Department of Spatial Sciences

Developing an Agent-Based Framework for Intelligent Geocoding

Matthew John Hutchinson

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

March 2010

DECLARATION

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment 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: ______________________________

Matthew John Hutchinson

Date: March 8, 2010

ABSTRACT

Geocoding is essential to translating a physical address such as a house, business or landmark into spatial coordinates which are used in a range of everyday activities. Geocoding is an active area of research, both within the literature and also in industry. Despite progress in the field, there remains a small portion of addresses which are difficult to geocode. The purpose of this research is to explore the use of agent-based techniques to add intelligence to the geocoding process. The importance of the research stems from its potential to move geocoding in a new direction, by complementing current theory and practice with control and knowledge improvements which will improve geocoding results. The investigation was undertaken by identifying the issues relevant to intelligent geocoding, designing an agent-based solution and building a prototype. The prototype was then evaluated using sample addresses to assess its quantitative performance, and its qualitative performance was evaluated based on the new functionality it provided. Results indicate that intelligence in geocoding is a product of both context and semantics (at a conceptual level) and control and knowledge (at an implementation level), where the two are "connected" by the agent paradigm which is *both* a representation and a solution. Other conclusions include that further development in learning and semantics in geocoding would allow the knowledge base to infer new knowledge and store insights regarding the spatial cognition of users.

ACKNOWLEDGMENTS

I am very grateful to my thesis supervisor, Professor Bert Veenendaal, who has been a constant source of support and mentorship throughout the research. He always had time to meet, provided constant and prompt feedback, was patient and understood how emotionally demanding it can be to write a thesis. It has also been a pleasure to work with the staff and students at the Department of Spatial Sciences at Curtin University, beginning with my original enrolment there in 1999. This research would not have been possible without the financial support of the Cooperative Research Centre for Spatial Information who provided a full scholarship, and various other opportunities during my studies. I would also like to thank the Geography Department at the University of Akron, who hosted me as a Visiting Academic and provided the support which helped me complete my thesis in the United States. Without the support of my family, both in Australia and the United States, I would not have been able to complete this research. My parents have always valued education; their guidance and encouragement has been a big factor in my life's achievements. My wife, Kathleen, is an inspirational person and her support during this research has been invaluable, I will never forget the contribution she has made.

Dedicated to my grandfathers, two ordinary men who demonstrated extraordinary courage at home and abroad to provide the freedom and privilege I enjoy today.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

GEOCODING is translating a physical address such as a house, business or landmark into spatial coordinates. The tendency of people not to use spatial coordinates for everyday tasks is reflected by the fact that people define locations represented in terms people can understand, such as place names (Hill, 2006), and often may only have a name to work with without knowing the location to which it refers. Addressing provides a convenient way for people to remember, and express, locations. Although addresses are "spatial", they contain no inherent connection or reference to an absolute location, other than the fact that an address does (or did) occupy a unique location on the Earth. Establishing this link between address and absolute location is the challenge in geocoding. The purpose of this research is to add intelligence to the geocoding process. Part of adding this intelligence is re-evaluating how geocoding is performed, and analysing how the process can be overhauled to improve geocoding and enable the process to be compatible with ongoing and significant improvements relating to computing and the Internet. The vision for intelligent geocoding involves advances in the areas of (i) flexibility, (ii) context, (iii) semantics, (iv) learning, (v) interaction and (vi) user perception.

This geocoding process is used in applications such as health, business, navigation and emergency services. Geocoding has found particular success in mainstream applications such as car global positioning systems (GPS), and increasingly developed mobile phone and smart phone location-based applications with mobile GIS. It is also a vital component of emerging technologies such as smart phones and services that are increasingly reliant on location. Street addressing was designed with no regard for absolute geographic coordinates. The availability of computing and geographic information systems (GIS) has meant more can be done with information that has a spatial component, by bringing together two fields: street addressing and absolute positioning. Because GIS performs operations using absolute geographic coordinates, any real world problem solving or analysis involving the location of buildings or other features requires a translation from address to absolute location.

1.1 Contemporary Geocoding

The geocoding process is well documented in the literature (Goldberg et al., 2007; Karimi and Durcik, 2004; Zandbergen, 2008; Wei et al., 2009; Lee, 2009; Jacquez and Rommel, 2009) and although the terms used to describe the phases can vary, the process itself is straightforward. The geocoding process relies on a combination of techniques from record linkage and GIS. The record linkage is necessary to take the original address submitted for geocoding and accurately identify corresponding records in one or more reference data sources which describe that address (Gu et al., 2003) for the purposes of correcting and verifying the address. After linkage occurs, techniques from GIS assist in assigning geographic coordinates to the address. The geocoding process can be divided into three phases (normalization, matching and locating) which are further described in Sections 2.1.1, 2.1.2 and 2.1.3.

1.2 Issues in Contemporary Geocoding

There are several categories of errors in the geocoding process and these relate to data input, reference data and the underlying process (Zandbergen, 2008; Karimi and Durcik, 2004; Goldberg et al., 2007; Bigham et al., 2009); these categories and the topics within these categories are presented in Table 1.1.

TABLE 1.1: Sources of Error in Geocoding

Data Input Errors (Zandbergen, 2008)	Reference Data (Zandbergen, 2008)	Underlying Process (Zandbergen, 2008)
Levels of Completeness	Completeness (Karimi and Durcik, 2004)	Assumptions made during interpolation (Goldberg et al., 2007)
Syntax	Currency (Karimi and Durcik, 2004)	Underlying Accuracy of Reference Dataset (Goldberg et al., 2007)
Semantic and Geographic	Correctness (Karimi and Durcik, 2004)	Uncertainty in the Matching Algorithm (Goldberg et al., 2007)
Iterative and Compounded	Consistency (Karimi and Durcik, 2004)	Uncertainty during Standardization
	Spatial Accuracy (Karimi and Durcik, 2004)	Choice of Areal Unit used for Geocoding (Goldberg et al., 2007)

Of particular interest are the errors associated with semantic, geographic, iterative and compounded problem addresses; these are given little attention in the literature, but are emerging as a problem type that could be catered for. Within the underlying process, the uncertainty in the matching algorithm is also of interest, as this could be a category for significant improvement, especially if the sub-process can be tied back to the concept of semantics and control. The categories of data input errors, reference data and underlying process are described in more detail in Sections 2.2.1, 2.2.2 and 2.2.3.

1.3 Rationale for the Research

Geocoding is at a transition point where its current capabilities are going to be confronted by advances in the Internet, artificial intelligence and geocomputation. For geocoding to continue to improve, it must not only handle its existing limitations but look to future advances for the opportunities which these bring. This section presents geocoding in light of these advances and the opportunities that are emerging.

1.3.1 The Bigger Picture for Geocoding Enhancement

Society is on the verge of a semantic geospatial web (Egenhofer, 2002), and this mindset needs to be embraced for the field of geocoding. Geocoding must follow the semantic geospatial web as it brings a totally new way of organizing information (Egenhofer, 2002; Passin, 2004), with the ability to find information based on the meaning of spatial and textual queries. The inventor of the web, Sir Tim Berners-Lee, has stated that:

> “The real power of the Semantic Web will be realized when people create many programs that collect Web content from diverse sources, process the information and exchange the results with other programs. The effectiveness of such software agents will increase exponentially as more machine readable Web content and automated services (including other agents) become available”. (Berners-Lee et al., 2001)

This has implications for both the creation and use of information, along with the control of how this information is used in software. With the opportunity afforded by these emerging technologies, the question is how will these affect the phases in the contemporary geocoding process and benefit the current limitations and difficulties in contemporary geocoding?

As reference data continues to improve with the use of authoritative geocode point data sets with verified positional accuracy, techniques such as interpolation will no longer be needed. Although it will take longer, complex sites and rural sites will eventually have reliable geocode data. The result of this is that one of the biggest issues in geocoding, accurate spatial reference data, will be minimized.

While current techniques for address standardization and matching do take into account the idea that users make mistakes, the types of mistakes catered for are limited to those involving language and the order of address elements. Very little consideration is given to mistakes they made with regards to semantics and ontological meaning or similarity.

Geocoding has an established, linear work flow with certain steps, and these should not be thrown out, but the way they are executed and controlled could be improved. Renovating the geocoding process with a new mechanism for control could not only prepare it for the burgeoning semantic geospatial web, but would also provide the opportunity for greater extensibility and modularity.

Emerging from this bigger picture of geocoding enhancement are specific opportunities for geocoder improvement. The idea is that these concepts will improve existing deficiencies and also provide a framework for future geocoder development as the semantic web becomes a reality.

1.3.2 Specific Opportunities for Improvement

Based on the current limitations of geocoding, several ideas have emerged which could be pursued for enhancing the process. These improvements relate to how knowledge is structured and stored, maintaining knowledge, how users perceive addresses, how reference data is accessed, the quality of data and the communication of this, the matching process and allowing the user to understand this process with its implications, and context. These topics are explained in Section 2.3.

1.4 Considerations for an Intelligent Solution

Definitions of artificial intelligence (AI) vary, but Negnevitsky (2002) states that "The goal of artificial intelligence as a science is to make machines do things that would require intelligence if done by humans", a definition originally presented by Boden (1977). Myers (2004) define intelligence as "the ability to learn from experience, solve problems, and use knowledge to adapt to new situations", which is very much in line with the vision for intelligent geocoding. The definition by Fogel (1995) that "Any system... that generates adaptive behaviour to meet goals in a range of environments can be said to be intelligent" reiterates that intelligence involves adaption and meeting goals. In looking towards what could shape an intelligent solution, it is proposed that the solution can be best described by three factors which are control, knowledge representation and learning.

Control relates to the cause, sequence and technique for executing programming code in software. There are different techniques for control in software and advantages associated with their use. Section 2.4.1 details the traits desirable for intelligent geocoding. Also presented in Section 2.4.1

are the properties of a particular programming paradigm (software agents) relating to control; there is commonality between these agent traits, the desired traits for control and traits associated with intelligent behaviour.

An intelligent geocoding framework requires *knowledge*, which are the concepts and data structured in a way the computer can use, needed specifically to store geocoding expertise (relationships and meta-information), query this expertise and create new knowledge. An intelligent geocoding solution will require a mechanism for acquiring and storing knowledge.

Learning is defined by Negnevitsky (2002) as involving "adaptive mechanisms that enable computers to learn from experience, learn by example and learn by analogy", also stating that "learning capabilities can improve the performance of an intelligent system over time". It is this type of capability which is needed for geocoders to dynamically update their own knowledge.

1.5 Objectives

The overall aim of the research is determining *how to add intelligence to the geocoding process*. The research objectives which emerge from this concept include:

1. **Identify issues and problems relevant to intelligent geocoding.** The topics of control, knowledge and learning identified in Section 2.4 provide the categorization for issues which will be analysed. It will be determined what are the most immediate and important issues that could be enhanced by intelligence. The question of what is intelligence as it pertains to geocoding will also be investigated. Other questions include (i) whether it is possible that there are other categories of problem addresses which have not been identified yet, (ii) what are the desirable properties of an intelligent geocoder, and (iii) what are the benefits of including control, knowledge and learning into the geocoding process?

2. **Investigate and develop a framework for how to build intelligence into the geocoding process.** If intelligence is needed, then how does the existing geocoding process have to be changed

and updated? Is intelligence required in all stages of the geocoding process, and what impact does this have on the correction techniques and reference data used? It will be determined how intelligence can be added now and provide immediate benefit, while also ensuring the framework is extensible. Is it possible to take a linear process (geocoding) and make it more flexible, event-driven and service oriented, as the Internet is? What roles will semantics and context play in an intelligent geocoding framework? A prototype intelligent geocoder will be built to incorporate aspects of the insights discovered during investigation.

3. **Examine and evaluate how control and knowledge can be used to build intelligence into a geocoding framework.** What are the knowledge requirements for an intelligent geocoding framework, and how do these compare for control knowledge and domain knowledge? Can domain knowledge be infused into the control process, and conversely can insights learned during processing be kept for future use? If knowledge is kept, how should it be structured and queried? Is learning possible within geocoding, and what are the implications for learning in this field? The best ideas for control and knowledge use in geocoding will be included in the prototype.

1.5.1 Overview for Pursuing Objectives

Objective 1 will be pursued via a literature review in which the opportunities for improvement in geocoding are identified, and priorities are determined. These findings will be used to shape the investigation and development in Objective 2, where the findings will help determine what "intelligence" is in terms of geocoding and how this can be built into geocoding. Three steps for evaluating Objective 2 include (i) conceptualizing a solution, (ii) designing a framework, and (iii) implementing a prototype. Objective 3 will examine the prototype developed in Objective 2, to evaluate to what extent control and knowledge were present. To do this three steps will be used which include (i) a quantitative evaluation of intelligent geocoder behaviour, (ii) evaluation of the intelligent paradigm for the geocoding process, and (iii) examining derived functionality. A more detailed method for examining all the research objectives is presented in Chapter 3.

1.5.2 Scope

Intelligence will be built into geocoding using an agent based approach because of the desirable traits that agents exhibit, particularly with regards to software control; these characteristics are presented in Section 2.4.1. The focus of the research for the framework will be establishing a mechanism for control (including context and semantics), and to a lesser extent knowledge and learning will included; the included knowledge and learning is more to demonstrate the potential that exists. The control is seen as the most fundamental component, which is why the focus is on it. Of the three phases in geocoding (normalization, matching and locating) the framework will cover all phases and the prototype will demonstrate a complete geocode work flow from start to finish.

Improvements in correction techniques and emerging address problem types will be focused in the matching phase. Improvements in control will be seen throughout all phases. The research will not include any improvement in the normalization phase *with regards to assigning meaning to the individual elements* nor in the locating phase *with regards to positional accuracy*. The rationale behind this is that advancements in these phases done by other researchers can be included in the intelligent geocoding framework if desired. In terms of handling problem addresses, the emphasis is put on handling semantic and geographic addresses, and iterative and compounded addresses. Figure 1.1 illustrates the scope of the research. The prototype is a sub-set of the functionality described in the entire system, as the purpose of the prototype is to not move beyond the "proof of concept" stage.

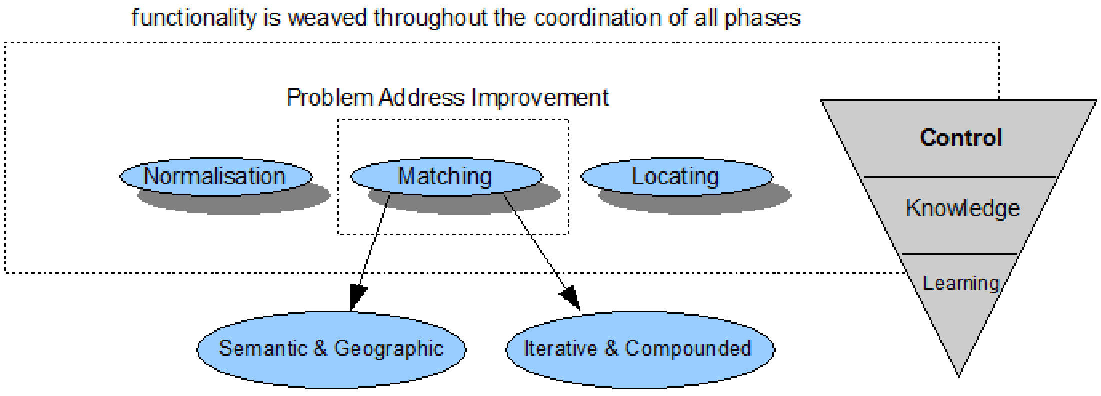


FIGURE 1.1: Scope of the Research

A key aspect of control will be adding parallelism to the geocoding process, and demonstrating that geocoding can performed and coordinated using distributed processing. Parallelism will be investigated at three levels, including: (i) intra-agent, (ii) inter-agent, and (iii) query. The intra-agent parallelism would allow a single agent to perform multiple tasks at the same time, within a single plan. Inter-agent parallelism would offer distributed processing which would utilize messaging between the agents to seek needed information and maintain awareness of other agents. Query parallelism is necessary for the geocoder to process multiple queries concurrently in the system. Associated with this query parallelism is the need to track the queries as they move throughout the system.

1.6 Contribution to GIScience

The research has several contributions, targeted both for the short-term improvement and some for longer-term improvement. In the short-term, the goal is to improve geocoding, which a vital part of GIS and especially location based services and navigation. In the long-term, geocoding will continue to evolve and improve, however it will also have to become "compatible" and integrated with the semantic web. Agents could assist with geocoding making the transition to the semantic web, by orientating the geocoding process towards an event-driven approach and aligning it with increased service usage on the Internet. At a broader level of contribution, the research also furthers understanding of modelling geographic entities with agent based techniques; specifically treating the geographic elements as "actors". Related to this is the use of messaging between agents to model the semantic relationships between address elements. The research provides a definition of what intelligence in geocoding is, and uses this to provide a framework containing numerous components of novel functionality. The framework is extensible, and provides a realistic model for distributed and parallel geocoding in software. In the field of geocoding, it appears to be the first time agents have been used (also in conjunction with a rule based system). In addition to adapting this paradigm successfully, the research has also introduced the idea that geocoders can dynamically improve over time based on previous experience (without human intervention).

1.7 Thesis Structure

With Chapter 1 establishing the need for geocoder improvement and providing the objectives for the research, Chapter 2 examines the suitability of agents for the purpose of architecture and implementation. Throughout Chapter 2 the strengths and suitability of the agent paradigm are presented, in addition to how the various agent aspects fit together for a single approach. The Research Design chapter (Chapter 3) presents how the research needed for the objectives presented in Chapter 1 will be pursued, including the metrics and observations used for evaluation. The Prototype Development chapter (Chapter 4) incorporates the geocoding issues and objectives from Chapter 1 in the design and implementation of a prototype. The scope for the prototype functionality is given in Chapter 1; also the method for developing the software is discussed in the Research Design chapter (Chapter 1). The Results chapter (Chapter 5) presents how the geocoder performed, both in quantitative and qualitative terms as established in the Research Design. The results also form the basis for addressing the objectives presented in Chapter 1. Interpretation of the results occurs in the Conclusions chapter (Chapter 6), where the metrics presented in the Research Design chapter are used in interpretation. Conclusions about the agent based approach and contributions to GIScience are also discussed. Ideas for further research and improvement are presented in the Future Work section (Chapter 6).

CHAPTER 2

SUITABILITY OF AGENTS FOR THE FRAMEWORK

THIS chapter provides a more detailed description of the agent paradigm, and highlights why the agent paradigm is suitable for an intelligent geocoding framework. First, the direct benefits are presented and then additional background detail about agent software is provided which builds on the benefits listed. The summary of agent suitability ties together the agent framework into a whole. The benefits of using agents presented in this chapter are for the most part common to any agents using the BDI model, and implementation level details (i.e. specific only to a certain project or product) are avoided unless common among several frameworks, or pivotal to the anticipated solution.

2.1 Phases in the Contemporary Geocoding Process

The three phases of the geocoding process are normalization, matching and locating. In general terms, normalization massages the address into a standardized form which can then be used for matching. Matching involves trying to find the address element values in reference datasets to confirm they exist. If and when an address has been matched to a sufficient level of confidence, coordinates are found for the address from reference datasets containing coordinates.

2.1.1 Normalization

Because it cannot be assumed that the address is error-free and that the address will exactly match with reference data sources, probabilistic record linkage is required (Gu et al., 2003). Within the record linking process, the address undergoes normalization and then matching (Whitsel et al., 2004). Normalization allows the address to be compared to reference data, using the same format as the reference data itself; the more closely an address resembles the format used for reference data the greater its likelihood of being correctly matched (Christen and Churches, 2005). As part of this normalization, the address must be cleaned and standardized. Cleaning typically uses hard-coded rules and look-up tables to alter an address for issues such as (i) variations in lower and upper case, (ii) expanding abbreviations, and (iii) punctuation (Nicoara, 2005). During standardization the address string is first parsed into its individual elements, i.e. the string is tokenized. Meaning is then assigned to the tokens, identifying which tokens represent the various address element types, such as street name and postcode (Lovasi et al., 2007). Using hard-coded rules, look-up tables and reference data to assign meaning is a deterministic approach (Tang and Clark, 2003). Another alternative is to use a probabilistic approach, such as a Hidden Markov Model, to assign meaning to the tokens (Churches et al., 2002). The deterministic approach for standardization is the norm for geocoding software.

2.1.2 Matching

Given an address that has been normalized, the matching process attempts to link the address to a corresponding record in one or more reference data sources (Drummond, 1995). Ideally the address will match exactly with a record in the reference data, based on individual address elements types, values and their order in the address. When trying to link addresses, the search space can be reduced by using a blocking technique (Zhao, 2007). To measure how closely the address matches the potential candidate addresses in the reference data, weights are used to gauge the relative importance of each address element type (Zhan et al., 2006). An element that is more important or influential in determining a match is given a larger weighting (Whitsel et al., 2006). Using these weightings, each potential candidate address in the reference data can be assigned a score; a higher score means the potential candidate is more likely to be a match (Rushton et al., 2006). A minimum match score can be enforced (Tang and Clark, 2003). Using these scores, the potential candidate addresses are designated as either a match, non-match or a tie (Whitsel et al., 2004). The occurrence of a tie would mean that human intervention is required.

If a match is not found, the selection criteria (such as the minimum match score) can be relaxed or the values of the inputted address elements can be modified with the goal that they will better match the reference data (Goldberg et al., 2007). Common techniques for modifying values include word stemming, the Soundex algorithm and Levenshtein algorithm; these algorithms focus on the spelling and grammar of the element values (Zobel and Dart, 1996; Cohen et al., 2003). For an areal address element (such as a postcode or locality), geographically adjacent areas can be substituted. All of these techniques have the motivation of correcting values for the potential mistakes that may have arisen from human input error. Depending on the geocoder implementation, some of these same techniques may also be used in the standardization process.

2.1.3 Locating

To return a geocode, the matched address needs to have geographic coordinates assigned to it. The reference data available to do this includes (i) polygons such as a state, postcode, or locality which have a coordinate assigned to their centroid, (ii) line segments which represent street networks, (iii) land parcels and (iv) address points.

The coordinates assigned will be the best available given the amount of information available in the address and the reference data available. If an address only contains state and locality information, the resulting geocode will be much more generalized than for an address containing locality, street name and street number information. Current geocoding returns a point coordinate for all of the address element types seen in Table 2.1, which means for a state or postcode, the coordinate represents a much larger area than for a street or street number (Strickland et al., 2007). In the case of a street name with no further information, the coordinates of the point halfway along the street are returned; the longer the street the less useful these coordinates are.

Coordinates for state, postcode and locality can typically be found in a gazetteer style data set (Goldberg et al., 2007), where coordinates are stored along with the name of the entity. The storage of this data can be in a GIS (stored spatially) or in a flat file (stored textually). Street data is typically stored in a GIS, which provides topological information. Coordinates for the street number can be interpolated using the street network, or can be looked up in an address point dataset. Address point datasets such as the Geocoded National Address File (G-NAF) (Richards and Paull, 2003) in Australia and ADDRESS-POINT (Ordinance-Survey, 2003) in the United Kingdom contain coordinates for the primary building located at the address; these datasets are the most accurate way of locating an address. In the case of an interactive geocoding session, the final geocode is returned to the user and displayed on a map. For batch geocoding or a service oriented query between two machines, the geocode is stored or returned without visualization.

TABLE 2.1: Address Element Types

Address Element	**Geometry Type**	**Desired Coordinate**	**Coordinate Search**
State	Polygon	Centroid	Look-up from gazetteer / GIS
Postcode	Polygon	Centroid	Look-up from gazetteer / GIS
Locality	Polygon	Centroid	Look-up from gazetteer / GIS
Street	Line	Mid-Point	Look-up from GIS
Street Type	-	-	-
Street Number	Point	On Feature or Parcel	Interpolation or look-up from GIS
Unit Number	Point	On Feature	Look-up from GIS
Landmark	Point, Line, Polygon	-	Look-up from gazetteer / GIS

2.2 Issues in Contemporary Geocoding

Within contemporary geocoding, the highest level issues include (i) data input errors, (ii) reference data, and (iii) the underlying process. Data input errors relate to the levels of completeness of an address submitted for geocoding, the syntax of the request and underlying semantic (including geographic) causes of error. Issues pertaining to reference data include completeness, currency and correctness (correctness includes positional accuracy and repeatability). Issues within the underlying process include uncertainty in the standardization process, uncertainty in the matching algorithm, match rates and assumptions made during interpolation.

2.2.1 Data Input Errors

The *levels of completeness* for an address relate to the number and types of elements that are in the address. The best case is having all address element types, however often there are varying street number types (duplex numbers, unit numbers, lot numbers), various permutations of missing elements, use of estate names which are not officially recorded, new localities which are not contained in reference data and aliases which are an alternative, non-official description of an address element type. Geocoders could be better at recognizing non-numeric characters and their meaning when used in street numbers. The degree of completeness can affect the uniqueness of the available information contained in an address which is important in determining the size of the search space for the final geocode. There is a balance between the desire to narrow the search space and prioritize possible matches but the questions arises of whether to eliminate a potential match if there is any chance the suggestion could be the intended result. This need for prioritization is why match rates are used, and why it is the norm in geocoding that not every possible match is considered equal (Tang and Clark, 2003). This is related to whether an otherwise incomplete address could be improved and geocoded if more information can be found. The completeness of an address is also directly related to another error, the choice of areal unit used for geocoding (Ratcliffe, 2001). If the amount of information in an address only allowed for a geocode at the postcode level, then using this for spatial analysis can have implications as it is much broader than a street or house level geocode.

Syntax relates to the characters (letters, numbers, punctuation) that make up words and the arrangement of words in an address (Zobel and Dart, 1996). This includes the number of the various address elements, the order they are in and errors that people commonly make based on language. An example is a street name with two or more words (e.g. “The Promenade”), where some geocoders interpret the second word as the street type. Punctuation such as the use of duplex numbers/letters (e.g.

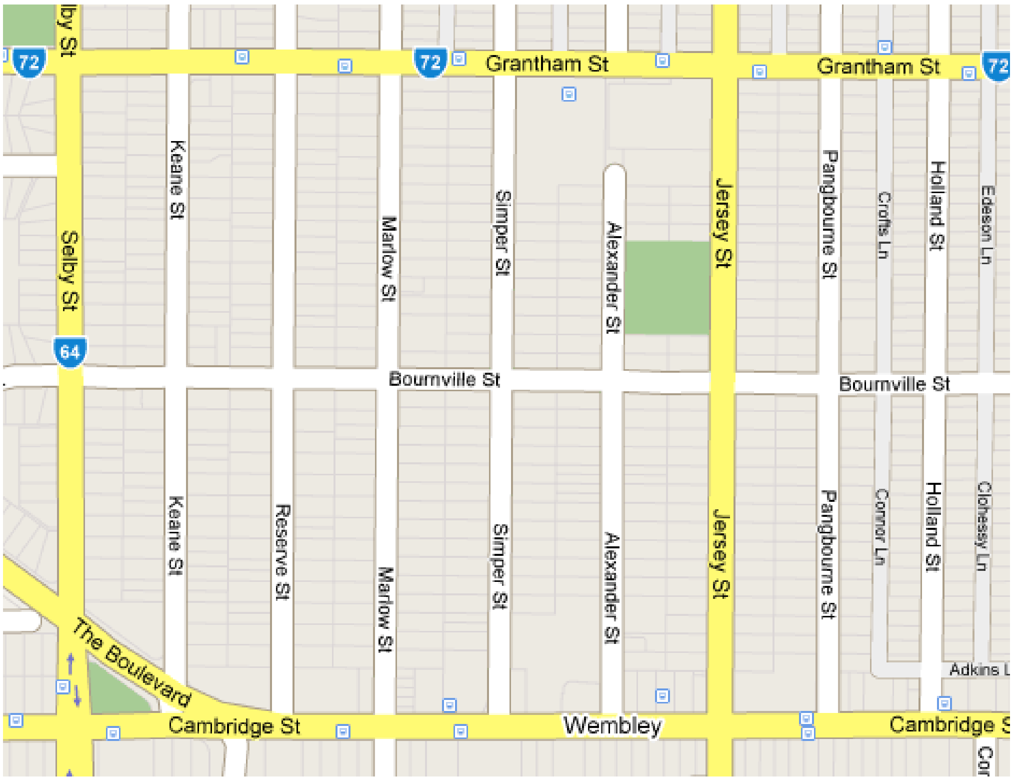


FIGURE 2.1: Spatial and Attribute Similarity of Streets

"48A" or "U3/154"), rural lot numbers (e.g. "Lot 310") and slashes can all make it harder to correctly parse an address and determine the meaning of elements, which is directly related to the standardization process.

The presence of particular address elements types (such as locality, street type) can be critical for determining the meaning of address elements, and is why syntactic mistakes can derail standardization and is troublesome when elements cannot be corrected. Other examples relating to syntax include similarity of an element value to other words ("Shiraz" and "Chiraz"), linguistic (unexpected spellings, e.g. "Nangarra" should be "Gnangara"), phonetic (i.e. verbal linguistic) (e.g. "Mushay" should be "Muchea"), spelling mistakes for various elements (typos, e.g. "Heskath" should be "Hesketh"), and also combinations of spelling mistakes with certain elements missing, conflicting elements (such as postcode and locality), abbreviation (e.g. "Vic Park" should be "Victoria Park"), non-abbreviation (e.g. "Mt Claremont" should be "Mount Claremont"), hyphens ("Pinjarra-Williams Rd"), numbers versus roman numerals ("Australia 2 Dr" should be "Australia II Dr"), reversed directional information ("Perth

South” should be “South Perth”), ranged addresses (“102-114 Normanby Rd”) and word stemming differences.

Semantic and geographic issues relate to data input errors based on user perception and relationships of address elements in physical space, respectively. Examples of geographic issues include that (i) parcels can have multiple addresses (complex sites and duplexes); (ii) rural and semi-rural address geocoding (such as using a lot number as a street number, and the difference between lot numbers and newer, revised rural number) has much poorer results than urban geocoding; (iii) different parts of the world have different addressing systems, such as that in Japan; (iv) correct street with neighbouring locality, (v) vanity addressing when an incorrect locality/suburb is substituted for another because it has higher socioeconomic status, (vi) corner addresses, (vii) roads with two names (e.g. “Jayes Rd” and “Bridgetown Boyup Brook Rd” are the same road), (viii) ontological similarity (e.g. “Small Creek Rd” should be “Little River Rd”), (ix) first and second order neighbouring localities, and (x) address elements that exhibit spatial similarity and semantics. Of particular note is how mistakes can be made where the entities involved (localities) have no similarity in language and the ultimate cause is the perception of the user.

An example of semantic and geographic errors is seen in Figure 2.1, where Grantham Street and Cambridge Street are very similar due to being parallel, having the same number of lanes, the same speed limits, and the same streets running between them. The idea here is that a user could get the streets confused based on their perception of the environment in their mind.

In Figure 2.2, a corner block is shown which could confuse a user when they submit the address for geocoding; in their mind they know where the house is located (and its house number) but may not know which street the house is associated with. Iterative and compounded addresses are those addresses which cannot be solved in their initial form, as there are too many conflicting elements or errors with the various element values. For example, an address has a misspelled street name, uses a neighbouring locality and the name of the neighbouring locality is also misspelled. As a consequence, there is not enough information in the original inputted address to geocode the address. The concept here is that correction for

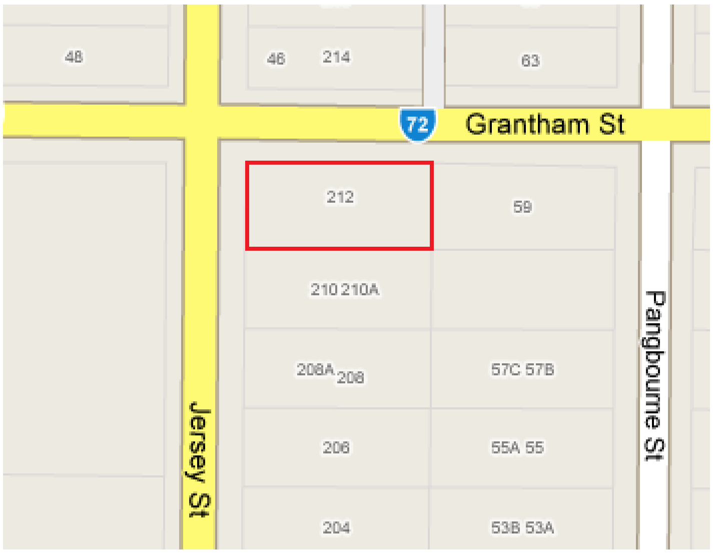


FIGURE 2.2: Corner Address

these types of addresses cannot be done in a single attempt, rather several iterations are required where one element must be corrected, after which this newly corrected element is used to correct another element. This can lead to having several possible addresses, which would have to be ranked. It would also require a mechanism in software to track each of the potential matches as they are pursued.

2.2.2 Reference Data

The overall quality of reference data affects many stages of the geocoding process, and directly affects the ability to standardize, match and locate an address. There may not be anything a geocoding engine can do to improve its reference data, rather the issue is accepting the error and identifying how it affects the geocoding outcome; it is also desirable to communicate this to the user.

The reference data errors of completeness and currency describe what quantity of reference data is available for given geographic regions and how up to date this data is, respectively. For example the ADDRESS-POINT data set includes geocodes for around 26 million addresses, and the product is updated for release every 3 months; the goal is to provide geocodes for all street addresses throughout Great Britain (Ordinance-Survey, 2003). A very similar product, the G-NAF (Richards and Paull, 2003) is available in Australia and also has an update schedule of 3 months. The G-NAF provides a geocode for all valid physical addresses, of which there are approximately 12.6 million. Google Maps has provided a mechanism for users to edit the location of addresses after they have been geocoded, i.e. correct the geocode location based on the user's opinion/experience. This is a prime example of user feedback, and bodes well for the idea that geocoding can be more of an interaction rather than a single query. In terms of completeness, individual geocode data for complex sites (such as blocks of apartments, universities or business offices/suites) and duplexes is not always available, and this issue is reflected in the G-NAF and ADDRESS-POINT data.

Correctness of geocoding data relates to the attributes (textual, non-spatial) and spatial data; correctness can be thought of as the degree to which the data correctly reflects reality. For example are the street names spelled correctly, and are the postcodes associated with the correct localities? The quality of the reference database can significantly affect matching frequencies (Dearwent et al., 2001).

Correctness for spatial data includes, but is not limited to, the issue of positional accuracy. Although positional accuracy is important for reference data (such as road locations and topological relationships), the most studied aspect of positional accuracy is that of the final geocode, establishing how comparable a geocode is to the equivalent position on the earth of a feature. No amount of normalization or matching will make this aspect better, it is reliant solely on the quality of the reference data.

There has been much attention given to positional accuracy in the literature (Bonner et al., 2003; Cayo and Talbot, 2003; Dearwent et al., 2001; Karimi and Durcik, 2004; Ratcliffe, 2001; Schootman et al., 2007; Strickland et al., 2007; Ward et al., 2005; Whitsel et al., 2006; Zandbergen, 2007; Zhan et al., 2006; Zimmerman et al., 2007) as this is inherently important for an accurate geocode. Ratcliffe (2001); Lovasi et al. (2007); Whitsel et al. (2004) also suggest that accuracy is poor, and in a study by Karimi and Durcik (2004), it was found that for the positional accuracy of three geocoding applications being tested only 55%, 43% and 40% of the geocoded addresses were within 45 metres of the GPS measurements. Zandbergen (2008) states that "typical positional errors range from 25 to 168m", and generally the positional accuracy of geocodes in urban areas are better than those in rural areas (Bonner et al., 2003; Cayo and Talbot, 2003; Ward et al., 2005). Cayo and Talbot (2003) go on to state that for rural areas "95 percent of the addresses geocoded to within 2872 metres of their true location", while the same percent for suburban areas "geocoded to within 421 metres". For urban areas, 95 percent of sites "geocoded to within 152 metres of their true location". In an attempt to improve these results, the authors of the paper also used land parcel coordinates instead of TIGER files to perform the geocoding. This technique succeeded by markedly reducing the positional error of geocoding: ninety-five percent (95%) of rural areas "were within 195 metres of the true location", while the same percentage of suburban sites were within 39 metres and urban within 21 metres. Positional error in geocodes can have a negative impact on spatial analysis (Zandbergen, 2008), reducing the ability to identify clusters and trends (Jacquez and Waller, 2000; Waller, 1996; Zimmerman et al., 2007; Burra et al., 2002) it has been shown that is not always accurate enough for analysis in health studies (Zandbergen, 2007; Zandbergen and Green, 2007; Whitsel et al., 2006). Authoritative data sets such as the G-NAF and ADDRESS-POINT provide the best option for spatial reference data in terms of positional accuracy as they are compiled from multiple sources and validated. As early as 1995, Drummond (1995) suggested the development of a nationwide database containing the latitude and longitude of every address in Australia. In addition to the positional accuracy itself, the user is not informed of the accuracy of the geocode (such as "plus or minus 20 meters" or that the geocode is an approximated centroid).

With regards to repeatability, a study by Whitsel et al. (2006) using 3615 sample addresses from 49 states in the USA found substantial differences between the four commercial vendors used (Zandbergen, 2008). Important differences were found between the vendors with respect to match rate, agreement of known census tracts and those from vendors, and the distances between known coordinates and those calculated. This reflects a similar conclusion in Whitsel et al. (2004) that the repeatability of commercial geocoding is not very good (Zandbergen, 2008). Comparing the study by Whitsel et al. (2006) and Whitsel et al. (2004) with a study by Karimi and Durcik (2004) where the three geocoding algorithms were compared and the differences found to be small (using the same reference data), it has been suggested that differences in reference data could be at least partly responsible for the differences in commercial geocoding results.

2.2.3 Underlying Process

Uncertainty in the Standardization Process occurs from the fact that the address elements can have the wrong type assigned to them during standardization. This can be due in part to reference data, but the predefined tables that are often used for cleaning can incorrectly alter words, resulting in the element being assigned the wrong type. The rules used to assign meaning based on the order of elements can also perform incorrectly, which is a reason why a probabilistic approach has been used by Christen and Churches (2005). The same techniques used to try and improve element values can also be used in the matching process, so improved techniques which look for errors based on geography or semantics could be used in the standardization process.

Uncertainty in the Matching Algorithm can occur for a number of reasons, some based on user preferences, the reference data used and techniques used to modify an element value to find a match. Weightings can vary for the elements, and can affect what is determined to be a match; although the greatest weight is usually attached to the street type, the choice of weights for other elements can make a difference. Correction algorithms such as Soundex (Zobel and Dart, 1996) and Levenshtein (Cohen et al., 2003) have their limitations, such as the dependence of the Soundex algorithm on the initial letter, mistyping, extra consonants, swapped consonants, silent consonants and the use of initials (Patman and Shaefer, 2001; Nicoara, 2005). The Soundex algorithm has shown in testing that it produces a high number of incorrect matches (Stanier, 1990). These correction techniques only deal with spelling and grammar, there is no thought given to correction based on semantics or spatial similarity. These and other correction techniques (word stemming, aliases) are important because they only mechanisms for ‘fixing’ an element value when it cannot be found in the reference data.

Match rates are a key quality indicator for geocoding, but have limitations with regard to the quality they communicate. For example, although a geocoder may return a high match rate for a particular set of addresses, the quality of the matches themselves could be low. This is because the match rate does not (by itself) describe the rigor of the match (i.e. the criteria used to determine a match) or the positional quality of the geocode. It is for this reason that match rates can be misleading (Whitsel et al., 2004), as relaxing the criteria for a match will result in a higher match rate. There are also implications for the match rate with regards to data used, as “a match is only obtained in parcel geocoding if there is a perfect match for the house number” (Zandbergen, 2008), which is why parcel geocoding can result in lower match rates; this is also the case for a point data set such as the G-NAF. Street geocoding is more relaxed, as it ”does not provide for a check if the house number actually exists and can therefore result in false positives” (Zandbergen, 2008). It should also be noted that the matching of duplexes and units is different from providing coordinates at the unit level; i.e. just because a match occurs at the unit level does not mean the geocode has positional accuracy at that level.

Assumptions made during interpolation occur when using the street geocoding method, as parcel sizes can be irregular and house numbers can be on the wrong side of the road, out of sequence. Work has been done to take these irregular parcel sizes into account (Bakshi et al., 2004). Despite its shortcomings, street geocoding remains one of the dominant geocoding technique. The choice of areal unit used for geocoding can have implications for how it is used, for example the centroid of a locality may not be suitable for the same spatial analysis (such as a health study) used for a point representing a house. There are also implications for navigation, where any geocode other than a building or parcel level result would be insufficient. Within contemporary geocoding, most processes (start to finish) are linear (Goldberg et al., 2007), and that results in few “optional pathways” able to be considered. This raises the question of whether this can bias outcomes? It would be useful to have the ability to pursue all potential address corrections.

2.3 Specific Opportunities for Improvement

Having discussed the phases in contemporary geocoding and the issues associated with them, there are opportunities for improvement which emerge. These opportunities for improvement are presented in Table 2.2.

TABLE 2.2: Ideas to Enhance Geocoding

Concept	**Description**
Ontologies and Semantic Understanding	Allow the meaning of data for greater flexibility of expression to be used by people and computers for querying
User Perception	Correct addresses for mistakes made by users based on their spatial cognition of the environment, using research in semantics and spatial similarity
High-level and Low-level Knowledge	Using an alias as an example, use "rules" to describe common mistakes both in terms of specific address element values (low level) and in terms of their element types (high level)
Knowledge Maintenance	A mechanism used to ensure reference knowledge (e.g. aliases and other rules) are all maintained at a given level of quality; it is desirable to occur without human intervention and focus on those factors affecting reference data
A Unified Geocoding Model	Defining the entities and relationships used in geocoding with such flexibility and extensibility that the same set of guidelines can be used to represent geocoding as it is implemented worldwide
Data Access	Knowing the context of data needed to solve a query, combined with web service discovery, the most appropriate data can be sought in real-time
Quality of Reference Data	Structure the metadata available in reference data to allow querying and make use of this in decision making at runtime; would allow for knowing what data is suitable for particular needs, and mediate multiple data sets. Other functional outcomes would include being able to (i) select the best dataset if there are several possibilities available, dynamically select using multiple criteria, (ii) store feedback about how well a data source performed, and (iii) update the original data source if it is found to contain errors.

Concept	**Description**
Communication of Quality	Move to a series of other questions which may help refine during the matching process, rather than present a list after the fact. To support this, the geocoder software cannot just take the initial query and process it once; the query must remain open and be able to incorporate more detail over time. This interactivity can continue with communicating the quality of the geocode in real-time while it is being processed, so user can see it be refined over time.
Match Quality	Improvements could include comparing the spatial resolution, the overall agreement of the individual elements within the matched address, and how different the matched address is from the original. The quality of a match should be a combination of (i) whether the query could be found in a reference dataset, (ii) an indication of the extent to which the criteria had to be "relaxed" for a match, (iii) how "similar" the matched address is to the original, and (iv) the associated spatial and temporal accuracy associated with the match.
Transparency in Processing	Part of communicating quality is being able to establish which steps were followed and in what order to arrive at a result. This would include the ability to track multiple queries through the system, which techniques were used and which address elements provided the catalyst for an address match. Extending this concept of "process pedigree" beyond matching, reasons could also be given regarding why certain reference data sets were used.
Reliability of Knowledge	Enhanced geocoding needs the capability to store several metrics, including when the knowledge was created, how often it is used, and whether new knowledge coming into the system repudiates existing knowledge. Correlations would be made to amount of use, creators of information and other indicators.
Context	Include user, application, geographic and jurisdictional context in the execution of geocoding phases (normalization, matching and locating), along with user interaction beginning, during and at the end of processing.

Concept	**Description**
Control	Needs to offer an event based paradigm, capable of the ongoing user interactions, the ability to infuse knowledge into the decision making process, merge multiple sources of information and status indicators into a clear decision making ability.
Learning	Approaches for an intelligent model would be to infer from specific case to another specific case (syllogism), or take a specific case and express it in terms of its general types as to make it applicable to other specific cases (i.e. other queries).

2.4 Considerations for an Intelligent Solution

The key aspects to consider for an intelligent solution include control, knowledge and learning. Control pertains to how the software receives new data, makes decisions and manages processing. Knowledge is storage of domain expertise used for decision making. Learning is essentially the accumulation (automatically) and storage of knowledge over time.

2.4.1 Control

After reviewing programming paradigms and language features in the literature, there are several which seem appealing, including (i) event based, (ii) goal directed, (iii) distributed, (iv) parallel processing, (v) non-deterministic, (vi) meta-programming, (vii) recursive and (viii) object oriented. These programming features are shown in Table 2.3.

