Spatially Motivated Dialogue for a Pedestrian Robot

Jamie Frost
Department of Computer Science
Oxford University

October 30, 2012

Abstract

In the field of robotics, there has recently been tremendous progress in the development of autonomous robots that offer various services to their users. Most of the systems developed so far, however, are restricted to indoor scenarios, non-urban outdoor environments, or road usage with cars. There is a serious lack of capabilities of mobile robots to navigate safely in highly populated outdoor environments. This ability, however, is a key competence for a series of robotic applications. We consider the task of developing a spatially motivated dialogue system that can operate on a robotic platform, where the purpose of such a robot is to aid pedestrians in urban environments to provide information about surrounding objects and services, and guide users to desired destinations.

In this thesis, we make a number of contributions to the fields of spatial language interpretation/generation and discourse modelling. This includes the development of a dialogue framework called *HURDLE* which builds on the strengths of existing systems, accompanied by a specific implementation for spatially oriented dialogue including disambiguating amongst objects and locations in the environment, and a natural language parser which combines an extension of Synchronous Context Free Grammars with a Part-of-Speech tagger. Our research also presents a number of probabilistic models for spatial prepositions such as 'in front of' and 'between' that make significant advances in effectively utilising geometric environment data, encompassing visibility considerations and being reusable for both indoor and outdoor environments. We also present a number of algorithms in which these models can be utilised, most significantly a novel and highly effective algorithm that can generate natural language descriptions of objects that disambiguates on their location. All these components, while modular, operate in tandem and interact with a variety of external components (such as path planning) on the robot platform.

Acknowledgements

I would like to thank my supervisor Professor Stephen Pulman for his support and encouragement throughout my research, in addition to Dr Phil Blunsom and Professor Paul Newman for their valuable advice.

Contents

Chapter 1

Introduction

1.1 Motivation

In the field of robotics, there has recently been tremendous progress in the development of autonomous robots that offer various services to their users. Typical services include support of elderly people, cleaning, transportation and delivery tasks, exploration of unaccessible hazardous environments, or surveillance. Most of the systems developed so far, however, are restricted to indoor scenarios, non-urban outdoor environments, or road usage with cars. There is a serious lack of capabilities of mobile robots to navigate safely in highly populated outdoor environments. This ability, however, is a key competence for a series of robotic applications. We consider the task of developing a spatially motivated dialogue system that can operate on a robotic platform, where the purpose of such a robot is to aid pedestrians in urban environments to provide information about surrounding objects and services, and guide users to desired destinations.

There has been much research in developing dialogue systems operating on a robot platform that involve discussion of the user's/robot's environment, with varying degrees of discourse flexibility. [Kruijff et al., 2007] presents a robot named *'PeopleBot'*, capable of augmenting autonomously acquired maps with quantitative information, by interpreting users' descriptions about objects in the environment. It also handles commands, such as requests to go to a destination, turn, and for the robot to follow the user (using the appropriate visual tracking systems). [Bos et al., 2003b] meanwhile presents *'Godot'*, which the user can direct to specific locations, ask for information about its status, and supply information about its environment. It uses an internal map for navigation, communicating its current orientation and accessible locations (e.g. doorways). [Matsui et al., 1999] similarly presents an indoor robot specifically designed for office use, such as answering queries about people's location, route guidance and delivery tasks. [Skubic et al., 2002] uses a particular representation of the environment known as an *Occupancy Grid Map* to generate simple spatial descriptions, such as *'the object is mostly to the left of me, but somewhat forward. The object is close'*. In [Skubic et al., 2001] this is applied to generate descriptions from sonar readings. [Dobnik, 2009] presents a mobile robot capable of learning simple spatial prepositions such as 'behind' and 'left of'.

However, there has been lack of such systems that involve extensive two-way interaction between user and system (known as *mixed initiative dialogue*), with typically the user taking most of the initiative in answering questions or issuing requests, and re-

ceiving immediate responses with no disambiguation. In addition, there are typically problems integrating spatial algorithms with robotic platforms, either because the representation of entities in the environment is too simplistic (e.g. represented as points, as in [Kelleher and van Genabith, 2006]), or because the generated descriptions are limited in complexity and ignore some of the more advanced approaches available. This is perhaps the product of a lack of integration; while there may be substantial research in the areas of robotics, spatial cognition and dialogue modelling, there is frequently a lack of cohesion between these, with the generality or complexity of the spatial algorithms or dialogue models often compromised to accommodate engineering constraints on a robotic platform.

As such, our core aims for this thesis are the following:

1. **The development of a dialogue framework** capable of providing flexible enough discourse to accommodate complex spatial conversations about a user's surrounding environment. This involves a number of requirements, such as supporting multimodal input (i.e. both natural language input from a user and input from a robotic platform), and sufficient generality to support a number of models of dialogue and be reusable in other task domains.

2. **The development of a semantic parser** capable of converting natural language to a semantic representation that the dialogue system is capable of processing.

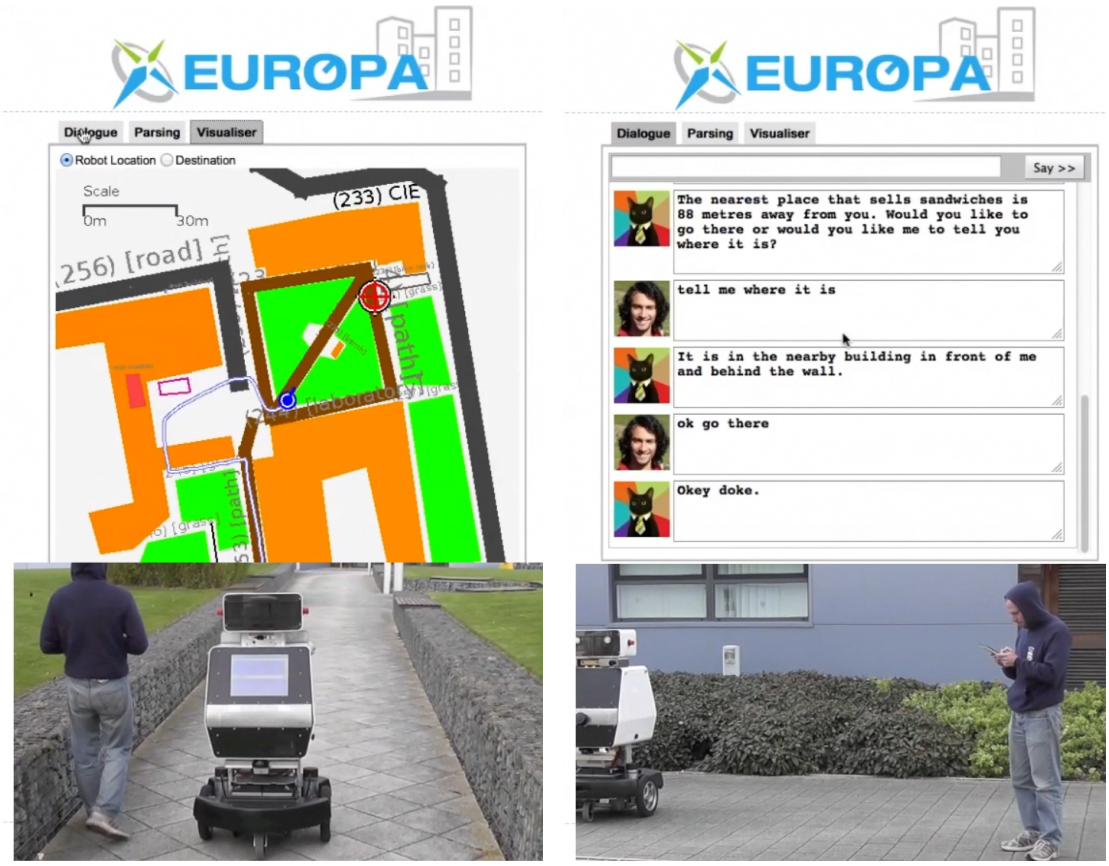
3. **The production of probabilistic models for various spatial prepositions** (such as *by*, *between*, etc.), and **algorithms which utilise these models**, such as searching for objects given a spatial description, and notably, the generation of spatial descriptions that disambiguate an object/location's identity or location.

