                             Point2 ⟵ get point of Intersection
13                             Translation ⟵ get translation methos of
                                 Intersection
14                             if offset between Point1 and Point2 within
                                 6.00m then
15                                 Point1 ⟵ update with Point2 coordinates
16                                 LineString ⟵ update with Point1
                                     coordinates
17                                 if Translation is method 1 then
18                                     Result[Road][Individual] ⟵ set
                                         LineString and translation method 3.1
19                                 else
20                                     Result[Road][Individual] ⟵ set
                                         LineString and translation method 3.2
21                                 end
22                             end
23                         end
24                     end
25                 end
26             end
27         end
28     end
29     return Result
30 End Function
```

A.5.3 Method 4

```
1  Function translateMrwaRoadNetwork(DataSet):
2      Result ⟵ set to DataSet
3      foreach Road in DataSet do
4          foreach Individual in Road do
5              if Individual is MRWA RoadSection and not left/right
                carriage way then
6                  LineString ⟵ get line string of Individual
7                  Points ⟵ get points of LineString
8                  foreach Point in Points do
9                      if Point is not translated then
10                         PointNearest ⟵ get nearest Landgate point
11                         if PointNearest is less than 25.00m offset then
12                             PointOne ⟵ nearest Landgate point
13                             PointTwo ⟵ nearest Landgate point (must be
                                different object as object from PointNearest)
14                             Line ⟵ between PointNearest and PointOne
15                             PointTranslated ⟵ nearest point to Line
16                             if PointTranslated equals PointNearest then
17                                 Translation ⟵ set translation method 4.1
18                             else
19                                 Translation ⟵ set translation method 4.2
20                             end
21                             if PointTwo and PointOne at same Road and
                                PointNearest and PointTwo are within
                                25.00m offset to Point and Point is within
                                polygon of PointNearest and PointTwo then
22                                 PointTranslated ⟵ set to Point
23                                 Translation ⟵ set translation method 4.0
24                             end
25                             LineString ⟵ update with PointTranslated
                                and add Translation
26                         end
27                     end
28                     Result[Road][Individual] ⟵ update LineString
29                 end
30             end
31         end
32     end
33     return Result
34 End Function
```

A.5.4 Method 5

```
1  Function translateIntersectionToNearestTranslatedMrwaRoadSection(DataSet,
   TranslatedRoadSections):
2      Result ⟵ set to DataSet
3      foreach Intersection in DataSet[MRWA][Intersections] do
4          Point ⟵ get point of Intersection
5          ConnectedRoadSections ⟵ get road sections connected to
            Intersection
6          Carriageway ⟵ get road sections with left or right carriageway
            of ConnectedRoadSections
7          if Carriageway count is 2 and all road sections are at the same
            road then
8              PointTranslated ⟵ create point of Carriageway mean
                values Translation ⟵ set translation method 6.2
9          end
10         if PointTranslated is not set then
11             foreach RoadSection in TranslatedRoadSections do
12                 PointTranslated ⟵ determine nearest translated MRWA
                    road section point
13                 Translation ⟵ set translation method 6.1
14             end
15         end
16         if PointTranslated is set and not equal to Point then
17             Result[MRWA][Intersections][Intersection] ⟵ update
                Point with PointTranslated and add Translation
18         end
19     end
20     return Result
21 End Function
```

A.5.5 Method 6

```
1  Function
     translateIntersectionToCentreOfSlipLinestring(DataSet):
2      Result ⟵ set to DataSet
3      foreach Intersection in DataSet[MRWA][Intersections] do
4          Point ⟵ get point of Intersection
5          if Point is not translated then
6              RoadSections ⟵ get Landgate road sections in 16.00m offset
7              RoadSectionGroup ⟵ group RoadSections by road
8              if RoadSectionGroup has road sections grouped in 3+2 or
                5+1+1 then
9                  MiddleElement ⟵ get the road section in the middle of
                    RoadSectionGroup (e.g. road section 2 of 3 or 3 of 5)
10                 LineString ⟵ get line string of MiddleElement
11                 if MiddleElement is set then
12                     PointTranslated ⟵ get mean point of LineString
13                     Translation ⟵ set translation method 7
14                     Result[MRWA][Intersections][Intersection] ⟵
                        update Point with PointTranslated and add
                        Translation
15                 end
16             end
17         end
18     end
19     return Result
20 End Function
```

A.5.6 Method 7

```
1  Function
     translateMrwaRoadNodeToTranslatedMrwaIntersection(DataSet):
2      Result ⟵ set to DataSet
3      foreach Intersection in DataSet[MRWA][Intersections] do
4          CurrentTranslation ⟵ get translation method of Intersection
5          if CurrentTranslation is translation method 6.1, 6.2 or 7 or
             intersection is not translated then
6              Point ⟵ get point of Intersection
7              ConnectedRoadSections ⟵ get MRWA road sections
                 connected to Intersection
8              foreach RoadSection in ConnectedRoadSections do
9                  LineString ⟵ get line string of RoadSection
10                 FirstPoint ⟵ get first point of LineString
11                 LastPoint ⟵ get last point of LineString
12                 if FirstPoint and Point are in 6.20m offset then
13                     PointTranslated ⟵ set to FirstPoint
14                     Translation ⟵ set translaton method 8
15                 else if LastPoint and Point are in 6.20m offset then
16                     PointTranslated ⟵ set to LastPoint
17                     Translation ⟵ set translaton method 8
18                 if Translation is set then
19                     Result[MRWA][Intersections][Intersection] ⟵
                         update Point with PointTranslated and add
                         Translation
20                 end
21             end
22         end
23     end
24     return Result
25 End Function
```