TABLE 2.3: Desirable Control Features

Feature	Advantage / Desirable Trait
Event based	Can compliment service oriented software, because the requests from incoming web requests are themselves the events. Although traditional batch geocoding on a desktop machine may have been procedural, geocoding has embraced and is well suited to a service paradigm, both for queries from humans and other machines via the internet.
Goal directed	Advantages from Cichelli and Cichelli (1977) include (i) the top level code is oriented around the user problem, (ii) there is an intuitive structure, (iii) ”control code is separated both logically and syntactically from function code”, that (iv) top down design can assist with nesting control code at lower levels, and (v) The top down approach can also be beneficial to defining data structures.
Distributed	Allows for the clustering of computers, which provides scalability in terms of processing, cost and the amount of requests it can handle. The system can be easier to manage, can be geographically decentralized and ensures there is no single point of failure. Each software entity has its own local memory. Entities communicate with each other by using message passing, communication occurs over the network.
Parallel	Multiple operations can be done in parallel on a single machine if it has multiple processors. This is especially useful if only one machine (i.e. not distributed) is used and it has many processors.
Non-deterministic	Ability to choose most appropriate action at runtime, with the ability to rollback a choice if it fails and choose from other alternatives, where the choice is not directly specified at design time (i.e. in code by the programmer). (Sterling and Shapiro, 1986)
Meta-Programming	Software can reprogram itself and because of this can handle new situations without recompilation
Recursive	Allows a function/method in code to call itself dynamically and (at design time) an unspecified number of times, where stopping is dependent on an exit criteria.
Object Oriented	Useful for modelling real-world phenomena and situations, and brings its own advantages including information hiding, data abstraction, encapsulation, modularity, polymorphism, and inheritance (Pierce, 2002).

Some of these techniques are not necessarily all contained within the field of AI, but nonetheless provide important benefits which could enable an intelligent solution. Event based, recursion and object oriented programming is available in most languages, such as Java (Horstmann and Cornell, 2007), C++ (Deitel and Deitel, 2009) and Python (Lutz, 2009). Non-deterministic programming is available in Prolog (Sterling and Shapiro, 1994), but is not found in more mainstream languages such as Java and C++. With extra frameworks, distributed and parallel processing are available for languages such as Java and C++. Goal based processing seems most common in systems developed around the belief, desire and intentions (BDI) model, such as PRS (Ingrand et al., 1992) and dMARS (D'Inverno et al., 2004). Agent oriented programming frameworks based on BDI also exhibit all of the desirable traits shown in Table 2.3 (Georgeff and Rao, 1995).

Just as object oriented programming is a software abstraction, so are software agents except they are *agent oriented*, not *object oriented*. Agents are seen as self-contained software entities, and have the properties seen in Table 2.4 (Jennings et al., 1998; Russell and Norvig, 2003):

TABLE 2.4: Agent Characteristics

Characteristic	**Description**
Autonomous behaviour	The agent pursues its own agenda and does not need to be explicitly told by a human or another agent how to achieve it.
Established objectives	Whether specified by a human at runtime, or hard-coded at design time, the agent has objectives which it pursues. It is through knowing these objectives, it is able to pursue them autonomously.
Control of its own actions and internal state	Unlike the objects used in object-oriented programming which are directly modified and have no choice in their state, agents have full control over the changes made to them and inside them.

Characteristic	Description
Perceive their environment	Although agents are often software based in a computer (some are hardware based too, such as in robots), they are considered to be more than a ”program” which is run once and then terminated. Rather they are situated in an environment, in which they perceive new information (percepts) and in turn act on the environment themselves (actions).
React accordingly	The agent has an ability to choose a course of action most appropriate to the situation at hand. The sophistication of an agent in choosing a response is a key aspect of the categories defined by Russell and Norvig (2003).
Opportunistic	Given a current situation, or changing situation, the agent can adjust its actions in the short-term to maximize the prospects of finding success in the long term.
Goal Directed	This means the agent knows what it should achieve but is not explicitly told how to do it. Part of being goal directed is the flexibility to be opportunistic, and to try alternative actions if those already tried do not succeed.
Interaction	Although agents do not require assistance to do their processing, they can choose to initiate communication with others (human or other agent) if it will advance their own agenda. Similarly, if an agent is asked a question it can choose to respond or not.

The goal-directed approach has advantages as outlined by Cichelli and Cichelli (1977) which include that the top level code is oriented around the user problem, there is an intuitive structure, “control code is separated both logically and syntactically from function code”, top down design allows for nested control code at lower levels, and that top down approach also applies to data structures.

2.4.2 Knowledge

To utilize knowledge, a knowledge level model can be developed to suit the domain of geocoding, which in this case is a “particular subject matter of interest in some context” (Uschold, 1998). A domain (in this case geocoding) is understood (i.e. organized by the human mind) concretely via a conceptualization. A conceptualization can be expressed in many ways, including simple written language, diagrams and ontologies.

As people we have a particular conceptualization of what geocoding (and more generally, space itself) is and what it involves. Our conceptualization of geocoding is defined by our experiences with visiting addressed locations, as well as our general knowledge of how street addressing operates and even the standards that define addressing in particular places. A knowledge level model for geocoding would have to include the relationships and entities common to it - at a minimum the address components (such as states, localities, post codes, streets, house numbers). Beyond this, as semantics is involved more, directions, non-spatial attributes and other factors could be modelled.

To make the knowledge level model more definitive, an ontology can be developed which is described by Gruber (1992) as *"a vocabulary of terms (names of relations, functions, individuals), defined in a form that is both human and machine readable. An ontology, together with a kernel syntax and semantics, provides the language by which knowledge-based systems can interoperate at the knowledge level: exchanging assertions, queries and answers."* Tecuci (1998) states that "An ontology contains the objects, concepts, and other entities that exist in an area of interest, as well as the relationships that hold them together". Passin (2004) explains that the ability to specify ontologies, vocabularies and represent knowledge is made possible by the use of logic. Advantages from using logic are described by Passin (2004) as (i) applying and evaluating rules, (ii) inferring facts that haven't been explicitly stated, (iii) explaining why a particular conclusion has been reached, (iv) detecting contradictory statements and claims, and (v) combining information from distributed sources in a coherent way. Welty et al. (1999) suggests various purposes for using ontologies, and these include (i) reuse and sharing, (ii) interoperability, (iii) structuring knowledge bases, and (iv) browsing and search.

With a knowledge level model defined, an ontology defined, and knowing that there are benefits to using logic a representation is needed for use in software. The main conclusion from Newell (1982) is that the knowledge level is a level above the "symbol level" (symbol level being the implementation level); the key to understanding this is that "a representation is the structure at the symbol level that realizes knowledge”. There are choices with regards to the approach used for representing the ontology. However, regardless of choice there is commonality amongst techniques in that both (i) will result in the creation of a knowledge base as concrete data is added to the ontological structure i.e. the knowledge base can be thought of as an instantiated form of the ontology, (ii) will require an inference engine to query the knowledge stored in the knowledge base, and (iii) rules can be written in terms of the individual assertions in the knowledge base. The two major choices for representation include the description logic (Russell and Norvig, 2003) based and frames based (Minsky, 1974) approach. For implementing the description logic based approach, the web ontology language (OWL) (Mcguinness and van Harmelen, 2004) has distinctive reasoning capabilities, while for frames there is no single standard for representation analogous to OWL (Wang et al., 2007). There are similarities and differences between the two techniques, in terms of expressive power, semantics, tool support and guidelines for usage; an comparison of the two is presented by Wang et al. (2007). An example of building an ontology is found in Noy and McGuinness (2001), where the focus is on a frame based approach but the methodology is still relevant to the OWL approach.

Van Rees (2003) describes the difference between a taxonomy and ontology, with a taxonomy being described as ”as a hierarchy created according to data internal to the items in that hierarchy” and more simply as a ”simple ontology”. Van Rees (2003) also suggests that “Once a lot of properties and relationships are added to a hierarchical structure, the term ‘ontology’ is better suited than ’taxonomy’ ”. Table 2.5 from Morris (2008) shows some of the differences between a taxonomy and an ontology.

TABLE 2.5: Taxonomy vs. Ontology (Morris, 2008)

Taxonomies	**Ontologies**
Usually are a single, hierarchical classification within a subject	Subsume taxonomies
Primarily focused on "is-a" relationships between classes	Include attributes with cardinality and restricted values
Limited in inference potential due to lack of relational expressiveness	Unlimited relationships between entities
	Superior inference support due to relational expressiveness

The inference engine used will depend somewhat on the knowledge representation used. Two types of inference engines are a semantic reasoner (Sirin et al., 2007) and an expert system (Friedman-Hill, 2003). Expert systems are rule based, and use a declarative programming paradigm, as opposed to typical procedural programming (Negnevitsky, 2002). Declarative software differs by being told what needs to be done, but is not given specific, pre-defined instructions on how to achieve it. Rule-based systems work by reaching conclusions and performing actions based on the premises it has. The system's ability to reach conclusions is only as comprehensive as the rules it has stored in its rule base, however new rules can be derived from existing rules. Both OWL and frame based representations can be reasoned over with an expert system, however expert systems have their origins in frame based knowledge representation (Mei and Bontas, 2004; Biondo, 1990). Additionally, there are things that can be done in a semantic reasoner that cannot be done in an expert system.

2.4.3 Learning

The field of learning within artificial intelligence is a vast one, but a critical fact is that there are three main types of learning, namely supervised, unsupervised and reinforcement learning. In addition to these is the process of logical inference, which can occur on the knowledge inside a knowledge base. The types of learning can be seen in Table 2.6.

TABLE 2.6: Types of Learning

Learning Type	Description	Training Data
Supervised	Inductive	Inputs and outputs
Unsupervised	Summarizes data	Unlabeled inputs only
Reinforcement	Receives reward input	Inputs only
Logical Inference	Deductive	Existing knowledge

As described in Russell and Norvig (2003), supervised learning has training data which includes both inputs and resulting outputs for a number of examples cases. Unsupervised learning has only unlabeled input data, i.e. the algorithm does not even know what the data is representing (no names are given, no categories, thresholds etc.), the objective here is to describe the data as best as possible using discovered patterns or commonalities in the data (a common technique is clustering). Reinforcement learning also has unlabeled input values, but has a performance critic which provides desirability feedback (e.g. "good" or "bad", "correct" or "wrong") associated with the inputs. Given these types of learning, the question arises of which type(s) are most relevant to geocoding, specifically as what is available in the way of training data and feedback.

One possibility for the use of supervised learning in geocoding is using decision tree learning, where each node in the tree is a test, and the branches are possible values (Russell and Norvig, 2003). The known input would be the incorrect address being geocoded and the known output would be the corrected address and optionally a label of the associated type of "problem" address. The idea would be to create a decision tree which could eventually be applied to incoming, incorrect addresses and by looking for specific (known) errors fix the address with the prescribed techniques (e.g. language, geographic, semantic correction). The decision tree could also be converted into rules for use in an expert system. The challenge would be to find a solution where the address elements values, their underlying cause and the correct addresses can be reconciled into decision tree learning. A potential problem with this idea is that solving a geocode involves not just the "face value" of address elements but the underlying relationships and causes behind the values; this would have to be accounted for in the learning algorithm. Using a supervised learning approach would only be effective if a sufficient number of address training data could be found; even finding the data of incorrect and correct address pairs could be difficult.

Reinforcement learning provides another opportunity, where the success of geocoding could be used as the performance critic. In this idea, the only inputs would be the incorrect addresses and whether the incorrect address could be corrected and geocoded. If the corrected address was the only solution (i.e. there were not multiple potential matches) then some conclusions could be made because the incorrect and correct addresses equate to each other. For example, if the locality was changed during correction, then an alias could be generated detailing this modification. Several address elements could be used, to be more specific. For example an alias could be made for a street name, but only in a particular locality and state. The idea is that these aliases are used in future geocoding and could be applied to incorrect addresses. An advantage to this reinforcement learning approach is it does not require training data of incorrect and correct address pairs; it also requires no human intervention. A disadvantage is that making any extrapolations (i.e. aliases) beyond the very specific incorrect address could be incorrect, and of course a newly stored alias may not be applicable in every future geocoding processing, it could only serve as a suggestion. Another aspect is that although the aliases are stored, the algorithm is creating these aliases from the correction techniques (e.g. language, geography, semantics) so what is stored is nothing that could have been determined anyway. However when done over time with enough input data, trends could be found in how these corrections are applied and the stored aliases could have worth (value added) beyond just the correction techniques.

It is the deductive reasoning made possible by logical inference which has the potential to create new knowledge from existing geocoding knowledge, by (i) using specific values from one query to help solve another query, and (ii) moving from specific values to expressing the same ideas as types.

An example of using specific values is combining two disparate pieces of information to find a new meaning, inspired by Aristotle who pioneered syllogisms, and described in Barnes (1969). For example, in an expert system two rules could be created which specify:

1. In the suburb of Armadale, the street Armadale Rd is contiguous with

Albany Hwy

2. Mt. Richon is a common vanity alias for the locality Armadale

If a geocode query included "Albany Hwy, Mt. Richon" then the expert system would use both rules and provide the suggestions of "Armadale Rd." and "Armadale". The key concept here is that the final outcome would not have been possible without combining the multiple pieces of disparate information. What is interesting with this approach is that these rules would not necessarily have to be added by a human; these could be added using insight from the reinforcement learning approach, i.e. a combined reinforcement and deductive approach. Another idea would also be to find specific instances of a trend or something happening, expressed in specific terms (for example "Pangbourne St", the locality of "Wembley") and express this in terms of "street" and "locality" to make the finding applicable to a wider set of future addresses that need correcting. Reasoning with types could also have uses if the general assertions could be acquired. For example:

- Some streets are within single postcode + some multiple localities are within single postcode, therefore *some streets are within multiple localities*
- Some streets in Churchlands have bird names + Some streets near water have bird names, therefore *Some streets in Churchlands are near water*

Some of these basic assertions could come from humans and some could be calculated by the computer, whether by looking at spatial data, attribute data or a combination of both. The overarching idea is that although these assertions may not solve queries individually, over time as many of these assertions and new conclusions intertwine they will together solve queries.

Artificial neural networks (ANN) are a potential tool for intelligent geocoding. The ANN can be used for supervised, unsupervised and reinforcement learning (Russell and Norvig, 2003). Three specific ideas for using ANNs in geocoding are:

1. Using for standardization, similar to how a hidden markov model (HMM) has been used; possibly used as a classifier where given the

input address as string, the ANN decides which tokens within the string are the various address elements. Perhaps this could be done using the values, positions and other information such as total number of tokens.

2. Using to determine the most prevalent geocoding mistakes for a given region, in trying to solve a query give more attention to certain problems types. This could make the difference in cases where two potential explanations for a given problem are equally as likely.

3. Classify an address into the type of problem address it is with the aim of correcting the address on the basis of knowing the problem type.

These ideas regarding neural networks would require much more research to even determine if the use of ANNs in these ways is feasible; this research falls outside the scope of the current research.

2.5 Agent Definition

Padgham and Winikoff (2004) provide a definition of an agent, which is taken from Wooldridge (2002) and originally adapted from Wooldridge and Jennings (1995); the definition is:

> "An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous* action in this environment in order to meet its design objectives" (p. 1).

Wooldridge (2002) defines an *intelligent* agent as also being *reactive*, *proactive* or *social*.

Intelligent agents have been used for spatial science tasks such as geospatial information retrieval and filtering, geospatial search engines, knowledge discovery, decision model assessment and optimisation (Shahriari and Tao, 2002), and the discovery and analysis of spatial information (Li et al., 2001). Agents have also been suggested as a tool for enabling geographic information systems (GIS) within an Internet environment (Tsou and Buttenfield, 1998).

Some applications use only one agent, but often more than one agent is required due to complexity or conceptual modelling; these are called *multi-agent systems* (Ferber, 1999). Multi-agent systems (MAS) are also referred to categorically as *distributed artificial intelligence* systems (Jennings, 1993); the defining concept is that "multiple agents interact to improve their individual performance and to enhance the system's overall utility" (Jennings, 1993).

Looking at some of the agent properties from Jennings et al. (1998), interaction and autonomous behaviour in particular are standouts with regard to multi-agent systems. Agents are capable of sending messages to each other, which can be as simple as a single message (to which a response may not even be sent) or as complex as a whole session of messaging, using established protocols, between one or more agents. The autonomy means that within a MAS, each agent is "doing its own thing", pursuing its own goals without regard to what the agents are doing. It is up to the individual agents whether or not they initiate messaging and respond to requests. Depending on how the system is designed, the MAS may have competitive or cooperative agents. Each have their own uses, although a cooperative system is very useful as *tasks can be achieved in parallel*.

A specific architecture within goal-based agents is the belief, desire, intention (BDI) model. Desires can be thought of as *goals* the agent wants to achieve, intentions as *plans* that dictate how to go about this and beliefs are *internal data* (Agent-Oriented-Software, 2003). The BDI architecture is shown in Figure 2.3.

The BDI system is modelled on ideas from psychology and philosophy, simplified into a version suitable for computer implementation (Howden et al., 2001).

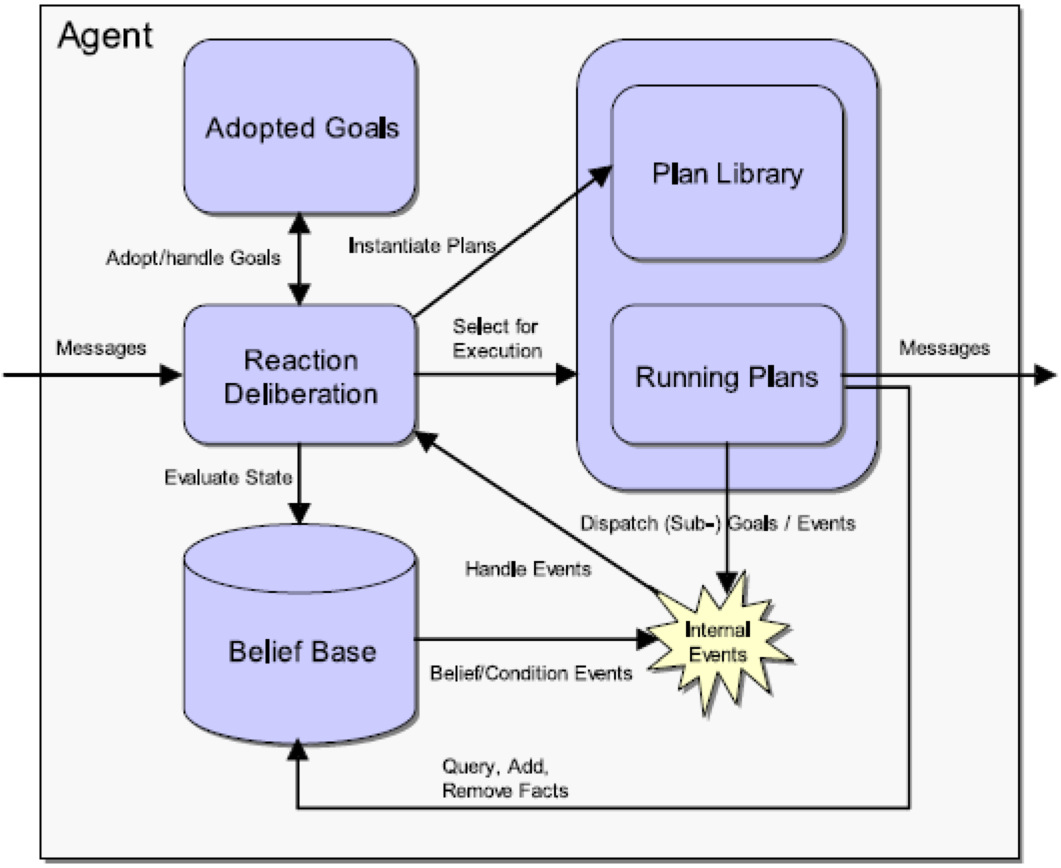


FIGURE 2.3: Architecture of a BDI System, from Braubach et al. (2004)

2.6 Beliefsets

A specific architecture within goal-based agents is the belief, desire, intention (BDI) model. Desires can be thought of as goals the agent wants to achieve, intentions as plans that dictate how to go about this and beliefs are internal data (Howden et al., 2001). Just as people have a view of the world, the BDI model sees agents also having a "view" of their world; this view is represented by the beliefs in the beliefset (Kinny et al., 1996). As the environment changes around the agent, beliefsets are updated to reflect this. The beliefsets provide the core of the BDI model, as most other functionalities are related to it. Beliefsets work hand in hand with events. Events have a twofold relationship with beliefs; an event can be the cause for updating a belief, and a change in belief can be the cause for creating a new event.

Two examples of beliefset structures are those used by the JACK and Jadex (Braubach et al., 2004) implementations. A JACK beliefset corresponds to a single database table, complete with a number of attributes (columns) and tuples (rows). Primary keys are also used to uniquely identify tuples (Howden et al., 2001). Each attribute has a data type, for example string, integer and boolean (there are others too). The databases are defined at design time, and are instantiated at run-time; many instantiations can be made from a single design. Sometimes tuples are referred to as facts, but these should not be confused with the concepts of facts used in rule based systems. Tuples are added, modified and removed using specific methods. Beliefset queries are used to select relevant tuples from a beliefset, based on parameters including beliefset name, attribute names and attribute values. The result of executing a beliefset query is a cursor. There are several different types, but essentially this is similar to a database cursor where the results can be iterated through. Using a relational database approach for the beliefset structure is not mandatory in Jadex, where "ordinary Java objects of any kind can be contained in the beliefbase" (Braubach et al., 2004). This means that Jadex beliefs can utilize any tool which can be represented as a Java object - including possibilities such as ontology reasoning engines and rule based systems, along with regular databases. Regardless of which implementation is used, a common feature in BDI is that triggers can be specified which activate when the beliefs reach a particular state. In JACK, this could be an attribute within a tuple being modified to a particular value, or in Jadex this could be the property of a Java object being set to a particular value. These triggers can be the catalyst for creating events; these triggers could be written in terms of geocoding data and conditions. For example, the triggers could be written in terms of the address elements (state, postcode, street etc.) and their spatial relationships (e.g. the street exists or does not exist, the state contains a particular postcode).

A benefit for geocoding is that beliefs provide a dynamic mechanism for maintaining awareness of events. Because the beliefs of an agent are ongoing, they can be updated at any time and likewise actions and selections can change accordingly which automatically ensures the agent is not operating under old information or assumptions. This means new geocoding information could be added at any time during geocoding, e.g.

additional input from the user, from other agents or web services. By having beliefs at the heart of the BDI agent system, a "belief driven" (data driven) design approach is available. This provides a single place for geocoding data (address elements, relations, results of tests), progress and metadata to be stored. It also provides the ability to track an address throughout its entire life cycle - "end to end tracking", and can keep track of information provided by the user or other agents. It also acts as an integration mechanism where disparate pieces of information can come together into the beliefset where they can combine to become the catalyst for something else happening. The beliefs support the pursuing of long terms objectives (i.e. "correct and geocode the address"), with a balance of proactive (e.g. find the data, try correction techniques, ask questions) and reactive behaviour (e.g. use an alternate data source, use information about other address elements). There is also the possibility of connecting the beliefsets within the agent to an external source such as a knowledge base; this would mean the beliefsets could be loaded with stored knowledge to be used for decision making in the agent, and would also provide a repository of knowledge for all agents to use.

2.7 Events

An event is used to signify a change of some sort has occurred, either inside an agent or within the environment (Georgeff and Rao, 1995). Regardless of the type of event, the event contains information. In this sense, an event in BDI can be thought of as an envelope; on the outside of the envelope is the destination of who the envelope should go to, and inside the envelope is paper with information on it. In the case of a normal event (i.e. an event used internally within the agent to initiate action), the agent "addresses" the envelope to itself. The information "inside" the envelope is a number of event properties or attributes, simple variables which each have a particular data type. The information contained within the event is accessible by the plan chosen to process the event.

Events are divided into internal and external events. Internal events allow the agent to monitor its own progress in relation to objectives while external events (percepts) allow the agent to receive information from its environment (other agents, from the user or a data source).

Events (along with plans) are not intended to be used as if calling subroutines - this procedural approach would counteract the benefits brought by events (Howden et al., 2001). Polymorphic events are possible, where although the core event itself stays the same, subtle differences in parameters and variable types can cause different outcomes in terms of plan execution. Because events are transient in nature, they require a concrete mechanism to store data that will "last" between events, this is the role beliefsets. Beliefsets have a close relationship with events as beliefsets represent the world, if the agent continues to use information that is outdated (i.e. new information has come into the system but it is not be used) its subsequent actions based on this information will be inappropriate and simply wrong.

This event based processing makes the agent well suited to working in a service oriented architecture, for example receiving geocoding queries on the Internet. Events and beliefs mean the agents are not "stuck" on a single line of processing, new events immediately affect processing without any time being wasted continuing to pursue an old agenda. Agents do not *have* to react to events (especially external events) so they still have choice and autonomy. Messaging can be synchronous or asynchronous, which for example could be used to broadcast new info/updates between agents or to the user. This also means an agent could ask a question where the answer is important and wait for reply (halting its individual processing), or could not wait for a reply and move on, only incorporating the reply when it arrives. Internal events can be triggered given predefined criteria (context), which is tied to values in the beliefset, for example when address elements have a particular value or a particular sequence of events occur. All of this activity can occur in multiple threads, and is very opposite to linear processing, although linear processes can still be catered for. With event based approach, the code does not have to list exactly in what order things will occur, it just needs the individual ways of dealing with it (i.e. plans) the order of processing could be different every time; this is good for geocoding due to the variability in input addresses and errors. Polymorphic events are possible, where although the core event itself stays the same, subtle differences in parameters and variable types can cause different outcomes in terms of plan execution; this enhances the context available. Because events are transient in nature, they require a concrete mechanism to store data that will "last" between events, this is the role beliefsets.

2.7.1 Automatic Events

Events which are started (also known as posted) as a direct result of a beliefset value changing can be thought of as automatic beliefs (Howden et al., 2001). These are completely reactionary in the sense than no deliberation needs to be done, only a pre-determined attribute-value condition must be met. The link between this reaction and deliberation is that as a consequence of the reaction, a goal can be posted.

2.7.2 Beliefset Callbacks

Although automatic events can be used for posting events when a given beliefset condition arises, in some cases it can be more useful to post events from within a beliefset callback. This is because the automatic events activate on the basis of particular values in the beliefset, whereas beliefset callbacks are activated on the manipulation of a beliefset. This would be useful for knowing when to update other agents or the user about progress being made in the geocoding of a particular element.

This is an important difference because there are times when the trigger value is not known, or is not of primary concern, and it is more important simply that the beliefset has been utilized. The other major benefit to using beliefset callbacks is that events can be posted from the callback method. There are several types of beliefset callbacks, for example when a tuple is added, modified or deleted. A benefit of the callback used when a tuple is modified is the ability to compare the "old" tuple with the "new" tuple that is replacing it, which means logical tests can also be performed on values.

2.7.3 Goals

Goals represent a particular situation or state of the environment which the agent would like to bring about (Howden et al., 2001). In pursuing a goal, there are many different possible alternative actions that the agent can take - but the ultimate measure of whether or not a particular goal has been achieved is defined by its beliefsets. The beliefset values are how the outcome of an agent goal is specified. The different courses of action that an agent can utilize in pursuing a goal are called plans, and these can perform any action the programmer wants (regular code is run); one of the critical actions that a plan serves in a BDI model is to update the beliefset. In this way a cycle emerges of the agent taking action based on beliefs and then changing these beliefs as a result.

Normal events differ from goal events (or simply "goals") in the way the task is pursued, and how success of the task is specified. With a regular event where a task is specified and an action taken - success is assumed regardless of outcome. a true goal (in the BDI model) has its essence defined in terms of the beliefset; a goal is complete when one or more beliefs in the beliefset reach a particular value. There are two main factors why some steps are represented as goals, and some as events, and these can be attributed to either time sensitivity and/or functionality. From a temporal point of view, events are immediate and goals are longer term; i.e. from an achievement point of view, goals relate to objectives while events relate to responses.

Because an event must be dealt with immediately, it makes sense that there is only time for one course of action - beyond that the event has passed and using the information contained in the event after this would mean the agent risks having an outdated view of its "world". An event is handled by choosing only one plan that is both relevant and applicable; this is a "one time chance" such that if the chosen plan fails then the event fails. In contrast to event handling, goal handling is characterized by a goal event potentially having several plans that are equally qualified for selection (regular events have just one), and also in the event of a plan failing others may be used in its place (where the normal event would fail on first plan failure). Another distinction between events and goals is that a goal use a beliefset to monitor whether or not it has achieved its goal; specifically the criteria for achieving its goal is specified by a logical condition constructed from attributes and conditions from one or more beliefsets.

As a guideline, a situation should be represented as a goal if (i) it is an objective that is to be achieved in the long term, (ii) there are many plans with equal relevance and applicability that could be selected, (iii) plan reselection and use (of a single plan) is required upon plan failure, or (iv) it would be desirable to use every plan that is relevant and applicable (not just the first that succeeds), or (v) the criteria for success of the situation can be expressed in terms of one or more beliefsets.

This means geocoding can be viewed from a goal and sub-goal perspective, with the ability to revisit steps as needed and not stay to a rigid linear pipeline. How would geocoding differ if the whole process could be divided into goals and sub-goals? This also enables an agent to balance the longer term goal of solving a geocode with the more immediate events which could affect processing.

2.8 Messages

Similar to events, messages are both the cause and the result of changes in beliefsets, except that messages deal with information originating from outside the agent, where events are focused internally within the agent (Howden et al., 2001). An agent can receive a message from another agent, or a message can originate from the environment (this is more correctly called a "percept"). With an incoming message or percept is information that may be used to update a beliefset; also, a change in beliefset may justify the creation of a new message leaving the agent which is communicating the change to another agent or affecting the agent's environment in some way.

2.9 Plans

Plans are an essential building block of the BDI model, and are atomic chunks of code which are executed under certain circumstances (Russell and Norvig, 2003). The circumstances under which the plans can be executed are specified by the event types for which the plan is suitable (a static criteria) the agents beliefs (a dynamic criteria). The ability to make decisions dynamically stems from using these small units of code grouped together and categorised by having the potential to solve a common problem (manifested in the agent system as events and goals). These plans are compiled at design time, and depending on conditions at runtime only certain plans will be applicable and others may also fail, meaning only a select sub-set are subsequently used to achieve the desired outcome (i.e. goal). Figure 2.4 shows how an agent has multiple plans, and how these plans are selected using two tiers of context, namely the *relevant* and *context* methods; these are specific to the JACK agent implementation. A possibility is that one of these layers of context could be connected to the domain knowledge stored in a knowledge base.

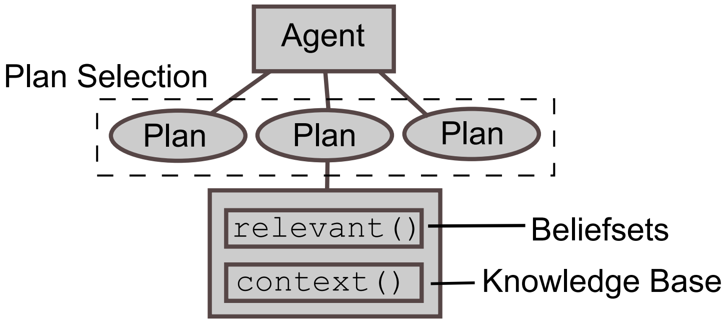


FIGURE 2.4: Context in Agents

An example of plan use is wanting to get a particular type of data, but needing to decide where to get the data from; the correct plan can be selected depending on contextual factors (e.g. user, application, geographic and jurisdictional). This use of plans to solve goals weaves a thread of these types of context through potentially every decision made in the system. Other uses of context via plans is ensuring an acceptable match rate and acceptable reference data quality are used when looking for data and performing the geocoding. Another example of potential plan selection for geocoding is when a particular correction technique (e.g. Soundex) yields no suggestions so another is used, or accessing a dataset does not work so another avenue for data is tried using a different plan, or agent waits until conditions change.

2.10 Summary of Agent Based Systems

Figure 2.5 brings together the various BDI components mentioned so far, and this is what forms the control of the proposed intelligent geocoding framework.

The key concepts seen in Figure 2.5 could be applied to geocoding such that:

- The messages will be sent between the outside world (the user, web services) and other agents

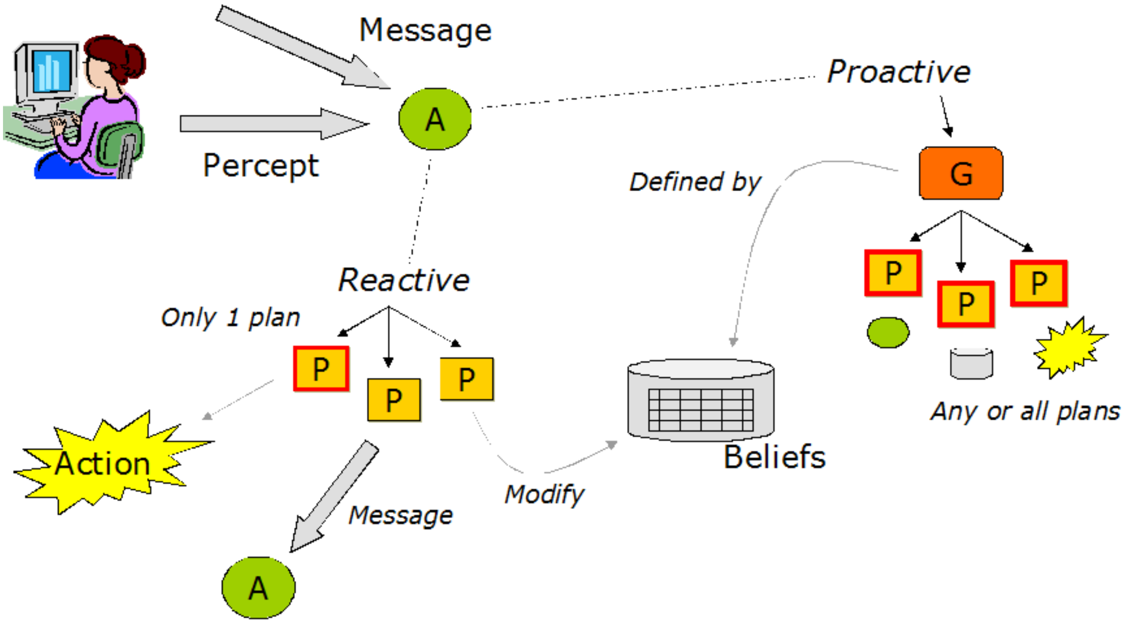


FIGURE 2.5: The BDI Life Cycle

- Agents will balance a proactive goal of geocoding with the reactive responses needed to deal with the messages and internal conditions that become fulfilled

- The whole control process will be data driven and real time via the beliefsets, which will simultaneously be the catalyst for spawning new tasks, monitoring their progress and provide the metric to measure completion

- Plans will be used to carry out the tasks required to satisfy regular events and goals

- Plans will be applied selectively and dynamically to find the best option

- The plans will in turn perform an action (a chunk of code), send a message, spawn another goal or update beliefs

This provided BDI cycle will continue until an address has a geocode determined for it or processing has been exhausted. This approach will be for control, but there remains a need for other techniques.

2.11 Conclusions

This section builds on the desirable control features and agent characteristics seen in Section 2.4.1. Agents contain many of the desired traits for intelligent geocoding, and many of the issues have the common thread of semantics running through them. A benefit of using agents for geocoding includes the ability to decompose the geocoding phases (normalization, matching, locating) into smaller problems (expressed as goals and subgoals) which allows for a less linear geocoding process and an ability to incorporate dynamic data coming from events internal and external to the agent. This decomposition is further complicated by the idea that processing can also be done in parallel. This leads to the question of how are tasks arranged to work in parallel. The goals and parallel tasks will also require conditions to define their success and failure. If designed correctly, the ability to process in parallel provides the opportunity for a scalable solution (using multiple processors and multiple machines).

There is a need to determine a way that the same linear geocoding process is not simply repeated using agents but that the process is enriched from using agents. Because agents are social, interactive and event based it is a possibility that the semantics and context could be incorporated into their behaviour. Relevant to this is the concept that the essence of semantics for addresses is in terms of the geographic entities and their relationships. Part of this behaviour is message sending, and considerations for design include (i) which agents are sending the messages, and (ii) the content of the messages being sent.

The event driven approach is also beneficial to geocoding as a service oriented architecture (SOA), because new geocoding requests coming in over the Internet from users or other machines *are* events. There is a choice to use one or more agents, using multiple agents could have benefits over just using one agent, but the design will need to determine what this benefit is and how to include it in the design. There is a “ripple” effect present in geocoding (at least there could be with an event driven approach) where although geocoding has many different steps and there several parts to an address, everything is intertwined; changes in one component can affect another. Whether it correction, scoring, or finding data all aspects are related - this is suited to event driven design.

Beliefs within the BDI model provide the “glue” which ties together the goals, messages, events, context and plan selection; this also provides a data driven approach to geocoding where multiple geocoding queries can be pursued simultaneously, advanced tracking of address progress is possible, and only the necessary geocoding phases are applied as needed. It is anticipated that beliefsets will be the cornerstone to control in intelligent geocoding, but there is a question regarding the type and richness of knowledge that a beliefset can represent.

Non-deterministic behaviour allows agents to deal with geocoding outcomes dynamically at runtime and retry if failure occurs in particular task. The use of the belief, desire, and intention (BDI) agent model provides the ability for many types of context to influence decision making in all aspects of processing. BDI provides the ability to take the currently linear geocoding process and make it more flexible where steps can be revisited or avoided, dependant on address values and context of the situation.

To utilize previous experience (i.e. learn) it will be necessary to read and write to a knowledge base, although only a reinforcement learning approach will be suitable because the only feedback is whether an incorrect address was successfully corrected with regards to reference data. The agent and knowledge base will need a common structure and vocabulary for describing addresses. The comprehensiveness of the knowledge structure will determine whether it is a taxonomy or an ontology.

Agents have the potential to offer a radically different way of designing a solution for geocoding, specifically enabling intelligent geocoding. Questions that arise include how to weave geographic semantics into the framework at a base level and throughout. From the issues discussed regarding user perception and geographic/semantic errors, there is a possibility that there are more types of geocoding mistakes being made than are currently known or corrected for. There is also a need to determine where in the geocoding process user perception is incorporated.

Is clear from reviewing the steps in geocoding that there is definition (with ordering and interdependence); these steps could be a natural fit with goals and sub-goals. Plans will contain code to perform the corrections, lookups and locating; their design will require thinking about the geocoding process differently to ensure the maximum benefit is achieved from using the agent paradigm. With interdependence between geocoding steps, along with the vision for greater interaction, there are several reasons why the geocoding process could be repeated several times for the same address (i.e. refinement of the solution).

Beliefs can be used to track progress, i.e. its beliefs will reflect what is happening in its "environment" of geocoding. The event driven design will require a mechanism for determining when to "stop". The traditional steps in geocoding are not going to change, but the paradigm used to control them and the order they execute in, how the traditional geocoding steps work together can be changed.

Levels of completeness will have implications for the quality of the geocode and if one or more elements within an address are missing then this will affect the processing of other elements, the design needs to ensure there is a flexible way of handling this. The design also needs to use the elements that are present in an address to narrow the search space for other address elements.

Although syntax only describes errors at the "surface" (i.e. not deeper semantic meaning), at least they are straightforward to store for correction and later use because the text is symbolic; the knowledge base can store correct and incorrect addresses. Storing of corrected names/syntax is related to the situation that the geocoder can only learn if exactly one correct result (assuming no more interaction from user) is found for an incorrect query.

Because the presence of particular address elements can be critical and they are sometimes unknown, the question arises of what would happen to processing if the missing elements could be inserted based on the elements that *are* present. It would be useful if the design could include a way of combining multiple misspellings, and also cater for the possibility that there are misspellings and other causes included in the same incorrect query. To cater for the perception of users would require research specifically into why users make mistakes spatially, this is beyond the scope of this research, however it would be good if the knowledge base could be extensible to cater for this. It would be useful if the knowledge base was extensible enough to cater for storing geographic errors (assertions stated in terms of actual geometric relationships).

Chapter 3

Research Design

The research methods shape the investigation of how to apply an agent-based framework to handle intelligent geocoding, given the issues of geocoding and the benefits of agent based systems. An overarching question and theme throughout the research design is how control and context is present and used.

3.1 Method for Identifying Relevant Issues

The key steps for this method are determining the current state of geocoding based on the literature and also commercial sources, distilling these concepts into a manageable overview and then determining priorities from these to guide the focus of the research. The findings and focus from this method are used in the method for developing the intelligent framework. The method for developing the intelligent framework (Section 3.2) is responsible for taking an initial idea, refining it, establishing a design and finally creating a prototype for evaluation. The method for examining control and knowledge determines what role these factors play in the prototype by analysing both the *behaviour* and *process* of the prototype. Looking at the behavior presents a quantitative perspective, while analysing the process provides a qualitative perspective. In addition to this analysis, other functionality also made possible by the prototype is examined. Figure 3.1 shows the various research methods used and the sequence they are used in.

Although there are many areas within geocoding that could be improved, the *uncertainty in the matching algorithm* aspect of the underlying process, along with data input errors attributed to semantic, geographic, iterative and geographic addresses all benefit from new research. This new research

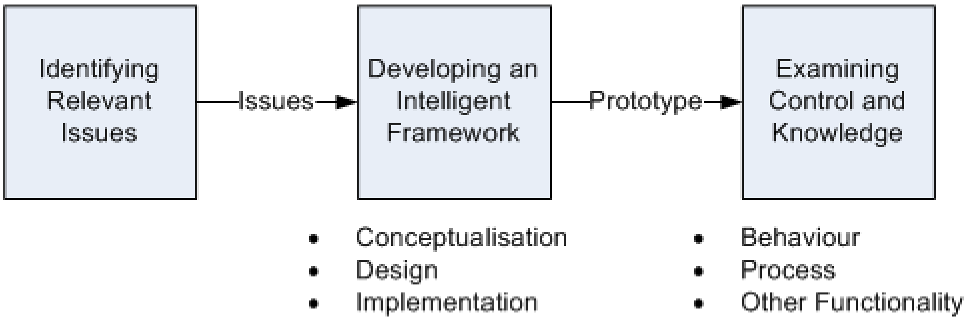

Identifying Relevant Issues
Issues
Developing an Intelligent Framework
Prototype
Examining Control and Knowledge
Conceptualisation
Design
Implementation
Behaviour
Process
Other Functionality

FIGURE 3.1: Research Methods within the Research Design

is based on where previous research has been done and where commercial efforts continue to improve. The geocoding component of the literature review is done first to build an understanding of this field which then directs the review of artificial intelligence techniques and how these fit into the solution. The literature on this topic provides an indication for what is and is not possible with current AI techniques, and also provides insight into which of the techniques are more suited for use in geocoding. Agent based programming was identified in Section 2.4.1 as a strong candidate due to both the traits it has and how these aligned with the anticipated, overarching desires for intelligence in geocoding.

3.2 Method for Developing an Intelligent Framework

To investigate and develop a framework for intelligent geocoding, an initial solution is sketched out which not only seeks to solve some of the issues in geocoding, but do so by adopting techniques from the field of artificial intelligence. With the basic solution sketched out, a more formal design is defined which includes a greater level of detail and a plan for the novel contributions to be created. An implementation tool is then selected which is capable of reproducing the design in program code. It is the implementation which produces the prototype (named "IntelliGeoLocator") which in turn is used to evaluate the research ideas proposed.

3.2.1 Conceptualization of the Solution

Preliminary ideas are established for a solution and AI paradigms are evaluated against preliminary ideas. This is where original synthesis of ideas occurs, and an informal design happens on paper. The background literature, the chosen focus for geocoding and adding intelligence all combine into what is a high-level solution. It was during this stage that the idea for modelling control using messages between agents (representing address elements) was formalized, along with using agent beliefsets to process and track all aspects of geocoder operation. Agents provide the control knowledge and a rule based system stores the domain knowledge. Storing the knowledge opens the pathway for learning. This step of the method provides a starting point for comparison of the initial design ideas to AI techniques to see how feasible the solution are from a software perspective. At this stage of evaluation, a theoretical evaluation is made to see whether the AI paradigms available would broadly support the preliminary ideas for adding intelligence to geocoding. The step is considered “theoretical” because at this point there could be several possible implementations of an agent or rule based system used.

3.2.2 Framework Design

This step includes designing the framework for intelligent geocoding, identifying and obtaining required data (both for geocoder operation and testing), and developing a testing framework. Design is applied before implementation to produce a better outcome in terms of software engineering (Pressman, 2004). To assist with the agent design process, the Prometheus methodology (Padgham and Winikoff, 2002) is used. Agent-based programming is a unique paradigm but just like other software paradigms it requires a software engineering methodology to ensure the design process is well thought out, structured and scalable. Prometheus was built specifically for the agent-based paradigm, and produces guidance for how to best approach the use of goals, beliefs, plans and events in design.

As early on as possible in the design, the data required for processing needs to be found as doing this can take time, and it is not worth waiting until implementation only to find a particular type of data needed is not available. Likewise with the testing framework (Section 3.3), thought was given early (during design) as to how robust the testing framework would be for testing the research ideas regarding building intelligence in geocoding.