4. **Integration:** A large emphasis is on how these systems, while capable of operating independently, can work together in tandem on a physical robotic platform. This manifests itself in two ways. Firstly, it guides various design decisions for each of the individual components; for example, the semantic parser must output a parse in a form that the dialogue system can robustly process. Secondly, this involves the development of a framework capable of connecting these linguistic systems to other components on the platform, such as path planning, appropriate simulator tools to test the systems together without need of a physical robot. This also includes the development of a remote web interface providing external communication with the robot while in operation.

These systems are shown working together in Figure 1.1.

1.2 The EUROPA Project

EUROPA (standing for *European Robotic Pedestrian Assistant*) is a collaborative project between a number of European institutions specialising in robotics and computer vision, aiming to develop such a robot. Its three aims of personal assistance, transportation tasks and guidance encompass a number of cross-disciplinary challenges:



(a) (b)

Figure 1.1: A trial of the robot effectively demonstrating the dialogue system, mobile interface and spatial algorithms working together, integrated on a robot platform.

Figure 1.2: The EUROPA robot.

Localisation is the process by which a robot (or indeed any agent) determines its location in the world. Location can be determined coarsely by use of GPS, but such a measure is inadequate when we consider the task of *mapping*, i.e. building up a representation of the surrounding environment, since we require that sensor readings from different locations are coherent with a small tolerance of error. The EUROPA robot primarily achieves localisation by use of a laser that scans the environment; by comparing the recorded laser measurement with that at the previous recorded position of the robot, we can determine the change in location with a high degree of precision. The robot is able to utilise the reflective properties of materials to spot vegetation and improve localisation [Bennewitz et al., 2009].

Object Tracking: The robot is capable of determining the trajectory of potentially multiple pedestrians in three-dimensional space using stereo camera input (i.e. two cameras in parallel to each other) [Ess et al., 2009a]. Using models which can predict the interaction of pedestrians relative to each other, it is possible plan the route of the robot in its vicinity that avoids collisions with moving pedestrians.

Path planning is the means by which a system can use its internal representation of the environment (which is presumed to be known) to find a route from the current location to a designated destination. Planning is composed of *global planning*, which considers the overall path of the robot, and *local planning* which deals with navigating local obstacles to reach the next goal point on the global plan.

Urban artefact identification is the process of recognising objects in urban environments, such as curbs, crossings and road signs [Maye et al., 2010] [Ess et al., 2009b]. The first of these for example is particularly useful in avoiding the traversal of roads (except when crossings are recognised).

The physical robot is shown in Figure 1.2.

1.3 Contributions of this Thesis

The main contributions of this thesis are as follows:

- The development of a dialogue manager (known as *HURDLE*) capable of accommodating a number of different models of discourse, accommodating multi-modal input, anaphoric resolution and handling multilogue/multiple utterances at once.
- An implementation of a dialogue system using this framework that allows conversations involving spatial reasoning according to the EUROPA domain.
- A number of probabilistic models for various components of spatial language, notably spatial prepositions such as *between*, *by*, *near*, *in front of*, etc.
- Providing a distinction between *validity* and *relevance/salience* models, and means by which we can measure the latter.
- A novel algorithm that generates natural language descriptions of a given object that disambiguates its location, given an environment of objects.
- An extension of an existing algorithm that generates natural language descriptions of a given object to disambiguate on its *identity*, with an investigation (and subsequent implementation) into how to yield the most natural expressions. This includes calculating the set of visible objects (using an efficient algorithm we present) for use in the description.
- Other algorithms for the interpretation of spatial language, including assessing the validity of longer spatial expressions and learning an object's occupancy in space based on descriptions about it.
- An exploration of the different methods for parsing natural language input in discourse system, and an advocation for Synchronous Context Free Grammars for effective integration with the tree structured represented expected by the dialogue manager as input. We also explore how a Context Free Grammar can be automatically generated by the dialogue manager for the target semantic language, and extensions to the SCFG formalism (with corresponding modifications to the *Earley Parser*) to accommodate more flexible translation.
- A number of tools produced to assist in the creation of environment representations, debugging of dialogue systems and a simulator to test the dialogue system effectively without the need of a physical robot platform.

1.4 Outline

In *Chapter 2*, we explore the various means by which spatial language can be represented, considering both qualitative and quantitative considerations. We present data structures and algorithms used that form part of how many prepositional models (i.e. models for *between*, *on*, etc.) can be defined, including the visibility of objects and using the

enclosure of objects within each other so that indoor and outdoor environments can be accommodated.

In *Chapter 3*, we determine how these prepositional models form part of higher level algorithms, for both the generation and interpretation of spatial language. In the task of language generation, we consider algorithms to describe an object both by disambiguating on identity and disambiguating on where it is. In the task of language interpretation, we consider a *search* algorithm to evaluate a large spatial description of an object on a database of objects in the environment, and a means by which we can learn the position and occupancy of an object in space based on descriptions made about it over time.