The prototype is evaluated using several criteria including (i) observation of the prototype behaviour, (ii) examining the results of address testing, and (iii) examining the additional functionality made possible by the new framework. The quantitative test is included to show capability and relevance compared to other established geocoders. The quantitative testing is intended to ensure the prototype is not grossly deficient in a sample of common problem address types. The goal of the research is to establish a new framework for geocoding with real implications, rather than to solely write a collection of problem address correctors. This is important as the main reason for the quantitative testing is to show an agent-based framework does not have barriers with regards to a wider adoption by industry. To conduct the quantitative test, five categories of test addresses are used, and three other geocoders are evaluated in conjunction with the prototype.

The categories of problem addresses for testing include (i) levels of completeness, (ii) syntax of addresses, (iii) semantic and geographic, (iv) iterative processing, and (v) compounding of errors. The geocoders selected for comparison include (i) Google Maps, (ii) Multimap, and (iii) Whereis. The Google geocoder was selected because of its broad usage and wide access, along with the fact it is constantly being updated. Multimap and Whereis were selected for their emphasis on Australia. All the geocoders selected for use are online geocoders, the rationale for this is that they are more advanced and are changing/updating more rapidly than the vendor-based geocoders which are more limited in scope and currency.

3.2.2.1 Identify System Components

The agent oriented component will be the hub of the entire geocoding process and maintain flow of control for the system, coordinating itself and the other components. The agent or agents will access the knowledge base and reference data as needed during processing. An agent will be capable of many different actions, some are basic while others (such as matching) require more sophisticated algorithms and supporting technologies. The components in the system can be seen in Figure 3.2.

In the course of completing a query, the agent will consult the knowledge base to help solve queries via knowledge stored from previous queries. An agent will also write to the knowledge base if any new knowledge is discovered.

An agent accesses reference datasets to make decisions regarding the address elements submitted in a query. It can be seen in Figure 3.2 that data is accessed both locally and over the Internet. The diagram also provides a prelude to the detail further contained in both the knowledge base and the agent system. The use of remote data via the Internet includes web services, and as more of these services become available, and as the semantic web further develops, agents will have richer data to draw from.

One of the main aspects of the design is that overall control of the geocoding process is based on the interactions between agents representing the geographic elements contained in an address. In other words, each address element is represented by a software agent and each of these agents pursue tasks associated with correcting and preparing their individual element for geocoding. Because there are multiple agents, these tasks occur in parallel. The result is the geocoding process in software running with multiple foci of control. Eventually when each agent is finished correcting its own element, all the elements from each of the agents are reassembled and coordinates are found for the address.

Because agents are used to represent the geographic address elements, the same geographic relationships that exist between the address elements *also exist between the agents*. Because the processing of individual elements does require information about other elements in the address, messages are used for the agents to send and receive data. However, because the agents represent geographic elements, these messages have content relating to the real geographic relationships. For example, the relationship that a state contains a particular locality, or that a street is within a given locality. The messages also allow the high level or conceptual level to be represented (e.g. in terms of “states” or “streets”) and the specific instances of these (e.g. “Western Australia” and “Marlow St”).

Messages also allow data outside the immediate system (e.g. from the

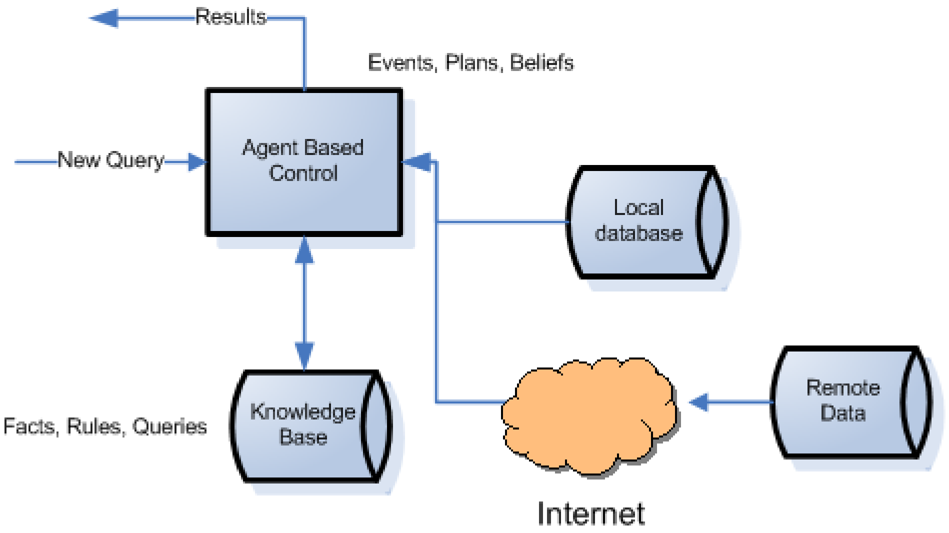


FIGURE 3.2: System Components

user) to reach the agents. Beliefsets enable the agents to internally store many different data values, relating to all aspects of its processing and providing the coordination it needs. For example, an agent stores its particular element value, that status of its progress in processing, its relationship with other address elements, and data describing selected other elements (plus other data). The tasks undertaken by each of the agents are modelled as goals, and the beliefsets allow the agent to know when, for example, an address component has been corrected and the agent can move on. Similarly, regular internal events (such as the status of progress being updated, an address element value being modified or a new piece of information from another agent) are also driven by the beliefset values. Plans are used by the agents to search for data, send messages, update a status score, query the knowledge base and other actions.

The agents do not persist their beliefsets to storage between sessions, instead using the knowledge base to store important data. The data stored is used in future queries as “knowledge” to help correct address element values, using the dynamically created aliases. The agents are able to query the knowledge base at runtime, and add content dynamically. There is a single knowledge base, which each agent contributes to.

The design of the prototype is suitable for all phases of the geocoding process, however the focus is on building intelligent knowledge and control into the geocoding process. As mentioned in the Section 1.5.2 of the research, the framework does not perform any normalization of the data with regards to the meaning of the address elements, i.e. the prototype assumes the incoming address elements have the correct meaning associated with them. There is nothing preventing the work of others such as Churches et al. (2002) to be used or embedded in the intelligent framework. Web services are used for the locating of actual geocode coordinates, and this action is performed by an agent.

The vision which underpins the design is that context and semantics will be built into geocoding control at a fundamental level. It is hoped that this combined with the event based and parallel nature will provide a scalable and innovative paradigm for geocoding which overhauls the existing geocoding approach for one more in tune with the semantic web and increasing use of service based processing on the Internet.

3.2.2.2 Agent Assignment

The geocoding process is the same, with a few differences, for each of the address elements and this lends itself to reusing large parts of agent behaviour for the various agents. The nature of geocoding means that, with some intercommunication between processes, the various address elements can be processed simultaneously, which ties in with the ability of agent software to do parallel processing. From a control and communication point of view, design is made easier by only having to consider the behaviour of each agent individually, and then putting these together via a common messaging approach. The agent types assigned within the system include a user agent, matching agent, and five specialty agents. Figure 3.3 shows the parallel behaviour of a single agent brokering the incoming queries and distributing the individual elements to the specialty agents.

The five specialty agents include the state, postcode, locality, street and property agents. Although five agents are mentioned specifically, the design is intended to cater for more. The geographic representation does not have to stop at the property level, and the design would allow drilling down to buildings, rooms in buildings and even objects in rooms. This adds flexibility and extensibility to the design, and is not constrained by the increasing complexity which would happen in a “linear” environment. Likewise, in terms of the higher level, areal features, it does not have to

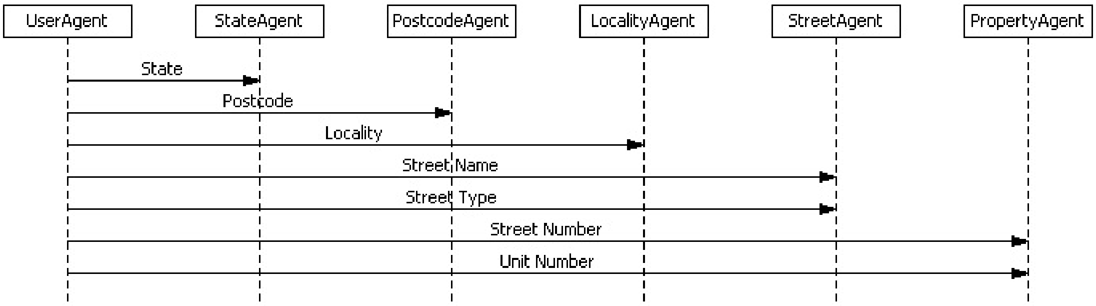


FIGURE 3.3: Address Components Distributed to Agents

stop at the state. Beyond state, there could also be a country representation, this in particular would work well with semantic knowledge detailing the idiosyncrasies of geocoding in other countries (e.g. geography, topology, temporal).

The street agent is responsible for processing both street name and street type. The property agent is responsible for processing street number and unit number. Because the internal beliefsets, reference data and much of the processing is the same for all geography element types, a single agent design has been made which will contain the functionality to process any element type.

When the system is run, multiple instantiations of this design are created and each assigned to a particular role. With so much shared functionality, this makes design much easier. The agent type created to incorporate these functionalities is named the *element agent*. The matching process seems distinctly different from the processing done by the element agent, both in its objective and also the data it uses. For this reason, a matching agent exists to perform the processing associated with matching.

A user agent was created to manage the brokering of queries from users, and then distributing these to the element agents. This user agent provides a neat start and finishing point for the parallel processing done by the element agents; it is the user agent which has the coordinating role for the multiple queries coming in. It is within the user agent that each agent is assigned a unique ID and kept track of so that the outcomes can be distinguished. The user agent is responsible for "distributing" these identifiers as needed by the agents, and when closing a query it retires the identifier.

3.2.2.3 Goals

The goals and sub-goals within the framework can be seen in Figure 3.4, where a goal is represented with a box, sub-goal as a coloured ellipse and a plan as plain ellipse.

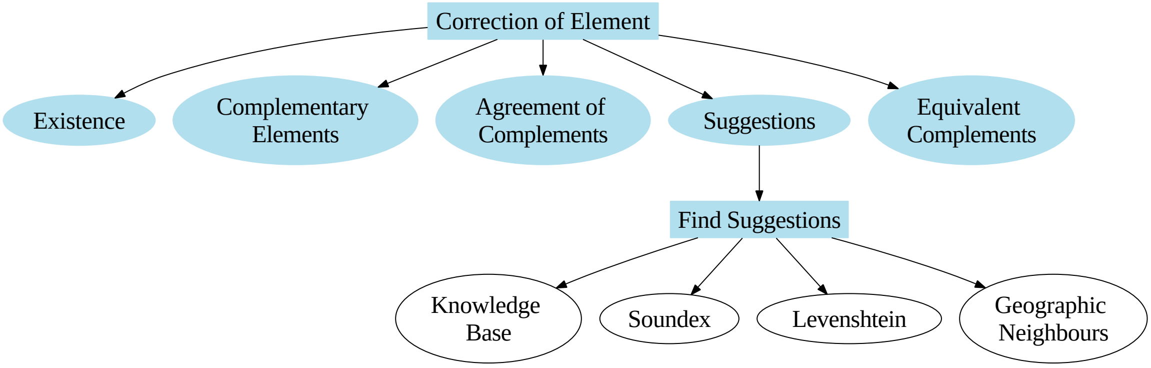


FIGURE 3.4: Goals and Sub-Goals within the Framework

The goal *Correction of Element* is the top-level goal which is pursued for every address element, and in turn each of the sub-goals are also pursued. Each of these sub-goals is a step in the element correction process, although depending on the status of the element, not all steps may be required at a given time in the geocoding process. The terminology used to describe the goals in Figure 3.4 corresponds with the definitions given in Section 3.4.

Looking at these sub-goals, and their role in the geocoding algorithm, it is seen that (i) goals are applied in a sequence where order is important, (ii) not all sub-goals will be needed in every situation, (iii) goals are subtly influenced by their environment, and (iv) a mechanism is needed to ensure one goal is completed satisfactorily before moving on to the next.

The goal to test whether an element is present and exists (“Existence”) is a simple lookup, no techniques are used in this step to try and find a possible replacement or correction. This information is stored in beliefsets. There are some special considerations to this step, such as when a particular type of element is required to check the existence of another because the other by itself is simply not possible; e.g. to check if a street number “exists” at the very minimum a street name is also needed.

The “Complementary Elements” goals sends messages to other agents (which each represent elements) to ask what *their values* are, which determines what the complementary elements are for the original element; these values are then stored in the beliefset of the original agent. Each agent exchanges messages with at least one or two other agents to do this, and this is an example of activity occurring in parallel.

It is important to mention that related to the use of these goals (and overall geocoder control) is the role of address element status scores. These scores are calculated for each address element and also for the overall address. The score of an individual agent affects messages sent out to other agents. When certain beliefs are true then the agent is "eligible" to perform certain actions and interactions. For example, once an agent has determined whether its address element is present and exists, it will respond to other agents requesting what its status is. Previous to determining this the agent being asked would not respond. If an agent reaches an individual score of 1.0 then this is the catalyst for the agent indicating they are "complete", and eventually when all agents are complete then the processing of the overall address is complete. Similarly, even when agents reach a final maximum score less than 1.0, this is communicated via messaging and the agents reach consensus indicating that overall processing is complete.

3.2.2.4 Messaging

Messages can be received and have the contents of the message written to its beliefsets, or the message can be acted on immediately and nothing stored in the beliefsets. In this design all the messages sent have their contents stored in the receiving agents beliefsets. Uses for messaging include: (i) bringing a geocode query into agent system, (ii) distributing address elements to the various element agents, (iii) Providing an element for use in matching, (iv) getting current information about complementary elements, (v) providing information in response to update request, (vi) correcting elements ready for matching, (vii) notifying other agents of updated status, (viii) providing arrays of element suggestions to Matching Agent, and (ix) providing arrays of complementary element suggestions to Matching Agent.

3.2.2.5 Inherent Semantics and Real-Time Quality

Two particular features in the architecture of the prototype are the abilities to have (i) inherent semantics in the geocoder because of the agent used and their messaging, and (ii) quality scores and geocodes calculated for addresses in real-time.

When working with the geographic elements, they are arranged conceptually as seen in Figure 3.5, where they are arranged in order of which elements "contain" other elements. This arrangement is what underpins the semantic relationships between the agents representing the address elements (referred to from herein as *element agents*).

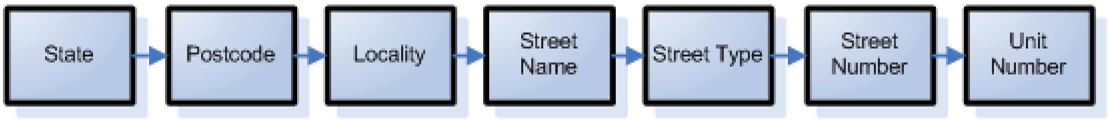


FIGURE 3.5: Elements Arranged in Descending Containment

There are several terms which describe the relationships between elements. An element is *present* if a value was submitted in the original submitted address, i.e. the element value is not blank. An element *exists* if it can be found in at least one reference data set, e.g. the locality "Floreat" exists in the gazetteer for Western Australia.

As seen in Figure 3.5, a given element has other elements on the "left" and "right" of it; these are referred to as *complementary elements*. For example, the complementary elements for the postcode are the state and locality. In a real query, if a particular element type is not submitted (i.e. it is blank) then the complement is the next present element type. The state and unit number elements do not have upper and lower complements, respectively. The complementary elements in this research are address elements, but there is no reason they could not be more generally used as "complementary components", any component that better contextualizes it and gives additional assurance.

This *agreement of complements* is a spatial agreement, for example the postcode contains a given street, or a street does have a particular street number on it. It is possible that two elements may not agree spatially, even if they both exist; this case would mean that one of the elements is incorrect.

An *equivalent complement* is the equivalent, complementary element value for a given element value; the result is selected from a reference dataset, using the given element value as the search criteria. For example, using the a street name, its equivalent postcode can be found. Using this approach, an element type finds the equivalent complementary element only to its "left" (the containing geographic element); this is an effort to perform a "one to one" search as often as possible (e.g. a street usually has few or a single locality associated with it, but a single locality would have many streets in it). This "one to one" idea is used simply to keep the list of equivalents as small as possible.

It is because the agent *is* the geographic element, and vice versa, that a commonality exists between control, knowledge and the phenomena being abstracted (the geographic element). This commonality is such that the geographic elements also benefit from receiving context and semantics. Figure 3.6 presents how the intelligent framework can be thought to have two imaginary tiers involved, where the bottom tier is the "behind the scenes" use of control and knowledge which in turn drives the context and semantics on the surface.

Figure 3.6 also shows how when viewed from overhead, only the context and semantics is apparent. Also, because the two "layers" of intelligent geocoding are parallel and aligned, then conceptually any effect or capability on the top layer has a corresponding cause or capability on the lower layer. The vertical column in Figure 3.6 represents that it is the agent paradigm which ties the two layers together and allows for the duality of the control and geographic representation.

The meaning of each agent is known to other agents, for example if there is an address element missing, the other agents can adjust accordingly. This leads into the established relationship between the agents, which at the most fundamental level is a "contains" relationship. For example, a postcode contains a locality. Additional relationships exist within the definitions presented previously, both individually ("present", "exists") and between elements ("spatially agrees with complementary elements"). As described with Figure 3.6, these geographic relationships are also represented at the control and knowledge level, by the messages each agent can send to each other, regarding their own status and also questions they

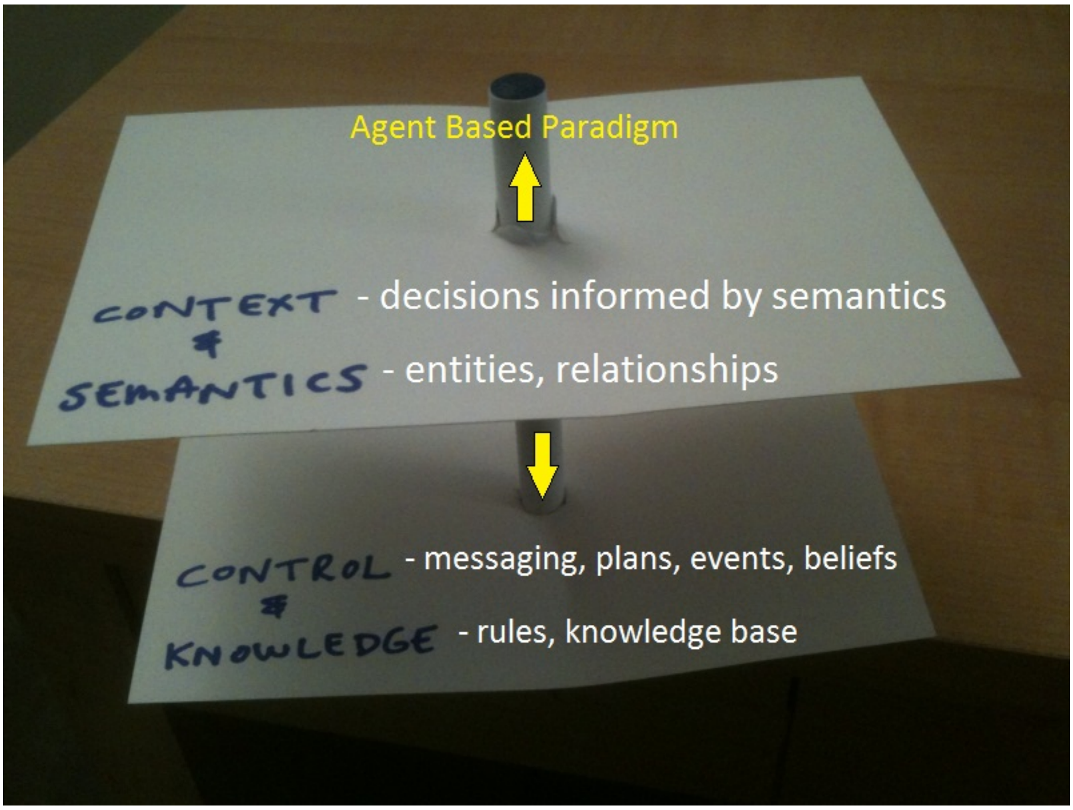

Agent Based Paradigm
CONTEXT & SEMANTICS
- decisions informed by semantics
- entities, relationships
CONTROL & KNOWLEDGE
- messaging, plans, events, beliefs
- rules, knowledge base

FIGURE 3.6: Two Layers within Intelligent Geocoding

may have for other agents.

Extensibility is also a feature of using agents in this way, for example having the ability to add more agents representing additional geographies or even other entities other than geography. The extent of the framework currently is street numbers (the lowest level of geography) but if buildings were desired then a "building agent" could be added in the future.

Because the agents will be processing their own elements in parallel, it means that some results will be ready sooner than others, due to the time it takes to query reference data. This means that as agents process their respective elements and have results, they can be displayed for the user (or sent to another program/machine) as they become available. If all address elements are processed quickly then the final result would be available almost immediately, but if there is lag then the user would benefit from seeing the result as it is so far. The user would also be provided with a geocode as soon as possible; this could initially have a low spatial resolution (e.g the state level), but would improve as more address elements are processed (in particular the lower level elements such as street and street number).

3.2.3 Prototype Implementation

Implementation was an iterative process which involved creating an initial skeleton of agent and knowledge base functionality, then connecting the agent with the knowledge base and adding more functionality to both. At each stage of development, the software was tested to ensure it worked. Time was needed initially to learn the specifics of the agent based framework and rule based system.

3.3 Method for Examining Control and Knowledge

At a high level, the methodology for evaluation aims to establish that (i) the new framework can perform tasks that current geocoders do, and (ii) the new framework can utilise semantics to improve geocoding using an intelligent approach. To do this will involve (i) comparison with a selection of existing geocoders, and (ii) any new functionality which has been made possible. Contrasting the prototype with existing geocoders provides an objective base of functionality which the prototype can be compared to; it will also indicate whether the new geocoder is too deficient to be used in industry. Testing will also evaluate the ability of the geocoder to: (i) resolve problem input addresses with invalid, missing or aliased information, and (ii) learn from previous geocode queries, create rules and use these in the future.

The testing methodology touches on the three objectives from Section 1.5, by using addresses identified as problematic (Objective 1) in a quantitative analysis, and using a qualitative analysis to evaluate how successfully intelligence was built into the geocoding process (Objective 2) and how control and knowledge are used together (Objective 3).

3.3.1 Evaluation of Intelligent Geocoder Behaviour

This approach is to look at the behaviour of the prototype from a quantitative perspective, gaining insight of how the system operated based on the values produced during processing. For example, how does the scoring process perform, does it work as expected? Do the agents cease processing correctly and how is the processing load spread amongst the agents? Other observations will include whether the real-time correction and geocoding worked as expected and also the parallel and distributed aspects. If these indicators perform as expected, then it is an initial sign that the geocoder is operating as expected. To carry out this observation, notes will be taken of the status of the prototype during address processing, the messages used and time taken to perform different tasks. In particular, focus will be given to the behaviour of agents individually and as a whole.

3.3.2 Evaluation of Intelligent Geocoding Process

Included in this stage is testing the prototype with known problematic addresses. The testing is not intended to be a rigorous test for positional accuracy, but more to show the prototype is not deficient. This testing follows that from Section 3.3.1 and provides the next validation that the prototype is or is not performing as expected. At the end of this stage, it would be known whether the prototype *is operating as expected and is on par with other geocoding products*.

To geocode a wider range of problem addresses, it is the correction techniques, not the control framework which is largely responsible - although the framework can help. More original correction techniques are needed, such as spatial similarity. It is expected that the geocoder will improve on existing problem addresses, however given the scope of the research, both the research and the prototype need to be explored further to demonstrate and achieve this, the result of the research is an extendible and expandable framework which can be further improved in the future.

However, for the newer correction techniques introduced there should be improvement in the new types of errors handled. Original functionality will emerge and this a contribution in itself, the address testing in this research is not about a massive comparison with other vendors, this research is a new framework and paradigm; this is about *how* the prototype does its processing, not so much the geocodes it gets back - this is exemplified by the fact that it uses another geocoding web service and it is not doing any standardization.

Results from the test addresses will be recorded in a table, with both IntelliGeoLocator and other geocoders being represented. For each address type, it will be noted whether the geocoder was able to resolve the geocoder entirely, partially, or not at all. The notes will also include insights into how the geocoders behaved. By using a variety of test addresses, it can be seen how suited the prototype and its paradigm are to different issues. The addresses considered "representative" will be chosen using the experiences of other experts (e.g. LandGate in WA) and the public, and the literature. Examples of different input addresses include the use of varying address elements, different ways of specifying addresses, and different errors. The test addresses are categorised in terms of completeness, syntax, semantic and geographic, and compounding of errors. These categories have been chosen because they include both the immediately visible and "invisible" aspects of an address, in other words those aspects apparent from the address string and those with a deeper causation. The category of compounded errors is a mechanism to cater for situations where several errors are present in an address.

3.3.3 Examine Derived Functionality

In addition to evaluating the behaviour of the intelligent geocoder (Section 3.3.1) and the effectiveness of the agent paradigm for the geocoding process (Section 3.3.2), there is also a need to evaluate other functionality made possible by the framework for intelligent geocoding. The concepts and questions which need to be tested include agent control, use and transfer of knowledge, inference, communication and more. To evaluate functionality defined at the conceptual/framework level (the left column in Table 3.1), practical demonstration is used at the prototype level; an overview of the evaluation methods are shown in Table 3.1.

TABLE 3.1: Derived Functionality to be Evaluated

Outcome to be Evaluated	**Method for Evaluation**
That an intelligent framework provides the ability for an address to flow through from start to finish with a final geocode	Demonstrate what happens when an address moves through all necessary beliefsets, plans and messaging
BDI (specifically events) acting as a form of control knowledge	Demonstrate sample events, what data was contained in them and which agents did the sending and receiving
BDI within the framework allowing the geocoding process to be less linear	Demonstrate with an example of how goals, plans and messaging interact in the geocoding process
Showing how knowledge from the KB was transferred to the agent (beliefsets) and used to make a decision	Demonstrate which plan accessed the KB, the knowledge which was retrieved and how this affected the agent
Re-writing the knowledge base dynamically (e.g. aliases) automatically	Demonstrate how a single result was written to KB at end of processing including what the fact was
Fundamental inference using disparate facts	Demonstrate how two pieces of information were used to solve a query
Use of context	Highlight and give examples of agent roles, plan selection, messaging between agents and knowledge base (including geographic context)
Tracking of multiple addresses made possible by framework	Demonstrate example of the tracking IDs assigned to queries throughout the geocoding lifecycle
Aggregation of results from different agents	Demonstrate sample message coming into MatchingAgent and how they were aggregated
Rule based reconstruction	Provide an example of how the rule based reconstruction performs

3.4 Conclusions

An important conclusion emerged after identifying the issues, which is that they have a common thread of intelligence weaved throughout them, this subsequently informed the method for developing the framework by providing a focus. The method for identifying relevant issues established there is a link between agent techniques and the issues in geocoding. When identifying relevant issues, limiting the scope means a foundation can be built (using intelligence) without deviating into the work being done by other researchers in complementary areas.

By conceptualizing the design and sketching out components, it was realized that there was a need for a knowledge base component to support the agent component. The design included planning for the future with regards to more and smarter web services that the agent will be able to utilize. Having multiple foci of control and parallel behaviour is a deviation from the norm for geocoding, but this is at the crux of the new approach to geocoding control; using agents in conjunction with web services (now and in the future) is a very complementary combination.

The semantics inherent in this design are directly related (enabled) by having the multiple foci of control, made possible by the agents. Extensibility has been considered, and just as individual agents are used to represent geographic entities, so too could additional contributing factors be added in the future as agents. Creating a "base agent" with the most common functionality and then tweaking (adding) to this for the various geographic elements makes design and code reuse easier. Focusing on what needs to be achieved rather than how to do it means new and additional techniques/plans can be added later without having to change the code which uses them.

Using messaging means that without any extra effort the software can be run on multiple machines, as messages can move across networks between agents. By representing address elements as agents, it has provided a way to fuse the practical (software) and conceptual (semantic geographic relationships) together into one.

Splitting the method for examining control and knowledge allowed three focus areas to be utilized and examine the geocoder from different aspects. The *behaviour* aspect analyses happens when real queries are processed, being a quantitative review of several of the unique aspects of intelligent geocoding. Geocoding *process* focuses on outcomes of different, representative address types and results of processing. Derived *functionality* looks at other desirable capabilities of the intelligent geocoding prototype, made possible by the intelligent paradigm.

CHAPTER 4

PROTOTYPE DEVELOPMENT

THIS chapter describes the prototype architecture, key design decisions, novel approaches to the geocoding process and the implementation using an agent based approach and a rule based system. The architecture is presented first to provide background necessary for understanding how the implementation is carried out.

The implementation is referred to as IntelliGeoLocator (IGL). Figure 4.1 shows an overview of the implementation, which includes the JACK agent framework (Agent-Oriented-Software, 2003), the knowledge base (implemented using the JESS API (Friedman-Hill, 2003)), and the supplementary (i.e. reference) data used in the algorithms to solve the geocoding queries. The Java interface in Figure 4.1 was written to simplify access to the knowledge base for the agents. The supplementary data includes databases accessed locally, and also web services accessed via the Internet. Also in Figure 4.1 are the facts and rules which are asserted and modified by the agents.

Both JACK and JESS are implemented in Java, so they are compatible with each other immediately. JACK was selected as it is a very close implementation of BDI theory and traces its roots back to the original proponents of BDI theory. The product is technically mature, stable and well documented. JESS was selected for use as it is the best known Java rules engine, also has technical pedigree (it is based on CLIPS) and is a faithful implementation of the original Rete algorithm used for rule based systems. Both JACK and JESS have been used in real industrial applications, which shows they are reliable and scalable. Using these tools would also provide documentation and support for industry sponsors if they decided to be further involved with the research and the prototype.

The following sections describe the implementation of IntelliGeoLocator,

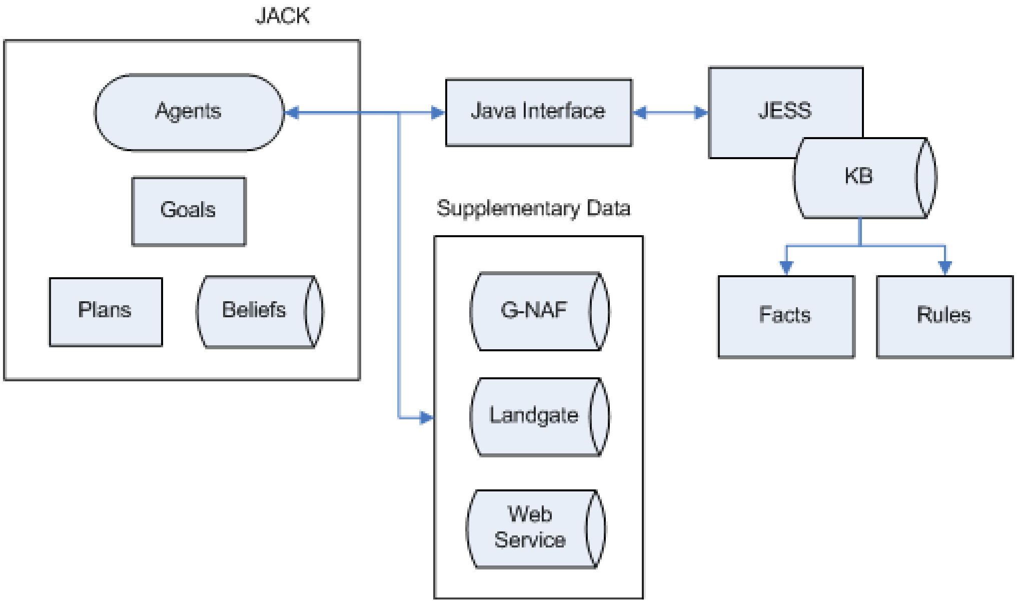


FIGURE 4.1: Overview of the Implementation

using the typical steps in process as a structure for the sections themselves. Every aspect of the geocoding lifecycle is included, from a new geocode query (i.e. address) entering the system, through to matching and locating.

4.1 Initialization and Setup

Because each agent is its own entity, they are each instantiated individually. Also, because these agents are individuals they cannot "see inside" the internal state of other agents, nor is there a global (i.e. shared) memory - each agent has its own protected memory. This means there is a need for sharing information, which is the role of messaging.

The agents in the prototype are only instantiated once, when the whole prototype is started up (which also occurs only once). This means that for every query that comes into the geocoder, new sets of agents are *not* created - rather the query "flows" through the existing instantiated agents; in this way many queries can be processed simultaneously. With this concurrency is the need to track and identify unique queries within the system; likewise when all processing is complete, the query needs to be closed. Only the agents themselves are explicitly instantiated in code by the programmer, components such as beliefsets, events and plans are each instantiated automatically as needed. A beliefset is instantiated once for each agent that uses it, the first time it is used.

In the intelligent geocoder design, there are three types of agent, and in the implementation there are also three *types*, however there are seven *instantiations*. These types include the `UserAgent`, `ElementAgent` and `MatchingAgent`. The `ElementAgent` is a "one size fits all" agent which, after it is instantiated, can take on the role of either `StateAgent`, `PostcodeAgent`, `LocalityAgent`, `StreetAgent`, or `PropertyAgent`. This is made possible by the fact that each of these element agent types perform the same tasks the majority of the time, they only differ in the geographic elements they need to compare and the dataset queries required. Also, a temporary `SenderAgent` is created and used for each new geocode query, sending the address information to the `UserAgent`.

4.2 Brokering and Closing a Query

The `SenderAgent` is used to input whole addresses into the system, which are read from file. The sender agent is the only temporary agent in the prototype; each time a new address is read from file, a sender agent is created and this agent is used to send a message "into" IntelliGeoLocator from the initial Java executable. Once the sender agent has sent the message, it is destroyed. It is feasible though that with an online system accepting queries from users via the web, the `SenderAgent` may not be necessary; instead the `UserAgent` could directly handle the queries from the website. The file which the addresses are read from are in a pipe delimited format; this input means that it is already known which parts of the address are of particular element types. This is contrary to other geocoding research, discussed in Chapter 1, which has focussed on initially identifying (through both rules and probability) which parts of an input string are which elements. Because of this, the assumption has been made that the user has not confused, for example, the value of the street name with the locality. Every time a new query enters the system, the user agent increments an integer stored in a beliefset, to keep track.

Figure 4.2 shows the `MessageAgent` sending the complete address to the `UserAgent`, via `SetAddressEvent`, and the `UserAgent` handling this event via the `IncomingAddress` plan. The `IncomingAddress` plan then sends each of the individual address elements to their respective agents, and also sends the all initial address element values to the `MatchingAgent` for preliminary matching (completely uncorrected).

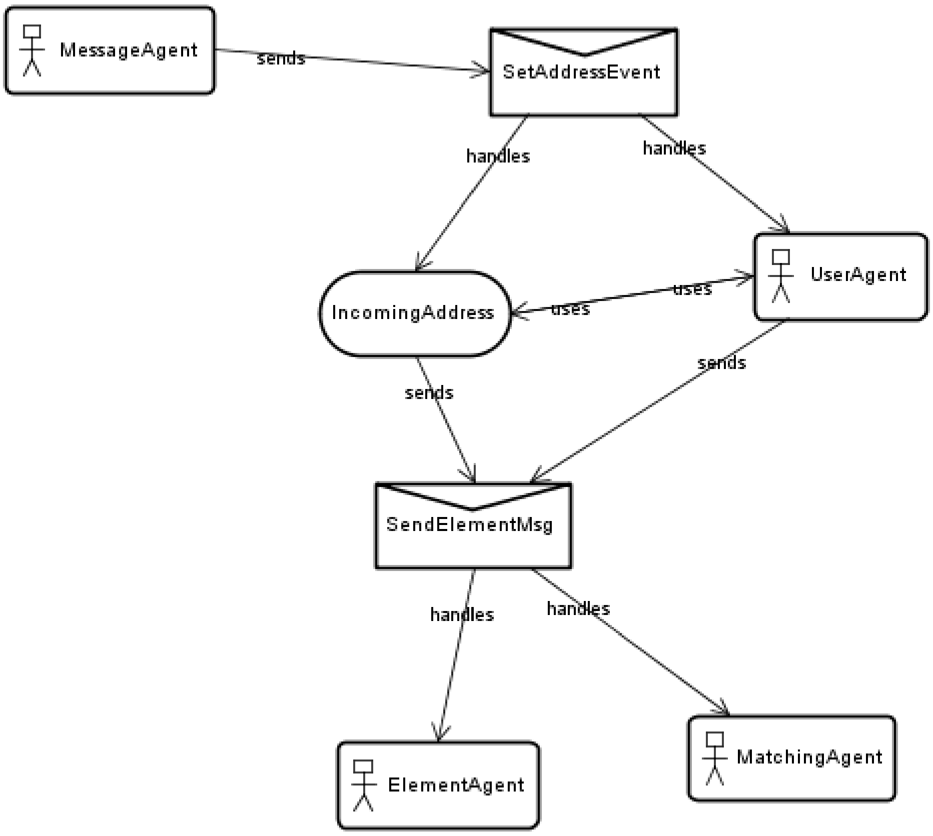


FIGURE 4.2: User Agent Distributing Address Elements

Three key plans involved in this stage are `IncomingAddress`, `ProcessExternalElement` and `InitiateGoal`. The `ProcessExternalElement` plan is used by the `ElementAgent` to add a new address element to its beliefset. `InitiateGoal` is also used by an `ElementAgent` to launch the overall goal of `FindBestValue`, which oversees the correction of the element. The `UserAgent` sends the various elements of the inputted address to each of the `ElementAgent`s in parallel.

4.3 Goal Coordination

As seen in Figure 3.4, the top level goal is *correction of element*, and the implementation of this is the `FindBestValue` goal. Each of the goals (including sub-goals) in Figure 3.4 are posted in sequence, beginning with *Existence*; this reflects that although a new paradigm is being used, the same contemporary geocoding steps are used. Posting these tasks as goals is the essence of using agents rather than a more procedural approach; the agent is approach is "this has happened, now work out how to handle it", i.e. what to do not how to do it (the most appropriate course of action is chosen by the agent at runtime).

Although the goals are posted in sequence, they are only pursued if a logical condition is not already true; this means *that not every goal is necessarily pursued*, for example, the *complementary elements* goal may not be required. This logical condition is included in the command used when the goal is posted, the command includes both the *goal* and an *exit condition*. The conditions are a combination of literal values (such as integers and strings) and also JACK `logical` variables which are bound to a particular value. The agent stops trying to pursue the goal when these conditions are met. Using the *Existence* goal as an example, the condition is that the `exist` attribute (in the belief table) becomes either "true" or "false", i.e. it does not remain at "unknown". In other words, stop processing when the outcome is determined to be either true or false. Other goals use more complex logical conditions by combining attributes from multiple beliefsets.

The `exist` attribute mentioned would have originally been set in the ProcessInfo plan, where initial values are added to the `Values`, `Stages`, `AgreementValues` and `AgreementStatus` beliefsets. This means setting all the attributes to the string value of "unknown". The only exception is identification ID numbers, and also the initial status score which is 1.0. The intermediary plans which post the sub-goals include the *relevance* and *applicability* selection methods within the plan, so that only those sub-goals which are suitable are used. The environment plays a role because the beliefsets dictate whether the sub-goals are used, and also when both the main goal and the sub-goals have succeeded; this highlights the close relationships between beliefs, goals and plans. Because the beliefsets are populated by messages from other agents and also the progress of the agent's own internal processing, the beliefs need to be updated as quickly as possible and thus be as up to date as possible.

4.3.1 Existence

When the existence goal is posted, only one plan will be both relevant and applicable. Element types including state, postcode, locality, street name and street type have a plan to check whether or not they exist. Because house number and unit require other information to determine if they are valid, these elements are excluded from the concept of *existence*. Instead, their validity is determined in the next goal which checks the spatial agreement of a particular element with its complements.

The reference databases used to check this existence are the G-NAF, state data from WA, and postcode data from Australia Post. The G-NAF is accessed via a JDBC connection to a remote MySQL server, while the state data and postcodes (obtained from LandGate) are stored locally using the Cloudscape database.

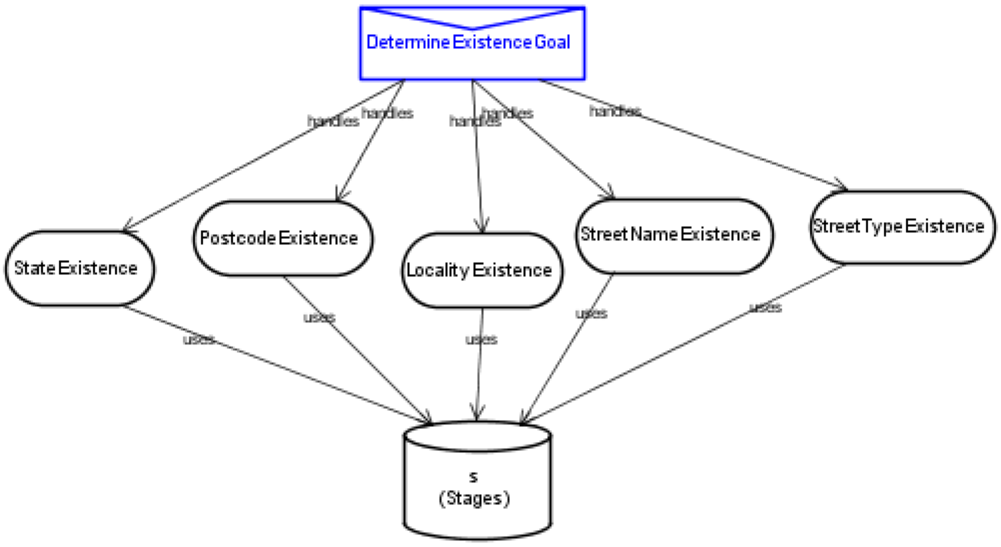


FIGURE 4.3: Plans Used to Determine Existence

A SQL statement is constructed (and stored as a string) using the information stored in the goal event. A custom class was written which then takes the string and executes this against the database; the class returns a boolean variable if at least one record is returned. The result of this query is then written to the InternalValues beliefset.

4.3.2 Complementary Elements

Each of the element types has a plan to determine the complimentary elements for it; only one of these plans is selected. Within the plan, both the complementary elements are found in parallel (i.e. simultaneously). If a particular element has not been submitted in the query, then it is not present and would not be the complementary element to anything. The agents send messages to each other to discover which element is the upper and lower. Each agent starts by messaging the other agents responsible for both of its complementary elements; the sending agent waits for a response from the destination agent. The payload of the reply message contains whether or not the element is present in the query, and whether it exists. If the destination agent has not yet determined this information, then the sending agent waits while it does so. If the sending agent determines that the destination agent is not its complementary element, then it proceeds to message the next possible element, and so on. The UpdateRequest message type includes just the identifying information and the element type it is requesting information about; this is essential, as some agents (e.g. street agent and property agent) handle multiple element types which have the same identifying integers.

Plans provide the full extent of their power and flexibility when there are several plans available to choose from, each with their own criteria for being selected. This becomes most apparent in this research with the use of plans to choose flow of control based on the geography type. This is what allows one element agent type to be designed and then used at runtime to take on the persona of any of the geography types. For example, if the element agent is instantiated as a *locality* agent, then when it comes to finding (for example) the complementary elements or suggestions, the plan specific to doing this for *locality* is chosen.

Each of the plans use the UpdateRequest message event, and that the results of the message interactions are stored in the AgreementValues and AgreementStatus beliefsets. All message replies are of the type UpdateResponse, which includes properties of unique identifier, and string, boolean and double variables which describe (i) the type of the element, (ii) the value, (iii) whether the element is present, (iv) whether it exists, and (v) the status score of the element.

Message sending in other situations is very similar to that of complementary elements; the only essentials are a destination address name (which agent the message is being sent to) and any extra parameters. When designing a message event, the header is similar to a method, and parameters can be defined. Within the message, members (i.e. variables) are assigned values by the *sending* agent, and when the *receiving* agent "reads" the message, it can access the same member values. The best analogy is an envelope, where the sending agent fills it with values, addresses the envelope, and send it to another agent who opens it and retrieves the values; this is why it is essential to include the `queryID`, `subID`, `parentID`, and element `type` for tracking purposes.

The SetAddressEvent message uses a custom object (of class *Address*) to transport it data. This class can be seen in Table 4.1. This class was a convenient mechanism to pass the data rather than setting many variables in the message itself.

TABLE 4.1: The Address Class

Class	Element	Setter	Getter
Address	Flat Number	setFlatNumber	getFlatNumber
Address	Street Number	setNumber	getNumber
Address	Street Name	setStreetName	getStreetName
Address	Street Type	setStreetType	getStreetType
Address	Locality	setLocality	getLocality
Address	State	setState	getState
Address	Postcode	setPostCode	getPostCode

Messaging in the geocoder uses both single messages, and also dialogues of request messages and responses. An example of the "one off" messages includes when an element agent sends a message out to its complementary elements with an update in its own values or status score; this message requires no response.

The logical condition used when waiting for a response or preparing to send a response can also be expressed in terms of a beliefset query. For example, in the `PostcodeRequest` plan, before the postcode agent responds to a request for information about its values and status, its needs to have these established already (operating under the assumption that "unknown" is not an acceptable response). So, the agent waits to reply until it has evaluated its own situation. The rationale behind this is that by waiting, only one request is made and in the long run this will reduce the volume of agent messaging. So, the `PostcodeRequest` plan uses a command to suspend processing until its "present" and "exists" attributes in the InternalValues beliefset are set to either "true" or "false". Once processing is unsuspended, the agent sends the reply. The parameters supplied to the reply include the incoming message to which is being replied, and an instantiated message event which is the response. The UpdateRequest message is the only message listed as having a reply, this is because after the request is sent, a response is expected.

4.3.3 Calculating a Status Score

The intelligent geocoding model has user-defined weighting for each address element (as per the ESRI model), but also includes a score indicating the quality of the element itself. A benefit of this is that both *importance* and a measure of *confidence* (at a given point in time) are included. There are two tiers of scoring, that which contributes to the score of the individual elements, and then the contribution of each element to the overall address score. Because the scoring mechanism inherently includes inter-reliance and communication between elements, it is suited to the parallel processing approach used by the geocoder; it also means the scoring can be updates in real-time as processing occurs. The scoring mechanism also allows for ambiguity in scoring, because there are distinct steps which include the element being present, existing and agreement with complementary elements. This means two elements can both exist yet not spatially agree with each other, however this is handled by the scoring approach. The intelligent geocoding model provides a continuous representation of the match, in conjunction with a weighting.

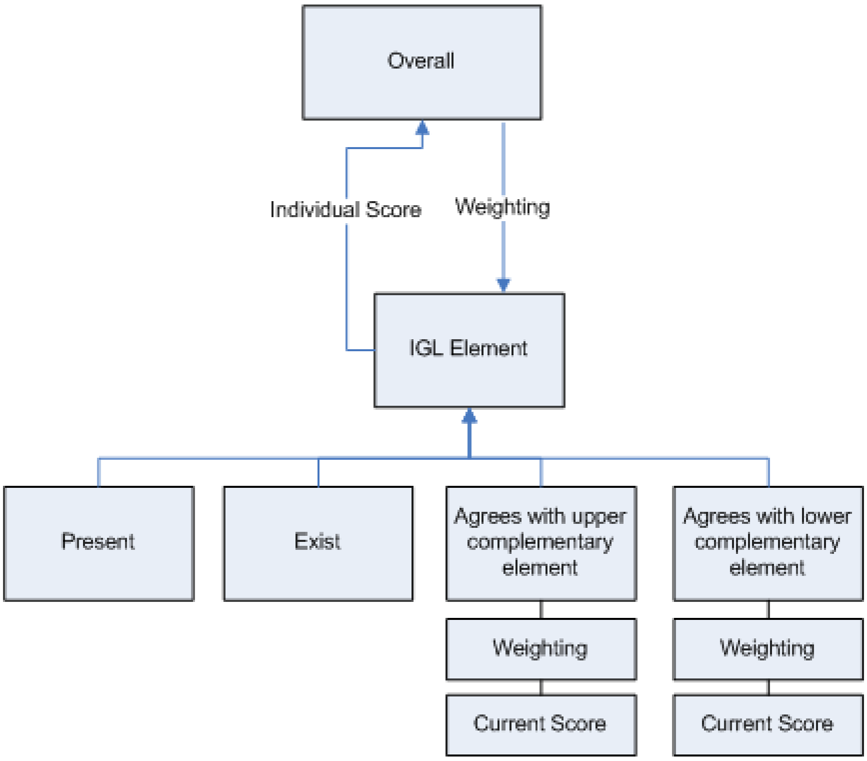


FIGURE 4.4: Quality Scoring in the Intelligent Geocoding Model

The importance of interdependence is shown by how the weighting accounts for this, e.g. it is more important that street name agree with locality than with street type. This importance is also included at the whole address level, with a weighting available indicating the importance of each element type to the overall score. This reflects the idea that ultimately locality is a more important indicator of a match than street type is. This quality scoring approach of IntelliGeoLocator also ties in to the correction philosophy it uses. The approach provides a finer granularity in terms of understanding the problem and helps to find potential solutions. It is also a gateway to a semantic correction (e.g. knowing that two address elements both exist but do not spatially agree may indicate the neighbouring locality was intended, or some other semantic cause). The concept of not treating the elements in isolation ties to the functionality in IntelliGeoLocator of providing equivalent complementary elements of suggestions, which can mean introducing element values not previously considered.

Approach of IntelliGeoLocator to matching is that even if the input query is incorrect, the user had a particular address in mind and that there is a finite number of reasons why they made a mistake (or that the address they had in mind defied convention), and that a solution can be found.

Going beyond this confidence score would be the ability to explain what the geocoder did to arrive at its conclusions and justify why it did this, in a way the user can understand. The user could stipulate whether they want to see this information or not. This provides another reason to use a rule-based system in the geocoder, as these systems can (inherently) provide the steps followed to reach a conclusion.

IntelliGeoLocator provides the following during operation:

- Calculates a score for the original submitted address (which provides the user with an initial idea of the quality of their input as a benchmark)
- Accounts for reduced spatial resolution in the resulting suggestions (relative to the original query). e.g. if the resolution of the original query was at the street level, but the result only matched to the locality level.
- Accounts for incorrect, individual element values when calculating a score (such as with the initial query)
- Uses a weighting system so different address elements can have varying importance in their contribution to the score
- The system has been designed to allow the weightings to be configured according to the preferences and context of the user; for example the confidence a user has regarding the individual elements they have submitted.

There is an event named `StatusRequired`, which is posted anytime a status score needs to be recalculated for an element. This event is a *regular event*, and has only one plan (`CalculateStatus`) which is used to handle it. Regardless of the element type, this plan works the same for all as it was programmed in a generic way. Calculation of the score is done via the formula presented in Appendix A.5.

The majority of code in this plan is used to access the needed values from beliefsets, and the rest performs simple arithmetic (e.g. addition, division etc.) to calculate the score. Some code is also used to round the final decimal value of the score (rounded up). The score is stored in the `Values` beliefset.

There is a consequence when the status score is updated in the beliefset, and is different from its previous value (i.e. its score has decreased or increased) as the beliefset `Values` posts the event `RevisedStatus` which in turn is handled by the plan `SendRevision`, which retrieves the complementary elements for the element with the updated score, and then sends a message to the respective agents with the new status score, and identifying information for the element.

The event is launched from the beliefset via a *beliefset callback*. At the time it runs, both the tuple that is being changed, as well as the new tuple are both available, which means comparisons can be done on both tuples. After comparing the two status scores and finding they are not equal to each other, the `RevisedStatus` event is posted via a beliefset posting method. The calculated score value is then written back to a beliefset (the `Values` beliefset).

4.3.4 Agreement of Complements

To determine whether elements agree, the situation is looked at from the point of view that each element needs to be checked against both of its complementary elements. Because of this, each plan available to handle this goal operates by comparing two specific element types. For example, the `StateLocality` plan checks the existence between the state and the postcode; the `PostcodeLocality` plan checks agreement between the postcode and locality elements. Figure 4.5 shows a selection of these plans (some are not shown).

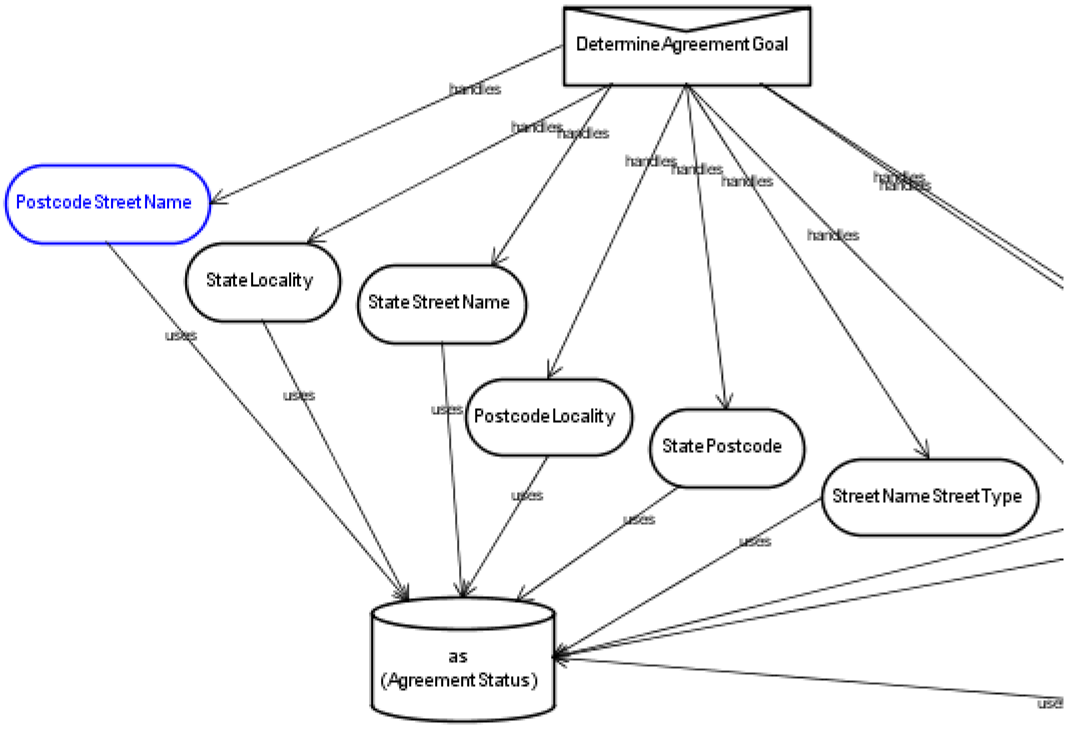


FIGURE 4.5: Determining Agreement Between Complementary Elements

For a given address query, `StateLocality` and `PostcodeLocality` would not both be used, as only one pair of elements can be the complementary pair. The actual code to do the check is a single SQL statement which uses the local and remote G-NAF, Australia Post and WA State data. For address elements such as house number and unit number, this is the first goal to test its validity (as these elements do not use the *existence* goal); the plan for evaluating these elements are the same as for any other elements. The plans for spatial agreement are presented in Table 4.2:

TABLE 4.2: Spatial Agreement Plans

Spatial Agreement Plans	
`PostcodeStreetName`	Verifies the spatial agreement of postcode and street name
`StateStreetName`	Verifies the spatial agreement of state and street name
`StateLocality`	Verifies the spatial agreement of state and locality
`PostcodeLocality`	Verifies the spatial agreement of postcode and locality
`StatePostcode`	Verifies the spatial agreement of state and postcode
`LocalityStreetName`	Verifies the spatial agreement of locality and street name
`StreetNumberUnitNumber`	Verifies the spatial agreement of street number and unit number
`StreetNameStreetType`	Verifies the spatial agreement of street name and street type
`StreetNameStreetNumber`	Verifies the spatial agreement of street name and street number

The same SQL statement is used for each combination of elements, because if one element agrees with the other, then the converse would also be true (assuming a single dataset is used). More research is required for the situation where multiple datasets are used and these differ in their results with regard to spatial agreement. Pairs such as (state, locality) and (locality, street) can be compared against each other in isolation, however some other element types are not this simple. To check the agreement for the street number and unit number it is necessary to use also street name and street type. The results of this processing is written to beliefsets.

4.3.5 Suggestions

Each element type has its own plan to coordinate the creation of element suggestions. In the design, each of these plans then post another goal (a *sub-goal* of the Suggestions goal) which in turn is handled by several plans, each plan corresponding to a suggestion technique (e.g. Soundex, Levenshtein, knowledge base, first order neighbours). In implementation, the use of a sub-goal has been skipped and instead the plan calls the suggestion techniques directly. This has not led to any decrease in functionality, and was done to save time and sufficiently shows proof of concept.

Each plan calls the techniques, using the element value as input, and puts the suggestions into an array. After all the suggestion classes have been used and the array is completely populated, the array is searched and any repeated suggestion values are eliminated. Although some techniques require only the element value itself, others require additional information such as a state or locality. For example, the Levenshtein technique (for finding street name suggestions) requires that a locality also be supplied. In some cases too, the suggestions from *other element types* are used as the additional information necessary to find suggestions for a given element type. An example of this is using the Levenshtein technique to find suggestions for street name - if the *original* locality element submitted in the address query does not exist, then there is no valid locality to use; to overcome this the *suggestions* of the locality elements (created in a different agent) are fed back to the street agent to provide the locality search space required for that technique. The benefit of this is that in some cases, an otherwise missing element is given a value.

For the Soundex and Levenshtein techniques, classes were written in Java, by Shyllon et al. (2007) which use the commonly available Soundex and Levenshtein algorithms. These algorithms were used in conjunction with the G-NAF database. For finding first order neighbours, a custom class consulted the G-NAF neighbours table. With the list of suggestions completed, the array is sent to the matching agent, via the message event SuggestionsForMatching. A regular event is also posted which contains the suggestion list and is the ultimate catalyst for EquivalentComplements being posted. The suggestions plans can be seen in Table 4.3:

TABLE 4.3: Suggestions Plans

Suggestions Plans	
`StateSuggestions`	Finds potential replacement values for state
`PostcodeSuggestions`	Finds potential replacement values for postcode
`LocalitySuggestions`	Finds potential replacement values for locality
`StreetSuggestions`	Finds potential replacement values for street name
`StreetTypeSuggestions`	Finds potential replacement values for street type
`StreetNumberSuggestions`	Finds potential replacement values for street number
`UnitNumberSuggestions`	Finds potential replacement values for unit number

Most of these plans are quite similar, although the techniques used to find suggestions for some element types do not work for other element types. For example the techniques of Soundex and Levenshtein work for locality and street names, but they do not work for street number suggestions. In the prototype, there are suggestions offered for locality and street name. Ideas for providing suggestions for street number and unit number would include providing a list of all the available numbers, or those that similar in terms of the digits.

The `LocalitySuggestions` and `StreetSuggestions` both use the `Suggestions` class, which use the traditional correction techniques in addition to the knowledge base. The results of these techniques are merged, any redundancies are removed and these are the suggestions. For the locality suggestions, the soundex, geographic proximity and knowledge base techniques are used, while for street name suggestions the Soundex and Levenshtein techniques are used. Once the plan has determined the suggestions, it posts a goal to find the adjacents for each suggestion. These adjacents are then sent to the matching agent. In the particular case of the `LocalitySuggestions` plan, it also sends the `BuildSuggestionsGoal` event to the street agent to initiate its building of suggestions (the street agent utilizes the reduced search space provided by the LocalityAgent).

4.3.6 Knowledge Base

A custom Java class was created to serve as a “black box” for use by the agent. This class is named `KnowledgeBase` and contains the methods `open()`, `save()`, `addAlias()`, and `queryAlias()`. The `KnowledgeBase` uses the Jess API, and specifically creates a `Rete` engine which is the primary Jess object and the engine in which the rule based system runs; using the Rete object facts and rules can be added to the engine and queries can be run. Manipulating the Rete engine with its methods is the programmatic equivalent to using Jess at the command line.

When the Rete object is first created, a text file is read in which contains Jess specific commands for initialization of the Jess environment. The commands include the templates, rules and queries created at design time; these are loaded in when the engine is initialized so they are then ready to use. The three main components used for knowledge base implementation (fact templates, rule definitions and fact queries) can be seen in Tables 4.4, 4.5 and 4.6.

The facts of type locality-alias are persistently stored in a separate file; these are aliases stored from previous geocoding sessions. The put-locality-alias is the fact type which is asserted using the values supplied from the agent, this is why the template includes slots for the `queryID` and `subID`. These ID values ensure all activity inside the rule engine is correctly tracked and all processing remains correct. The street-alias fact template is identical to locality alias, except in the address element type it represents. The implementation for rule based reconstruction follows the example shown in Appendix A.3.

TABLE 4.4: Fact Templates

Fact Templates	
`locality-alias`	A locality alias already stored in the KB
`street-alias`	A street alias already stored in the KB
`put-locality-alias`	Incorrect locality value used to search
`put-street-alias`	Incorrect street value used to search
`found-locality-alias`	Asserted when locality alias found
`found-street-alias`	Asserted when street alias found
`potential-element`	An address element asserted by the ElementAgent
`agreement`	Indicates that two address elements spatially agree
`postcode-state-match`	Indicates that the given postcode and state spatially agree
`locality-state-match`	Indicates that the given locality and state spatially agree
`street-state-match`	Indicates that the given street and state spatially agree
`locality-postcode-match`	Indicates that the given locality and postcode spatially agree
`street-postcode-match`	Indicates that the given street and postcode spatially agree
`street-locality-match`	Indicates that the given street and locality spatially agree
`street-sttype-match`	Indicates that the given street and street type spatially agree
`address-match`	Asserted when an address (minimum one pair) is matched

The rules which are loaded include locality-alias-match and street-alias-match. Using locality as an example, the rule compares the locality-alias and put-locality-alias, checking if the values in the `alias-name` and `state` slots of the former match with those of the latter. If they do, then the found-locality-alias fact is asserted; this fact also includes a queryID and subID so results can be tracked within the knowledge base. Also loaded from the text file are two custom functions, named query-locality-alias and query-street-alias which look for facts of type found-locality-alias, with the supplied queryID and subID. These functions can also be called using the Jess API, and the results can be taken from Jess working memory and retrieved for use in the `KnowledgeBase` object, which can then return these to the agent.

TABLE 4.5: Rules in the Knowledge Base

Rule Definitions	
`locality-alias-match`	An alias has been found for the inputted locality
`street-alias-match`	An alias has been found for the inputted street
`postcode-state`	Asserts the `postcode-state-match` fact
`locality-state`	Asserts the `locality-state-match` fact
`street-state`	Asserts the `street-state-match` fact
`locality-postcode`	Asserts the `locality-postcode-match` fact
`street-postcode`	Asserts the `street-postcode-match` fact
`street-locality`	Asserts the `street-locality-match` fact
`street-sttype`	Asserts the `street-sttype-match` fact
`match-4-1`	State, postcode locality, street and street type match
`match-3-1`	State, postcode, locality and street match
`match-3-2`	State, postcode, street and street type match
`match-3-3`	State, locality, street and street type match
`match-3-4`	Postcode, locality, street and street type match
`match-2-1`	State, street and street type match
`match-2-2`	State, postcode and locality match
`match-2-3`	State, locality and street name match
`match-2-4`	Postcode, street and street type match
`match-2-5`	Locality, street and street type match
`match-1-1`	State and postcode match
`match-1-2`	State and locality match
`match-1-3`	State and street name match
`match-1-4`	Postcode and locality
`match-1-5`	Postcode and street name
`match-1-6`	Locality and street name
`match-1-7`	Street name and street type

Most of the methods in the KnowledgeBase object are straightforward; open() is used to read the facts from file and load these into the knowledge base. Similar to this, the save() method writes any locality-alias facts to file. The addAlias() method has parameters including `type`, `aliasName`, `realName` and `stateName`; this is all the information needed to add a new alias. When saving the facts, Jess uses a method to write to plain text; if needed, Jess can also write to XML (facts and rules) although this is not used in this research.

From the point of view of the agent, the parameters used with the `queryAlias()` include *queryID*, *subID*, *type*, *alias*, and *state*. The parameter *type* is used as a switch to specify whether a locality or street name element type is the subject of the query. The *alias* parameter is the term (a string) being queried, and the *state* is the Australian state being searched within. Results are returned to the agents a Jess `QueryResult` object, which is similar to a `ResultSet` object in Java database programming.

TABLE 4.6: Fact Queries

Fact Queries	
`query-locality-alias`	Checks if any `found-locality-alias` facts have been asserted
`query-street-alias`	Checks if any `found-street-alias` facts have been asserted
`find-matches-all`	Finds all matches of any quality (number of pairs)
`find-matches-level-4`	Finds matches comprised of four pairs
`find-matches-level-3`	Finds matches comprised of three pairs
`find-matches-level-2`	Finds matches comprised of two pairs
`find-matches-level-1`	Finds matches comprised of one pair

4.3.7 Equivalent Complements

The plans used for determining the equivalent complements are dependent on *what the complementary elements are*. Because of this, there are several plans which cater for each possibility. For example, there are the plans `LocalityEquivalents_State` and `LocalityEquivalents_Postcode` which find the equivalent state or equivalent postcode for a locality, depending on which is the complementary element for locality. Several Java classes were written which interrogate the state data and postcode data to find the equivalent complementary element for a given element. The classes return an array, in case there are multiple results.

The resulting arrays are sent to the matching agent to be used in the reconstruction process. The equivalent elements plans can be seen in Table 4.7. These plans all have the same purpose, but differ in the address elements being used for each. Their purpose is to find the equivalent element for each of the suggestions previously found by the geocoder.

TABLE 4.7: Equivalent Element Plans

Equivalent Element Plans	
StateEquivalents	*Not used*
PostcodeEquivalents	Find the state equivalent for a given postcode
LocalityEquivalents_State	Find the state equivalent for a given locality
LocalityEquivalents_Postcode	Find the postcode equivalent for a given locality
StreetNameEquivalents_StreetType	Find the street type equivalent for a given street name
StreetNameEquivalents_Locality	Find the locality equivalent for a given street name
StreetNameEquivalents_Postcode	Find the postcode equivalent for a given street name
StreetNameEquivalents_State	Find the state equivalent for a given street name
StreetNumberEquivalents	*Not used*

For a given suggestion, the equivalent element is always the first available element of a larger geographic size. If a given value has several complementary elements (e.g. a locality is in several postcodes) then all of these are retained and used. To implement this, a call is made to the reference database to select the values (the equivalent element) based on the query.

At runtime, the equivalent element will be based on which complementary elements that are present. For this reason, flexibility in the code is needed to cater for this situation. For example, the locality element could have to find equivalent elements of either postcode or state, depending on which is available; the agent knows which complementary element is available based on previous messaging with other agents. There are several plans, and these cater for different cases amongst element type pairs. An example of this is having both LocalityAdjacents_State and LocalityAdjacents_Postcode; these plans are also cases where both `relevant()` and `context()` are used for plan selection.

Once the equivalent elements have been found, the values are placed in an array and sent (via a message) to the MatchingAgent.

The `relevant()` method is the first tier of filtering whether the plan should be selected, and if the plan makes it past this first tier then the `relevant()` method is used. An important consideration is also that the `relevant()` method is static, while the `context()` method is not. This means the relevant() method is useful for selecting plans based on element geography types (which are strings) or any other event methods that included in the event "envelope". The `context()` provides the critical link to the beliefsets, which because it is not static can query the beliefset in realtime and compare the result cursor to the logical conditions defined in the `context()` header itself. This means `context()` is useful for checking whether (for example) an element already "exists" by querying the `InternalValues` beliefset. The `context()` method can also be used for static tests, it does not have to be used for testing beliefsets. This means that when finding the equivalent complements for an element, the appropriate plan be found by first checking (using `relevant()`) which element type is being is being processed, and then secondly (using `context()`) the element type which it has as its complementary element.

4.4 Matching

As seen in Figure 4.6, the matching agent receives address elements from up to three different message events, including `PerfectsForMatching`, `EquivalentsForMatching` and `SuggestionsForMatching`. Elements are only sent via the `PerfectsForMatching` if the element both exists and agrees spatially with its complementary elements.

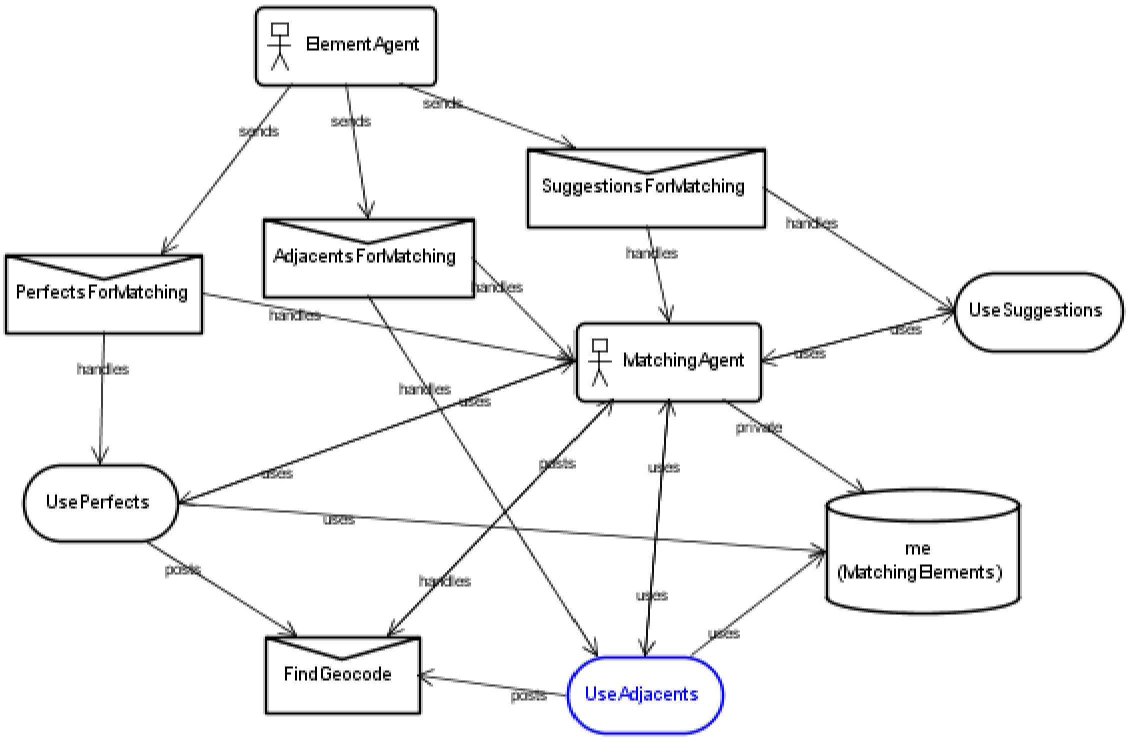


FIGURE 4.6: Agent Matching Process

The plans `PerfectsForMatching`, `EquivalentsForMatching` and `SuggestionsForMatching` assert their respective arrays into the Jess engine object, where each element within the array is a unique fact. The Jess engine instance was created when the original IntelliGeoLocator executable was run, and this object is passed to the matching agent when it is instantiated. To assert facts “inside” the Jess engine, the `executeCommand()` method of the `Rete` object is called and the parameters of this method include the Jess command `assert`, which places a fact in working memory. The values for the slots in the assert command are taken from the elements contained in the message arrays.

As more facts are asserted into the Rete engine, the `run` command is used to ensure all the required rules are fired. As these rules fire, new facts are asserted which represent the matches found. To access the solution facts, a *query* is used inside Jess to select all the facts and return these as a QueryResult object. Parameters to the query include what quality level is desired. By running the query and putting the results into a QueryResult object, it means the values contained in the facts can be accessed in "regular Java", outside the Rete object itself. The QueryResult object is similar in use and concept to a ResultSet object used in Java to query a database. This QueryResult is then iterated through, and the various elements from the address are put into a single string and an event is posted (`FindGeocode`) begins the geocoding process. The final category of plans are the Matching plans, seen in Table 4.8.

TABLE 4.8: Matching Plans

Matching Plans	
`ProcessElement`	Adds new element to matching beliefs and posts `FindGeocode` event
`Geocode`	Queries web service with address
`UseAdjacents`	Asserts complementary elements and executes rule based sysem
`UseSuggestions`	Asserts suggestions into the rule based system

The ProcessElement plan is used every time a new address element enters the geocoding system. When the UserAgent sends a new geocode query to the ElementAents, the ElementAgent also sends an immediate copy to the MatchingAgent. With a new address, the elements will reach the MatchingAgent at slightly different times, but the ProcessElement plan will aggregate this as they come in; this is how the geocoder derives its dynamic geocoding refinement of the raw elements in a new query. It performs the geocoding by posting the FindGeocode event. This concept of different pieces of information arriving at different times, and refining results is well suited to the idea of fusing agents for interactive geocoding sessions with users.

The Geocode plan is critical, it actually performs the query to find the coordinates. From an implementation perspective, it is sent all the address elements it needs inside the FindGeocode event. It tests each of these address elements to see which have values and builds a string using them.

A custom class named QueryEngine was created to work with the Java web service library provided by LisaSoft. This custom class is for convenience, and sets settings such as username and password, and other parameters such as the country. An instantiation of the custom class is created and the address string is passed to it. The web service query is invoked, and the resulting latitude and longitude is retrieved from the object.

UseSuggestions is more simple in terms of code than UseAdjacents, as it just asserts each of the values in its array. In addition to asserting it values also, the UseAdjacents plan also run the rule based engine; this is what finds the matches.

A query has been developed to query the rule engine, and retrieve any results from the matching. This particular query (named "find-matches-all") is used to find any results; there are other queries written to retrieve only those results with a specified number of matching pairs; this is another form of quality measure. The UseAdjacents plan also posts the FindGeocode event, which in turn is handled by the Geocode plan mentioned above.

Also within this plan is the code to write new address elements to the knowledge base, for both street names and locality names. If the geocoder has an address element that is originally incorrect, and then finds a single, final solution, the geocoder writes the "incorrect / correct" pair to the knowledge base.

The other important assertion it writes to the knowledge base is that the suggestions and adjacents "agree". There is a specific fact type to represent this.

4.5 Geocoding

The plan which handles the geocoding is named simply `geocode`, and this uses a Java class called `Address`, which stores the element values in properties and includes a method named `.geocode()` to access a web service provided by LisaSoft, and then returns the coordinates (via `.getLon()` and `.getLat()`). LisaSoft is one of the industry sponsors involved in the geocoding prototype. The service is XML based, and result codes are provided in the web service, provides an indication of result resolution. For example, it specifies whether the match was at the house, street, locality, postcode or state level.

The authoritative dataset used to get coordinates is the Geocoded National Address File (G-NAF) developed by PSMA Australia (Richards and Paull, 2003); for the prototype a web service was used which accessed the dataset stored at another location (a partner company, LisaSoft). To use the web service, first a URL was built using the address of the geocoding engine along with several parameters attached to the end of the address.

The parameters included *engine*, *countryCode*, *freeFormAddress*, *flatNumber*, *levelNumber*, *buildingName*, *number*, *streetName*, *streetType*, *locality*, *state*, *postalCode*, and *geocode*. Not every parameter was used, the most important were the element types; the more element types included in the query then the better the resolution of the result (assuming the elements were correct).

Each agent has its own local connection to the Cloudscape database containing LandGate and Australia Post data. Cloudscape is a pure Java relational database that can be embedded in projects with an extremely small footprint. This database is opened when the geocoder is first run and accessed as needed by the agents.

4.6 Agent Based Approach

Based on experiences from the development of the prototype, agent-based design seems like an effective way to design software. To measure this effectiveness, a paper by Jennings (2001) was used, in which the author suggests that although data is simply not available to show the quantitative benefits of agent design, there are qualitative justifications. To argue the affirmative, Jennings (2001) states it is essential to show that (i) “agent-oriented decompositions are an effective way of partitioning the problem space of a complex system” (p. 37), (ii) “the key abstractions of the agent-oriented mindset are a natural means of modelling complex systems” (p. 37), and (iii) “the agent-oriented philosophy for modelling and managing organizational relationships is appropriate for dealing with the dependencies and interactions that exist in complex systems” (p. 37).

For the decomposition, Jennings (2001) summarizes that "the natural way to modularize a complex system is in terms of multiple autonomous components that can act and interact in flexible ways in order to achieve their set objectives" (p. 38).

- The components have been modularized in terms of the objectives they achieve. In the prototype this is seen from each agent having a different role according to their geographic type; there is also the `UserAgent` and `MatchingAgent` which have different objectives too.
- Complex systems have multiple loci of control. This is true for Intel-liGeoLocator, and especially ties in with the idea that the prototype try to avoid the linear geocoding process; in the prototype, control is distributed amongst all the agents. In order to achieve this, each agent is active and autonomous.
- Interaction is essential and with very complex systems it is impossible to know ahead of time all the interactions that will be needed; it is beneficial then that the intelligent geocoding model is extensible meaning additional beliefs and plans can be built into each agent. Jennings (2006) indicates this is due to unpredictable times, reasons and the components themselves. To handle this, the agents require an ability to make decisions at runtime; and although an agent may not expect a particular type of message they can determine what to do when required. Although the prototype is not this complex, it did assist design by not having to explicitly define the timings and order of messaging. It was shown that the quality measures involve many and varied sequences of messaging, where new statuses and scores are shared between agents; none of the interactions *between* agents were specified at design time, only the messages as far as each agent was concerned. This approach of runtime decision making about messages has the benefits of reducing coupling and the burden of determining control relationships at design time.

With regard to the abstraction, Jennings (2001) suggests that it is desirable to minimize the difference between the problem being solved and the paradigm of the tool used to solve it - and that subsystems are well suited to being modelled by agents because "they involve a number of constituent components that act and interact according to their role within the larger enterprise" (p. 38). Other benefits for abstraction include (i) High-level social interactions between subsystems and components, (ii) Viewed from different level of abstraction, (iii) Changing webs of relationships, (iv) Can represent collectives, and (v) Agents suited to the cooperation, coordination and negotiation found in systems.

When it comes to managing organizational relationships, the benefits of using the agent design approach include (i) The ability to flexibly form, maintain and disband organizations (Jennings, 2001), and (ii) The ability for "individual agents or organizational groupings can be developed in relative isolation and then added into the system in an incremental manner".

An interesting aspect, and a benefit, of using the agent programming paradigm was that the focus is on building the agent and the messages it can send and receive; the actual timings of when to send messages and exactly what will happen was not explicitly defined. In this way, the system is more than the sum of its parts - when the system is run there is no way of telling exactly what messages will occur in what order, but in the end the system arrives at the answer via messages. One of the core outcomes of the research has been that functional geocoder can be built using an agent paradigm, demonstrated by the prototype running from start to finish. As part of this, it was shown that the key agent constructs of goals, plans, beliefsets, messages and events can all serve a purpose in agent based geocoding.

Beliefsets provided time-sensitive structure, forces processing to be up-to-date, persistence allowed for coordination of multiple geocode queries through system, efficient querying, beliefset callbacks allowed for mechanism of performing operations when adding, modifying and deleting tuples, provided control by storing the status of processing, weights, values used for distributed processing, "empty" structures populated as processing continued, provide the storage needed for the transient nature of event-driven operation.

Factors that were dynamically monitored included whether the immediate address element was "present" and "existed", along with it's spatial agreement with the elements above and below it, also monitored was the rating value and element type of the elements above and below; this was in flux in real-time. Because the agents automatically notify each other of updates, it is almost a "set once and forget" situation where as a programmer the algorithm is not completely set, the agent will communicate and modify as needed.

4.7 Conclusions

From a programming perspective, JACK and JESS integrated together in a straightforward way. It was also intriguing to watch the overall system in operation. In particular watching the interactions/messages between agents was interesting because of the self-determining sequencing and the fact that messaging was different for each address tested.

Using the same "base" agent with minor tweaks for the role of the agent lends itself to extensibility. Tracking is so essential because agents are their own entities with protected memories, and they need any data for what they are expected to decide on (via messaging) there is a lot of information being sent throughout the geocoder.

It was determined that there is such a close relationship between goals and beliefs because beliefs are the measure for goal success. The beliefset was a useful data structure for geocoding, and insights discovered during development. An example of this was not to iterate through beliefsets, instead they change from instant to instant and acting on them should not involve time (i.e. iteration) but rather reaction based on triggers. It is the temporal aspect of beliefsets which set them apart from other data structures; the beliefset is not a common data structure in regular programming languages, the closest would be an in-memory embedded database.

Goals mean the steps can be revisited only as needed, this is flexible and means the developer can have many possible courses of action without having to worry about how this will affect each other (side-effects, sequence) as at runtime the agent will decide which to use. The temporal aspect and time sensitive focus of the whole agent tool makes it suited to real-time queries from the Internet. In addition to sending messages is the extra option of waiting for a response, this is a useful technique as it provides a balance between autonomy and social behaviour.

Broadcasting/sending updates is more efficient than other agents constantly asking for them. The scoring system is useful because it simultaneously considers the score of the address and its elements. Because an element's score is calculated in its own status and that of its complementary elements, a ripple effect between elements occurs however messaging is regulated (self-regulated) by agents to ensure there are no artificial/false frenzies where scores increase for no reason.

Spatial agreement between elements in combination with tracking/multiple iterations means other possible addresses can be pursued and at least see if one turns out to be decidedly better. Having a dedicated sub-goal for suggestions means there is a clear place to insert new suggestion/correction techniques, and it is here that correction techniques which take spatial cognition/perception of the user into account (now with semantic and geographic category, more in the future with further research). Correction techniques are only applied to the appropriate element types, and this is use of context in processing. The relevant() and context() methods in a plan provide a significant form of context for the agents. Because rule based systems have no fixed algorithm, it is suited to pulling in data from different agents and then aggregating it. The address and element relationship is analogous to the rule and fact relationship, both have parts and a whole which is why it is so suited.

CHAPTER 5

RESULTS AND DISCUSSION

THE results include the presentation of how the prototype behaved when it processed a range of real addresses, the robustness of the prototype when geocoding a range of sample addresses, and a summary of the functionality which emerged from the prototype. These results are obtained using the testing methodology in Section 3.3. Results indicate that a quality score was used successfully when implemented when using an agent oriented approach, real-time updating of status scores was successful and the agent approach does provide a means for processing many queries simultaneously in a parallel and distributed way. Address testing shows there are no significant shortfalls by using the agent approach. Results also show that there is novel behavior in the prototype, which is underpinned by context, semantics, control and knowledge.

5.1 Evaluation of Intelligent Geocoder Behavior

This section focuses on observations made about the prototype as real data was being processed. Some of the results were unexpected (such as the behavior of agents when finding the appropriate quality score) while others confirmed what had been expected (success of real-time geocodes and parallel processing).

5.1.1 Geocode Match Quality

The scoring algorithm as a basis of geocoding quality was tested using several types of addresses (of varying correctness), however for the purposes of illustration and explanation in this section, an address known to eventually be scored at 1.0 (i.e. fully resolved and completely correct) was used. This was to highlight the number of steps and communication required to reach this score. An interesting finding was that for addresses which do not reach 1.0 (i.e. their quality is lower), the individual agents (and subsequently the overall score) reach a maximum score between 0 and 1. This is interesting because there is no overall control mechanism telling the scoring process to stop, instead it is the consensus of agents (based on messaging and updates to individual agent scores) which controls when the process stops; more is mentioned about this later in this section. The results are presented in detail because the scoring and storage of the scores in beliefsets is a fundamental aspect of agent control for IntelliGeoLocator.

The formula for calculating a match score (see Sections 4.3.3 and A.5) has some conditions with it such that when calculating a score when elements do not exist (such as during processing before the final address solution is found), those elements that do not exist are not used in the calculation. This is because when calculating a score during processing, the elements which do not exist *should not* have any contribution. Each element type has its initial score calculated "internally", in the sense that no other data is used other than data known about itself; this includes whether or not the element is present and whether or not it exists. This is demonstrated using the same example address from the section on iterative processing. The abbreviations used in the Tables 5.2 through to 5.12 are listed in Table 5.1.

TABLE 5.1: Abbreviations Used in Score Tracing Examples

T	Type
V	Value
S	Address Element Score
P	Present
E	Exist
AU	Agree with Upper Complementary Element
AL	Agree with Lower Complementary Element
WP	Weight assigned to element being present
WE	Weight assigned to element existing
WU	Weight assigned to Upper Complementary Element
SU	Current Score of Upper Complementary Element
WL	Weight assigned to Lower Complementary Element
SL	Current Score of Lower Complementary Element

Table 5.2 shows the initial scores for the elements based on whether the elements are present and exist; at this point of calculation, the agents are not aware of information about the other elements. Even when the various elements are sent the information (in particular the score and value of its complementary elements), it does not affect the element scores because the elements do not spatially agree.

TABLE 5.2: Initial Scores for Address Elements

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.60	Y	Y	n/a	N	1	2	n/a	n/a	2	0
locality	shentin park	0.13	Y	N	N	N	1	2	3	0	2	0
st name	glosster	0.14	Y	N	N	N	1	2	3	0	1	0
st type	st	0.43	Y	Y	N	n/a	1	2	4	0	n/a	n/a

Table 5.3 shows that when an agent calculates the initial score, only the criteria of whether the address element is present and exists is used, i.e. complementary elements do not contribute to the score at that point of processing.

TABLE 5.3: Scores of Complementary Elements have No Effect

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.60	Y	Y	n/a	N	1	2	n/a	n/a	2	0.13
locality	shentin park	0.13	Y	N	N	N	1	2	3	0	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0	n/a	n/a

In Tables 5.4, 5.5, 5.6, 5.7, 5.8 and 5.9 it is shown how the scores of the elements increase as a result of other agents updating their scores and sending these updates as messages. Table 5.4 shows the state spatially agreeing with the locality (row 2, column 7 there is a “Y”), the locality existing, and the locality spatially agreeing with the state; each of these criteria contribute to the score of the element. Note that only a few changes are shown in each table to indicate the transitions that are passed through.

TABLE 5.4: Spatial Agreement and Existence Affect Scoring

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.65	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.13
locality	shenton park	0.60	Y	Y	Y	N	1	2	3	0.60	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

Table 5.5 shows how once an agent updates the score for one of its own elements, it sends this to the other relevant agents. In this case, the state agent incorporates the updated locality score and in turn updates its own score.

TABLE 5.5: State Store Increases based on Locality Increase

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.84	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.6
locality	shenton park	0.60	Y	Y	Y	N	1	2	3	0.60	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

In the next several tables, the only changes which occur are the state agent and locality agent sending updated scores to each other, and in turn updating their own scores. Table 5.6 shows the locality score increasing as a result of the current score of its upper complementary element increasing *its* score. Interestingly, the increase in locality score from Table 5.6 plays a role in Table 5.7 where it is the catalyst for an increase in score for the “state” element. This is because locality is the lower complementary element of the state, and this demonstrates the ripple effect of how an increase in one element affects others. This is why the message-based scoring design ensures that agents do not enter a “race condition” where they continue to update the scores of other agents continuously. Table 5.8 shows that the locality score increased as a result of the state score increasing (seen in Table 5.7) however the state score does *not* subsequently increase (seen in Table 5.9). Avoiding this race condition is made possible by the agents knowing whether they are the “only two” involved in the increments; this is achieved by messaging and storing information in beliefsets.

TABLE 5.6: An Increase in Locality Score

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.84	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.6
locality	shenton park	0.69	Y	Y	Y	N	1	2	3	0.84	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

TABLE 5.7: An Increase in State Score

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.88	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.69
locality	shenton park	0.69	Y	Y	Y	N	1	2	3	0.84	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

TABLE 5.8: An Increase in Locality Score

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.88	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.69
locality	shenton park	0.71	Y	Y	Y	N	1	2	3	0.88	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

TABLE 5.9: An Increase in Locality Score did not Increase State Score

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.88	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.71
locality	shenton park	0.71	Y	Y	Y	N	1	2	3	0.88	2	0.14
st name	glosster	0.14	Y	N	N	N	1	2	3	0.13	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

It should be noted that in Tables 5.6 to 5.9, the affect of the street name and street type have not been shown, so that the interaction between just two agents can be understood. In particular, a behavior has emerged where although the agents are contributing to each other's score, the score for each element reaches a limit after which no further change occurs. In this example, the limit for the state is 0.88, and for the locality it is 0.71. This can be seen in Figure 5.1.

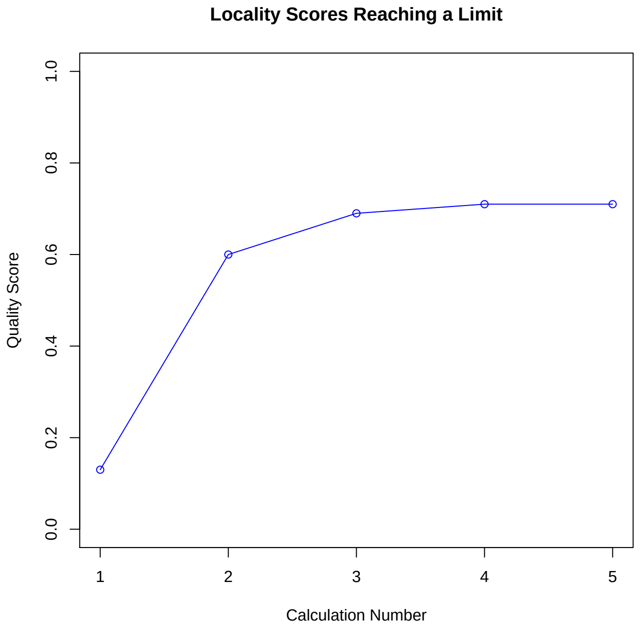


FIGURE 5.1: Locality Element Score Change Reaching a Limit

This means that although the state agent and locality agent are contributing to the score of each other, the score does not become useless by simply increasing every time to 1.0 regardless. Instead the algorithm "sorts itself out" by reducing the increment in score each time. The results indicate that there is an initial increase in score, which can be attributed to spatial agreement occurring and the incorporation of the score of a complementary element in its own score calculation. The rapid decrease in the rate of scores sent between agents can be attributed to *division* being used in the formula for element scores. The use of division in the formula actually seems to be an essential ingredient, if the formula had used just *addition* for example, the agents would not have come to an organic stop short of 1.0. The uniqueness of this algorithm is that the formula itself, not programming code, is the main reason behind the agents finding the appropriate score and not incrementing beyond this. It is also important to remember that updates of scores are sent from an agent only to its complementary elements; this means that although a change at one "end" of an address can ripple to the other (e.g. state through to street type), it limits the direct increasing of an element score by too many agents.