In *Chapter 4*, we analyse the various models for representating discourse state and determining discourse behaviour, and evaluate existing dialogue frameworks which employ these models. We draw on these weaknesses, along with the requirements of the EUROPA task domain, to produce a dialogue framework that is domain-independent, supports multi-modal input, encourages reuse of generic rules for discourse controls, and accommodates a number of the different possible dialogue models. We examine an implementation using the framework that supports spatially-motivated dialogue for the EUROPA domain.

In *Chapter 5*, we turn our attention to the associated task of processing natural language input for use in the dialogue system, as well as the converse task of generating natural language from outputs of the system. We analyse the past approaches to input parsing specifically within the scope of discourse systems, and propose a means by which we can produce a rich but rigid representation of semantic content that allows effective and robust extraction and matching of constituent data. We examine if such translation can be learnt and the inherent challenges and problems associated with it, and present the means by which our dialogue framework can automatically generate the target semantic grammar.

Lastly, in *Chapter 6*, we examine the overall system architecture which connects the various linguistic components on the robotic platform, present details of the user interface used to interact with the robot, and provide more technical detail that underpins some of the components, such as building the compiler that interprets our dialogue language within *HURDLE*. We also evaluate the performance of the system as a whole.

Chapter 2

Prepositional Models

2.1 Overview

The interpretation and generation of spatial natural language has been studied in a wide range of practical settings. These include the generation of weather descriptions from geographic data [Turner et al., 2008], the generation of route directions [Wei et al., 2009], and spatial search engines designed to evaluate queries against databases of geographically organised objects [Jones et al., 2003]. It has also been applied to visual search, where video clips can be retrieved from large databases of CCTV surveillance data by interpreting spatial descriptions involving prepositions such as *down*, *through* and *towards* [Tellex et al., 2010]. There have also been a number of robotic applications. [Tellex et al., 2011] uses probabilistic graphical models to analyse components of spatial expressions for a forklift truck to interpret expressions such as *"Go forward and drop the pellets to the right of the first set of tires"*. [Skubic et al., 2004] presents a robotic platform for human-robot dialogue intreaction capable of using a laser scan of the environment in order to describe objects involving prepositions such as *near* and *in front of*.

In this chapter, we present models to enable individual components of spatial language to be interpreted and examine a number of ancillary challenges in order to produce flexible models that are adaptable to a variety of environments. We investigate a number of applications of these models in Chapter 3, including how natural language expressions can be generated for objects, and how we can accommodate spatial queries to search amongst amongst objects in the environment.

Space occupies a privileged place in language and our cognitive systems, given the necessity to conceptualise various semantic domains. One might initially claim that such cognition is in keeping with an 'objectivist' view of reality [Lakoff and Johnson, 1980], that things in the world can be encompassed by universally accepted and objectively definable categories. The world seems to consist of particularly well-delineated objects (and language consisting of words and morphemes are used to refer to these categories), yet as [Herskovits, 1986] observes, aspects of spatial language seems to often transcend such physically definable categories and use conceptual mental constructions which do not exist in the real world. In just using the simple expression 'left of the road', it is only via idealisations, conceptualisations and selections that permits a mediation between this use of language and a canonical view, i.e. some universal agreement on aspects of

its interpretation, even if varying in individual, linguistic and cultural variations. This inevitably leads to a subjective rather than objective view of the interpretation of spatial language, but one can still provide a mapping that embodies both quantitative and qualitative properties, even if only approximate.

Attempts to model such cognition originates from the cognitive psychology and psycholinguistic domains. [Lindkvist, 1950] provided early work on interpretation of prepositions such as 'at', 'in' and 'on'. Similarly, [Denofsky, 1976] provides a quantitative analysis of the preposition 'near' based on contextual factors such as object size and the 'range' over which the positioning of the the objects are considered (for example, the range of a chair and table might be that of the room they are situated in). We will explore this in more detail in Section 2.7.4.

Older approaches to modelling prepositions however are typically characterised by a more qualitative, often logic-based approach. A particularly seminal work is that by [Herskovits, 1986], who identifies a number of significant factors in identifying the canonical interpretation of spatial expressions:

1. *Normal situation type:* A 'normal' situation is one that adheres to the laws of physics, is within the field of gravity (if on the Earth) and the interactions between objects, or of a human with an object, conforms to its usual 'function'. This would allow us to identify for example that in the expression *'The woman walked through the wall'*, the implied meaning is as such: that some 'gap' in the wall was present, the walking took place at floor level, and the rigidity of the wall provided static bounds for which her path could take. As Searle humorously points out in [Searle, 1978], were a mat to be stiffened so as to be perpendicular to a floor, and a drugged cat balanced on its upper edge, the expression *the cat on the mat* would be *literally* true, but would fail to match an application of 'common sense' given our knowledge of the behaviour of objects and physical laws.

2. *Purpose:* A prototypical use of spatial expressions is as a 'locative expression', that is, one whose intent is to provide spatial constraints on its possible region, rather than a non-spatial interpretation. Locative expressions consist of two parts. The object whose location is under discussion is known as the *referent* or *located object*. The other, the *landmark object*, represents some entity that the referent makes reference to, such as with 'The teapot is *on the table*'. There are also a number of non-prototypical purposes. The expression may aim to to disambiguate identity rather than location (e.g. *'The shop at the corner sells cigarettes'*). They may also have some intended inference: *'I bought this present in a store on Oxford Street'* for example is not intended to identify the particular store but draw attention to the consequences of the specified location or provide some characteristic of the present. Additionally, locative expressions may have no spatial motivation at all, but instead provide some physical properties of an object, as in *'the man in the hat'*.

3. *Ideal meaning:* The 'ideal' geometric interpretation of a spatial relation is generally between two or three geometric objects, such as points, lines, surfaces and volumes, according the constraints as defined in the normal situation type, and possibly subject to other transformations. While [Herskovits, 1986] provides no quantitative explanation for such spatial relation, they recognise that they are more akin to

prototypes, some measure of membership to some accepted standard (as with *'is a whale a fish?'*), rather than truth-conditional meanings. There are two transformations which typically occur in such ideal meaning. *Sense shifts* shift one relation to another relation to a similar relation. *'The deep wrinkles in his forehead'* for example is interpreted with the substituted preposition 'on'. The use of 'in' expresses the relation of a gap to the 'interrupted' object, yet our interpretation is still such that some contiguity between the wrinkles and face is required. This effect is particularly evident when we consider expressions with the same landmark object but different meanings; *'the crack in the vase'* for example is suggesting the crack is on the surface, whereas *'the water in the vase'* clearly indicates the entire volume enclosed within the vase, but technically excluding the vase itself. *Tolerance shifts* meanwhile allows relations to be approximately true under certain conditions. With *'the book on the table'* for example, the relation would still be considered to hold even with other books between the book in question and the table (i.e. with the requirement of contiguity violated). It might be argued however that [Herskovits, 1986] is misleading in equating tolerance shifts with 'approximate truth'. The relation still fully conforms to the prototypical case even if a large stack of books are involved; instead it is relaxation (or 'tolerance') in the contiguity constraint by introducing transitivity.