Figure 5.2 shows the complete graph of which Figure 5.1 is a part. Figures 5.2 and 5.1 show the different scores which the locality element has during its processing. The purpose of including Figure 5.2 is to show how the score moves through phases, and how the steepness is tied to significant events (e.g. existing, or spatial agreement with another element); the flatter segments of the graph correspond to where the score was "tweaked" (increased slightly) due to a slight increase in another element. Figure 5.3 shows the scores of the locality element at *every* point of processing where it or another element updated its score. The purpose of including these two types of graphs is to (i) focus on the scores of the individual element and the transitions between these (Figure 5.2), and (ii) focus on the element scores over time, where "time" is the total number of increments that occurred in the system (Figure 5.3). When these two types of graphs look similar, it means that the number of updates (i.e. number of increments) to the element's own individual score was similar to the number of updates in the overall system; this means the element was a major contributor to the overall address score.

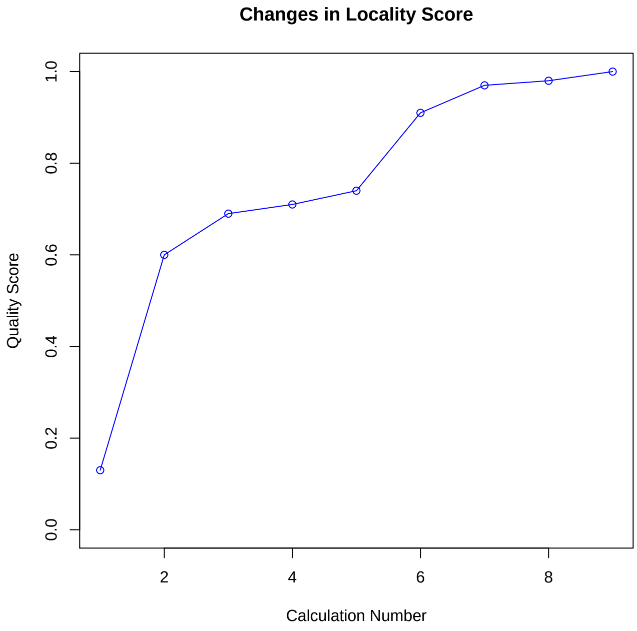


FIGURE 5.2: Complete Graph of Changes in Locality Element Score

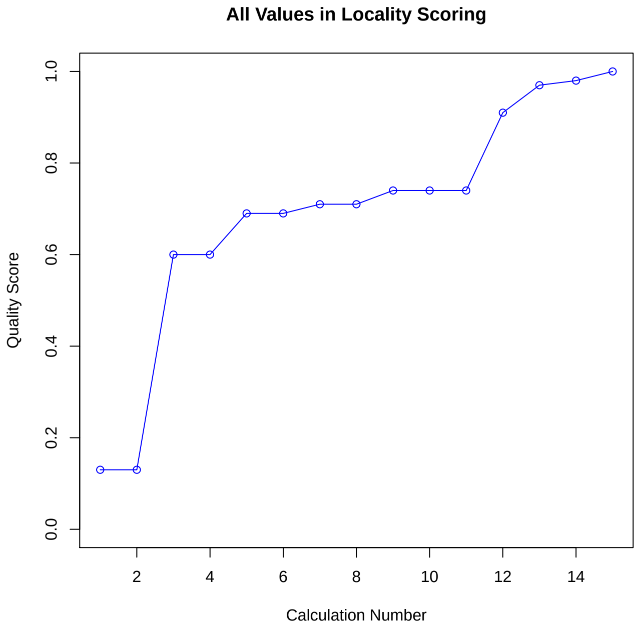


FIGURE 5.3: All Values in Locality Scoring

Figures 5.4 and 5.5 show that the change in score for the "state" element is steady, and occurs mostly uniformly across the life of the query; this occurs from a combination of the state element (i) having only one complementary element, and (ii) weighting its own presence and existence the same as spatially agreeing with the locality. The weightings used for the example are shown in Table 5.10, where the → arrow points to the element being weighted. For example, "State → Locality" means that state uses a weighting of x for locality when calculating the state score. The larger the weighting, the more important the element is in the calculation. For example, when calculating the score for locality, a weighting of 3 is used for state and a weighting of 2 is used for street name; this means agreement with state has more influence than street name. The rationale for this was that within a state in Australia, the locality names would be unique, and that there could be multiple streets with the same name in the same locality. These weightings can be configured as needed.

TABLE 5.10: Weightings Used for the Scoring Example

State → Locality	2
State ← Locality	3
Locality → Street Name	2
Locality ← Street Name	3
Street Name → Street Type	1
Street Name ← Street Type	4

Note that every element type has a weighting of 1 for being "present", and a weighting of 2 for "existing". These weightings reflect that an element existing (i.e. it is found in a reference dataset) is much more important than the element simply being present in the geocode query (being present says nothing about whether the element is correct).

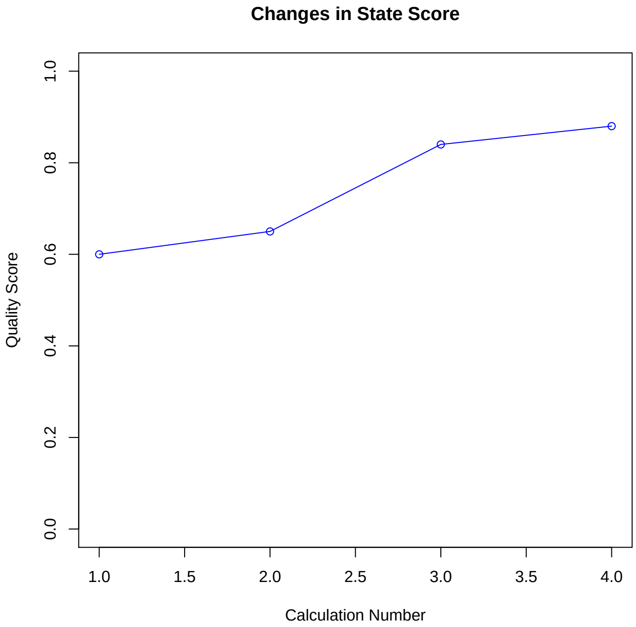


FIGURE 5.4: Changes in State Score

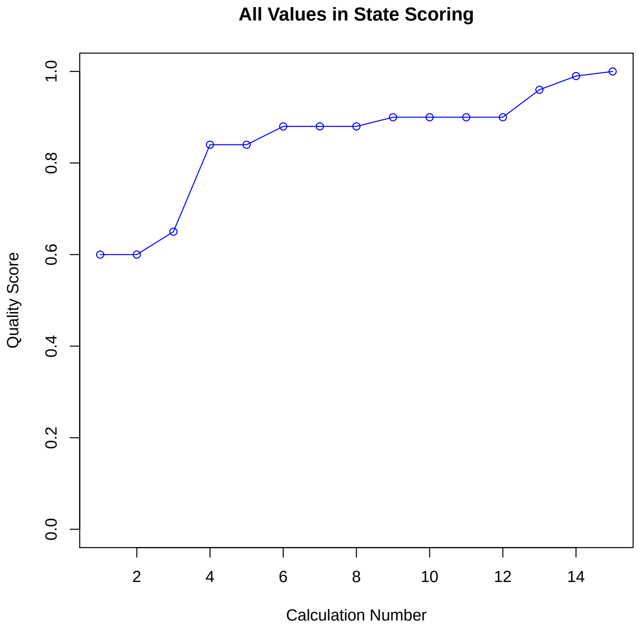


FIGURE 5.5: All Values in State Scoring

It can be seen from Figures 5.6, 5.7, 5.8 and 5.9 that the street name and street type elements each have a steep increase in their scores, and this is related to their reliance (in terms of weighting) to the locality. Specifically the street name has a weighting of 3 for the locality (i.e. quite important) and a low weighting of 1 for the street type. The street type element is heavily reliant on the street name (and indirectly, the locality), as it has a weighting of 4 for the street name, and this is its only complementary element.

The weightings are stored in beliefsets, which can be modified at runtime. This means that with a suitable user interface, the user would be able to alter these, or an application could alter these as parameters in a web service. In the future, other factors affecting quality could also be added, and would flexible in the same way.

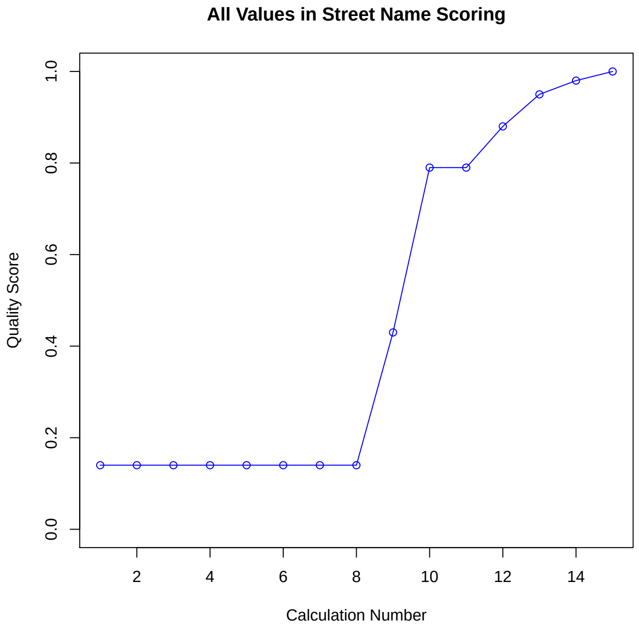


FIGURE 5.6: All Values in Street Name Scoring

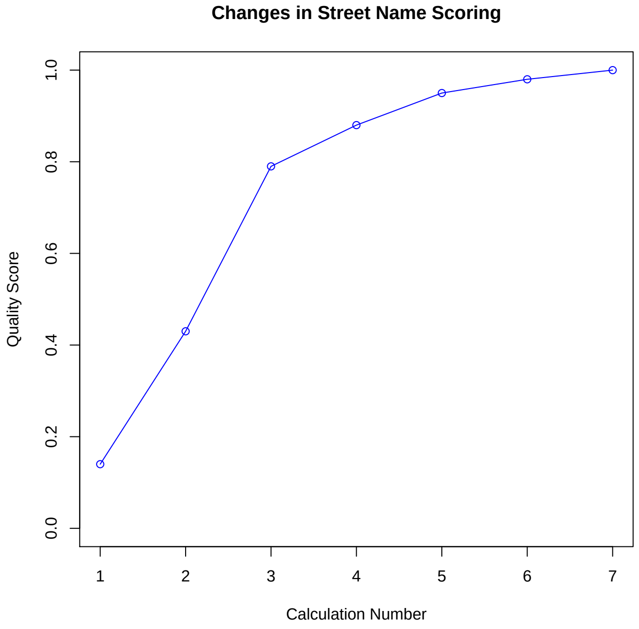


FIGURE 5.7: Changes in Street Name Scoring

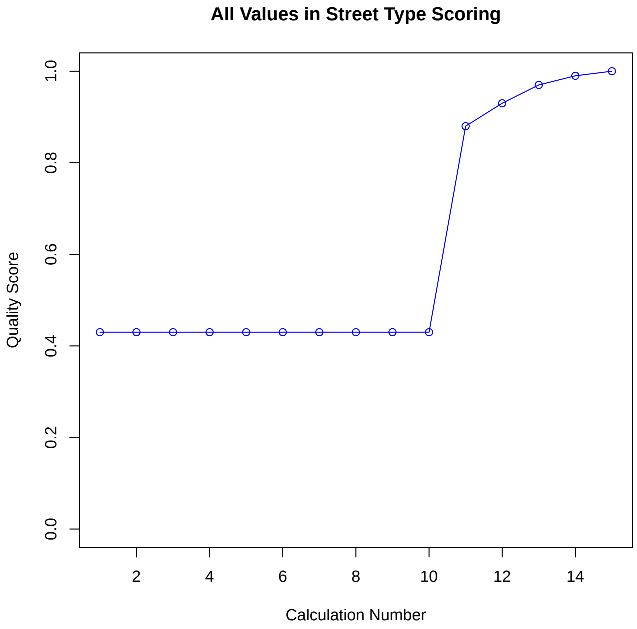


FIGURE 5.8: All Values in Street Type Scoring

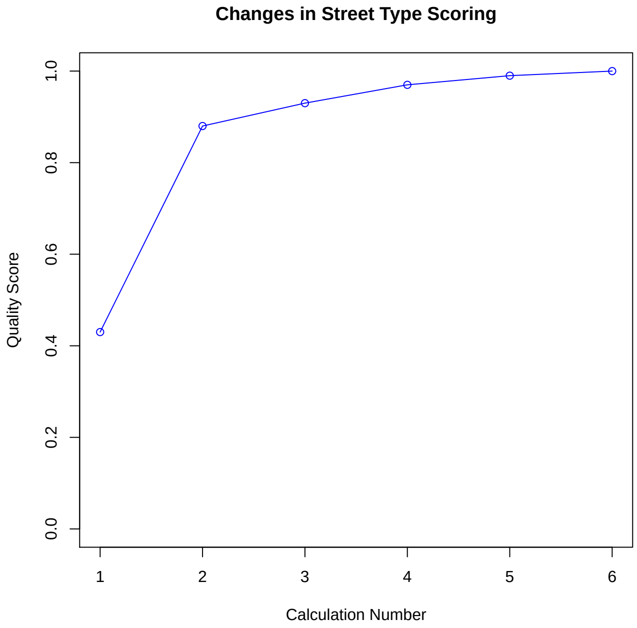


FIGURE 5.9: Changes in Street Type Scoring

Figure 5.10 shows the progress of the overall address score during the lifetime of the query. There are several definite spikes, and the causes of these are shown in Table 5.15 where scores and their contributing scores are presented. A graphical representation of the message sending between agents which is the basis for increases in scores as seen in Figure 5.11; this reinforces the idea that every increment described in Table 5.15 is performed via messaging.

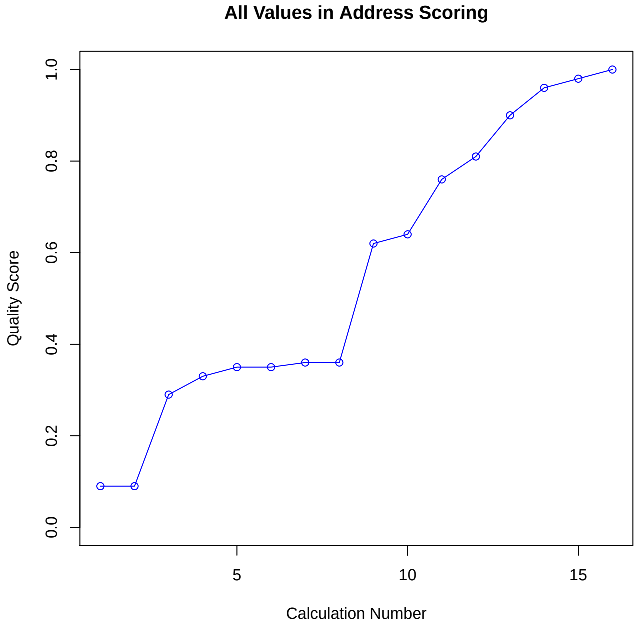


FIGURE 5.10: All Values in Address Scoring

During processing, the calculation of element scores and the sending of these scores to other agents is performed by all the agents concurrently.

The prototype is able to correct address elements during processing, and can actually branch off into many simultaneous possibilities (discussed in Section A.2). Reflecting this in the example, Table 5.11 shows what happens when the value of the locality is changed to the correct value of “Subiaco” (this is achieved from a geographic proximity correction). The implications of this is that the locality spatially agrees with the street name, which increases the locality score slightly (by 0.03), and this in turn raises the score of the state (by 0.02). The status score of street name is still quite low, but the real changes occur *after* this. The increase of score for street name and its interaction with locality is a major cause for subsequent increases.

TABLE 5.11: An Increase in Several Elements

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.90	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.74
locality	subiaco	0.74	Y	Y	Y	N	1	2	3	0.88	2	0.14
st name	gloster	0.43	Y	N	N	N	1	2	3	0.71	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

The street name agrees with both its upper and lower complementary elements, and the results of this can be seen in an increase of its score, which is shown in Table 5.12. The presentation of Table 5.12 is similar to the beliefset structure used in the agents, and is intended as a reminder for the reader as what occurred during processing (updating of fields and these acting as a catalyst for increase in other elements).

TABLE 5.12: An Increase in Street Name Score

T	V	S	P	E	AU	AL	WP	WE	WU	SU	WL	SL
state	WA	0.90	Y	Y	*n/a*	Y	1	2	*n/a*	*n/a*	2	0.74
locality	subiaco	0.74	Y	Y	Y	N	1	2	3	0.88	2	0.14
st name	gloster	0.79	Y	Y	Y	Y	1	2	3	0.71	1	0.43
st type	st	0.43	Y	Y	N	n/a	1	2	4	0.14	n/a	n/a

The remaining changes in score can be seen in Table 5.13; this table shows that eventually the address elements each reach a score of 1.0 because the address is fully corrected and each element spatially agrees with the others.

TABLE 5.13: Various Element Scores as Processing Continues

Type	Value	Score	Score	Score	Score	Score
State	WA	0.90	0.90	0.96	0.99	1.0
Locality	Subiaco	0.74	0.91	0.97	0.98	1.0
Street Name	Gloster	0.79	0.88	0.95	0.98	1.0
Strert Type	St	0.88	0.93	0.97	0.99	1.0

In addition to the status score (and subsequent indicator of quality) of individual elements, also included is the status score of the overall address. During processing (as the overall score is incrementing) because the variable *m* relies on elements spatially agreeing, the overall score for the address increases when an address element reaches the status of existing and also spatial agreement. The overall scores for the address in the example can be seen in Table 5.14, the values 1 to 16 indicate each of the changes in score of the overall address (i.e. the address score was calculated 16 times). The values in Table 5.14 are the same values used in Figure 5.10.

TABLE 5.14: Overall Address Scores

	Score		Score
1	0.09	9	0.62
2	0.09	10	0.64
3	0.29	11	0.76
4	0.33	12	0.81
5	0.35	13	0.90
6	0.35	14	0.96
7	0.36	15	0.98
8	0.36	16	1.00

The increases in overall score seen in Table 5.14 can be attributed to several factors, such as elements existing, reaching spatial agreement and individual elements increasing in their scores. These overall score increases and factors can be seen in Table 5.15.

TABLE 5.15: Factors in Overall Score Changes

Transition	Factors
0.09 to 0.29	Locality exists Locality agrees with upper State agrees with lower
0.36 to 0.62	Street name exists
0.64 to 0.76	Street name agrees with upper Street name agrees with lower
0.81 to 0.90	Street type agrees with upper

The weightings used in determining the overall score can be seen in Table 5.16.

TABLE 5.16: Weightings used for Overall Score

Element	Weighting
State	2
Postcode	4
Locality	3
Street	3
Street Type	1
House number	2
Unit Number	1

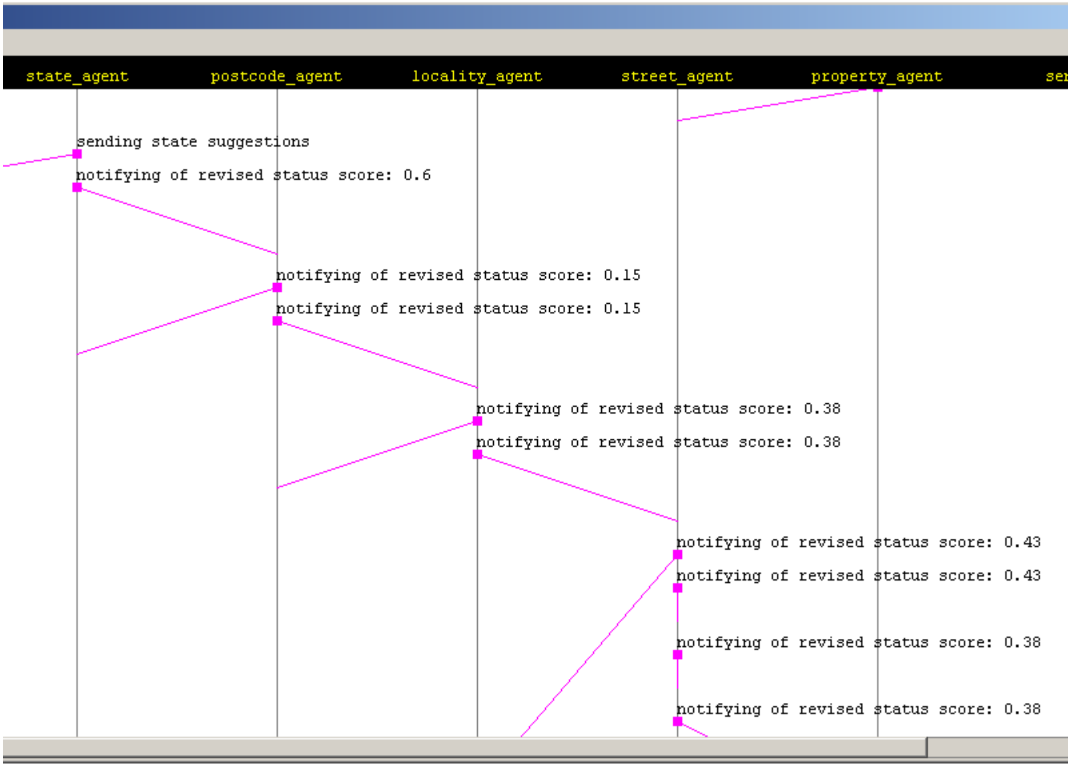
Address scoring as shown in this section provide a measure of quality, including both the score of individual elements and the overall address. Both scoring techniques recognize that although an element may not exist now, it may do in the future. This feature has relevance especially to the levels of completeness category of test addresses, for example where (i) an element does not exist at all, (ii) it is present in the address but does not exist, and (iii) where two elements both exist but do not agree with each other. In these cases, the algorithm for element suggestions and address reconstruction worked well as often these missing elements could be “filled in”. By incorporating an element’s complementary elements in the calculation right from the start, it highlights the importance of not just considering an element in isolation. By implementing these formulae in beliefsets, the values can be in constant flux and the calculation be updated dynamically; it can both increase and decrease depending on what information comes into the system, which keeps in tune with the concept of an agent being in a dynamic environment.

The weighting of each address element adds a level of flexibility which is also useful for providing a quality score which includes some context. However the implementation stores the weightings in instantiated beliefsets which means that at run time they can be modified without problem; the implications of this is that the weightings can be adjusted to suit the context of the user or applications that have put forward the query. Users could also set weightings according to how confident they are about the query they have submitted, or in the case of an administrator their knowledge of a particular geographic area and the mistakes people commonly make. Examples of this weighting being important are when two or more elements may both exist, but may not spatially agree with each other; the weighting in this case would see the one with the higher rating being used. Another example is related to the iterative functionality of the system, and the situation where several suggestions may be available. By having weightings, these suggestions may in some cases be ranked.

Because the formula for overall address score takes into account the resolution of the address, it means a result address with less elements is necessarily given a higher score than another suggestion with more elements; this has relevance for the test addresses which had elements missing. Calculating the quality rating was straightforward with beliefsets, not just for the persistence, but because they are so tightly integrated with the whole agent functionality; instead of having to worry about data and process as separate considerations, beliefsets blend in with the whole paradigm. Measuring the quality of a query is an ongoing result throughout the whole geocoding process, as it occurs when a query first enters the system, during the correction stages, and also at the end of processing when an address has its quality rating calculated. This real time processing has been an important part of the framework, and it is the internal catalyst for the events which occur inside the system.

Context was used successfully to modify execution based on address element types, the quality rating, and the factors used to calculate the quality rating. This shows that geocoding is capable of using plans and the sort of "polymorphic event handling" that they bring. The close relationship can be seen between context and control, where context is essentially making a flow of control choice based on the given value. Context can be seen in the dynamic calculation of the quality score. Keeping in mind that in the formula, the element types each have their own weighting, and so even though the same *number* of elements could be present in different cases, the *type* will determine they have different scores; these weightings were shown in Figure 5.10.

Figure 5.11 shows a messaging diagram created during processing of quality score updates; it is a graphical depiction of the updates and subsequent beliefset modifications shown in tables previously throughout this section. Each agent type is labeled across the top, and each vertical line represents an agent. The sloped, interconnecting lines represent the messages sent between agents. The small block on each line designates where the message was sent from, and the other end of the line terminates at the agent the message was sent to. It can be seen that typically an agent will receive an update from another agent, after which it recalculates its status and send two new messages to its complementary elements (agents).



```
sending state suggestions
notifying of revised status score: 0.6
notifying of revised status score: 0.15
notifying of revised status score: 0.15
notifying of revised status score: 0.38
notifying of revised status score: 0.38
notifying of revised status score: 0.43
notifying of revised status score: 0.43
notifying of revised status score: 0.38
notifying of revised status score: 0.38
```

FIGURE 5.11: Agents Sending Messages with Updates Scores

There is a relationship between the quality score of the address and the geocode determined, such that it is possible to have a high quality score, but not be able to geocode the address. It depends on the reference data, because as is the case with IntelliGeoLocator, non-spatial textual reference data (i.e. lists of localities, streets etc.) are used to perform the address correction and quality assessment. It is after this that coordinates are found for the address. So if the textual reference dataset (e.g. locality and street names) used for address correction is different from the coordinate/spatial dataset then this “verified but unlocated” situation could arise. For IntelliGeoLocator, this would only happen if the street name or locality was not in the spatial reference dataset, as the non-spatial reference data does not include street numbers and so there would be no conflict between reference datasets.

To reiterate, the important point is that the score calculated by IGL is intended to be a measure of confidence that the match is what the user intended, and also how valid the address is as defined by its existence in the physical world. If the quality score was intended to provide a detailed assessment of positional accuracy, the IGL would have had to interrogate the metadata of the reference data, and this was not done. This would be an excellent addition to any geocoder though, effectively distinguishing the *granularity* of the geocode (e.g. locality or street level versus house level) from its *positional accuracy* (e.g. 5m, 10m or 50m). This would require additional research and work.

The testing of IGL has also raised issues regarding what the geocode represents, especially in cases where an arbitrary point is chosen to represent a line or area. This is an issue of context, but could also be considered an issue of quality depending on what the user is expecting to be presented or what they were going to use the result for. Depending on user context, different types of users may have different expectations and these could be defined in the "user" agent which assigns "contexts" relating to the type of user or the type of application, eg. student, emergency management, finding states and countries. For example, for a point representing a state (e.g. Western Australia), would the user prefer the centroid (in the desert) or state capital (Perth)? When testing IGL using the LISAsoft web service for coordinates (which in turn uses the G-NAF dataset from PSMA), results for states, localities and post codes were all centroids. For street level results it was the mid-point of the road which was used. In the case of having multiple results, each with possibly the same quality score, it would be good if there was a way to choose the one "most resembling the original query" but this can be a hard concept to design because the question arises of whether this is based on syntax (e.g. both have 80% the same characters in the their strings), physical proximity (one choice is 5km closer to the query value), semantic concept ("both the query and one option are named after birds" - e.g. Falcon Ave. and Pelican Ct. in Churchlands, Western Australia) etc.; this would have to be the focus of further research. This concept would have most impact in situations where several candidate addresses are tied. With all the processing and messaging that occurs in calculating a quality score, the questions arises of how this load is distributed amongst the various elements, and causes behind any variation in how the processing occurs.

5.1.2 Activity Load for the Address Elements

Figure 5.12 shows the number of increments needed for each address element to reach a final score of 1.0. The address used for Figure 5.12 is "perfect", in that all the elements exist and agree. This address was used specifically to see how messaging activity (i.e. load) varied for each element type. These increments in the address elements are almost synonymous with messages between agents. The only difference is that one agent (the street agent) is responsible for two address elements, specifically street name and street type. So when there is an update is required between these, there is no message required, as it is only an internal event used. But the increment happens nonetheless. That is why, for the purpose of analysing the algorithm, Figure 5.12 shows *increments*.

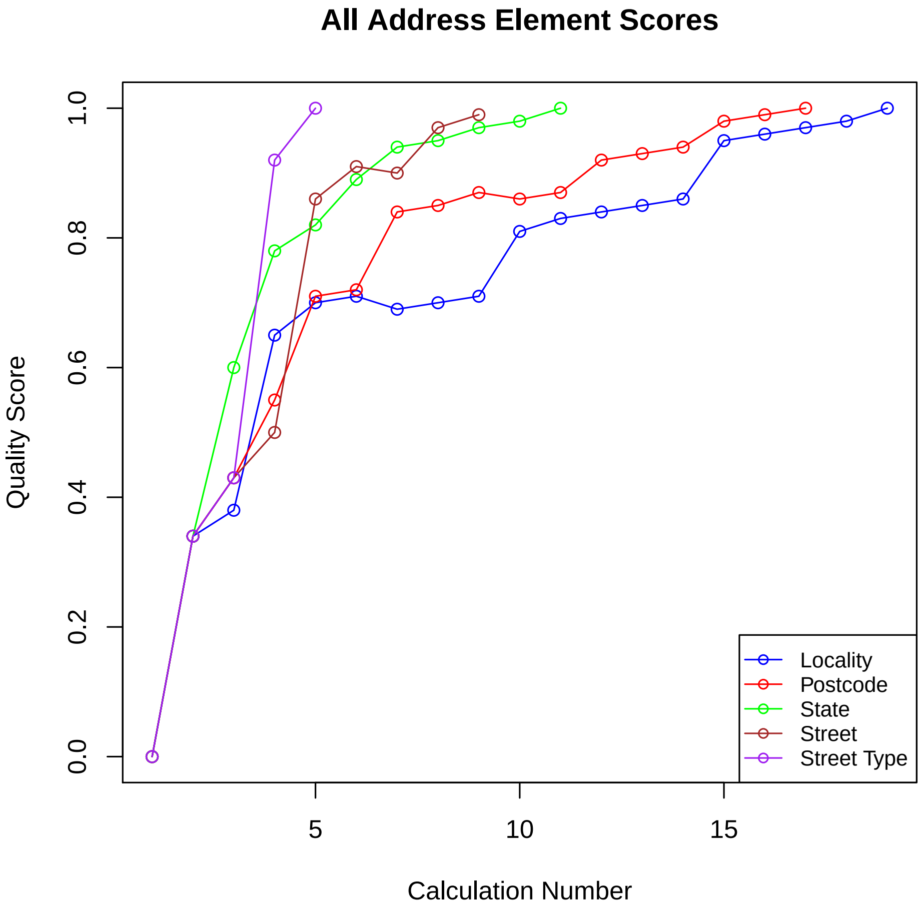


FIGURE 5.12: Each Address Element with all Score Increments

One significant observation is that there is more activity for the postcode and locality elements. In this example, state has 11 increments, postcode has 17, locality has 19, street has 9, and street type has five increments. There are two possible reasons for this, including:

- The number of other address elements that provide updates to a given element. For example, state and street type only have one source (i.e. one complementary element) of increments; conversely they also only send updates to one address element.
- Because updates for a given element come only from the lower and upper complementary elements, the number of increments that *they* have will affect the number of the given element. For example, because the postcode element has great activity, this feeds the activity of the locality element.

So overall, locality has slightly more (two) more increments than postcode because although both elements receive updates from two other address elements, and have an increment "frenzy" from updating each other, it is the fact that the locality element has a lower complementary element (street) which itself has two sources of updates. Postcode however has the complementary element of state; this difference means locality gets a few extra increments from the street element.

The longer it takes for an element to reach its final score, the more involved it is in the overall process; this is shown by postcode and locality. The difference in the number of increments for each element highlights how the algorithm (and in particular the ordering and time used for processing) is non-deterministic and address-driven. Not only are postcode and locality active in the address processing, but they have the largest weightings associated with them. The number of interactions (i.e. time) is a function of the number of other elements it relies on and the weightings of all elements. Because agent messaging is used for the scoring, adding additional factors in the future would be possible without having to re-architect the design because the factors could be agents and more messages could be used along with weightings.

5.1.3 Initial Real-Time Correction of Elements

Initially in the processing of an address, the prototype builds up its geocoding quality for each element as they arrive from the `SenderAgent` (where they were read from file), and then from there the concept of distributed processing is shown to work as the various agent types begin evaluating and processing their elements in parallel. This processing starts with the state level and progresses to the property as the data comes in - it happens very quickly and not every level in between is necessarily shown if the next piece of data comes in quick enough. For example, the various levels for the address "32 Marlow St, Wembley 6014 WA" are shown in Figures 5.13, 5.14, 5.15 and 5.16.

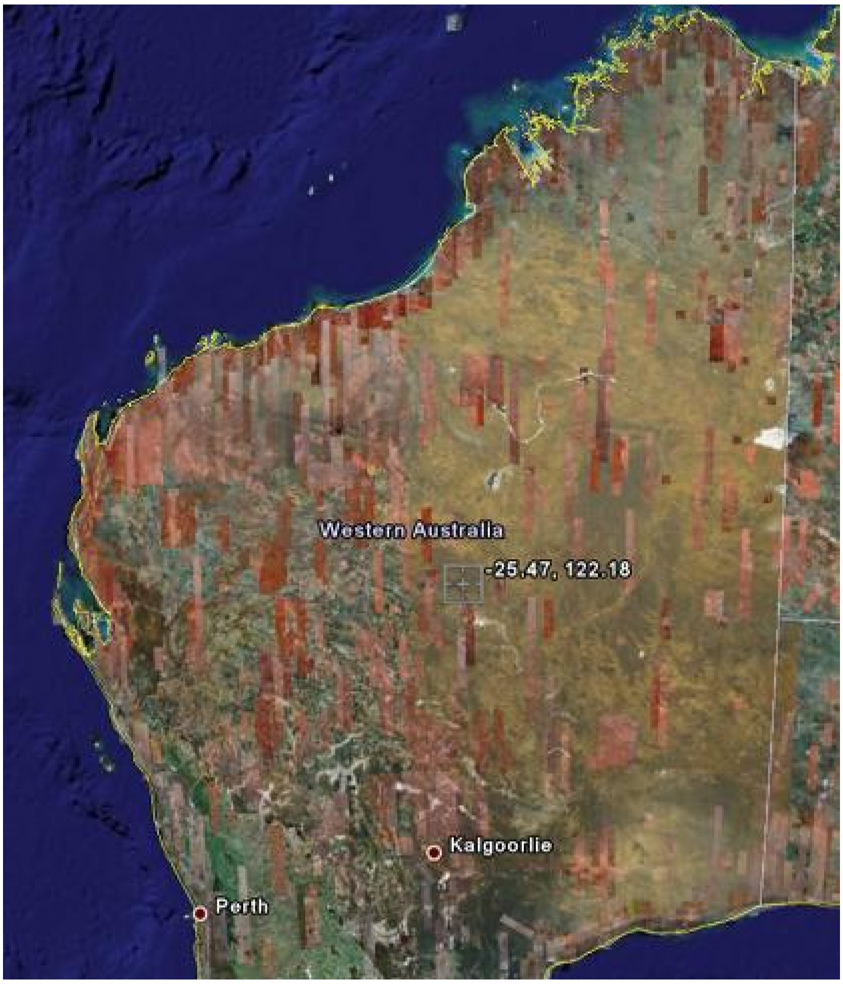


FIGURE 5.13: Geocode at the state centroid level

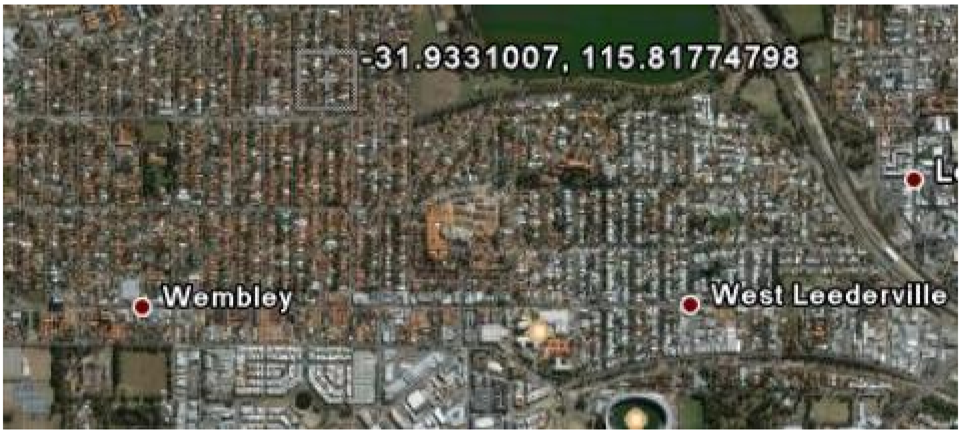


FIGURE 5.14: Geocode at the locality centroid level

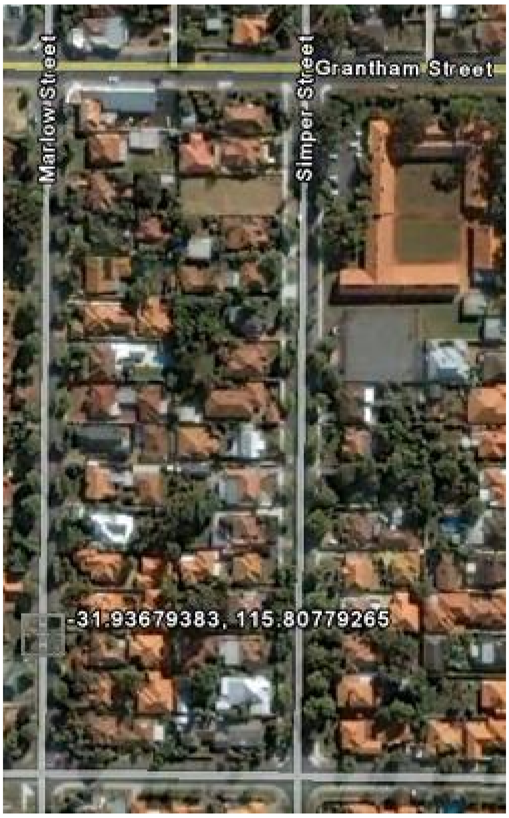


FIGURE 5.15: Geocode at the street centroid level

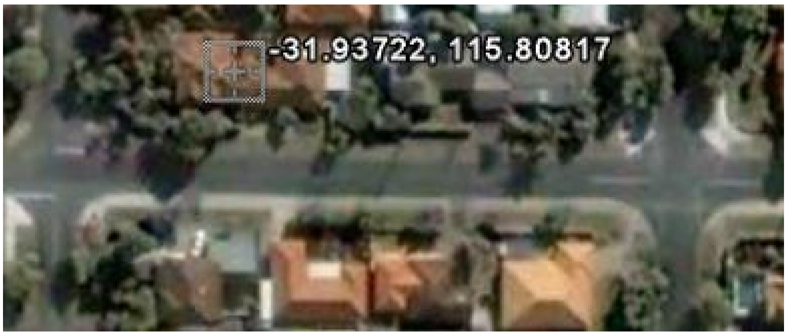


FIGURE 5.16: Geocode at the property level

The example in Table 5.17 shows the raw processing is an address which is already correct; if an address is submitted in which one or more of the elements do not exist, then the agent still shows raw results for those elements that do exist. The idea here is that while the user is waiting for the correction process to correct the elements with errors, at least they can get *some* preliminary geocode information; this is only in the case where a real user sits at the front end, this is less critical for batch jobs or where applications are at the front end. An example of an address entering the system and having its initial element values be geocoded can be seen in Table 5.17.

TABLE 5.17: Geocodes for the Raw Elements Entering the System

Geocode	Address String
-25.47, 122.18	(wa)
-25.47, 122.18	(6014 wa)
-31.9331007, 115.81774798	(wembley 6014 wa)
-31.93679383, 115.80779265	(marlow wembley 6014 wa)
-31.93679383, 115.80779265	(marlow st wembley 6014 wa)
-31.93722, 115.80817	(32 marlow st wembley 6014 wa)

5.1.4 Parallel and Distributed Processing

Figure 5.11 is a good representation of the messages that are sent between agents; the order of these messages is not determined at design time. As it was used in the prototype, parallel processing allowed the `UserAgent` to distribute the initial geocoding query in parallel. The most significant result was that the tasks of processing each element type could be done simultaneously, and intermediate results could be sent between agents. Because a different agent was responsible for each type of address element, the situation allowed for the creation of one "generic" agent (`ElementAgent`) which was then given the persona of a particular type (e.g. `LocalityAgent` and `StreetAgent`). Then in parallel, each agent with its corresponding persona could access its own algorithms, local data, message sending or web services as it needed. On a single CPU system, this parallelism is "pseudo-parallelism" as the agents operate on different threads and very rapidly switch between these. When the agents run on different machines this is very much "true" parallel and distributed processing. When running on a multi-core CPU, properties can be set in the agent runtime environment to have agents run in the different cores. As user demand increases (i.e. server load grows) more machines can be added to scale accordingly.

The prototype handles many simultaneous queries coming into the geocoder, and this was no harder than designing the system to handle just one address, as the event driven paradigm forces a synchronized and concurrent system design from the start. Because of the tracking system described in Appendix A.2, the unique IDs allow any number of simultaneous geocoder queries to be handled. It should be noted that for every additional query coming into the system, no additional agents are being created. The same six agents (`UserAgent`, `MatchingAgent`, `StateAgent`, `PostcodeAgent`, `StreetAgent` and `PropertyAgent`) are instantiated the whole time, and remain instantiated between queries so there is no overhead in creating new agents. Memory is used to process events, send messages, and add beliefs to beliefsets (and other operations), but it seems logical that this processing would scale as additional agents of different types could be added. For example, there is no reason there could not be five locality agents (or five of *every* agent) and when new geocode queries enter the system they could be assigned appropriately (load balanced) to an agent. The parallel processing also ties in with the processing of the initial "raw" inputs, as both provide a way to return results to the user while they wait. Because the quality scores (both for individual elements and overall) are calculated on an ongoing basis, these results are displayed for the user along the way.

5.2 Evaluation of Intelligent Geocoding Process

Testing indicated that overall performance of IntelliGeoLocator was generally on par with other geocoders, although all geocoders had types of addresses they performed poorly on. This assessment of performance is based on a range of address types used for testing. The following sections identify the range of address types and provide a representative example of an address type to illustrate the nature of the performance.

5.2.1 Levels of Completeness in Addresses

The levels of completeness were compiled using results from four geocoders (see Section 3.2.2 regarding geocoder selection), including Google Maps, Multimap, Whereis and IntelliGeoLocator. This is tabulated for a range of address types and representative addresses are shown in Table 5.18 where the vendors are abbreviated as *G*, *M*, *W* and *I* respectively. The addresses provided in the tables are *examples* of such address types and are *representative* of that address type; these are just some of the many addresses tested. The test addresses used in this chapter were either provided from other organisations familiar with geocoding, from personal experience, or were examples picked randomly from the phone book (with the only criteria being it was an example of the particular address type). The test addresses used to illustrate the results here are representative of their types. Addresses in this category are missing at least one address element, or some derivation of this. Table 5.18 and the other tables presenting address testing in subsequent sections use the result abbreviations of "P", "Yes" and "No", note that "P" indicates "partial".