The ideal meanings of elementary spatial concepts can be categorised loosely into 5 classes: topological, geometrical, physical, projective and metric. Topological concepts are those which do not employ some form of 'axis', such as the prepositions 'at', 'across' and 'inside'. Geometrical concepts (referred to in later literature as *projective prepositions*) have some implicit or explicit axis, whether the axis be a line (as with 'down' and 'along'), plane (as with 'in front of') or involve orthogonality with a line (as with 'to the left of'). Physical concepts involve integration of the actual physical dimensions of the landmark object rather than some inferred conceptual plane, volume, line or point. This might involve some kind of contiguity with the landmark object, such as 'on' (in the sense of supporting an object) or the volumes constrained to the horizontal cross-section of the object (i.e. 'above' and 'below'). Projective concepts are similar to geometrical ones, but involve the axis of the observer rather than that of the object. Lastly, metric concepts involve some measure of distance between objects, such as 'near' and 'close to'.

This taxonomy however is not widely adopted, and generally only a distinction is made between topological and projective concepts. Geometrical and projective concepts can be conflated into a single concept by distinguishing between their *frame of reference*, that is, the entity to which the axis of the relation is assigned. With the *deictic* frame of reference, the axis is positioned at the observer, aligned with the line of sight [Kelleher and van Genabith, 2006] (such as with 'the tree in front of you'). With the *intrinsic frame*, the axis is based on some landmark object's canonical position, or salient orientation (as with 'in front of the shop'). With the *absolute frame*, the axis is based on some static environment influence, whether based on the magnetic poles or gravity (as with 'above the house' or 'north of The Equator'). Curiously, there are cross-linguistic variations in such frames. In particular, some languages lack a distinction between the deictic and intrinsic frames, adopting solely the former [Levinson and Wilkins, 2006b].

In the rest of this chapter, we explore in more detail the practical building blocks that prepositional models can be composed of, as well as presenting specific models for

various prepositions. A short summary of each section is as follows:

Prepositional Models We explore further the quantitative means by which acceptability of spatial relations has been modelled in literature.

Data Representation We present our choice of representation for how objects in the environment can be modelled, to encompass both their semantic and spatial properties.

Validity versus Relevance Models We analyse how some 'relevance' assumptions have been considered in previous models, and argue for a dual model representation for spatial prepositions; one to represent validity, and the other to represent relevance.

Determining and Leveraging Visibility We explore how visibility of objects from a point of observation can be used to improve our models, and present an algorithm to efficiently determine object visibility.

Using B-Trees We explore use of the B-Tree data structure to allow us to simultaneously model indoor and outdoor environments, and discuss how this data can be used for a variety of different purposes.

Data Collection To determine suitable parameters for each of our prepositional models, we conducted an online experiment. We explore the data collection process here.

Models The culmination of these above considerations is the development of number of models for numerous different spatial prepositions, which we present.

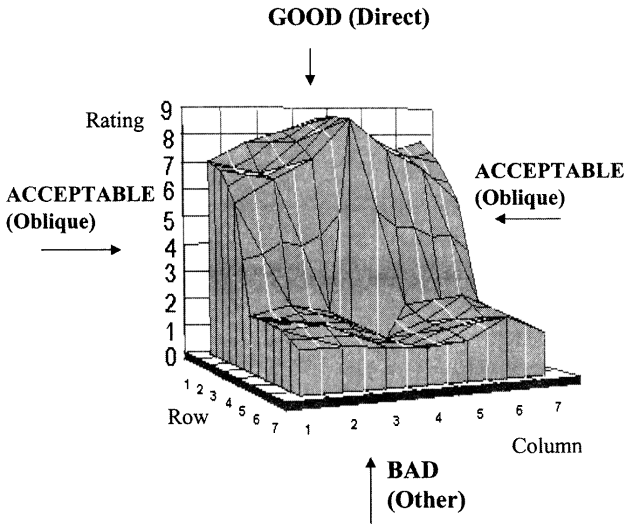
2.2 Prepositional Models

During the 90s considerably more research emerged concerning a quantitative analysis of the 'acceptability' of spatial relations, as alluded to in [Herskovits, 1986] but never explored. Perhaps the clearest embodiment is the concept of *'spatial templates'* introduced by [Logan and Sadler, 1996]. This presented participants with a small O in the middle of an invisible 7x7 grid. Across trials, a small X was placed in each of the remaining cells, and participants were asked to rate the acceptability of the sentence "The X is above the O". The resulting plot of acceptibility ratings across participants was referred to as the *spatial template* encompassing the predicate 'above'. This data is pictured in Figure 2.1.

A simplified outlook is one where the region surrounding an object can be partitioned into 3 classes: *good* regions where the acceptability rating was high, *acceptable* regions for intermediate ratings, and *poor* regions for low ratings. The task is then one of regression to find a suitable model that best explains the experimental data obtained. The bulk of current research into spatial cognition involves ascertaining these models for a variety of primitive spatial relations, and determining the factors that influence such models.

Models are largely categorised into two types, *Proximal* and *Centre-of-Mass* [Regier and Carlson, 2001]. Centre-of-Mass models disregard shape data of landmark objects in favour of representing the object simply by a single point, namely its centre of mass or some other salient centre. In the context of the model for 'above', any deviation

Figure 2.1: The spatial template for the preposition *above*, as presented in [Logan and Sadler, 1996].



with the upright vertical of the centre-of-mass will result in a diminution of acceptability, regardless of whether the object appears within the area bounded by the horizontal extremes of the landmark object. [Gapp, 1995] initially suggested a linear drop-off of acceptability based on this angular deviation, but a lack of correlation with the experimental data resulted in a revised model based on proximity to the bounds of the object, that is, the closest points of the located and landmark objects. The use of Centre-of-Mass models is also problematic in topological relations such as 'near'. [Kelleher and Costello, 2008] uses an equation for proximity that linearly degrades between the centre of the landmark object to 0 for a point in the horizon. This is fraught with numerous problems, as with much similar research. Firstly, the setup of the associated experiments is such that shape of the landmark objects (i.e. small polygons) used will inevitably have no influence on the user, thus providing a poor comparison to reality. In real-life settings however, the physical bounds of objects, particularly large ones, do indeed have a significant effect on spatial interpretation (distance on a park for example is generally based on proximity to its edge, not its centre). [Kelleher and van Genabith, 2006] attempts to improve the choice of the origin of the frame, that is, the directional axis vector from which angular deviation is calculated for projective relations, by choosing a point on the surface that is at the centre of the 2D silhouette observed from the user's viewpoint. Despite the complexity involved to calculate the point from a three-dimensional mesh, it fails to address the intrinsic problems of the centre-of-mass model.

The second problem leads to some discussion to the choice of value for a relation's 'acceptability'. Is it adequate for a mere *relative* measure of acceptability across candidate positions of the referent, or is an absolute measure required? We will see the importance of this later in Chapter 3, where many algorithms consider multiple spatial models at once (for example, the product of their outputs). A relative measure is only valid when considering a single model in isolation; in which case, we would only concern ourselves with the function being monotonic with regards to some property, for example we may as well use linear degradation (where the gradient can be chosen arbitrarily) of validity

with distance for the *by* model for example, since the exact shape of the function would be irrelevant.

We lastly explore the *Potential Field Model* [Olivier and Tsujii, 2004] whose provenance was as a means to model spatial occupancy in path-planning for robot manipulators [Khatib, 1986]. Objects in the environment each generate a potential field, such that the strength deteriorates with distance from the object. The field is conceived as a repulsive force, and the sum of such forces across multiple objects generates an overall field for which the minimum potential contour can generate a path most likely to avoid collisions. Potential fields can capture spatial relations based on proximity via the use of an elastic potential function, analogous to that of a mechanical spring:

$$P_{prox} = \frac{K_{prox}}{2}(\sqrt{((x - x_0)^2 + (y - y_0)^2)} - L_{prox})^2 \tag{2.1}$$

where K_{prox} is the spring constant (i.e. strength of the spring), L_{prox} the original length of the spring without contraction or extension, (x_0, y_0) is the position of one end of the spring at the landmark object (the choice of a single point is indicative of a centre-of-mass model) and (x, y) the position of the referent. We can capture directionality by the square of the horizontal displacement from the axis (which differs from the angular deviation described previously). In addition to the potential field model and simple linear degradation is applicability, some models make use of the *sigmoid function*, which more effectively represents the fact that there are regions of full acceptibility with little variation, and regions of little/no acceptibility with little variation, with potentially the transition between the two only representing a small region.

2.3 Data Representation

The manner in which objects and locations in an environment are conceptualised and represented has been the subject of much research. Most are based on humans' hierarchical view of spatial organisation, where objects are conceptualised based on their enclosure within another [McNamara, 1986]. Other analyses contrast a *space*-based approach, where an environment is conceptualised by the space occupied by the local environment, and an *object*-based approach which focuses more on the organisation and relations between objects in the space [Yeap and Jefferies, 2000].

The storage/organisation mechanism used is particularly pertinent in the field of robotics, where frequently multiple spatial maps derived from multiple sources need to be fused. A *metric map* focuses on low level sensor data, using for example laser scanning to generate lines representing surfaces and obstacles. These maps are typically used to generate further maps for a robot to navigate its environment; a *navigation graph* is a graph consisting of nodes representing accessible points in the environment, with edges indicating traversal paths enabling a robot to move in its environment without collision with static obstacles (we generate this for example in Figure 6.3). Unlike metric maps, toplogical maps represent objects in an environment explicitly, with nodes representing individual objects, potentially grouped by area, and potentially linked by spatial prepositions that pairs of objects satisfy [Vasudevan et al., 2007]. Finally, a *conceptual map* represents an ontology of object types. Rather than specific to a particular environment, it allows us to model hyponymy, where we can identify that object classes are subtypes

of one another, e.g. that chairs and tables are furniture and that a kitchen is a room. [Galindo et al., 2005] considers the topological and conceptual maps in parallel, so that objects in a specific environment can be anchored to entity classes in the conceptual map, thus inheriting the properties associated with that type of object. There have been several practical systems which have fused metric information with semantic. [Diosi et al., 2005] creates a map via guided tour, with the map segmented according to labels provided by an instructor. [Anguelov et al., 2004] is able to automatically generate a topological map from a metric one via supervised learning.

In light of [Herskovits, 1986]'s discussion of how we idealise the relationship of objects by treating them as points, lines and volumes, we adopt this treatment for our representation of objects. As such, all objects in the environment are associated with a list of two-dimensional points (or single point) from a birds-eye-view perspective, and an indication of whether it is a point, line or polygon. We can also extend this data to the third dimension by optionally specifying a height, so that polygons are treated as prisms. We do not however model more complex spatial information in the third dimension, and therefore assume either outdoor environments or single floor indoor environments where we can legitimately conceptualise objects as protrusions from the ground plane, with height but no elevation. However, all the spatial models which we subsequently present could be easily extended to three-dimensional object data provided such data was available.

On a semantic level, we adopt [Galindo et al., 2005]'s approach of associating semantic categories (e.g. laboratory, building) to each object in the environment. We also represent an ontology of entity classes by maintaining a tree of such classes. These classes can be assigned properties in the same way that entities can, in addition to a default width and height. In cases where this data isn't available for a specific entity (particularly for points and lines, where using coordinate data alone does not permit any inference to be made regarding the mass of the object), we can fall back on these values specified for the class. For example, if no coordinate was available at all except for the objects' positions, by treating them as point entities with some width associated with the class of object, then from the perspective of the spatial models, such models are modelled as circles. Classes have a number of optional flags, such as whether it is a mass or countable noun (which can be accessed by the natural language generator, so we can distinguish for example *some furniture* from *a tree*), and whether it is a salient enclosure (which we explore later in Section 2.5). Objects may also be assigned a salient axis if applicable, indicating the direction from the centre of the object to its 'front' face.

Each object has a unique identifier. For certain *dynamic entities* we reserve special identifiers, such as references to the observer (e.g. the lab in front of *you*) modelled as a point with height, and the current path (*"take the first left"*), implying the user is bounded to some particular line based entity such as a road. These are resolved to temporary concrete entities before any spatial model is evaluated.

Unlike approaches such as [Kuipers, 2000] that consider all types of maps in one monolithic multi-hierarchical framework, we consider only the topological and conceptual maps working in tandem. This is mostly motivated by the modular nature of the EUROPA platform, where the robot's global and local path planning functionality, making use of (and creating) the navigation and metric maps, operate entirely independently.

2.4 Determining and Leveraging Visibility

A number of the spatial models that will be presented in Section 2.7 utilise visibility information in their validity and relevance variants, whether in computing the visibility of objects from each other, or more commonly, the visibility of objects from the perspective of the observer. In addition, the information may be used directly in the task of presenting the surrounding objects to the user.