TABLE 5.18: Test Table for Levels of Completeness

Address Example	G	M	W	I
1 - Complete Address				
197 Holbeck St Doubleview WA 6018	Yes	Yes	Yes	Yes
13 Paris Rd Australind 6233	Yes	Yes	Yes	Yes
163 Alfred Rd Mt Claremont 6010	Yes	Yes	Yes	Yes
2 - Complete Address (Duplex)				
58a Cresswell Rd Dianella 6059	Yes	No	No	Yes

Address Example	G	M	W	I
224b Marmion St Palmyra 6157	Yes	P	No	Yes
262b Selby St Wembley 6014	Yes	P	No	Yes
3 - Complete address with unit number				
4 / 133 Bourke St Leederville WA 6007	P	No	No	P
6/14 Alora Drive Port Kennedy 6172	P	P	No	P
44/70 Marlboro Road Swan View 6056	P	P	No	P
4 - No Postcode				
103 Wellington Rd Dianella WA	Yes	Yes	Yes	Yes
39 Warringah Cl Kallaroo WA	Yes	Yes	Yes	Yes
4 Comer St Como WA	Yes	Yes	Yes	Yes
5 - No state or postcode				
119 Stirling Hwy Nedlands	Yes	No	P	Yes
169 Delgado Pde Iluka	Yes	Yes	Yes*	Yes
31 Backhouse Rd Kingsley	Yes	Yes	Yes*	Yes
*6 - No state, postcode or locality **				
40 Pennant St (North Perth, 6006)	P	P	No	Yes
157 Acton Ave (157 Acton Ave, Rivervale 6103)	Yes	Yes	No	Yes
70 Roberts St (Bayswater)	P	P	N	P
*7 - House number and street name only **				
10 Forrest	P	No	No	P
31 Fordham	P	No	No	P
177 Ocean Keys	No	No*	No	No
*8 - Street name and type only **				
Parnell Ave	P	P	No	P
Murray Bend Rd (Murray Bend Rd, Ravenswood)	Yes	No	No	No
Anaconda Drv (Gosnells)	Yes	Yes*	No	Yes
9 - Street name only				
Hampstead	P	P	No	P
Sir Charles Court	Yes	No	No	No
Woodbridge (Woodbridge Drv, Greenmount)	No	P	No	Yes*
*10 - State name only **				
WA	Yes	Yes	No	Yes
NSW	Yes	Yes*	No	No
NT	Yes	Yes	No	No
*11 - Postcode only **				
6021	Yes	Yes	No	No

Address Example	G	M	W	I
6001 (post office box, non-geographic)	No	No	No	No
12 - Locality name only *				
Malaga	Yes	Yes	Yes	Yes
Tamala Park	Yes	Yes	Yes	No
South Lake	Yes	Yes	Yes	Yes
13 - Complete, Semi-Rural Address				
Lot 310 Arborfield Way Bullsbrook 6084 WA	No	P	P	P
Lot 2 Ranford Rd Armadale WA 6112	P	P	No	P
Lot 1 South Western Hwy Mundijong WA 6123	P	No	No	P
33 - Lot numbers				
1 Cockburn Rd, South Fremantle is also Lot 50 Cockburn Rd, South Fremantle	Yes	No	No	No
Lot 36 is also 425 Victor Rd Darlington WA 6070	No	No	No	No
38 - Use of Estate Names				
Brighton Estate (replacing Butler)	No	No	No	No
Floreat Waters (replacing Churchlands)	No	No	No	No
Landsdale North (replacing Landsdale)	No	No	No	No
- New Localities				
Dardadine, WA	No*	No	Yes	No
Meeking, WA	No*	No	No*	No
- Alias Examples				
Old Yorkrakine Road, Tammin (old)	No	No*	Yes	No*
Russell street, Tammin (new, renamed)	No	No*	No*	No*
Bournville St, Wembley WA (old)	Yes	No	Yes	Yes*
Ruislip St, Wembley WA (new, renamed)	Yes*	Yes	No*	Yes
Hockley Street, Mount Barker (old)	No	No*	Yes	No*
Hockley View, Mount Barker WA (new, renamed)	No	No*	No*	No

In Table 5.18 there are asterisks which appear for address types and for the results of individual addresses. For the address types, the asterisk indicates that for IGL, data was restricted to the state of Western Australia (WA). This means that the search space of the geocoder was restricted to WA. This was taken into account, given that other geocoders would search all states and could therefore have more potential matches. For the specific, representative test addresses an asterisk was used to indicate that the result is what is stated (e.g. “Yes”, ”No”) but there was some caveat; in other words the asterisk detracts (in the case of a "Yes") or enhances (in the case of a "No") from the stated result. For example, where a geocoder found the correct result but the result was one of several suggestions (as would be the case when searching other states in addition to WA) the geocoder was given a "Yes" but with an asterisk. This allowed geocoders to receive an evaluation consistent with their performance. The use of these asterisks is used in a similar way for the address types in Sections 5.2.2, 5.2.3, 5.2.4 and 5.2.5. In some cases in Table 5.18, some addresses were geocoded despite a positive result not being expected. For example, concerning the *for state name only* category, IGL did not have reference data to geocode "NSW" or "NT". Similarly, it was not expected that any geocoder would geocode a postcode for a post office box but it was tested nonetheless to ensure this.

An advantage of the approach used by IGL for this overall category of address is how it handles the dilemma with ambiguous results and the balance between the desire to narrow and prioritize but not eliminate if there is any chance the suggestion could be the intended result. IGL uses complementary address elements to narrow the search space which increases processing speed and narrows the result set of possible geocodes. IGL also does also not mandate that certain element values *have* to be present. For example, even if a critical element type such as locality is not present, the equivalent values based on the elements that are present are used. Even if there are multiple possibilities they are all explored.

Figure 5.17 shows an overview of address evaluation for this category. Even though Google shows a higher hit rate than IGL, there are reasons, including that IGL was not able to separate the unit numbers and slashes from the house numbers (natural language processing) which Google can do, and also that because a web service was used by IGL for locating coordinates it (IGL) was not privy to how the web service manipulated/processed reference data in regard to some of the unusual street names. Future versions of IGL could use multiple web services for locating, and use consensus from these services to provide increased confidence. The ratio of "yes" (successful) compared to "partial" geocode results should also be noted, specifically because there is a relatively large amount of partial results. This was largely due to the geocoders returning multiple results based on the ambiguity of the query. In other words there were multiple results each of which were equally likely. This indicates that there could be potential in the future for (i) asking the user additional questions to refine the query, and (ii) possibly storing the most common choices of the candidates dynamically in the knowledge base. The last option though would still need to cater though for the fact that the more uncommon of the candidates *could* be the actual result; more research would be needed to explore this. This type of knowledge could be stored as rules (which the knowledge base used in the prototype caters for) at a higher level of knowledge than the facts stored in alias tables.

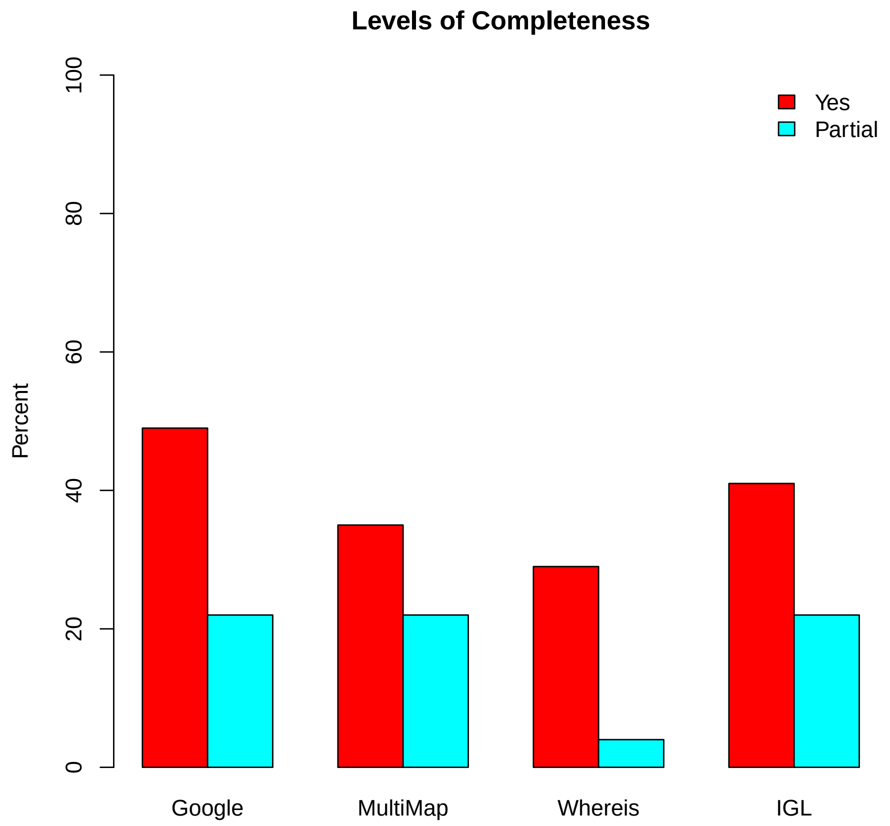


FIGURE 5.17: Results for Levels of Completeness Category

The approach of IntelliGeoLocator to matching is that even if the input query is incorrect, the user had a particular address in mind and that there is a finite number of reasons why they made a mistake (or that the address they had in mind defied convention), and that a solution can be found. The concept generalizes to demonstrate that basic assumptions and processing of strings in addresses are still subject to error and the expected spacing and words typically associated with particular element types do not always correspond to what is expected. With computing power continuing to increase, it can also be asked why a brute force approach is not used more frequently; for geocoding this means trying different combinations of address elements together, simultaneously.

In the example of “177 Ocean Keys” why not try every possibility? For example, the following permutations could exist: (i) “Ocean Keys” could be the street name, (ii) “Ocean” could be the street name, (iii) “Keys” could be the street type, (iv) “Keys” could be the locality, (v) “177” could be the street number, or (vi) ”177” could be part of a postcode. Although IGL (the prototype) has the meaning of its elements pre-established, it has the ability already for (i) pursuing multiple possible address possibilities using a tree mechanism, and (ii) iterate through generations of address possibilities as potential matches “evolve”.

An example of the postcode concept is “Doveton, Victoria” which has the postcode “3177”, where it is conceivable a user could mistakenly not type the “3”. Effectively each of the element possibilities forms another result suggestion. This is the approach taken with IntelliGeoLocator, as explained in Section A.2. The key here is that a “match” could be found in multiple components of an address (i.e. street, locality) but that the first one found (in a linear algorithm) is not necessarily the correct one. IG would attempt all options in parallel.

Given the current state of geocoding with regards to queries which have multiple geocodes it seems that the use of interaction between the geocoder and the user will continue to be important - for example asking additional questions to narrow down lists of suggestions. These questions would be in terms of things that users can relate to, such landmarks, events or possibly distance and direction; these would require semantics and geographic context beyond what is present in the prototype but which IGL can provide with further expansion.

Agents are ideal for this social, event-based and dynamic behaviour. IGL obtains geocode queries from a source (eg. file, user, etc.) and processing begins there. The intelligent geocoding framework has been designed so that as long as the geocode query is still “open” (which would be until the user closes their browser) then new information could be included into an existing query. This is because all the information used in processing the query is stored in beliefsets and these beliefsets can be modified at any time. Because the framework includes the use of automatic events (i.e. posted from beliefsets), the updated information is perpetuated through the system and processing is updated, the geocode result would change accordingly.

5.2.2 Syntax of Addresses

Addresses in this category are those which have an error regarding the spelling or arrangement of the characters and tokens within the address string. These types of errors are "superficial" in the sense there is usually no underlying, deeper cause. The address types and representative addresses are shown in Table 5.19.

TABLE 5.19: Test Table for Syntax

Address Example	G	M	W	I
14 - Similarity to commonly known words				
Shiraz St Greenmount WA (should be Chiraz St)	No	No	Yes	No
Glide St East Fremantle WA (should be Glyde St)	No	No*	Yes*	Yes
Queue Ct Swan View WA (Cue St)	No	P	No	Yes
Navy Base (Naval Base)	No	Yes	Yes	Yes
15 - Linguistic (written)				
Wanneru, WA (should be Wanneroo)	Yes	Yes	P	Yes
Nangarra, WA (should be Gnangara)	Yes	No	No	No
Lewin, WA (should be Leeuwin - locality)	No	No	Yes	Yes*
16 - Linguistic (verbal)				
Mushay (instead of Muchea)	No	No	Yes	P
Coeburn (instead of Cockburn)	No	No*	No*	No*
Gardner (instead of Gairdner)	No	Yes*	Yes*	Yes*
17 - User makes mistake typing street				
Waratag Ave, Dalkeith 6009	Yes	P	P	P
39 Heskath Ave Seville Grove 6112 (should be Hesketh)	Yes	No	Yes	Yes
19 Birtonia Way Forrestfield 6058 (should be Burtonia)	No	Yes*	Yes	Yes
18 - User makes mistake typing street (no postcode)				

Address Example	G	M	W	I
Waratag Ave, Dalkeith	Yes	P	P	P
39 Heskath Ave Seville Grove	Yes	No	Yes	Yes
19 Birtonia Way Forrestfield	No	Yes	Yes	Yes
19 - User makes mistake typing locality (with postcode)				
Shoveler Tce, Joondaluo 6027 (should be Joondalup)	Yes	No	Yes	Yes
Lakor St Scarborough 6019 (should be Lalor)	No	Yes	No	Yes
Binacle Rd Ocean Reef 6027 (should be Binnacle)	Yes	Yes	Yes	Yes
20 - User makes mistake typing locality (no postcode)				
Stock Rd Attadalw (should be Attadale)	Yes	No	Yes	Yes
Littlefield Rd High Wycome (should be Wycombe)	Yes	Yes	Yes	Yes
Stephen St Abany (should be Milpara, near Albany)	No	No*	No*	No*
21 - Incorrect postcode (correct locality and street)				
Aberdeen St, Northbridge 6014	Yes	Yes	Yes	Yes*
Irian Gr Riverton 6108 (should be 6148)	Yes	Yes	Yes	Yes
High St South Perth 6051 (should be 6151)	Yes	Yes	Yes	Yes
22 - Conflicting locality and postcode				
Floreat 6015	1st	1st	1st	Both
Victoria Park 6107	1st	1st	1st	Both
Rockingham 6172	1st	P	1st	Both
34 - Abbreviation of locality name				
8 Marmion St East Freo (East Fremantle)	No	Yes	Yes	Yes
11 Leonard St Vic Park (Victoria Park)	No	Yes	Yes	No
144 Raleigh St Carlisle S (Carlisle South)*	Yes	No	Yes	No
35 - Long form				
Saint Michael Terrace, Mount Pleasant WA	Yes	Yes	Yes	P
Mount Claremont, WA (correct)	Yes	Yes	Yes	Yes
Mt Claremont, WA (incorrect)	Yes	Yes	Yes	Yes
Mount Henry Road, WA (correct)	Yes	Yes	No	No
Mt Henry Rd, WA (incorrect)	Yes	No	No	No
Mounthaven Street, Kalamunda (correct)	Yes	Yes	Yes	Yes
Mt Haven Street, Kalamunda (incorrect)	No	No*	Yes	Yes

Address Example	G	M	W	I
36 - Freeway				
Mitchell Fwy, Perth WA	Yes	No	Yes	Yes
South West Hwy WA (South Western Hwy)	No	No	No	No
Australind Bypass	Yes	No	No	Yes*
37 - Hyphenated names				
Pinjarra-Williams Rd, WA	Yes	Yes	No	Yes
Wandering-Narrogin Rd, Cuballing	Yes	Yes	No	P
Ongerup-Pingrup Rd, Pingrup	Yes	Yes	No	P
Baronhay Court, Kensington (should be Baron-Hay)	Yes	Yes	Yes	Yes
Belair Place, Connolly (should be Bel-Air)	Yes	No*	Yes	Yes
Gin-Gin, WA (should be Gingin)	No	No	Yes	Yes
40 - Punctuation Missing				
Oconnor WA	No	No	Yes	Yes
Odea Gate, Canning Vale (should be O'Dea)	Yes	No	Yes	Yes
Adale Wy, Dalyellup (should be A'Dale)	Yes	No	Yes	P
Break Oday Dr, Australind (should be O'Day)	Yes	Yes	Yes	Yes
Allsaints Way, Churchlands (should be All Saints)	Yes	No	Yes	Yes
Backbeach Rd, Onslow (should be Back Beach)	Yes	No	Yes	Yes
Lagrange Rd, Stoneville (should be La Grange)	Yes	No	Yes	Yes
- Full Stop				
C.W.A. Avenue, Useless Loop	Yes	No	No	No*
S.E.C. Road, Rosa Brook	No	No	No	No*
Old Haul Road No. 1, Karrakup	Yes	No	No	No
- Ampersand				
Cobb & Co Road, West Pinjarra	Yes	No	No	Yes
41 - Punctuation not wanted				
Smith's Beach Rd, Yallingup WA (Smiths Beach)	No	Yes	P	Yes
View Way, Swan View (should be Viewway)	No	No	Yes*	No
Kings Way, Nedlands (should be Kingsway)	No	No	Yes*	No
42 - Number in street name				
Australia II Drv, Crawley WA	Yes	No	Yes	No
Australia 2 Drv, Crawley WA	No	No	Yes	No
John XXIII Ave, Mt Claremont WA	Yes	Yes	Yes	No
John 23 Ave, Mt Claremont WA	Yes	No	Yes*	No
44 - Directional Reversed				

Address Example	G	M	W	I
Perth North, WA (North Perth, WA)	Yes	No*	Yes	No
Swan Middle, WA (Middle Swan, WA)	Yes	Yes	Yes*	P
South Carlisle, WA (South Carlisle, WA)	Yes	No*	No*	P
4 - Streets with "The" in them				
The Cove, Yallingup WA	Yes	No*	Yes*	Yes
The Boulevard, Floreat WA	Yes	No	Yes	Yes
The Summit, Yangebup WA	Yes	No	Yes	Yes
- Road name could be the road type				
The Avenue, Nedlands WA (correct)	Yes	No	Yes	Yes
Avenue, Nedlands WA (incorrect)	No	No	Yes	Yes
The Promenade, Mt Pleasant WA	Yes	No*	Yes	Yes
Promenade, Mt Pleasant WA (incorrect)	Yes	No*	Yes	Yes
The Crescent, Helena Valley WA (correct)	Yes	No*	Yes	Yes
Crescent, Helena Valley WA (incorrect)	Yes	No*	Yes	Yes
- Ranged addresses				
102 - 114 Normanby Rd, Inglewood	Yes*	Yes*	No	No*
19E - 23E Johnston St E, Boulder	No	No*	No	No*
Unit 35, 227 - 237 North Rd, Centennial Park	P	No	No	No*

In Table 5.19 the presence of an asterisk (*) means there was some caveat, or in more general terms, the geocoder was "given the benefit of the doubt" or performed such that it belonged more in one category than another. For example, with the sample address of "Glide St East Fremantle WA", Whereis provided just two suggestions, but the first suggestion was the correct geocode; Multimap returned the geocode of "Fremantle, WA". So for the latter, it did not meet the requirement but was not a complete failure to geocode. For some address types, such as the *ranged addresses*, IGL did not provide a result because the functionality had not been implemented.

There were relatively few "partial" results in Figure 5.18, because mostly the geocoders either geocoded the query correctly or not at all.

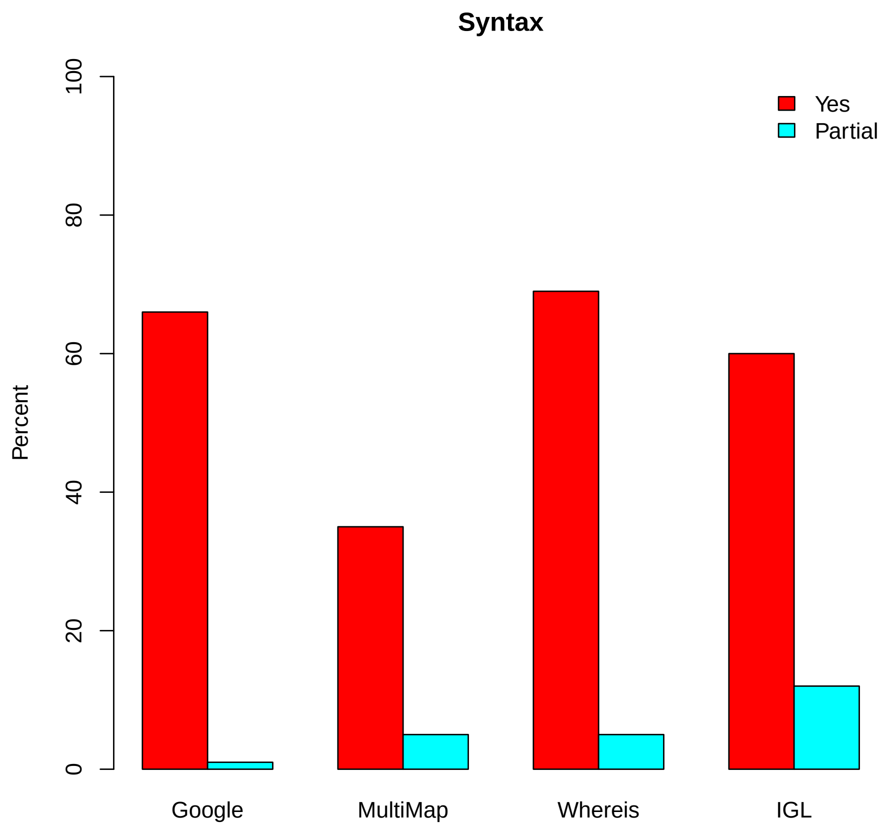


FIGURE 5.18: Results for Syntax Category

IGL had slightly more partial results than the other geocoders, which can be attributed to it producing a result at reduced spatial resolution than the original query. For example, when given a street name and locality, IGL may have only produced a geocode at the locality level. The performance of IGL within the category was heavily dependent on its string manipulation and correction techniques. IGL was designed to be extensible so that additional string correction techniques can be added, because by default IGL uses Levenshtein and Soundex. A proof of concept was shown in the prototype, however with additional correction techniques (which can be added to the agent as plans) the performance of IGL would increase.

5.2.3 Semantic and Geographic Addresses

The address types in Table 5.20 deal with the underlying causes of some address errors. It should be noted that "geographic" types can also be "semantic", but the title of the section is used to reflect that semantic errors can have several causes (spatial, meanings of words etc.), while geographic errors are firmly rooted in spatial relationships.

TABLE 5.20: Test Table for Semantic and Geographic

Address Example	G	M	W	I
23 - Correct street with neighbouring locality				
Hurstford cl, cottesloe wa (should be Peppermint Grove)	Yes	No	Yes	Yes
Inwood Pl, Winthrop (should be Murdoch)	Yes	No	No*	Yes
Doric St, Rossmoyne (should be Shelley)	Yes	No*	Yes	Yes
28 - Vanity address				
17 Thomas St, Mt Richon WA (should be armadale)	Yes	P	No	No
129 Edinboro St, mt hawthorn WA (should be Joondanna)	Yes	Yes	Yes	Yes*
22 Money Rd, Attadale WA (should be Melville)	Yes	Yes*	Yes	Yes
29 - Display unique locations for duplexes				
103A Flinders St, Mt Hawthorn WA 6016	No	No	No	No
5A Davy St Wembley Downs 6019	No	No	No	No
164A Stock Rd Attadale 6156	No	No	No	No
30 - Rural addresses use LGA rather than locality				
304 Timber Creek Crs Toodyay (should be Coondle)	No	No	P	Yes
23 Backland St Esperance (should be Sinclair)	Yes	Yes	Yes*	Yes
6 Albatross Drv Albany (should be Bayonet Head)	Yes	No	Yes*	Yes
31 - Corner addresses				
2 McKay St Bentley is the same as 50 Marquis St	No	No	No	No
240 Bournville St, Wembley is same as 30 Keane St	No	No	No	No
74 Alexandra Rd, East Fremantle same as 9 Coolgardie Ave	Yes	Yes	No*	No

Address Example	G	M	W	I
32 - Rural roads with two names				
Jayes Rd, Boyup Brook WA and Bridgetown Boyup Brook Rd	Yes	Yes	Yes	Yes
Great Eastern Hwy through Kellerberrin is Massingham St (58 Great Eastern Hwy, Kellerberrin)	No*	No	No*	No*
Wubin-Mullewa Rd through Perenjori is Fowler St (26 Mullewa-Wubin Rd, Perenjori)	P	No	No	P
Coalfields Rd through Collie becomes Throssell St becomes Cameron Rd becomes Gibbs Rd (119 Coalfields rd, Collie WA)	P	No	No	No*
39 - Complex sites with private roads				
104 Kununurra Way Coogee Beach Holiday Park 3 Powell Rd Coogee	No	No	No	No
Jackson Ave, Bentley, WA (Curtin University)	Yes	No	Yes	P
Victoria Drive, Royal Perth hospital, Shenton Park WA	No	No	No	No
43 - Ontological Similarity				
Small Creek Rd, Denmark WA (Little River Rd)	No	No	No	No
63 Cambridge St, Wembley WA (should be Grantham). Note 63 Cambridge St is in West Leederville - *Discussion*				
21 Pearson St, Herdsman (21 Pearson Way, Osborne Park) - *Discussion*				
Swan Ct, Yangebup (Cygnet Ct, Yangebup) - *Discussion*				
King Regent Dr, Connolly (Prince Regent Dr)	No	No	Yes	No
46 - Historic change				
Floreat Park, WA (should be just Floreat)	No*	No*	Yes	Yes
Maniana, WA (was a pseudo-suburb in Queens Park)	No	No	No	No
- Rural Address (Location number)				
1073 Abbott Rd, Arthur River WA 6315 (old)	No*	No	No*	No*
21 Abbot Rd, Arthur River WA 6315 (new)	No*	No	No*	No*
3284 Boyup Brook-Arthur Rd, Moodiarrup WA 6393 (old)	No*	No	No*	No*
4512 Boyup Brook-Arthur Rd, Moodiarrup WA 6393 (new)	No*	No*	No*	No*
2738 Rajander Rd, Bowelling, WA 6225 (old)	No*	No*	No*	No*
814 Rajander Rd, Bowelling, WA 6225 (new)	No*	No*	No*	No*
- Rural Address (Lot number)				
Lot 21 Albany Hwy, Arthur River, WA 6315 (old)	Yes?	No	No	Yes?

Address Example	G	M	W	I
16882 Albany Hwy, Arthur River WA 6315 (new)	No*	No	No*	No
Lot 3 Boscabel-Chittinup Rd, Moodiarrup WA 6393 (old)	No*	No*	No	No*
1226 Boscabel-Chittinup Rd, Moodiarrup WA 6393 (new)	No*	No*	No*	No*
Lot 3 Coalfields Rd, Darkan, WA 6392 (old)	Yes?	No*	No	Yes?
10371 Coalfields Rd, Darkan, WA 6392 (new)	No*	No*	No*	No*

The graph in Figure 5.19 shows a low performance for all geocoders in this category. The "partial" results were relatively low as the geocoders either solved the queries completely or not at all. The performance of Google and IGL were similar, with one stand-out feature of IGL was its ability to correct an LGA name with the corresponding locality name. This idea that a locality is within an LGA was added at design time, but further to this are the representative test addresses which involved additional ontological considerations. The inclusion of the rule based system in IGL means that the prototype could be extended to include enhanced ontological processing and reasoning. With the results in mind, it is important to remember that the goal for the prototype was to perform the geocoding process intelligently, which means the focus is not primarily on *what* the results were but rather *how* they were obtained. In other words, the prototype does intelligently (one example is using the dynamic knowledge base) what other geocoders do using lookup tables and static aliases. Also the goal was not to build only address correction utilities but rather a framework for the whole geocoding process.

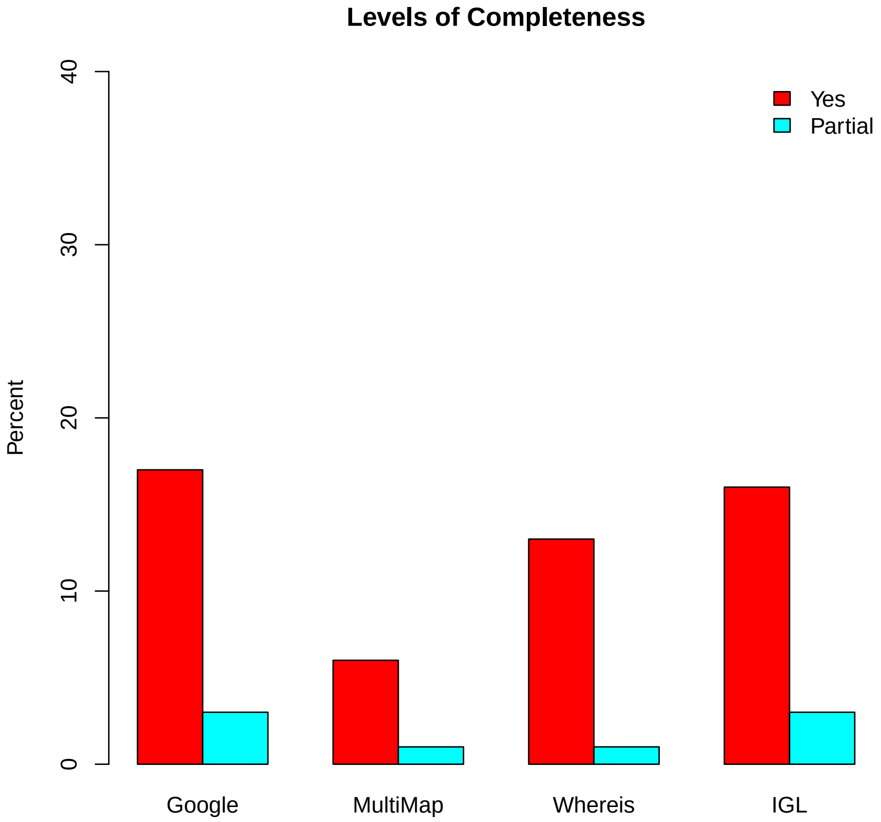


FIGURE 5.19: Results of Semantic and Geographic Category

An example of semantics is the use of a neighbouring locality in place of the correct one, as seen with Cottesloe and Peppermint Grove in Figure 5.20. This type of example occurs frequently, is easy to explain, but most importantly *it highlights how mistakes can be made where the entities involved (localities) have no similarity in language and the ultimate cause is the perception of the user*. The geocoding of 1st order neighbours is well established technique, and handles well, although MultiMap performed poorly in this regard, as it did not have the internal mechanism to check for neighbouring localities, the other geocoders did; this category of address can be seen in Table 5.20 (item 23). IGL was able to identify 1st, 2nd and 3rd order neighbours through a combination of neighbour tables built from spatial data (these spatial operations could also be done directly in the future) and the ability to iterate through several generations of potential solutions (hence the "neighbours of neighbours").

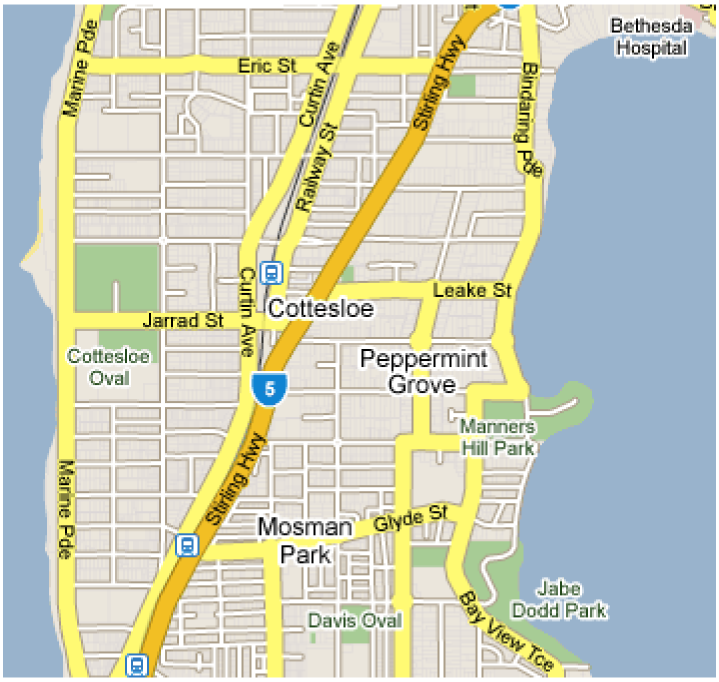


FIGURE 5.20: Neighbouring Localities: Cottesloe and Peppermint Grove

Following the criteria of *no similarity in language* and the *role of perception*, other examples can be found which move from specific (expressed using values) to more "abstract" (expressed using types). Examples of semantics with a geographic component include Cambridge and Grantham streets (seen in Figure 2.1), and Pearson St and Pearson Way (seen in Figure 5.21). The first is a case of two streets which are the same size (in terms of lanes and traffic), are parallel with each other and have the same streets intersecting them (in a grid fashion); as a result these streets are commonly confused. Although the example of Pearson Way and Pearson St involves the same street name, there is more to this, as both streets are metres from each other, run in the same directions and are both near a major lake (a defining landmark). The key to this concept is that users could make mistakes when submitting a geocode based on these semantic and geographic relationships, causes which are not currently included in geocoders. The intelligent framework has provided a foundation via the knowledge base which currently can have this *alias* stored as a rule, and with future work the *relationships* (distance, direction and landmark proximity) could be stored. The tool selected for the knowledge base would not have to change, the ontological structure would just have to be made richer. However the prototype has demonstrated the ability to make a conclusion using two disparate facts, so in this case it would be ensuring these facts can describe the causes (distance, direction and landmark proximity).

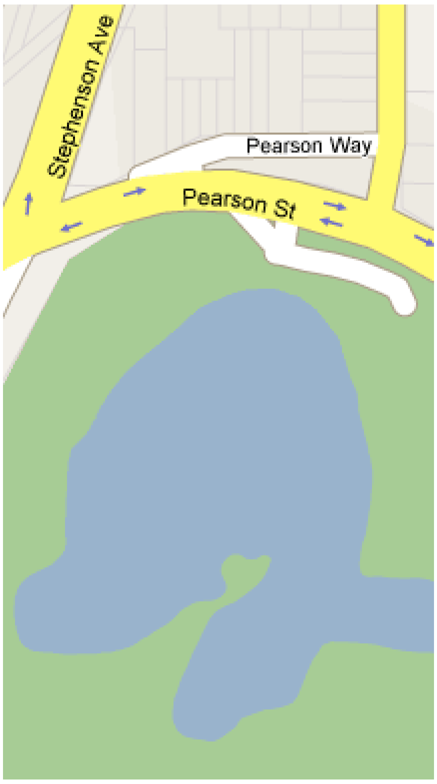


FIGURE 5.21: Geographically and Syntactically Similar

The examples of “Swan Ct” and “Cygnet Ct”, both in Yangebup represents the case where two entities are semantically similar based on what they are named after, in addition to the fact they are in very close geographic proximity to each other and also have the same street type; in this case both street names describe very similar objects. Another example is “King Regent Dr”, which for testing was purposefully submitted as “Prince Regent Dr”; which only Whereis was able to geocode. It seems likely that Whereis derived this solution via word similarity (i.e. the word “Regent”), it also indicates that Whereis uses all tokens in an element string to do similarity (i.e. Levenshtein and Soundex) comparisons. An example of a semantic mistake which cannot be solved from word similarity is “Small Creek Rd, Denmark WA” which should actually be “Little River Rd, Denmark WA”; in this case the confusion arose between “small” and “little”, along with “creek” and “river”. To solve a query like this, a reference is needed which represents the relationships between the elements and can be queried. If geocoders used this approach it would provide a much fuller understanding and better performance. For IntelliGeoLocator to offer this would require the ability to reason over terms such as "small" and "little". This type of ontological reasoning was beyond the scope of the research but the intelligent framework is extensible so that additional semantics (and learning) can be added which could be used for this purpose. This information could also be fitted into the rule-base if it was extended to accommodate this. So if the user "taught" it one time, it could use the new knowledge another time. If the user had been able to identify this relationship and build in this rule by hand, the knowledge would be available for future queries (i.e. not fully automated, but the "proof of concept" is there).

The vanity address problem is really no different from the problem of using the wrong (often neighbouring) locality. All the geocoders tested handled this well. The design of IntelliGeocoder with its ability to run in iterations means that it can search for localities which are actually second or higher order neighbouring localities. An example of this is seen in Figure 5.22 where it is shown how the locality Daglish is a second order neighbour to Wembley.

Further research is required and may involve building a rule comprising geographic functionality, possibly utilizing a web processing service function. A GIS could be used to dynamically check whether an address is a potential corner address, by checking its proximity to street corners; if it was a street corner, by checking the cadastre information around it, it could possibly determine what its "other" address might be.

5.2.4 Iterative Processing

To demonstrate the result of using iterative processing, one of the test addresses which has compounded errors can be used. The address in its incorrect form is "Glosster St, Shentin Park, WA"; the correct equivalent is "Gloster St, Subiaco, WA".

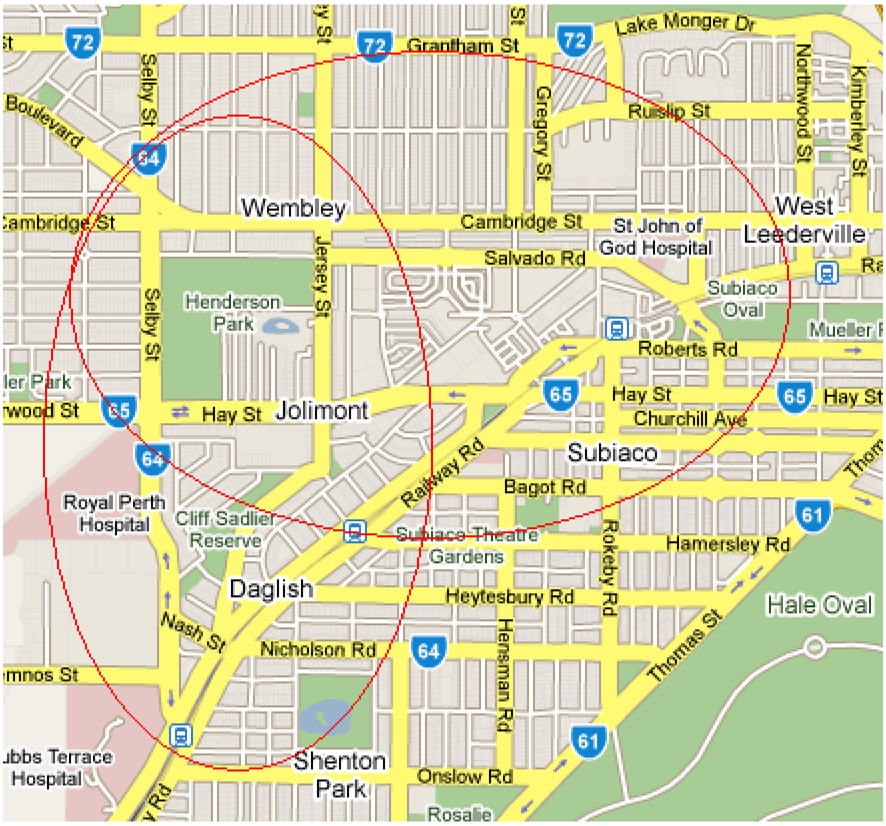


FIGURE 5.22: Daglish is a Second Order Neighbour to Wembley

TABLE 5.21: Sample Address with Compounded Errors

Glosster Present: Yes Exist: No	St Present: Yes Exist: Yes	Shentin Park Present: Yes Exist: No	WA Present: Yes Exist: Yes

In this case, as seen in the list of sample addresses, the address has a misspelled street name, uses a neighbouring locality and the name of the neighbouring locality is also misspelled.

Because the street name does not exist, the only techniques which can be used with it are the knowledge base, Soundex and Levenshtein; the same is also true for the locality. In particular, the “neighbouring locality” technique cannot be used for the locality because it does not exist. The original address is given the query-ID, sub-ID and parent-ID of 1, 0 and 0, respectively (the notation 1:0:0 can be used to show this).

The Soundex and Levenshtein techniques were used on the street name, and the results can be seen in Table 5.22; this table also shows the suggestions for locality. The Levenshtein technique was also used for the street name but there were no results.

TABLE 5.22: Suggestions found during First Iteration

Type	ID	Technique	Value
street	1:0:0	Soundex	Galston Place (Duncraig)
street	1:0:0	Soundex	Ghooli South Rd (Ghooli)
street	1:0:0	Soundex	Glastonbury Way (Wattle Grove)
street	1:0:0	Soundex	Glastonbury Rd (Armadale)
street	1:0:0	Soundex	Gloster St (Subiaco)
street	1:0:0	Soundex	Gloster Way (Woodvale)
locality	1:0:0	Soundex	Shenton Park

It should be noted too that the most important relationships in this example are *street and locality* and *locality and state*. The street type is an unusual element in that it is most useful for “breaking a tie” in the event that there are two streets in the same locality with the same name. Aside from that happening, the street type is obviously so non-discriminatory that it does not affect the street chosen; however it is still used as a factor in matching.

The suggestions are sent to the matching agent but no matches exceeding one element are found. As described in Appendix A.2, each “match” (in this case just single elements) are assigned updated tracking IDs and distributed back to the agents.

Table 5.23 shows the element values as they are sent back to the agents; note that each of these now has a unique sub-ID.

TABLE 5.23: Element Values with Updated IDs

Type	ID	Value
street	1:1:0	Galston Place (Duncraig)
street	1:2:0	Ghooli South Rd (Ghooli)
street	1:3:0	Glastonbury Way (Wattle Grove)
street	1:4:0	Glastonbury Rd (Armadale)
street	1:5:0	Gloster St (Subiaco)
street	1:6:0	Gloster Way (Woodvale)
locality	1:7:0	Shenton Park

As explained in Section A.2, each agent receives back the elements of its particular element and begins processing this; as a consequence its complementary agents realize they have no element with that corresponding sub-ID, and because of this take the default value of the parent address. In the example, the street agent would realize that it has no value with a sub-ID of “7”. Also, separately, the locality agent would realize that it has no value with a sub-ID of “5”. These are just two of the “new” addresses spawned from iteration; if we were to do a “join” across the agents the two addresses (amongst many) would be seen, as detailed in Table 5.24.

TABLE 5.24: Selected First Generation Addresses

ID	Street	Street Type	Locality	State
1:5:0	Gloster	St	Shentin Park	WA
1:7:0	Glosster	St	Shenton Park	WA

These are two of many new address combinations being processed, and these results indicate that there are at least two combinations which should eventually lead to the same result, although the steps taken to get there may be in a different order.

Of the two addresses shown in Table 5.24, the second can be followed further into its iterations. The suggestions found this time for the values with ID 1:7:0, can be seen in Table 5.25.

TABLE 5.25: Suggestions Found during Second Iteration

Type	ID	Technique	Value
street	1:7:0	Soundex	Galston Place (Duncraig)
street	1:7:0	Soundex	Ghooli South Rd (Ghooli)
street	1:7:0	Soundex	Glastonbury Way (Wattle Grove)
street	1:7:0	Soundex	Glastonbury Rd (Armadale)
street	1:7:0	Soundex	Gloster St (Subiaco)
street	1:7:0	Soundex	Gloster Way (Woodvale)
locality	1:7:0	geo neighbour	Daglish
locality	1:7:0	geo neighbour	Floreat
locality	1:7:0	geo neighbour	Jolimont
locality	1:7:0	geo neighbour	Karrakatta
locality	1:7:0	geo neighbour	Kings Park
locality	1:7:0	geo neighbour	Mount Claremont
locality	1:7:0	geo neighbour	Nedlands
locality	1:7:0	geo neighbour	Subiaco
locality	1:7:0	geo neighbour	West Perth

When these are sent to the matching agent, a significant match is found between the street, Gloster St (in Subiaco), and the locality Subiaco. Although not mentioned explicitly in the example, element matches would also occur (using the same approach) between the street name and street type, and also the locality and state.

TABLE 5.26: Second Generation Element Values with Updated IDs

Type	ID	Value
street	1:8:7	Galston Place (Duncraig)
street	1:9:7	Ghooli South Rd (Ghooli)
street	1:10:7	Glastonbury Way (Wattle Grove)
street	1:11:7	Glastonbury Rd (Armadale)
street	1:12:7	Gloster St (Subiaco)
street	1:13:7	Gloster Way (Woodvale)
locality	1:14:7	Daglish
locality	1:15:7	Floreat
locality	1:16:7	Jolimont
locality	1:17:7	Karrakatta
locality	1:18:7	Kings Park
locality	1:19:7	Mount Claremont
locality	1:20:7	Nedlands
locality	1:12:7	Subiaco
locality	1:21:7	West Perth

Note in Table 5.26 that the same sub-ID value is shared by the street and the locality (1:12:7). The street type and state used in the matching/reconstruction would also have the same ID. When these are sent back to their respective agents, if a "join" was to be done, it can be seen that a complete address is distributed between the agents. At this point, a complete match has been found and there is no need to continue processing other possibilities, unless this is desired. This is an example of a contextual choice which could be set by the user, do they want to stop with the first completely correct match, or instead continue processing in the event that there are several complete matches? Also relevant to this is the situation where a complete match is never found, and instead several partial matches are found; depending on the number and type of elements they would have different quality scores.

5.2.5 Compounding of Errors in Addresses

This section describes a small number of problem address types, but they are worth considering, as they reflect the situation where a user makes multiple mistakes in a single address. A subset of the addresses tested are presented in Table 5.27.

TABLE 5.27: Test Table for Compounded Errors

Type of Address	G	M	W	I
—— [24] Misspelt street with neighbouring locality				
Jup Ln, Belmont (should be Jupp Ln, Rivervale)	No	No	No	Yes
Dorofy St, Bassendean (should be Dorothy St, Ashfield)	No	No	No	No*
Rensure Blvd, Mindarie (Renshaw Blvd, Clarkson)	No	No	Yes	No*
—— [25] Correct street, misspelt neighbouring locality				
Hampton Rd, Handsdale (should be Hampton Rd, Darch)	Yes	No*	No*	Yes
Fletching St, Mirabooca (should be Fletching St, Balga)	Yes	Yes	Yes	Yes
Willis St, Wykiki (should be Willis St, Warnbro)	Yes	No*	No*	Yes
—— [26] Misspelt street, misspelt neighbouring locality				
48 haleswoorth rd, floreatt wa (Halesworth, Jolimont)	Yes	No	Yes	Yes
Glosster St, Shentin Park wa (Gloster, Subiaco)	No	No	Yes*	Yes
Erwin Rd, Morley WA (should be Irwin Rd, Embleton)	No	No	Yes	No
—— [27] Misspelt street, wrong street type, misspelt neighbouring locality				
Ozborn St, Palmyra WA (Osborne Rd, East Fremantle)	No	No	No	Yes*
Longfjord St, Wide Gum Valley WA (Longford Rd, Beaconsfield)	Yes	No	No*	Yes
Russ St, Bealiar WA (Rhus Ct, Yangebup)	No	No*	No*	Yes*

The results from this category are shown in Table 5.23, where there are no "partial" results, only successful. Also, it can be seen IGL performed better than the other three geocoders in this category.