We will also see in Chapter 3 how we can use visibility to restrict the domain of objects we need to consider. In the case of identifying an object from a description, then a visible object which can be uniquely identified among visible objects is sufficient for overall identifiability, even if there are non-visible objects that match the description. For example 'go to the tree' would be clear if there was one visible tree, regardless of other occluded trees in the environment. Conversely, in *generating* a description for an object, we can restrict the 'distractor set' (i.e. the objects our description must distinguish the referent from) to visible objects, provided the referent is visible. Similarly, we can partly restrict entities that can feature in our description to visible ones, since it would be counterproductive to refer to landmarks that are both not visible and not known to the user.

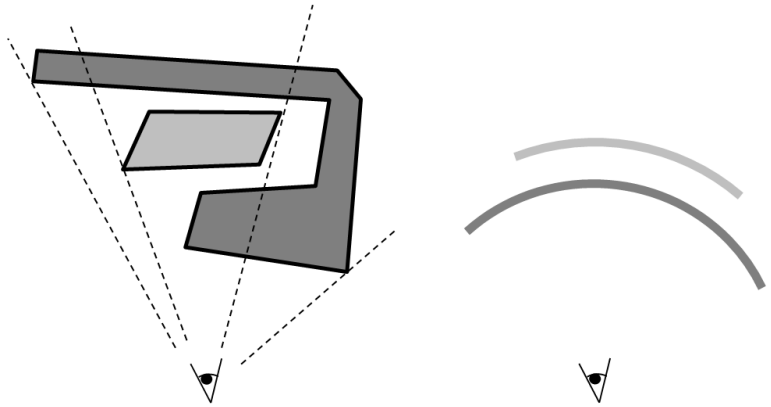
Computing visibility in an efficient manner is of paramount importance, especially tasks such as that in Section 3.3 when visibility needs to be ascertained for a large number of points in a lattice, or more generally when we require calculating the validity for a large number of spatial models at once, such as in the algorithm in Section 3.2.

A naive approach to determine visibility would involve firing rays from each vertex of each object in the environment to the point of reference (usually the observer), detecting collisions with any other objects. For n objects each with an average of m vertices, this would require $O(n^2m^2)$ time; intractable for anything other than small numbers of objects. Our algorithm computes the visible regions of all objects in $O(n^2)$ time in the very worst case, but in practice is closer to linear time with respect to n.

The efficiency of our algorithm is based largely on modelling objects as *polar ranges* with respect to some observation point. A polar range embodies that an object has some visible angular range, with some perceived distance which we assume to be constant. More specifically, a polar range is defined as a triple (r, θ_1, θ_2) where r is the radius of the arc, and θ_1 and θ_2 are the bearing range of the arc. The modelling assumption is based on the fact when considering a pair of objects, we'd usually expect one to appear unobscured in front of the other. There are scenarios involving complex shapes where this modelling assumption yields the incorrect visibility result, as indicated in Figure 2.2, but these examples are extremely contrived and would occur very rarely in practice. r is calculated as the distance to the Centre Of Mass of the object, and the angular range by taking the minimum and maximum angle from the observer to each point of the shape. We can also include the height of each object, thus expanding our representation to 4-tuples $(r, \theta_1, \theta_2, h)$. From a geometric perspective, this is the equivalent of modelling our objects as strips of a cylinder.

Given this representation, our task is then to determine which angular ranges for each arc are obstructed by the others. The output of the algorithm is to indicate for each object whether it is fully occluded, fully visible, or partially occluded. In the case of the latter, we also wish to indicate the object which occludes it to the largest degree.

Figure 2.2: An example of how shapes can be modelled as polar ranges, where the radius is the distance to the Centre Of Mass. In this contrived example, the modelling assumption breaks down, as the enclosed object is visible on the actual scene, but not in the polar representation on the right.



The algorithm to determine these occluded ranges is presented as Algorithm 1. We can summarise it as such:

1. We order the arcs by the left-most angle θ_l, and hold it in a list $DISTRACTORS$. The motivation behind such ordering is that neighbouring arcs are more likely to obscure each other, and we therefore adopt a *sliding window approach*, where we consider only a window of arcs that could possibly obstruct each other, and conversely, do not obstruct any objects outside the window except arcs previously processed.

2. This is the purpose of $ACTIVELIST$, which has the invariant that for any arc in the list, we have recorded which portions of it have been obstructed by all other arcs in $ACTIVELIST$. We can remove any elements with right extreme θ_r from $ACTIVELIST$ when some new arc with left extreme θ'_l is added such that $\theta'_l > \theta_r$, because the ranges do not overlap, and thus cannot possibly obstruct each other. We also know due to our invariant that the removed arcs have already been compared with the other arcs in $ACTIVELIST$.

3. We start therefore with an empty $ACTIVELIST$, with our pointer to the current element of $DISTRACTORS$ as the index j. We add the current element to $ACTIVESET$, remove any arcs as previously described, and compare the element to all others remaining in $ACTIVESET$.

4. However, since angles are cyclic, it's possible that arcs towards the end of $DISTRACTORS$ may obstruct arcs towards the beginning of the list. We therefore loop back, and can continue adding to $ACTIVELIST$ from the start of $DISTRACTORS$. We can finally terminate when the arc with the largest θ_r value is no longer in the active set.

How then, given a pair of arcs, do we update visibility? We first find the intersection of the two ranges to get the segment of the farther arc (based on the distance r) that may

Algorithm 1 An algorithm to approximate the obscured ranges of each object in an environment.

$DISTRACTORS \leftarrow$ A list of distractors, in the form $(\theta_l, \theta_r, r, h)$ where $[\theta_l, \theta_r]$ is the angular range of the arc (s.t. $(\theta_l < \theta_r) \wedge (0 \leq \theta_l < 360) \wedge (\theta_r - \theta_l < 360)$), r is the distance from the observer, and the list is ordered by θ_l.
$ACTIVELIST \leftarrow \langle\rangle$
$LOOPBACK \leftarrow false$
$LASTONE \leftarrow \arg\max_x \{\theta_r | x = (\theta_l, \theta_r, r, h) \wedge (\theta_l, \theta_r, r, h) \in DISTRACTORS\}$
$j \leftarrow 0$
while $\neg$LOOPEDBACK $\vee$ LASTONE $\in$ ACTIVESET **do**
  if $|ACTIVESET| > 1$ **then**
    for $i = 0 \rightarrow |ACTIVESET| - 2$ **do**
      updateVisibility(ACTIVESET[i], last(ACTIVESET))
    end for
  end if
  if $j = |DISTRACTORS| - 1 \wedge \neg LOOPEDBACK$ **then**
    $LOOPEDBACK \leftarrow true$
    $j \leftarrow 0$
  end if
  (a,b,d) $\leftarrow$ DISTRACTORS[j]
  for $(\theta_l\prime, \theta_r\prime, r\prime, h\prime) \in ACTIVESET$ **do**
    if $(\theta_l, \theta_r, r, h) \neq (\theta_l\prime, \theta_r\prime, r\prime, h\prime) \wedge \neg\theta_l \in [\theta_l\prime, \theta_r\prime]$ **then**
      $ACTIVESET \leftarrow ACTIVESET - (\theta_l\prime, \theta_r\prime, r\prime, h\prime)$
    end if
  end for
  $ACTIVESET \leftarrow ACTIVESET \cup \{(\theta_l, \theta_r, r, h)\}$
end while