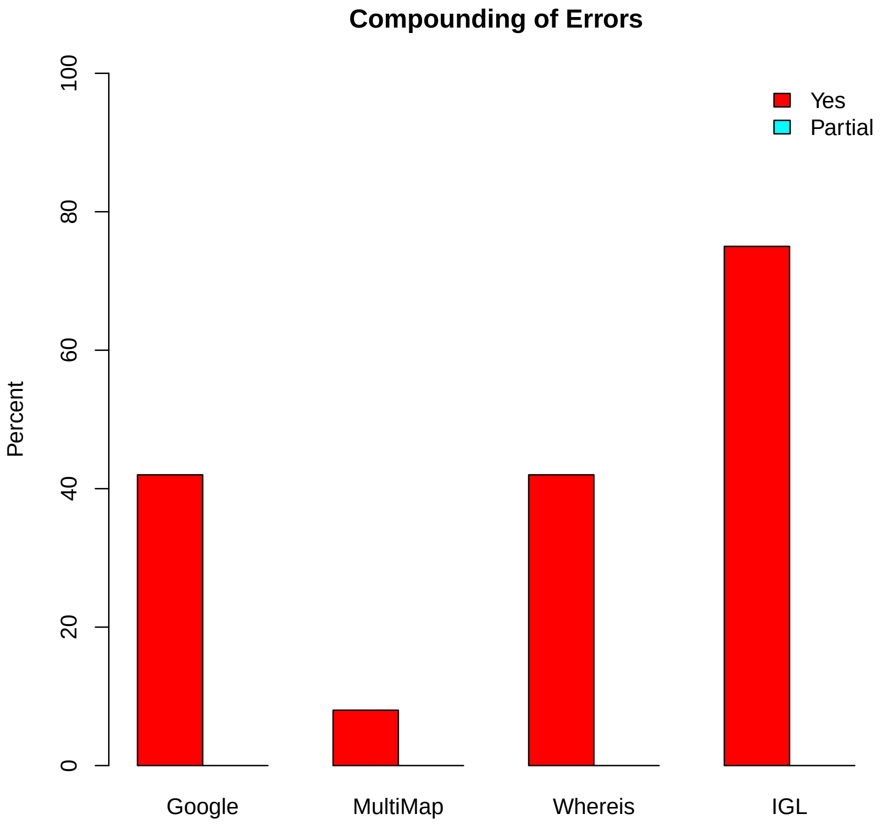


FIGURE 5.23: Results for Compounding of Errors Category

Because IntelliGeocoder uses an iterative execution design, it is suited to dealing with compound errors. This is because at each “generation” of correction, *each correction technique* is applied. This results in permutations expressed in what is effectively a “suggestion tree’. Assuming that all of the sources of error in the address can be corrected by the techniques used in IntelliGeocoder (Soundex, Levenshtein, neighbouring locality, knowledge base), the order of how these corrections should be applied is not a constraint with the design.

5.3 Derived Functionality

The use of a complete address demonstrates that an address can successfully flow from start to finish through a geocoder using the control knowledge embedded in the events, goals and plans used in processing. Strengths can be highlighted, in particular the almost endless possibilities provided by an agent utilizing its strengths in control, along with the ability to rewrite the rules and facts in the knowledge base at runtime (due to the rule based system being interpreted).

5.3.1 Fusion of Knowledge and Control

The results of fusing the knowledge (contained in the knowledge base *and* the agent) with the agent, and its associated control, can be categorized in two ways. The first is that because the knowledge base is interpreted, it can be altered dynamically at runtime; the second is that the agent (which is compiled) can have its behaviour altered by it accessing the knowledge base in its compiled code. Essentially this means that the compiled code accesses the knowledge base in a set way, but the knowledge it accesses enables the agent to exhibit "pseudo-interpreted" behaviour.

The knowledge base in the prototype stores facts which are at a fundamental level of knowledge, similar to those in the G-NAF (Richards and Paull, 2003). However these are stored as facts and they have the capacity for inference. The sample aliases seen in Program 1 are representative of the locality aliases created in the prototype, and were created as a result of the correction techniques (e.g. Levenshtein, Soundex, geographic neighbour). Specifically, these were created when the original query was successfully resolved to a single corrected result; however with minor improvements to the prototype in the future several geocoding suggestions could be provided, they can all be stored as aliases, but with the quality codes indicating the confidence that they are.

There is no measure of confidence to the correctness of the alias *stored* in the knowledge base (not to be confused with the scoring in the agent component), if it is present in the knowledge base it is assumed to be correct and reliable. In the prototype, aliases were not stored for street names, however if they were, it would be possible to use two disparate facts

Program 1 Aliases Stored in the Knowledge Base

```
(MAIN::locality-alias (alias-name ``SUBI'')
                      (real-name ``SUBIACO'')
                      (state ``WA''))

(MAIN::locality-alias (alias-name ``FREO'')
                      (real-name ``FREMANTLE'')
                      (state ``WA''))

(MAIN::locality-alias (alias-name ``EAST FREO'')
                      (real-name ``EAST FREMANTLE'')
                      (state ``WA''))
```

(one regarding locality, and one regarding street name) to solve a single incorrect query with those two elements being incorrect. An example of this is in a first query ("Runner Ave, West Perth") an alias for "Walker Ave" (correct) was stored as "Runner Ave" (incorrect). In a separate query ("Ord St, Western Perth"), an alias is stored for "West Perth" as "Western Perth". In each of these individual queries, they were solved because they had other complementary information. In the first query a semantic search found a street suggestion and had a correct locality to confirm this; in the second query the street was found, and its corresponding locality ("West Perth") was considered a confident match with the incorrect input ("Western Perth"). With these aliases now stored, a third query ("Runner Ave, Western Perth") was easier to match. The knowledge base is one of the correction techniques which can be called *after* or *before* another correction technique.

The control of the agents in the prototype is affected by the retrieval of these aliases. When the agent finds an alias in the knowledge base, it uses the entry and does not need to use the other correction techniques; this is because the entries in the knowledge base are derived using the other techniques. Further work could allow the knowledge base to synthesise its own knowledge, then it would be used *in conjunction* with other correction techniques.

The test addresses which had incorrect or abbreviated localities demonstrate the transfer of domain knowledge to control knowledge, where the locality aliases stored in the knowledge base become asserted in the agent's beliefsets.

The use of relational, tuple-based beliefsets in the agent system is sufficient for storing operational knowledge, and for the bulk of test addresses it is suitable for storing the domain knowledge. This is because (i) the level of knowledge being stored in the knowledge base is low level aliases, and (ii) the issues associated with many of the problem addresses are syntactic. But the test addresses with an ontological similarity reveal that a different approach (other than relational storage) is needed to store, and infer, the relationships needed to geocode these queries; the rule based system serves this purpose.

For the levels of completeness category often there is no way to "repair" something that is missing (and the associated ambiguity), although sometimes the missing information can be solved from other complementary elements or other datasets, and from users. The intelligent geocoding model does take this into account, it was just not within the scope of implementation of the prototype. Rules in the knowledge base relating to aliases have a geographic context because the aliases are only relevant in given areas (i.e. one area, many areas or universal).

IntelliGeoLocator demonstrated how an agent which displays dynamic behaviour from static code can connect with a rule based system to provide the ability to dynamically rewrite code; this itself is the link between control knowledge and domain knowledge. From this it was shown that domain knowledge could transition to control knowledge when the agent accessed the knowledge base, took the results and put them into agent form (beliefsets, messages, plans) where they could be acted upon. An example of this was the aliases in the knowledge base being used to solve queries. Also demonstrated was the use of a basic ontology. Context in the knowledge base included the state associated with a corrected locality, and also the locality associated with a corrected street.

The highlight of learning in the system is the automatic generation of rules which are stored in the knowledge base; this makes full use of the rule based system. This prototype did this successfully when a single suggestion/match was found for a query. This was because (according to the processing of the prototype) there was only one possible solution. During testing an address with a correct street name and incorrect (but valid) locality was corrected, and a rule created. The correct locality was a neighbouring locality, and it contained the street name. It is important to look to the future and recognize that using the knowledge base will provide functionality that a file-based solution or database could not. The use of a rule based system for matching was unique. Included with the development of the knowledge base was the associated ontology and high level rules; if these had reached a greater level of sophistication it is anticipated that better results in the area of learning would have also been possible.

5.3.2 Context

The results of using context can be seen in the different roles that the agents have, the plan selection inside agents, the messaging between agents, and in the knowledge base.

Having agents with different roles allowed for parallel processing between agents, but in terms of context, the most benefit of separate agent roles came from the agents being able to specialise in their own element type, and the fact that by having separate agents *allowed for* the design of plan selection in the agents and messaging. Having the `UserAgent` provided a common entry point for new queries, and allowed for the creation of tracking data for each new query. This tracking data was critical for the operation of the prototype and segmenting this functionality into one agent was a convenient way of ensuring the tracking information was created before distribution. Because the user agent manages all interaction with the user, the element agents do not need to do this which made design easier and in the future if there is increased interaction with the user (e.g. via a website) then it would be easier given that there is a single point for communication with the user.

The `MatchingAgent` also provides a single point of coordination, with regards to bringing together the results of the various `ElementAgent`s. This worked without any problems, and was essential for aggregating the different elements; it also provided a single point for using the combined elements to execute a query to the database or web service to retrieve coordinates. Although the `PresentationAgent` was not implemented in the prototype, it has been planned for and would customise presentation according to context based on the needs of the user; in this sense it would be an intermediary between the `UserAgent` and `ElementAgent`s.

Because there is just one `ElementAgent` and it is set at runtime to be responsible for a particular element type, context is important for it to perform the processing relevant to its element type. This is where the *relevant()* and *context()* methods provide static and dynamic choice with regards to plan selection. Examples of this in the element include selecting plans based on the element type, and also the combination of element type and the complementary element type.

Program 2 *Relevant* and *Context* in the Agent Language

```
static boolean relevant(BuildAdjacentsGoal ev)
{
boolean returnBool = false;
  if ( ev.element_type.equalsIgnoreCase("locality") )
  {
     returnBool = true;
  }
  return returnBool;
}

context()
{
  ev.upper_neighbour_type.equalsIgnoreCase("state");
}
```

In Program 2, it can be seen that the language-standard methods of *relevant()* and *context()* are used to test if the element type is “locality”, and the upper complimentary element is “state”. It is important to note that the relevant() method is static in terms of the Java language, and so tests in this method must be consistent with those expected in a Java static method; in this case it means this is where the payload of agent messages are examined. In the example, another value in the message is also tested. The power of the *context()* method is its ability to not just test with a static limitation, but also with dynamic objects - especially beliefsets. It is in the *context()* method that the agent can evaluate a beliefset at the time of plan selection, and use the results of the beliefset query to select a plan. This has significant potential power because it means that the *context()* method could (as it is appropriately named) be used for modifying the processing of the geocoder based on the context of the user or application. For example, the particular context of the user could be captured via the website and each particular preference stored in a beliefset. The code sample in Program 2 is presented to reinforce how context is so specifically catered for in the agent language. In the prototype, use of the knowledge base was from agents plans, and direct calls can be made to the knowledge base in the *context()* method.

The messages used in the prototype are mainly “well-known”, in the sense that each agent has the code it needs to adequately process any messages it receives at runtime. The best example of context in the prototype messaging is the “message overloading” which is used to send a particular *type* of message but have their elements vary in their type (seen in Program 3); this is the same concept as method overloading in regular object-oriented programming.

Program 3 Message Overloading

```
#posted as
    attributes(int queryID_v, String element_type_v,
    String element_value_v)
    {
     ...
    }
#posted as
    attributes(int queryID_v, String element_type_v,
    String element_value_v, boolean suggested, ArrayList loc_list)
    {
      ...
    }
```

The code in Program 3 is included to highlight that context exists not just in where the message came from (and where it is going), but also the structure and contents of the message. This provides a means whereby geographic semantics represented in a message could be used to determine the action to take. Further research would be required to investigate how to achieve this.

There is context in the knowledge base, as the aliases are only applicable in certain situations. For example the locality aliases are only applicable in certain states. The same could be done with streets in states, streets in localities and so on. The results from the prototype indicate a basic capability, but there is so much more that could be done using this approach in future research. There is a powerful complement where the knowledge base provides results which incorporate geographic context, and the agent uses the results in a way that utilizes control context. The agent-based and rule-based paradigms have provide a novel and effective control mechanism. Plans are the embodiment of context, as seen by the three levels of granularity available.

5.3.3 Rule-Based Address Reconstruction

In explaining the results for rule-based reconstruction, the same example address is used as presented in Section 5.2.4, which is “Gloster St, Subiaco, WA” (incorrectly inputted as “Glosster St, Shentin Park, WA”). The initial suggestions from the ElementAgents can be seen in Table 5.28.

TABLE 5.28: Initial Suggestions from the ElementAgents

[wa] 1:0:0
[*wa*, shenton park] 1:0:0
[*duncraig*, galston place] 1:0:0
[*ghooli*, ghooli south rd] 1:0:0
[*wattle grove*, Glastonbury way] 1:0:0
[*armadale*, Glastonbury rd] 1:0:0
[*subiaco*, gloster st] 1:0:0
[*woodvale*, gloster way] 1:0:0

These values can also be seen in Table 5.21, except here the emphasis is placed on the “equivalent” element values which are looked up, and paired with the suggestion. Note that WA does not have an equivalent element as it is the “top most” element.

With these suggestions asserted in the matching engine, it runs, and once the matching engine has run, the IDs assigned to the matches (of which there are not very many) can be seen in Table 5.23. The only match (i.e. two or more pairs) is:

$$[\text{wa}] + [wa, \text{shenton park}]$$

This match, along with the various suggestions (which did not overlap with any other elements) are sent back to agents with new IDs. Although the majority of the suggestions were not matched, they are still given unique IDs.

As mentioned in the design chapter, for each of these unique IDs, a new “thread” is weaved which constitutes a new address - where necessary the default (i.e. original) address elements are used to fill the “blanks”. From the example, two of the addresses can be seen in Table 5.20. From

Tables 5.23 and 5.24, we can see that “Shenton Park” and “WA” were assigned the ID 1:7:0 - this is because they (the pairs) matched within the reconstruction. When the elements were sent back, street and street type lacked a value so the values of these elements (“Glosster” and “St”, respectively) were taken from the original input.

Following the progress of the address with ID 1:7:0, it means that any suggestions from *this* address will also have ID 1:7:0. These can be seen in Table 5.25. It is important to note which correction techniques produced these suggestions; Soundex was used on the street name with ID 1:7:0 because no technique had been applied to it previously. For the locality element with ID 1:7:0, only the “geo neighbour” correction can be used because it was derived from locality 1:0:0 using Soundex, and so it cannot have Soundex used twice. As mentioned in the design chapter, only certain particular correction techniques can be used in sequence. The suggestions from the address with ID 1:7:0 can be seen in Table 5.25, and reveals several pairs, seen in Table 5.29.

TABLE 5.29: Suggestions from the Address with ID 1:7:0

[*subiaco*, gloster st.]
[*wa*, subiaco]
[wa]

It can be seen that the overlap from these three provide the complete address of gloster st, subiaco WA. The same rule-based address reconstruction approach is used for street names and street types, but is not shown in the example. Street numbers and unit numbers were not implemented, but the same concept would apply and they could be implemented in a similar manner. The difference with street number and unit numbers is that the “direction” of the complementary “lookup” would be different. Remember that for street name, locality, and postcode they all “lookup” the adjacent element which is a larger aggregation. For street number and unit number, the street name would be required to look these up and discover each possibility - then these could overlap with the inputted values.

The matching worked as expected, with every possible match being found, including the single elements. With each suggestion having its own unique

ID, the matching is not confused between the different addresses, despite the addresses being all in the same JESS runtime memory. This runs fine, however it means that the facts and rules build up over time. If the main memory in JESS is reset then all of the facts and rules are lost. There are two ways this could be mitigated, although these were not implemented in the prototype. These include using JESS modules to partition the execution space so only certain rules run in an area of memory and the rules will only match facts in the same given area of memory. The other option is retracting the facts after the matching rules have been applied; this can be done because the right-hand side (RHS) of the rule can reference the facts used to activate it.

The code created to extract the results of the matching from within the JESS memory and bring these “back” into the agent memory also worked. This code provides a parameter which determines what level of matching results should be fetched. This level is the number of pairs which constitute a match. In this way, an address with five, four, three pairs and so on can be found; this means the code can be called and the best match can be found. Aside from the domain knowledge stored using the rule-based system, there is also control knowledge as seen in this address reconstruction. The results of using a rule-based system have indicated how a rule based system is useful for situations where no single algorithm exists.

5.3.4 Flexibility

The existing flexibility that comes in the system is made possible by the combination of beliefsets, plans and rule based systems. Included within the existing flexibility are two ways to terminate processing dependent upon the context, (i) stop when a certain quality is reached, or (ii) continue until the maximum number of children levels is reached. Inside the rule based matching system, the ability to set which elements are passed on and which are not was a useful feature. Additional flexibility is provided by the ability to adjust whether the MatchingAgent does the immediate, raw geocoding. The prototype also demonstrated how weightings can be tied to importance or user confidence of element.

When a new geocode query is submitted, it would be a useful to enable the user (which could be a person or another computer) to have a “dialogue” for geocoding or only a one time data entry. Improvements to processing flexibility would include which geocoding algorithm to use for scoring, either a conservative or broad choice; an example of the algorithm choice considerations are seen in Table 5.30. In addition to this, the geocoder could be configured to only use domain knowledge which has been designated only for a particular use.

TABLE 5.30: Properties of Emergency and Business Geocodes

Emergency	Business
Critical to have only 1 match	Multiple matches are acceptable
Tied matches and choice of scoring algorithm is important	Scoring algorithm serves as a rank
High level of confidence required	More assumptions and lower confidence acceptable
Near real-time is desirable	Less time sensitive

Presentation information could also be enhanced, given the incoming address and by including extra information such as platform and use captured from the user and have the presentation agent make decisions based on this; this would be presentation context.

5.3.5 Extensibility

The design of the intelligent geocoding model can easily extend the framework to accommodate other syntactic and semantic considerations in resolving addresses. This was the rationale behind having plans execute the correction module; if a new correction technique is desired it can be put in a plan, and added with little modification to existing code or changing of the current design. This of particular interest for semantic correction techniques which may developed in the future.

The knowledge base is also ready to be extended, extending the different geocoding rules (eg. different jurisdictions, applications, countries, etc.) to support the concept of the unified geocoding model. For different countries, the idea would be that the same geographic primitives would be used to represent any country, but the relationships between these would be specific to that country. The facts and rules in the knowledge base would reflect this and the geocoding engine could then reason over these.

5.3.6 Information Discovery and Access

The datasets used in the project were pre-determined in the software, and called when needed from within the plans. In the case of a local database, the SQL was formed dynamically, and for the web service query the URL was also formed dynamically. The foundation is present in the intelligent model to allow settings about the datasets to be stored as facts and rules in the knowledge base. For example, this would mean there could be different plans for the different datasets, and then within these plans the code accesses the most suitable dataset of that type. An example of this would be plans for local database and web service, and then within the database plan it queries the knowledge base using a particular geographic region (for example Bentley, WA) and address type (for example complex type) and then finds the best dataset. There is significant potential for storing settings of how to access web services (and metadata describing these services) in the knowledge base, which could be updated dynamically. This would eliminate the need for "hard coding" where to find data. This is related to the overarching theme of the semantic web, and the proliferation of services on the Internet and the ability to find and broker previously unknown services.

5.4 Conclusions

Results confirmed that the algorithm for scoring led to self-determined behaviour, as it stopped when it found a natural maximum. The scoring used the social aspect of agents, performing coordination and communication with the agent messaging framework; this also means that the scoring approach embraces parallel and real-time processing. The scoring approach allowed for flexibility and quality by including weightings for address elements measuring both (i) importance of the individual element, and (ii) the confidence associated with that element at a given point in time (in real-time). Relevant to scoring is that future implementations could add as many factors for determining geocode quality as desired (it is expandable), these can be physical (additional geographic features) or virtual (e.g. cultural).

The event driven paradigm is well suited to the real-time nature of web service requests on the Internet, and the agent lifecycle is also well suited to geocoding, in several ways. First is the reactive aspect of responding to a new geocode request, this is followed by the pro-active aspect of doing everything possible to find a solution. The constant cycle of observe, plan and act is also very relevant where there is constant and dynamic updating of beliefs.

A controlled ripple effect in scoring is useful and manageable, where some of the address elements contribute to the scores of some other elements but not all. IGL has demonstrated parallel processing for geocoding and is a scalable system.

The use of user perception in conjunction with semantic and geographic correction has merit, however a richer ontology is needed for this. The knowledge base has shown a proof of concept for storing geocoding rules, and this could be extended to include greater sophistication. It was shown that there can be compounded errors in geocoding, and this presents the idea that "deeper" causes of error could be at work in some incorrect geocoding queries. The agents were able to access knowledge stored from previous experience and use this to complete a subsequent query, which shows learning for geocoding in a prototype form.

With multiple tiers of context built into the agent framework, it is well suited to modifying geocoder behaviour based on varying situations. Context was included via (i) the different roles of agents, (ii) plan selection inside agents, (iii) messaging between agents, and (iv) the knowledge base.

Because the research was focused on getting the framework (agent and knowledge base) to work as a whole, there was less effort directed towards *correction* algorithms. If more correction modules were added to the agent (to cater for all types of problem addresses) then performance would increase. The BDI processing approach could also be applied specifically to correction algorithms.

The prototype demonstrated a unique approach using beliefsets to create a mechanism which simultaneously (i) tracked the various elements being processed in parallel and coordinated their reconstruction, (ii) enabled multiple "generations" of addresses, (iii) allowed compound errors in addresses to be discovered during iterative processing, and (iv) pursue multiple possible solutions for a single address (and its elements) then choose the best.

The idea of a compunded error in an address and the fact that this problem type was not handled by other geocoders is preliminary evidence that other types of problem addresses may exist. It also hints that user cognition could be a legitimate source of address errors. To handle these sort of potential errors, it makes sense that a framework for geocoding which has iterative processing, inherent geographic semantics, a dynamically rewritable rule base and extensibility would be useful. The compounded error type benefited from the iteration ability of IGL and the way it can pursue multiple possibilities.

The way in which IGL processed addresses in the geographic and semantic category is unique in that the agent representation and messaging provided inherent geographic semantics. This approach of having the software paradigm be intertwined with the problem concept sets IGL apart from other geocoders. The significance of this is not just tied to the results from testing, but the power and extensibility the intelligent framework provides.

The dynamic rewriting of knowledge in the rule base has demonstrated that a geocoder can add and update its own domain knowledge at runtime. This has significance for future geocoders, because it provides an opportunity for less human intervention and for the knowledge base to increase its own knowledge autonomously. Another significant outcome is that IGL has demonstrated that domain knowledge can be transferred to control knowledge; there is actually a loop where after the agents use domain knowledge to make control decisions they then use the outcomes to add new knowledge to the knowledge base.

Using a rule based system for aggregating data from multiple agents is useful because it does not fire until all needed data is asserted, so two functionalities include (i) having many rules to determine control based on content of the data, and (ii) temporal control where the engine waits for all agent input before proceeding. Using a paradigm which enabled inherent semantics which was (i) had more functionality, (ii) enabled intelligence, (iii) was very apt to the event driven and service oriented nature of geocoding.

It has been concluded from the testing of Section 5.1 that there is a fundamental, quantitative difference in how the prototype performs geocoding and that results indicate there are intriguing properties unique to agent geocoding. It was seen from Section 5.2 that the prototype handles a variety of addresses, is suitable for use as a geocoding engine now, and with further research it could provide enhanced capabilities. The functionality described in Section 5.3 indicates there are other benefits from using an agent paradigm, in particular control and context.

CHAPTER 6

CONCLUSIONS

INTELLIGENCE in geocoding is a product of both context and semantics (at a conceptual level) and control and knowledge (at an implementation level), where the two are "connected" by the agent paradigm which is *both* a representation and a solution. This has provided a foundation for intelligent geocoding and the conclusions presented in this section cover the objectives originally presented in Section 1.5, namely identifying relevant issues, developing an intelligent framework and examining both control and knowledge.

6.1 Issues Relevant to Intelligent Geocoding

The specific opportunities for improvement which arise from the issues in contemporary geocoding (Section 3.1) have a common theme which is that intelligence provides solutions for all of them. Based on the flexible capabilities of the prototype, the idea still stands that if the normalization and reference data are as good as they can be, then benefit will come from working with the user, understanding what they want and also making the most from prior experience. Specific types of issues that are relevant to intelligent geocoding including (i) the underlying process, specifically within matching (Sections 3.1 and 2.2.3), (ii) a mechanism in software to interact with increasingly available web services (Sections 1.3.1 and 5.3.6), (iii) coping with an increasingly event driven and real-time nature of the Internet (Section 1.3.1), (iv) robustness of natural language processing and availability of reference data will continually increase (Section 1.3.1), (v) greater attention needed for problem addresses based on semantic and ontological errors (Section 2.2.1), (vi) determining and expressing match quality, (vii) context (user, application, geographic, jurisdictional) as in seen in Sections 2.3, 2.9, 3.2.2.5 and 5.3.2, (viii) reducing human intervention in maintenance and resolving problem addresses (Section 1.6). The rationale for the intelligent framework for geocoding is that the existing geocoding process would benefit from a new paradigm, providing the improvement needed to cater for these current and emerging needs. Each of the issues identified and subsequently pursued in the research found their way into the design and subsequently into the prototype, which shows that the issues had real solutions and that they also contributed to the new functionality presented in the results chapter.

6.1.1 Intelligence as it Pertains to Geocoding

Intelligence can be used for improving geocoder performance over time and reaching new conclusions. This can be done by using a rule based system to combine disparate facts describing address elements to solve new queries (Section 5.3.1). For geocoding, intelligence includes making decisions based on subtle differences that are situationally dependent. An example of this is the use of context for control as seen in Sections 2.3, 2.9, 3.2.2.5 and 5.3.2. The ability to adapt, and the flexibility to make choices and operate in new situations is a property of intelligence and in geocoding this is made possible by accumulating dynamic aliases over time, and subsequently increasing knowledge (Section 5.3.1). An intelligent geocoder has the ability to form outcomes catered to the subtleties of the address being submitted. This non-deterministic behaviour is described in Section 2.4.1 and means the framework can handle subtle differences in queries and form results accordingly. Social behaviour with users and other software can be seen in the intelligent framework via the messaging capability (Section 2.8) and event-based processing (Section 2.7) which both contribute to providing intelligence in geocoding. Having an understanding of the data and concepts the geocoder uses is part of intelligence, and an approach to this is a shared geographic ontology that can be used for control and domain knowledge (Sections 2.4.2 and 3.2.2.5). Intelligence means being able to consider multiple possible solutions, i.e. multiple directions of inquiry and using the best result (Section 5.2.4), which IGL does. IGL is also able to use past experience to influence current decisions (and improve its performance), which is the fusion of knowledge with control (Section 5.3.1) and the use of domain knowledge (Section 4.3.6). When accessing this knowledge, IGL also demonstrates intelligence by understanding that only certain knowledge is relevant in particular situations (use of context). Intelligence in geocoding also includes the ability for justification, and IGL is capable of explaining the steps and rationale it used in finding result (Section 4.3.6) because it uses a rule based system.

6.1.2 Desirable Properties of an Intelligent Geocoder

The three main properties of intelligent geocoding include control, knowledge and learning. Intelligent control in geocoding results in behaviour which is event based, goal directed, distributed, parallel, non-deterministic, metaprogramming, recursive and object oriented (Section 2.4.1). Two knowledge properties which are desirable for intelligence in geocoding include ontological structure and logic. An ontology provides the structure for knowledge in the database, in addition to the ability for reuse and sharing (of the ontology and for the knowledge it describes), interoperability, structuring knowledge bases, browsing and search (Section 2.4.2). Logic utilises the structure provided by an ontology, and allows for applying and evaluating rules, inferring facts that have not been explicitly stated, explaining why a particular conclusion has been reached, detecting contradictory statements and claims, and combining information from distributed sources in a coherent way. Not all learning types are suitable for geocoding, for example, ideas for using supervised learning have been considered but could be intractable (Section 2.4.3). Reinforcement learning provides another opportunity, where the success of geocoding could be used as the performance critic, and this was the approach used in the prototype. It is the deductive reasoning made possible by logical inference which has the potential to create new knowledge (combining two disparate pieces of information) within the knowledge base. Also useful would be the ability to move from specific values (e.g. “Smith Avenue”) to expressing the same ideas as types (e.g. “road”) as seen in Section 2.4.3, which would enable additional conclusions to be reached via generalisation.

6.1.3 Benefits of including Control, Knowledge and Learning

As presented in Section 5.3.1 benefit of using control, knowledge and learning together is that they are complementary and form a cycle, as seen in Figure 6.1.

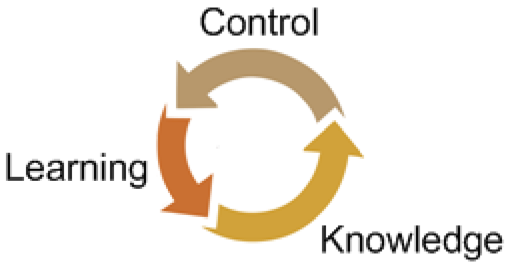


FIGURE 6.1: Complementary flow of control, knowledge and learning

By including the reinforcement approach of learning (using a successful, unique match and geocode as the performance critic) as part of control, learning is provided. This learning creates new knowledge which is stored in the knowledge base. This knowledge is then used by the control component when making future decisions; this demonstrates the transfer of domain knowledge to control knowledge. The implications of this are that a system using this intelligent framework would continue to store knowledge over time and because of this continue to increase its abilities. As more knowledge is autonomously stored, the less human intervention is required and the software control has a richer set of knowledge to draw on.

6.1.4 Unidentified Categories of Problem Addresses

Although identifying new types of problem addresses was not a major focus of this research, an outcome of the research is an indication that there could be additional types of geocoding errors relating to the spatial cognition of users. The compound errors in Sections 2.2.1 and 5.2.5 along with the ontological errors in Sections 2.2.1 and 5.2.3 indicate there are errors in geocoding beyond just those handled by algorithms such as Soundex and Levenshtein. For the semantic and geographic problem address types, all the geocoders tested below 20% (Section 5.2.3), which indicates that this is an area for improvement by the geocoding community. IGL performed well with regards to the iterative and compound errors, which shows the new approach is warranted and also reinforces the question of whether there could be other problem addresses which are unknown. Also of relevance is the fact that improvements in reference data does not solve errors based on human spatial perception/cognition. Similarly, improvements in natural language processing may improve processing of the query string, but do not solve ontological problem addresses.

6.2 Development of the Intelligent Framework

The most significant aspect of the intelligent framework development was the decision to represent the individual address elements as agents (Section 3.2.1). This influenced almost every other aspect of the geocoder, and is essential to the added functionality provided by event-based processing and parallel processing (plus other benefits). Relevant to this is the representation of the relationships between agents as messages, with messages sent throughout the system referring to geographic relationships and content. The goal-based and event-based capability of the framework means that unless there is specialised data required (which had to be stored locally), it is very realistic that a geocoder would be able to rely entirely on the use of other web services and have no data stored locally (Section 3.2.2.1). Agent reuse is another benefit of the framework development, as most agents are very similar except for a few specifics (Section 3.2.2.2).

6.2.1 Updating the Existing Geocoding Process

The intelligent framework uses a design and subsequent paradigm which differs from other approaches in geocoding, and this was done to explore the idea that there was a better way to coordinate geocoding and to embrace the increasingly service oriented nature of the Internet. The iterative processing within the framework is enabled by the multiple foci of control (Section 5.2.4) within the agent paradigm. The messages used in the framework allowed a new approach for geocoding to be tested, instead of the linear flow approach. This enables distributed processing without extra effort, and illustrated that additional types of geocoding errors could be handled. The goal based approach (Section 3.2.2.3) provides flexibility and the ability to specify a goal without explicitly defining how to achieve it. This has positive implications for working with web services in the long term, and in the short term allows context to be utilised for decision making. Real-time correction (Section 3.2.2.5) and notification was enabled by the agent paradigm and presents a new approach. This approach is also extensible, allowing multiple factors (geographic, cultural, abstract) to be added and contribute in real-time. The scoring mechanism used also embraces the parallel approach, and has the capability to work correctly within this new paradigm, using techniques such as the limit for scoring which reaches a natural limit.

6.2.2 Correction Techniques and Reference Data

It has been determined in this research (Section 4.3.5) the order which correction techniques are used in is important (nearest neighbour needs to be used after the element name is correct, for example using Soundex or Levenshtein). Similarly, not every correction technique is suitable (or used in the same way) for every geography type, for example the street element type may not have a first order or second order "neighbour" the same way a locality does. The reasons for this are based on the meaning associated with the address elements and the fact that their fundamental geometry types (point, line and polygon) are different. As more research is done into user cognition in geocoding, it could be discovered that the address element types, the order they are analysed in and their values all play a role in the mistakes users make. Despite a new paradigm being used, no new types of reference data were required, which means there is no additional burden from a data perspective to using the intelligent framework. It is the sum of these benefits (iteration, messaging/parallelism, goals and web service usage) that allow the intelligent framework to pursue the geocoding process *intelligently*. The goals allow the geocoder to pursue goals using the BDI technique from AI, while the knowledge base allows for dynamic storage, modification and inference. The inherent semantics made possible by the agent assignment are also reflected in the knowledge base, where knowledge is stored in terms of the same relationships. With additional research it would be possible for the agent to choose from multiple web services using BDI reasoning which would enhance performance even more.

6.2.3 Role of Semantics and Context

There is inherent semantics within the intelligent framework such that one address element can "contain" another spatially (Section 3.2.2.5), which is made possible by the agent paradigm. In addition to this, two layers exist where context and semantics (between geography) relationships are also represented at the control and knowledge level, made by possible by the agent framework Section 3.2.2.5. Context was included in the intelligent framework and prototype via (i) the different roles of agents, (ii) plan selection inside agents, (iii) messaging between agents, and (iv) the knowledge base. The role of semantics and context is at a higher level conceptually than the control and knowledge used to implement it; it is the semantics and context which deals with the geography and relationships between geographic elements.

6.3 Using Control and Knowledge to Build Intelligence

Control and knowledge builds intelligence because it provides the underlying mechanism that powers the semantics and context mentioned in Section 6.2.3. The benefit of using an agent-based approach is that it enables the two tiers of (i) semantics and context, and (ii) control and knowledge to be *simultaneously* represented, because the agents *are* the geographic elements. Control knowledge is utilised in both the agents and knowledge base in different ways, and can be characterised differently. Knowledge in the agent is stateless (it is not stored between sessions) but the knowledge in the knowledge base is persistent. When the prototype is run for the first time, the knowledge base is empty, but increases over time (because it is re-writable), while the knowledge in the agent remains the same. Also, the knowledge in agents is expressed in terms of the code and context in plans, events and other constructs; the knowledge in the knowledge base is stored a completely different approach. Yet it is these differences which ultimately make them stronger as a combination.

6.3.1 Requirements for Control and Domain Knowledge

Following from the mention in Section 6.3, control knowledge can be stateless, while domain knowledge does need to be stored persistently. Given the distributed nature of the intelligent framework, the ability to track address elements and reassemble them throughout processing is essential. With event-based processing being such a large part of the intelligent framework, it is important to have control data structures which are suited to event-based processing, with a significant part of this ensuring that the structure is temporally and thread safe. In the prototype, the beliefsets (represented internally in the agent as relational tables) used provided this temporally and thread safe capability (Section A.1). The knowledge structure used in the control needs to be the same as that used in the knowledge base, i.e. the same ontological structure; this allows for the same concepts to be expressed in both which ultimately allows transfer of domain knowledge to control knowledge. A relational structure suffices for control knowledge, but more expressive storage is needed for domain knowledge. This more expressive structure provides the ability for inference.

6.3.2 Structuring and Querying Knowledge

From a development point of view, it was useful to abstract out what occurs in the knowledge base, providing simplified software methods for the agent. The "query ID" and "sub ID" used for tracking and aggregating address elements are important to include in the facts used in the knowledge base, as they are used to associate the elements from the various agents and correlate them into one address. The facts used in the knowledge base for indicating that two address elements spatially agree use the same geographic relationships as those in the agents, this reinforces the importance of having similar schemes in both. The rule based system provides the mechanism to "layer" rules upon rules, for geocoding this means atomic spatial agreement pairs (e.g. locality and state spatially agree) can be combined to represent matches at any match success level. Because there are individual match rules for each level of match (levels one through four) it means there is a quality measurement metric already in the rule based system. Because a rule based system is used for the aggregation and matching, there is also direct lineage of what occurred to get the result; this could be shown to a user to explain or justify the processing steps used.

6.3.3 Acquiring New Knowledge

The key to acquiring new knowledge is the ability to store new knowledge dynamically and also the ability for inference. This ability for inference allows for combining multiple disparate pieces of information to solve an otherwise unsolvable query. Although it was not implemented in the prototype, the framework allows for moving from specific instances of address elements (e.g. street name) to the equivalent general concept (e.g. type, such just "street"), this abstraction could be a way of learning new rules that apply to the "general" based on the "specific". Currently the knowledge stored in the knowledge base prototype is derived only from the correction techniques, this demonstrates learning but ideally a future version needs to create knowledge from a mechanism other than those original correction techniques (this would be possible via inference).

6.4 Future Work

The suggestions for future work are varied in terms of where they fit into the geocoding process. Suggestions include modifying the agent to include a rule based system to control the posting of goals, expressing facts in the knowledge base in terms of geography, dynamic storage of web service description information, and modelling errors made by humans due to spatial similarity.

6.4.1 Rule Based System in the Agent

One option is to use the rule based system for control, and not only as the repository of domain knowledge. The beliefs would be seen as the raw beliefs (percepts) about its environment, while the rules in the RBS would be another level on top of the beliefs. As the "raw" beliefs come into the agent, this causes facts to be written in the rule based system - but not everything is asserted (this would be redundant), but rather facts at a slightly higher level; these might be referred to as aggregations. As seen in So and Sonenberg (2004), where rules are used in agents. Dietrich et al. (2003) suggests using semantic web technologies to build rule-based agents.

One concept is that the rules actually provide the agent with a sense of perspective, in the sense that via the rules the agent can “recognize” its own behaviour. For example recognizing that the agent is performing better in locality than another, or that it has processed the same query several times in the last week (which may be unusual). Because the RBS has the backchaining functionality, it can also post goals, which the agent can then pursue and when the goal is finished the result can be posted back into the rule based system - which may in turn make another rule fire. Looking at the strengths of the BDI and rule-based paradigms, it can be seen that the hybrid idea could work. This is because with a regular BDI system the criteria for defining when a goal is pursued is set at design time. By using a RBS within the agent, the rules can be used to launch the goals. However this means over time the rules can change, and the catalyst for posting goals can change. This is analogous to people, where there is a finite set of goals and actions people can do, but there motivations for doing them (and combinations) may change over time.

6.4.2 Reference Data

Datasets commonly used for geocoding could be expressed using formats compatible with semantic reasoning, for example the G-NAF could be stored using RDF triples which would provide additional options for inferencing over the data. Another related question is whether the Open Geospatial Consortium (OGC) could release ontologies for various fields, such as geocoding; this would tie in with the triples.

6.4.3 Geographic Facts Derived from GIS

For an area, street or point of interest, geographic facts could be asserted about the location. These facts would be found via traditional GIS processing functions. This would mean that reasoning could then occur regarding geography. An example is a corner address, where the spatial components of this phenomenon are spatially computed and asserted as facts; a rule could then stipulate that this a corner address and assert another fact accordingly. These geographic relationships would be determined at design time, and would be the geographic computation equivalent to a developed geocoding ontology.

6.4.4 Dynamic Data Source Knowledge

Because web services provide an opportunity to access data that is current, varied and in some cases specialized, aggregating these services for geocoding would be beneficial. If the geocoder could discover these web services (via web service registries) autonomously as they become available, they could be stored as facts in the knowledge base. This would mean the new web services could be added and modified dynamically. Also, if meta-data and keywords were available to describe the web service, these could be stored and rules could be used for selecting suitable web services in the future.

6.4.5 Spatial Similarity in Geocoding

The work by Bruns and Egenhofer (1996) describes the searching of databases using the concept of spatial similarity. In particular, the factors of topology, distance and direction have been described, quantified and used for determining spatial similarity. Further research could determine whether types of factors could be useful in geocoding to describe errors made by humans in their cognition of addresses. Further research could include a correction module for the geocoder which uses these geographic criteria for suggestions. A GIS could be used to perform the spatial queries in real-time.

REFERENCES

Agent-Oriented-Software (2003). Jack agent manual. Available at: www.agent-software.com.

Aldemir, T. (1994). *Reliability and Safety Assessment of Dynamic Process Systems*. Berlin: Springer-Verlag.

Bakshi, R., C. A. Knoblock, and S. Thakkar (2004). Exploiting online sources to accurately geocode addresses. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, Washington, D.C. Association for Computing Machinery.

Barnes, J. (1969). Aristotle's theory of demonstration. *Phronesis 14*(2), 123–152.

Berners-Lee, T., J. Hendler, and O. Lassila (2001). The semantic web. *Scientific American* (284), 34 – 43.

Bigham, J., T. Rice, S. Pande, J. Lee, S. Park, N. Gutierrez, and D. Ragland (2009). Geocoding police collision report data from california: a comprehensive approach. *International Journal of Health Geographics 8*(1), 72.

Biondo, S. J. (1990). *Fundamentals of expert systems technology: principles and concepts*. Norwood, New Jersey: Ablex Publishing Corporation.

Boden, M. A. (1977). *Artificial Intelligence and Natural Man*. New York: Basic Books.

Bonner, M. R., D. Han, J. Nie, P. Rogerson, J. E. Vena, and J. L. Freudenheim (2003). Positional accuracy of geocoded addresses in epidemiologic research. *Epidemiology 14*, 408–412.

Braubach, L., A. Pokahr, and W. Lamersdorf (2004, 9). Jadex: A short overview. In *Main Conference Net.ObjectDays 2004*, pp. 195–207.

Bruns, H. T. and M. J. Egenhofer (1996). Similarity of spatial scenes. In *7 th Symposium on Spatial Data Handling*, pp. 31–42.

Burra, T., M. Jerrett, R. Burnett, and M. Anderson (2002). Conceptual and practical issues in the detection of local disease clusters: A study of mortality in hamilton, ontario. *Canadian Geographer 46*(2), 160–171. cited By (since 1996) 19.

Cayo, M. R. and T. O. Talbot (2003). Positional error in automated geocoding of residential addresses. *International Journal of Health Geographics 2*(10).

Christen, P. and T. Churches (2005). A probabilistic deduplication, record linkage and geocoding system. In *ARC Health Data Mining Workshop*, University of South Australia.

Churches, T., P. Christen, K. Lim, and J. X. Zhu (2002). Preparation of name and address data for record linkage using hidden markov models. *BMC Medical Informatics and Decision Making 2*(9).

Cichelli, R. J. and M. J. Cichelli (1977). Goal directed programming. *SIGPLAN Not. 12*(7), 51–59.

Cohen, W. W., P. Ravikumar, and S. E. Fienberg (2003). A comparison of string distance metrics for name-matching tasks. In *Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico.

Dearwent, S., R. Jacobs, and J. Halbert (2001). Locational uncertainty in georeferencing public health datasets. *Journal of Exposure Analysis & Environmental Epidemiology 11*(4), 329–334.

Deitel, P. J. and H. M. Deitel (2009). *C++ How to Program* (7 ed.). Boston: Prentice Hall.

Dietrich, J., A. Kozlenkov, and M. Schroeder (2003). Rule-based agents for the semantic web.

D'Inverno, M., M. Luck, M. Georgeff, D. Kinny, and M. Wooldridge (2004). The dmars architecture: A specification of the distributed multi-agent reasoning system. *Autonomous Agents and Multi-Agent Systems 9*(1).

Drummond, W. J. (1995). Address matching: Gis technology for mapping human activity patterns. *Journal of the American Planning Association 61*(2), 240–51.

Egenhofer, M. J. (2002). Toward the semantic geospatial web. In *Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems*, McLean, Virginia.

Ferber, J. (1999). *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. New York: Addison Wesley.

Fogel, D. (1995, Nov). Review of computational intelligence: Imitating life [book reviews]. *Proceedings of the IEEE 83*(11), 1588–.

Friedman-Hill, E. (2003). *Jess in Action: Rule Based Systems in Java*. Greenwich: Manning.

Georgeff, M. and A. Rao (1995). Bdi agents: from theory to practice. In *Proceedings of the first International Conference on Multi-Agent Systems (ICMAS-95)*, pp. 312–319.

Goldberg, D., J. Wilson, and C. Knoblock (2007). From text to geographic coordinates: The current state of geocoding. *Journal of the Urban and Regional Information Systems Association 19*(1).

Gruber, T. R. (1992). Ontolingua: A mechanism to support portable ontologies. Technical report.

Gu, L., R. Baxter, D. Vickers, and C. Rainsford (2003). Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences.

Hill, L. L. (2006). *Georeferencing*. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Horstmann, C. S. and G. Cornell (2007). *Core Java Volume 1 - Fundamentals* (8 ed.). Boston: Prentice Hall.

Howden, N., R. Ronnquist, A. Hodgson, and A. Lucas (2001). Jack intelligent agents - summary of an agent infrastructure. In *5th International Conference on Autonomous Agents*, Montreal, Canada.

Ingrand, F. F., M. P. Georgeff, and A. S. Rao (1992). An architecture for real-time reasoning and system control. *IEEE Expert: Intelligent Systems and Their Applications 7*(6), 34–44.

Jacquez, G. and R. Rommel (2009). Local indicators of geocoding accuracy (liga): theory and application. *International Journal of Health Geographics 8*(1), 60.

Jacquez, G. M. and L. Waller (2000). The effect of uncertain locations on disease cluster statistics. In H. T. Mowrer and R. G. Congalton (Eds.), *Quantifying spatial uncertainty in natural resources: Theory and applications for GIS and remote sensing*, pp. 53 – 64. Chelsea, Michigan: Arbor Press.

Jennings, N. R. (1993). Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review 8*(3), 223–250.

Jennings, N. R. (2001, April). An agent-based approach for building complex software systems. *Commun. ACM 44*(4), 35–41.

Jennings, N. R., K. Sycara, and M. Woolridge (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems 1*(1), 7–38.

Karimi, H. and M. Durcik (2004). Evaluation of uncertainties associated with geocoding techniques. *Computer-Aided Civil and Infrastructure Engineering 19*(3).

Kinny, D., M. Georgeff, and A. Rao (1996). A methodology and modelling technique for systems of bdi agents. In *Agents Breaking Away: Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW96)*, pp. 56–71.

Lee, J. (2009, January). Gis-based geocoding methods for area-based addresses and 3d addresses in urban areas. *Environment and Planning B: Planning and Design 36*(1), 86–106.

Li, D., Q. Lei, Z. Ying, L. Ying-wei, W. Xiao-lin, and X. Zhuo-qun (2001). Geo-agents: Design and implementation. *Journal of Wuhan University 6*(1-2), 451–458.

Lovasi, G., J. Weiss, R. Hoskins, E. Whitsel, K. Rice, C. Erickson, and B. Psaty (2007). Comparing a single-stage geocoding method to a multi-stage geocoding method: how much and where do they disagree? *International Journal of Health Geographics 6*(12).

Lutz, M. (2009). *Learning Python* (4 ed.). Sebastopol, California: O'Reilly Media Inc.

Mcguinness, D. L. and F. van Harmelen (2004). *OWL Web Ontology Language Overview*.

Mei, J. and E. P. Bontas (2004). Reasoning paradigms for owl ontologies. Technical report, Institut fur Informatik, Freie Universitat.

Minsky, M. (1974). A framework for representing knowledge. Technical report, Cambridge, MA, USA.

Morris, J. (2008). The role of ontology in modern expert systems development. In *October Rules Fest*, Dallas, Texas.

Myers, D. G. (2004). *Psychology*. New York: Worth Publishers.

Negnevitsky, M. (2002). *Artificial Intelligence - A Guide to Intelligent Systems*. Harlow, England: Pearson Education Limited.

Newell, A. (1982). The knowledge level. *Artificial Intelligence 18*, 87–127.

Nicoara, G. (2005). *Exploring the Geocoding Process: A Municipal Case Study using Crime Data, Master's Thesis*. Ph. D. thesis, The University of Texas at Dallas.

Noy, N. F. and D. L. McGuinness (2001, March). Ontology development 101: A guide to creating your first ontology. Technical Report Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford University.

Ordinance-Survey (2003). Address-point user guide. Technical report.

Padgham, L. and M. Winikoff (2002). Prometheus: a methodology for developing intelligent agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, pp. 37–38. ACM.

Padgham, L. and M. Winikoff (2004). *Developing Intelligent Agent Systems: A Practical Guide*. West Sussex, England: John Wiley & Sons Ltd.

Passin, T. B. (2004). *Explorer's Guide to the Semantic Web*. Greenwich, CT: Manning Publications.

Patman, F. and L. Shaefer (2001). Is soundex good enough for you? on the hidden risks of soundex-based name searching. Technical report, Language Analysis Systems, Inc.

Pierce, B. C. (2002). *Types and programming languages*. Cambridge, MA, USA: MIT Press.

Pressman, R. S. (2004). *Software Engineering: A Practitioner's Approach*. New York: McGraw-Hill.

Ratcliffe, J. (2001). On the accuracy of tiger-type geocoded address data in relation to cadastral and census areal units. *International Journal of Geographical Information Science 15*(5), 473–485.

Richards, P. and D. Paull (2003). A geocoded national address file (g-naf) for australia: The g-naf how. Pdf, Public Sector Mapping Agencies (PSMA) Australia Limited.

Rushton, G., M. P. Armstrong, J. Gittler, B. R. Greene, C. E. Pavlik, M. M. West, and D. L. Zimmerman (2006). Geocoding in cancer research. *American journal of Preventative Medicine 30*, S16–S24.

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Upper Saddle River: Prentice Hall.

Schootman, M., D. A. Sterling, J. Struthers, Y. Yan, T. Laboube, B. Emo, and G. Higgs (2007). Positional accuracy and geographic bias of four methods of geocoding in epidemiologic research. *Annals of Epidemiology 17*(6), 464 – 470.

Shahriari, N. and C. V. Tao (2002). Gis applications using agent technology. In *Symposium on Geospatial Theory, Processing and Applications*, Ottawa, Canada.

Shyllon, E., B. Veenendaal, and M. Hutchinson (2007). Building knowledge from address information for intelligent geocoding. In *Spatial Sciences Institute International Biennial Conference 2007*, Hobart, Tasmania.

Sirin, E., B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz (2007). Pellet: A practical owl-dl reasoner. *Web Semant. 5*(2), 51–53.

So, R. and L. Sonenberg (2004). Situation awareness in intelligent agents: Foundations for a theory of proactive agent behavior. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04)*.

Stanier, A. (1990). How accurate is soundex matching. *Computers in Genealogy 3*(7), 286 – 288.

Sterling, L. and E. Shapiro (1986). *The art of Prolog: advanced programming techniques*. Cambridge, MA, USA: MIT Press.

Sterling, L. and E. Shapiro (1994). *The Art of Prolog* (2 ed.). Cambridge: The MIT Press.

Strickland, M. J., C. Siffel, B. R. Gardner, A. K. Berzen, and A. Correa (2007). Quantifying geocode location error using gis methods. *Environmental Health 6*(10).

Tang, A. and K. Clark (2003). *ArcGIS 9 Geocoding Rule Base Developer Guide*. Redlands, CA: ESRI.

Tecuci, G. (1998). *Building Intelligent Agents: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies*. San Diego, California: Academic Press.

Tsou, M.-H. and B. P. Buttenfield (1998). An agent-based, global user interface for distributed geographic information services. In *7th International Symposium on Spatial Data Handling*, Vancouver, British Columbia. Taylor and Francis.

Uschold, M. (1998). Knowledge level modelling: Concepts and terminology. *The Knowledge Engineering Review 13*(1), 5–29.

Van Rees, R. (2003). Clarity in the usage of the terms ontology, taxonomy and classification (paper w78-2003-432). In *Construction Informatics Digital Library*. Ljubljana University (http://itc.scix.net/).

Waller, L. (1996). Statistical power and design of focused clustering studies. *Statistics in Medicine 15*(7-9), 765–782. cited By (since 1996) 19.

Wang, H. H., N. Noy, A. Rector, M. Musen, T. Redmond, D. Rubin, S. Tu, and T. Tudorache (2007). Frames and owl side by side. In *10th International Protege Conference*, Budapest, Hungary.

Ward, M., J. Nuckols, J. Giglierano, M. Bonner, C. Wolter, M. Airola, W. Mix, J. Colt, and P. Hartge (2005). Positional accuracy of two methods of geocoding. *Epidemiology 16*(4), 542–547. cited By (since 1996) 33.

Wei, R., X. Zhang, L. Ding, H. Ma, and Q. Li (2009, October). A knowledge-based agent prototype for Chinese address geocoding. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Volume 7146 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*.

Welty, C., F. Lehmann, G. Gruninger, and M. Uschold (1999). Ontology: Expert systems all over again? invited panel at aaai-99. In *The National Conference on Artificial Intelligence*, Austin, Texas.

Whitsel, E., K. Rose, J. Wood, A. Henley, D. Liao, and G. Heiss (2004). Accuracy and repeatability of commercial geocoding. *Practice of Epidemiology 160*(10).

Whitsel, E. A., P. M. Quibrera, R. L. Smith, D. J. Catellier, D. Liao, A. C. Henley, and G. Heiss (2006, July). Accuracy of commercial geocoding: assessment and implications. *Epidemiologic Perspectives & Innovations 3*, 8+.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. Chichester, U.K: John Wiley and Sons.

Wooldridge, M. and N. R. Jennings (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review 10*(2), 115–152.

Zandbergen, P. (2007). Influence of geocoding quality on environmental exposure assessment of children living near high traffic roads. *BMC Public Health 7*(1), 37.

Zandbergen, P. and J. Green (2007). Error and bias in determining exposure potential of children at school locations using proximity-based gis techniques. *Environmental Health Perspectives 115*(9), 1363–1370. cited By (since 1996) 12.

Zandbergen, P. A. (2008). A comparison of address point, parcel and street geocoding techniques. *Computers, Environment and Urban Systems 32*(3), 214–232.

Zhan, F., J. Brender, I. DE Lima, L. Suarez, and P. Langlois (2006). Match rate and positional accuracy of two geocoding methods for epidemiologic research. *Annals of Epidemiology 16*(11), 842–849. cited By (since 1996) 11.

Zhao, H. (2007). Semantic matching across heterogeneous data sources. *Communications of the ACM 50*(1).

Zimmerman, D. L., X. Fang, S. Mazumdar, and G. Rushton (2007, January). Modeling the probability distribution of positional errors incurred by residential address geocoding. *International Journal of Health Geographics 6*, 1+.

Zobel, J. and P. Dart (1996). Phonetic string matching: lessons from information retrieval. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, pp. 166–172. ACM.

Every reasonable effort has been made to acknowledge the owners of copyright material. I would be pleased to hear from any copyright owner who has been omitted or incorrectly acknowledged.

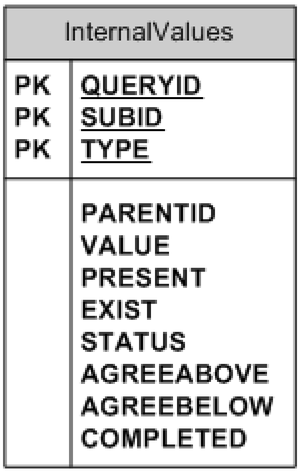
APPENDIX A

ALGORITHMS

A.1 Beliefsets

Design of the beliefsets is based on the goals specified in Section 3.2.2.3. Specifically, the goals of *existence*, *complementary elements*, *agreement of complements*, *suggestions* and *equivalent complements* each have their own requirements in terms of the attributes needed to store their data. It has been mentioned that Prometheus is an iterative process, and the design of beliefsets reflects this. The first "draft" of beliefset design assists with agent selection, and can be improved later. The beliefset design is presented below, in terms of the goal types. All of the beliefsets below are presented in their final form, and in finalising their design, feedback was taken from later steps in the Prometheus methodology.

Existence The main criteria for building a beliefset to work with this goal is the ability to store whether an element is present and exists. These fields can be seen in Figure A.1.

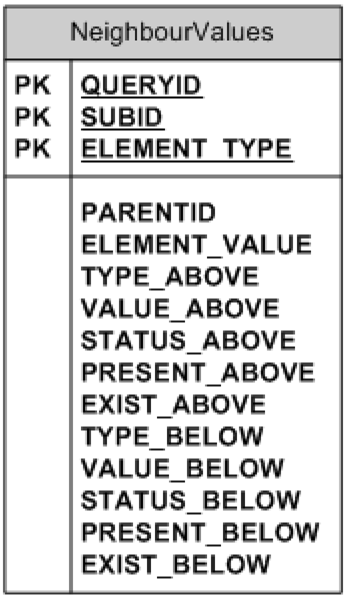


	InternalValues
PK	QUERYID
PK	SUBID
PK	TYPE
	PARENTID
	VALUE
	PRESENT
	EXIST
	STATUS
	AGREEABOVE
	AGREEBELOW
	COMPLETED

FIGURE A.1: Relational Table Design for ”InternalValues” Beliefset

The possible values for the present and existing fields include true, false, and unknown. There is an important difference between “false” and “unknown”; with existence for example, false means *no it does not exist*, which is quite different from *it has not been verified yet*. The other fields will be explained in following sections. Also in Figure A.1 are the “Query ID”, “Sub ID” and “Type” fields; these are all primary keys for the beliefset. These key fields are found in almost every beliefset used in the geocoder, as it provides a way to uniquely identify and track any address element throughout the system.

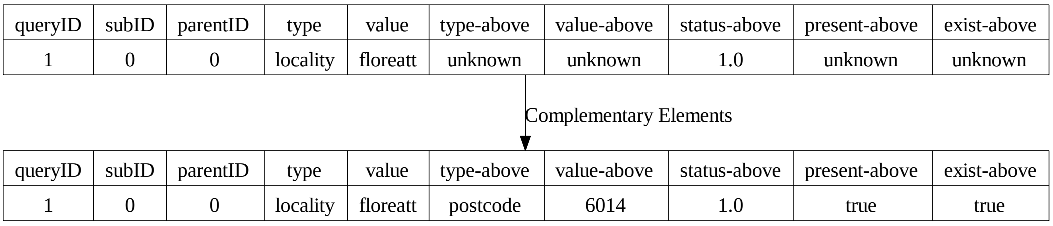
Complementary Elements To clarify, the use of the word *neighbour* in this beliefset name refers to the given element’s complementary elements. Likewise, the terms *above* and *below* refer to the complementary elements. Although this wording could have been changed for presentation in this thesis, the original terms were used due to complexities and inter-relationships in the actual design. The information about the complementary elements is stored in the “Neighbour Values” beliefset, seen in Figure A.2.



NeighbourValues	
PK	QUERYID
PK	SUBID
PK	ELEMENT_TYPE
	PARENTID ELEMENT_VALUE TYPE_ABOVE VALUE_ABOVE STATUS_ABOVE PRESENT_ABOVE EXIST_ABOVE TYPE_BELOW VALUE_BELOW STATUS_BELOW PRESENT_BELOW EXIST_BELOW

FIGURE A.2: “Neighbour Values” Beliefset

The type and value for each complementary element is stored, and the potential values for these attributes are either true, false or unknown. The other attributes are used for calculating a status score (discussed in Section A.5). Figure A.3 shows how at the beginning of processing the type and value of a complementary element may be unknown, but once the information is found, it can be updated.



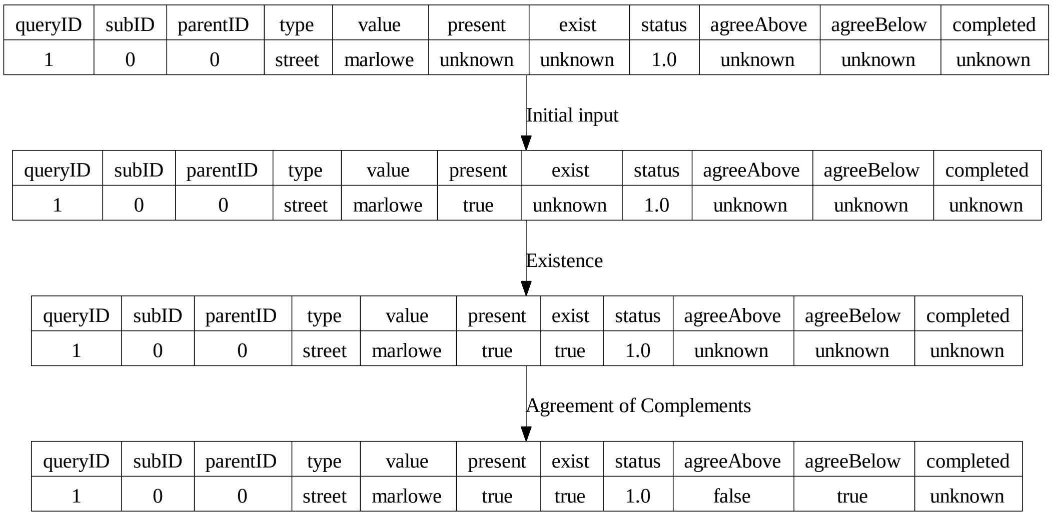
queryID	subID	parentID	type	value	type-above	value-above	status-above	present-above	exist-above
1	0	0	locality	floreatt	unknown	unknown	1.0	unknown	unknown

queryID	subID	parentID	type	value	type-above	value-above	status-above	present-above	exist-above
1	0	0	locality	floreatt	postcode	6014	1.0	true	true

FIGURE A.3: “Neighbour Values” Beliefset being Updated (upper portion)

The example in Figure A.3 shows the type and value of the complementary element being set from unknown to postcode and 6014.

Agreement of Complements The storage of whether an element agrees with either of its complements is in the InternalValues beliefset, already shown in Figure A.1. From a design point of view, all of the information relating to the particular element type being analyzed is put in the InternalValues beliefset, while information in regards to "other" address elements (from the perspective of the element being analyzed) is stored in the NeighbourValues beliefset. An example of how a record in the NeighbourValues beliefset can update is shown in Figure A.4.



queryID	subID	parentID	type	value	present	exist	status	agreeAbove	agreeBelow	completed
1	0	0	street	marlowe	unknown	unknown	1.0	unknown	unknown	unknown

Initial input

queryID	subID	parentID	type	value	present	exist	status	agreeAbove	agreeBelow	completed
1	0	0	street	marlowe	true	unknown	1.0	unknown	unknown	unknown

Existence

queryID	subID	parentID	type	value	present	exist	status	agreeAbove	agreeBelow	completed
1	0	0	street	marlowe	true	true	1.0	unknown	unknown	unknown

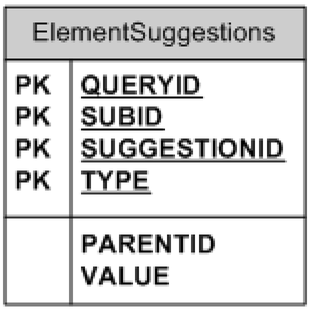
Agreement of Complements

queryID	subID	parentID	type	value	present	exist	status	agreeAbove	agreeBelow	completed
1	0	0	street	marlowe	true	true	1.0	false	true	unknown

FIGURE A.4: Agreement of Complements Information being Updated

Note how in Figure A.4 the final agreement is stored as true and false, while at the start of a new geocode query this processing has not been done and is initially set as unknown.

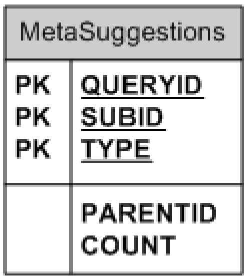
Suggestions A beliefset is needed to store all the suggestions arising from the suggestions goal. The requirements are similar to the other beliefsets, except that an additional primary key field is needed (SUGGESTIONID) to uniquely identify suggestions; this is because there can be multiple suggestions of the same type. The value of the suggestion is a regular attribute. The beliefset design purpose is named ElementSuggestions, and is shown in Figure A.5.



ElementSuggestions	
PK	QUERYID
PK	SUBID
PK	SUGGESTIONID
PK	TYPE
	PARENTID
	VALUE

FIGURE A.5: “Element Suggestions” Beliefset

In addition to storing the actual suggestions themselves, it was also anticipated that another beliefset may be needed to store information about the suggestions. The design started with a count of the suggestions, and the design stayed this way. In the future, other attributes cold be added to store statistics or other information used to examine performance. The MetaSuggestions beliefset is shown in Figure A.6.



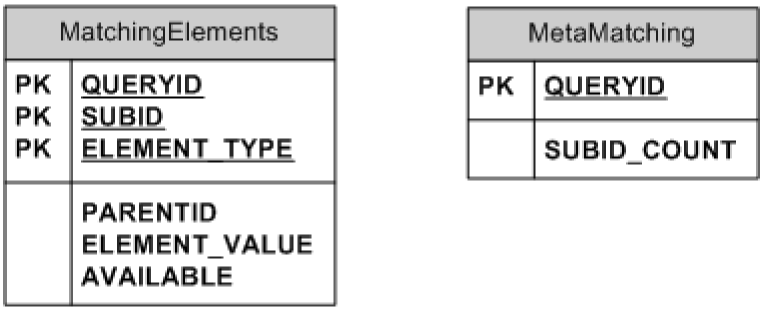
MetaSuggestions	
PK	QUERYID
PK	SUBID
PK	TYPE
	PARENTID
	COUNT

FIGURE A.6: “Meta Suggestions” Beliefset

Together, the ElementSuggestions and MetaSuggestions provide the information needed to find the equivalent complements for a given element.

Equivalent Complements There is actually no beliefset created specifically for the equivalent complements. The details why will be covered in following sections, but it is largely because it is the last goal, and as long as it can pass its results onto the next step in the geocoding process, it does not need persistence.

Matching and Scoring It was anticipated that beliefsets would be needed for storing the elements used in the matching process, storing any explanatory information, and also the weights used in calculating a status score. Figure A.7 shows the beliefsets used in matching.

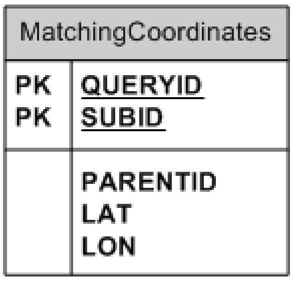


MatchingElements	
PK	QUERYID
PK	SUBID
PK	ELEMENT_TYPE
	PARENTID
	ELEMENT_VALUE
	AVAILABLE

MetaMatching	
PK	QUERYID
	SUBID_COUNT

FIGURE A.7: "Matching Elements" and "Meta Matching" Beliefsets

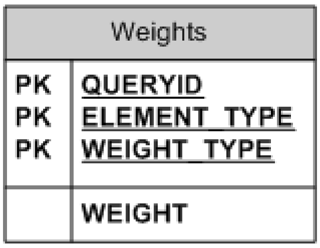
The attribute type "available" in the MatchingElements beliefset provides the ability to store an address element in the beliefset but not use it yet (used for when a brand new query enters the system). It was also anticipated that a beliefset would be needed to store the results of the final stage in geocoding - obtaining the coordinates. The beliefset created for this, MatchingCoordinates, is presented in Figure A.8.



MatchingCoordinates	
PK	QUERYID
PK	SUBID
	PARENTID
	LAT
	LON

FIGURE A.8: "Matching Coordinates" Beliefset

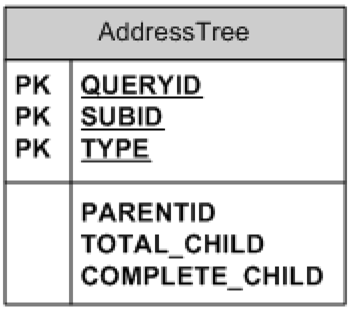
Weightings are used to calculate the status score, where each different element type that contributes to the score has its own weighting in relation to the element type having its score calculated. Instead of having these weightings hard coded, the weightings are set at runtime and stored in a beliefset. The structure allows for different element types to have different weightings relative to other element types. For example, the `street type` may have a higher weighting than `state` when calculating the score for the `street element`. The beliefset for storing these weights is seen in Figure A.9.



Weights	
PK	QUERYID
PK	ELEMENT_TYPE
PK	WEIGHT_TYPE
	WEIGHT

FIGURE A.9: “Weights’ Beliefset

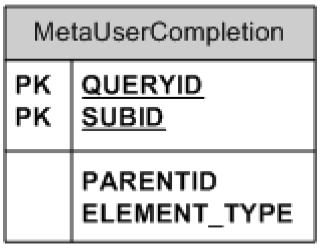
Coordination It was recognized that coordination and tracking would be an issue with so many addresses being processed, and also the need to determine when processing is completed. To assist with this, the beliefsets AddressTree (Figure A.10) and MetaUserCompletion (Figure A.11) were designed.



AddressTree	
PK	QUERYID
PK	SUBID
PK	TYPE
	PARENTID TOTAL_CHILD COMPLETE_CHILD

FIGURE A.10: “Address Tree” Beliefset

The AddressTree beliefset provides a mechanism to track the number of suggestions that have been created from a given element value, and whether the “child” element itself is done with processing.



MetaUserCompletion	
PK	QUERYID
PK	SUBID
	PARENTID ELEMENT_TYPE

FIGURE A.11: “Meta Completion” Beliefset

`MetaUserCompletion` is a beliefset which gets updated when a particular element is completely finished processing.

A.2 Address Element Tracking

There is a concept of “parent” and “child” addresses, and more specifically, child elements. The original address is considered to be the initial parent, or root node, of the system. During geocoding, several iterations can occur depending on how incorrect the original address is. Suggestions are made for each element that does not exist, or exists but does not spatially agree with its complements; each of these suggestions adds to the number of possible addresses that could be the “corrected” version of the original query. The iteration within the system stops after three child “branches” have been pursued, or when a complete match is found. It was the opinion of the researcher that beyond three branches, there may be a risk of straying too far from the original query element.

Each suggestion that emerges from the matching and is kept becomes a child node. Each element within the original address is essentially the parent node of its own tree; and the parent node along with each of the child nodes have corresponding, complementary elements on the other trees.

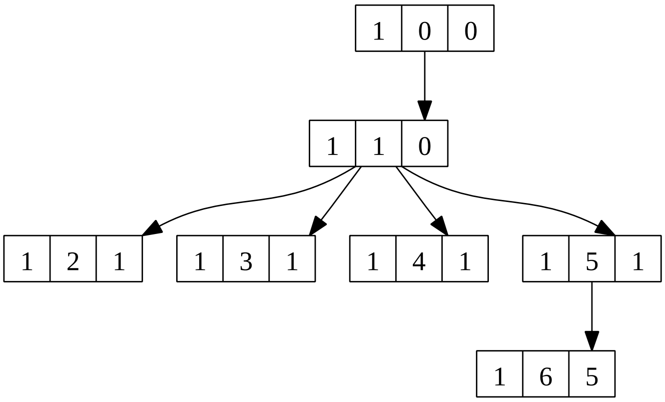


FIGURE A.12: Unique Identifiers for Each Child Node

Figure A.12 shows that each element suggestion has three unique integers which together distinctly identify the element. These three integers, from left to right, are the *query-ID*, *sub-ID* and *parent-ID*. The query-ID is the same for every element contained/generated from the same original query submitted by the user; in Figure A.12 this value is 1. The sub-ID is unique for every element, this simply increments for every additional element suggestion that is added; in Figure A.12 this value ranges from 0 (in the root element) to 6 (the final element or third generation). The parent-ID is the sub-ID of the element "above" (i.e. its parent) any given element.

As described in the Appendix, the FindBestValue goal has several criteria for success, one of these criteria utilizes this concept of parent and child nodes. The goal succeeds when a parent element has more than zero child elements, and the number of total child elements equals the number of completed child elements. As a design decision, no more than three "generations" of child element suggestions are allowed to ensure the suggestions are not too far removed from the intended value inputted by the user. This means that processing would stop when an element writes to a beliefset a `parentID` of 2. Of the different criteria for termination, it is the first to occur which causes termination. In Aldemir (1994), Friedmann Mattern's idea of "sticky state indicators" is provided as a solution for this distributed termination detection problem which useful for termination in agent-based geocoding.

This tree structure and the FindBestValue is recursive, in two ways. The first is that suggestions originate from suggestions, and this continues for three generations. As a result of this, the second form of recursion is the that the goal FindBestValue will be nested. This is because a parent element will not have completed its goal until its child element has, and so on; once the child element has finished, the parent is free to complete their goal.

The steps after the posting of FindBestValue are evaluated every time the FindBestGoal is posted, although not every sub-goal will necessarily be pursued.

This capability for tracking address elements throughout the system, and knowing which elements are the parents or children makes possible the concept of iteration. Iterations allows the cycle of goals to be used several times. Each of the goals are visited one after another, which at first inspection seems to be the same as the linear geocoding process used in current geocoders, but the *whole sequence of goals* can repeated if needed. The rationale behind this is that some addresses may need several transformations in order to be fully corrected. For example, a user may have a locality confused with its neighbour *and* may also spell that neighbour incorrectly. It is expected that in the future that the use of in-depth semantics in geocoding will further validate this concept. If the whole geocoding process can be thought of a "pipe", the iteration essentially takes the first set of results and drops these back into the pipe.

Figure A.12 is both a data structure and a process, and conceptually is applicable to both whole addresses and individual elements. In other words each node shown in Figure A.12 could represent an address or specific element (*e.g.* a locality or postcode). If used for a whole address, then only one tree would be needed, as presented in Figure A.12, but if used for individual elements, there would be many trees used in conjunction with one another. Any number of suggestions could be found for a given element, which means there can be more suggestions for a particular element type than another. Root nodes will always have the exact same IDs; for the leaf nodes though, it depends how the subIDs are assigned. If issued indiscriminately then there would be no commonality of numbering between the different trees.

Alternately, a mechanism could be used to coordinate the distribution of subIDs such that the element with a particular subID *in one tree* belonged to the same address as an element with the same subID *in another tree*. This option would mean that if the elements contained in all trees needed to be reassembled, a query to gather all the elements with the same subID would belong to the same address.

A.3 Address Element Reconstruction

It was shown how the goals constituted a large part of the overall system operation. Each element in the address, unless it exists to begin with and also spatially agrees with its complementary elements, will move through each of these steps. The last goals in the sequence were *Suggestions* and *Equivalent Complements*, and after these steps have concluded, the result is several lists containing pairs of the form [suggestion, equivalent] created by each of the agents, operating in parallel.

It is in this section that the overall algorithm resumes, and the following five steps are given consideration:

1. Element candidates and adjacents are sent to the matching agent with the queryID, subID and parentID of the element value for which the

suggestions and adjacents were created from.

2. Suggestions and adjacents are processed in rule based system, and none, one or many matches may be found.

3. The matching agent maintains a counter to store which subID integers can be used.

4. Each match is given a unique subID and the counter is incremented accordingly, suggestion elements which do not match are also given a subID. The respective parentID is also attached. It is necessary to give subIDs to the elements which do not match, because in the worse case scenario they will all be sent back as suggestions and therefore need a subID.

5. Each of the elements in the rule based system are sent back to their respective agents, and written into the InternalValues beliefset using the queryID, subID and parentID provided by the matching agent. Included in the messages back to the element agents is the total number of elements being sent (this becomes the number of children number for the parent element). From a contextual point of view, if the user wants feedback during geocoding, the MatchingAgent would send the matches it finds in the RBS to the UserAgent for display to the user.

This reconstruction algorithm provides a coordinated numbering of leaf nodes. Because it is the MatchingAgent which generates, assigns and distributes the subIDs, it possible to ensure each element belonging to a particular address each receive the same number.

To demonstrate what is happening, consider the lists created for several elements, where the known element is in regular font, and the suggestion is in italics. This would be [street name, *locality*], [locality, *postcode*], [postcode, *state*] and also [state]. Note that state does not have an equivalent element suggestion, as it is at the “end” of the elements; in terms of its agreement with other elements, the state has only one complementary element.

If these lists of pairs are grouped together it can be seen that there is “overlap” between the suggestion of one pair, and the equivalent of another. Using this overlap means the lists can be chained together, and if enough of these pairs are chained together then a complete address will be formed. Even if a complete address is *not* found, addresses comprised of varying numbers of elements will be found. In order to reconstruct and analyze these lists of suggestions and equivalents, a rule-based system is used.

Within the rule based system, each of the suggestions is asserted as a fact, of type `potential-element`. This fact has the slots `query-id`, `type` and `value`. The slot `query-id` is used only to keep track of the various facts in the system, so that facts from different queries do not interfere with each other. The `type` slot is used to identify whether the suggestion is a state, postcode or street name *etc*. The `value` is the actual character string representing the postcode, street name etc. In Figure A.13 it can be seen that potential elements include a state ("WA"), postcode ("6014"), locality ("Wembley") and street ("Marlow").

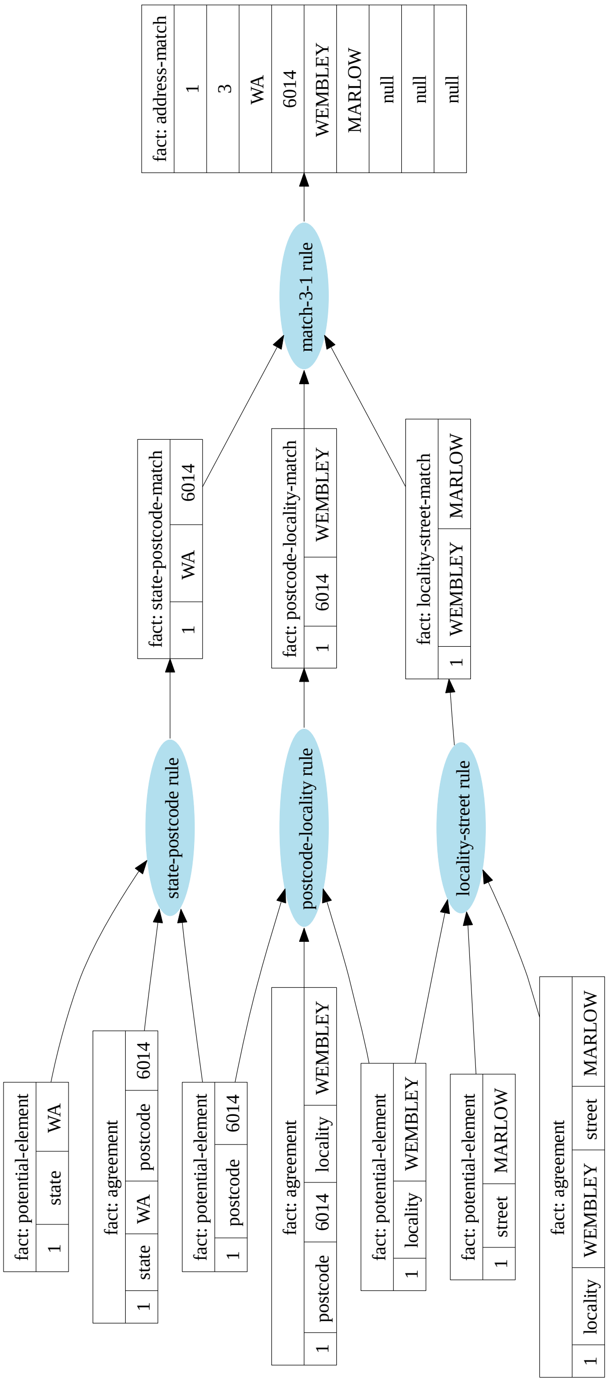


FIGURE A.13: Rule Based Address Reconstruction

To represent the relationship between the suggestions and their equivalent elements "to the left", another fact template, `agreement`, is used. This template has the slots `query-id`, `of-type`, `of-value`, `with-type` and `with-value`. The `of-type` and `of-value` slots describe the suggestion, and the `with-type` and `with-value` slots describe the element type which is equivalent to the suggestion. In Figure A.13, agreements can found between state and postcode, postcode and locality, and locality and street. The example shows how for each of these agreements, the two `potential-element` facts are mentioned in the `agreement` fact, which ultimately means the two elements match with each other (resulting in, for example, the `state-postcode-match fact`). Even if there are two potential elements, there is no match without the `agreement` fact.

The way rules are used in the reconstruction process can be thought of as "finding pairs", and then "groups of pairs". The rules are used for matching and consolidating facts, as seen by the three tiers of facts which are processed in two stages. The first stage rules (such as `state-postcode`, `postcode-locality` and `locality-street`) bring together the otherwise disparate fact types of `potential-element` and `agreement` and explicitly define any pair matches found as facts (*e.g.* `state-postcode-match` and `locality-street-match`).

These facts representing matches of pairs are then further combined (if they match) and unified into a single fact representing a complete address. This unification can be seen in Figure A.13, where the `address-match` fact contains values for the state, postcode, locality and street name; the values for street type, street number and unit number remain *null* in this case, as no information (in the form of facts) was matched for these. The name of the rule `match-3-1` has meaning; the first number (3) indicates it is a rule used for matching three facts, and the second number (1) is a unique identifier (i.e. this naming convention allows for many rules which match three facts). It should be noted that the first two slots in the `address-match` template are `query-id` and `match-value`; the latter is used to store the number of pairs combined to populate the `address-match` fact, this number also serves as an initial quality indicator.

The address represented by `address-match` in Figure A.13 is one of the potentially many addresses. This address is also a "first generation" child of the original address (root node). Only the address matches with the highest quality values are "kept" for subsequent reiteration. For example, if there were many `address-match` facts with quality scores of 3, 2 and 1 - only those with quality 3 are kept; at this overall stage of the geocoding process, no weighting is given to the various element types that may comprise that address. Unless this child address is a perfect match, it will be resubmitted back into the system, and follow the same process that the original address (its parent) did when it was first submitted.

A.4 Reiteration of Address Elements

In the processing algorithm, the initial query passed through a series of goals, where various assessments were made, status scores were calculated and where necessary suggestions were found and messages were sent to the matching agent with the information needed for matching. Matching then occurs and this leads to the next steps, including:

1. The elements which are sent from the `MatchingAgent` back to the element agents are added to the respective InternalValue beliefsets, using the queryID, subID and parentID supplied by the matching agent.
2. Adding the elements from the `MatchingAgent` is (mostly) treated no differently from when the *original* element entered the system and was itself added to the `InternalValues` beliefset. This addition to the beliefset begins the process of pursuing goals that are relevant.
3. Any elements being sent from the `MatchingAgent` already "exist", so the first goal *Existence* is not needed. The other goals may all be suitable.
4. The *Complementary Elements* and *Agreement of Complements* goals are pursued for the agent, and results are written to beliefs as usual.
5. As usual, any time during pursuing goals, status scores are recalculated if needed.
6. If needed, the Suggestions plan is activated and the usual process follows where suggestions and equivalent complements are found and sent to the matching agent. The queryID, subID and parentID reflect that these originated from next generation elements.
7. Overall processing ends when elements reach the third generation is reached or a match at a given quality level is found.

One of the outcomes from the tracking system, messaging and beliefset design is that throughout the various `InternalValues` beliefsets contained in the element agents, each of the element types has corresponding equivalents in other agents; this means in each beliefset within the different element agents are elements each with the same `queryID` and more importantly the same `subID`. The best way to think of this is if a "join" (the database definition) were to be performed across the different beliefsets using the `queryID` and `subID`, then a complete address would be the result.

When an agent is determining its *complementary elements* and *agreement of complements*, it will request information from its complementary elements; this request will include the queryID, subID and parentID of the desired element. If the agent *being asked* has no tuple with that queryID and subID then it adds a corresponding tuple and for the value uses the "default" value of its parent (determined using the parentID).

When finding suggestions for an element (*Suggestions* goal), given two complementary elements, if one element exists and the other does not then suggestions are only found for the element that does not exist. If both the elements existed, but did not agree spatially, the suggestions would be found for *both* elements.

The nature of the system means that there can be many combinations of potential elements forming many potential addresses. However with the tracking techniques used, this is not a problem.

Only certain correction techniques can be used at various generations of suggestions, also there are restricted orders in which the techniques can be used. Restrictions include that, for example, the finding of neighbouring localities can only be used if the original locality exists. Also, the opinion of the author is that a technique cannot be used twice - soundex used twice would provide erroneous suggestions. The knowledge base can be used at any generation. A selection of valid combinations of correction includes Levenshtein-Soundex-Geo (where "Geo" is neighbouring geographic localities), Soundex-KB, Levenshtein-Geo and Geo-KB.

The whole process is a cycle, with the bulk of the processing done by the various element agents, with the matching agent providing an essential integration mechanism followed by subsequent redistribution. The tree of element suggestions provides a novel (and exhaustive) technique to look for potential address matches.

Figure **??** shows how elements from the Matching Agent are sent back to the ElementAgents and added to the InternalValues beliefset, in a way very similar to a new address element (from an entirely new geocode query). By using this approach, it means that the same process can be used for either a new address element or an element being used as part of an iteration.

A.5 Quality Scores

The focus of quality in the agent system is on the quality of the address results, specifically to what extent the address elements agree with each other and the overall spatial agreement of the address. Because suggestions are found in separate agents at element level, and so the "inward looking" scoring scheme is relevant and valid because when a suggestion is found, not all other element types are automatically modified to reflect or agree with the suggestion value. Using the agent-based quality means that a measure is provided to the user so they understand the quality of the geocode they are using, and has a focus on the address itself.

The quality score for an individual element in the address has a value between zero and one. The score is a measure of how well an element "fits" with the address it is in. A value of zero (0) indicates no agreement, while one (1.0) indicates that the element agrees completely with its neighbouring elements. Combining these individual scores together can also provide an overall measure of quality for the address.

The individual score is calculated by using several factors, including whether the element value is present, exists, and agrees with both its upper and lower neighbours (where applicable). Each of these factors is also given a numerical weighting, which reflects the relative importance of the factor. For example, when calculating the score for street name, it would be considered much more important for the street name to agree with the locality than with the street type; this is reflected by giving locality a larger weighting in the calculation. Each particular element being calculated can have different weightings for the same element type. Also taken into account is the current score of the element's complementary elements. This means that the current status score of one element will affect the score of the elements that rely on it, because of this the element scoring process is interdependent and gives an inherent measure of the overall address quality. The calculation be seen in Equation A.1, where w is the weight and s is the score.

$$\text{score} = \frac{\mathrm{w}_{present} + \mathrm{w}_{exist} + (\mathrm{w}_{upper} \times \mathrm{s}_{upper}) + (\mathrm{w}_{lower} \times \mathrm{s}_{lower})}{\mathrm{w}_{present} + \mathrm{w}_{exist} + \mathrm{w}_{upper} + \mathrm{w}_{lower}} \tag{A.1}$$

When calculating the numerator, the weight for *present* and *exist* are only added if the element is present and exists. When calculating the denominator, the weights for being present, existing and agreeing with the neighbours is added regardless. Ultimately, this means that the status score for an element is penalised via the numerator. When a new query is submitted, each element is given an initial quality of 1.0. This can be thought of as providing the element with the "benefit of the doubt" regarding its score; not until proven otherwise is the element penalised.

A score for the overall address can be calculated by combining these individual address elements. This formula can be seen in Equation A.2, where n is the number of elements submitted in the original query, and m is the number of elements in the final, matched address result for which a geocode is returned to the user; when calculating the overall score before processing has *completely* finished, m is the number of elements which are present, exist and spatially agree with their complementary elements. The score of the element is the same score calculated for individual elements in Equation A.1. It can be seen in Equation A.2 that the score is penalised if the number of elements in the final matched address is less than the number originally submitted. This penalty is applied because the resulting address contains less information than the original. This penalty is useful to quantify the fact that although a result address may have a better score than the original submitted address, its geocode may have a reduced resolution than the original query.

$$\text{score} = \frac{m}{n} \times \frac{\sum_{i=1}^{m}(\mathrm{w}_{\text{element } i} \times \mathrm{s}_{\text{element } i})}{\sum_{i=1}^{n} \mathrm{w}_{\text{element } i}} \tag{A.2}$$

Each element used in Equation A.2 has a weighting which denotes its importance to the overall address. For example, the *postcode* can be configured to be more important in determining a geocode than a *street type*.

The linking algorithm used to find address matches was presented in Section 4.9.2, and this algorithm also provides a basic source of quality information. The matching agent ranks the results it obtains using the linking algorithm, in order to pass on only the best matches to the respective specialty agents. When the linking algorithm is used, the number of pairs that agree spatially is recorded. For example, an address that has the state, postcode, locality, street name and street type in agreement would have 4 *pairs* in agreement.

As well as providing the user with an indication of how reliable their results are, the quality measures are also important in internal system processing. When multiple address matches are found, regardless of whether one or many results are expected, these matches need to be sorted. Some sorting occurs within the matching agent (using the “pair score”), and after this further sorting occurs at the specialty agents, where addresses considered to be of “equal quality” by the matching agent can be sorted based on the whole address score calculated for each. For example, although there could be two matched addresses returned by the matching agent, each with three pairs that spatially agreed, each address could have different sets of pairs. Because of the weightings associated with the different element types, this means that one of the addresses could have a higher whole-address score calculated for it and subsequently considered a better result.