be obscured. We then consider whether a line projected from the eye-level of the observer (say at height above of the ground of h_o) to the top of the farther arc (of distance r_2 and height h_2). By simple geometry this line is $y = (\frac{h_2 - h_o}{r_2})x + h_o$. If the farther object is obscured from view by a nearer object of distance r_1 and height h_2, then:

$$(\frac{h_2 - h_o}{r_2})r_1 + h_o < h_1 \tag{2.2}$$

If the further object was obscured, we indicate that the intersection of the two angular ranges was obscured, and leave a reference to the obscuring object.

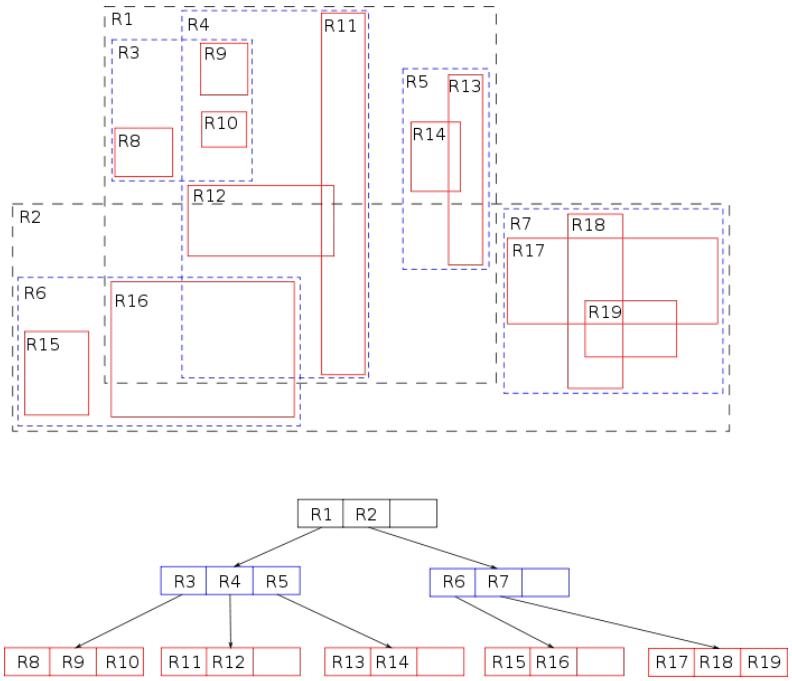
We can see that the complexity of the algorithm depends on the extent to which the arcs representing each object overlap. It is possible to a construct a scenario where the complexity is quadratic in the number of objects; if say after ordering the arcs by θ_l, each arc is contained within the range of the arc immediately before it in the list. The consequence is that *ACTIVESET* can never discard its elements, eventually filling up to the total number of arcs. Given on each addition to *ACTIVESET* we compare it to all other elements in it, we do a quadratic number of visibility comparisons. In the opposite extreme, where none of the arcs overlap, *ACTIVESET* is always of size 1, and no comparisons are required.

2.5 Using R-Trees

A distinction is commonly made in robotics between indoor and outdoors environments. Indoor environments usually imply a greater density of objects, narrower spaces to traverse and a bounded, enclosed environment. In the context of metric mapping, as we explored in Section 2.3, no explicit distinction between indoor and outdoor environments need be made, given that solid enclosures (i.e. walls of buildings and rooms) can be explicitly represented in the map, with the map concerned only with occupied and unoccupied space. Considering both types of environment simultaneously is more problematic for topological maps that consider the relationship of discrete objects in the map, such as evaluating prepositional models between pairs of these objects. [Vasudevan et al., 2007] for example deals only with indoor environments, with ad hoc groupings of objects into rooms and corridors, connected with doorways. All the spatial models explored in Section 2.2, such as [Kelleher and van Genabith, 2006], [Logan and Sadler, 1996] and [Regier and Carlson, 2001], merely treat all entities as solid objects enclosing no others, thus permitting only simple environments. Ideally, we'd wish to model the enclosure of objects within each other, whilst indicating whether an enclosing region consists of solid edges (e.g. a room or a building) or represents some conceptual/nominal region such as a town or city. By doing so, we greatly increase the generality of prepositional models in terms of the referent and landmark objects we can consider.

[Jones et al., 2004], [Vaid et al., 2005] and [Jones et al., 2003] use a data structure known as R-Trees in the context of indexing geographically positioned entities (e.g. countries, towns and buildings). An R-Tree, as illustrated in Figure 2.3, is a tree structure where the children of each node are those physically enclosed by the parent entity. Maintaining a structure for entities in an environment is an effective indexing method for spatial entities, since we can readily select entities within a region by restricting the set

Figure 2.3: An example R-Tree structure.



to those which are descendants of a node corresponding to that region. For example, we could efficiently evaluate queries such as "Find all museums within 2km of my current location". We can root the tree with an entity representing the entire environment, which we subsequently refer to as the *root entity*. We consider an entity within the enclosure of another provided that it is within the region on the ground plane of the other. A cup therefore might be the child of a table, or a tree of an area of grass, despite having an 'on' rather than an 'in' relationship.

We extend the definition of a R-Tree such that objects are assigned a flag indicating whether they have an outer obstructing shell. For example, a building has an outer shell, whereas a region of grass would not.

Using a R-Tree to store the entities in the environment has a number of beneficial consequences:

1. **The domain of objects that may be visible can be restricted.** If we follow the ancestors in the R-Tree upwards from the observer, until we reach an object with an obstructing shell, we know that any objects that aren't in the subtree rooted by this node cannot be visible. This leads to a large efficiency saving, but also allows us to correctly identify objects as obstructed from view where they otherwise may have not been. To determine whether two arbitrary objects are visible from each other, we find the path through the R-Tree connecting the two objects. If any object on this path has an obstructing shell, with the exception of the object at the highest level in the tree, then we have an obstruction. Otherwise, we'd apply our computationally more expensive algorithm of Section 2.4.

2. **Prepositions such as *in*, *on* and *through* can be evaluated more efficiently.** We can directly determine the validity of these relations by a depth-first

or breadth-first search of the object's descendants in the R-Tree, eliminating the need for repeated expensive geometric computation. The computation of determining whether one object is enclosed within another instead need only be performed once when inserting the entity into the R-Tree.

3. **A measure of *relevance*.** In Section 2.6 we will see that we distinguish relevance models from validity models, which in part is based on the relatedness of objects. This relatedness is largely determined by the number of edges in the R-Tree that separates the referent and landmark object. *"The chair is east of China"* for example, while possibly true (i.e. is *valid*), is not regarded as relevant due to the number of enclosures crossed within the R-Tree to get from the chair to China (when one considers towns, continents, etc).

4. **The *near* model.** For the *near* prepositional model, the validity is dependent on the *range* of the two objects, where for example we would interpret the nearness of a chair to a desk not by an absolute measure of distance, but with respect to say the room they were enclosed in. We explore this model further in Section 2.7.4.

An algorithm for adding a new entity to the R-Tree is presented in Algorithm 2, executed each time a new entity is added to the environment. We first use the method *getLowestEnclosingNode* in Algorithm 3 to obtain the node that minimally encloses the span of the entity we wish to add. We can then add the new entity at this correct level. However, since the new entity may itself enclose the children of the current node, we need to move the subtrees to become children of the added node as appropriate (lines 3 to 8). The In relation is computed geometrically via the method presented in Section 2.7.6.

When creating dynamic entities, such as references to the observer (i.e. for relations such as *behind you*), we require such entities to temporarily be inserted into to the tree. This can be achieved using the same method, and returning the tree to its previous state when such entities are no longer needed.

Algorithm 2 function addNode(*toAdd*, *root*)

Require: *toAdd* is the node to add.
Require: *root* is the root of the tree.
1: $p \leftarrow getLowestEnclosingNode(toAdd, root)$
2: Add *toAdd* as child of p.
3: **for** e ← children(p) **do**
4: **if** In(e, toAdd) **then**
5: Add e as child of *toAdd*.
6: Remove e as child of p.
7: **end if**
8: **end for**

Algorithm 3 function getLowestEnclosingNode(*toAdd*, *node*)

Require: *toAdd* is the node to add.
Require: *node* is the node in the tree currently under consideration.

```
1: for e ← children(node) do
2:     if In(toAdd, e) then return getLowestEnclosingNode(toAdd, e)
3:     end if
4: end for
5: return node
```

2.6 Validity versus Relevance Models

2.6.1 Introduction

Our explorations of models for spatial prepositions so far have encapsulated the *validity* of prepositions only; that is, giving a notion of their correctness. However, one might wish to consider the *informativeness* of relations, giving a measure of its value of inclusion in a generated description. In the extreme case, relations may be entirely valid but irrelevant; consider for example *'The chair is west of China'*. Determining a measure of relevance is critical in providing natural descriptions when we consider algorithms that produce descriptions that either disambiguate on identity or in location, as described in Sections 3.2 and 3.3, since in language generation, informativeness is equally as important as correctness.

[Pattabhiraman and Cercone, 1990] describes the role of salience in text planning, where the prominence of objects and properties governs how generated textual content is both selected (i.e. choosing which information to include) and organised (i.e. how such information is structured and linguistically realised). A distinction between salience and *relevance* is made: the latter relates to objects/properties external to the speaker, while relevance operates more macroscopically, concerning dialogue participants' utterances and interpretations in communitative contexts. [Leech, 1983] defines relevance as such:

> An utterance is relevant to a speech situation if it can be interpreted as contributing to the conversational goals of the speaker or hearer.

This relates to one of the Gricean maxims (which we explore in Chapter 4), notably the Principle of Relevance, which states tersely, 'be relevant'. There have been a number of attempts to computationally model salience and relevance. [Hajičová, 1993] for example models discourse salience by the *recency* in which an object was mentioned. [Kelleher and van Genabith, 2004] examines an object's *visual* salience by examining its size and centrality in the observer's field of view.

It should be noted that salience is not entirely independent from validity. [Gapp, 1994] for example bases the tolerance of distance from a landmark object for topological prepositions on its salience, and indeed, our model for the *by* preposition in Section 2.7.2 uses the landmark's height in calculating validity. [Kelleher and Costello, 2008] uses only a single prepositional score for topological relations based on the validity based on distance and its landmark object's salience. However, while certain factors like landmark height may influence both prepositional validity and relevance/salience, we argue that these two

measures should be computed separately due to their differing functional roles in discourse. We have seen for example that relevance/salience is most prominent in language generation. Similarly, for certain converse tasks, such as *verification* of a spatial expression (e.g. *"is the car between the two trees?"*), interpretation is contingent on the *validity* of the preposition between alone.

2.6.2 Types of Model

Consider a relation aRb that merits potential inclusion in a spatial description, where a is the referent, b is the landmark object, and R is the spatial preposition relating the two objects. There are 3 different aspects of this relation we can partition salience/relevance models into:

1. **Models based on the landmark** b. This includes the visual salience as discussed previously, in addition to other semantic properties, such as the type of object and contextual factors; a famous landmark would have a large intrinsic salience regardless of its spatial dimensions or visual properties.

2. **Models based on the relationship between** a **and** b: Here we consider the appropriateness of using the two objects together in a relation. For our "chair is west of China" example, we would expect the appropriate metric relating a particular chair and China to be low.

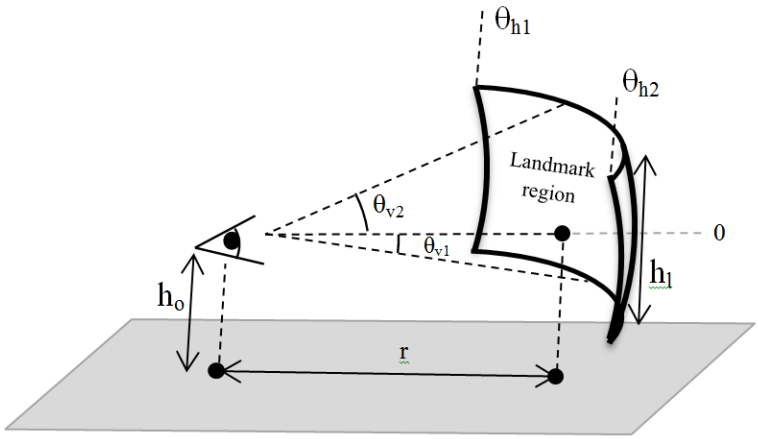
3. **Models based on the specific preposition** R: Salience of a relation may be specific to the function of the preposition. For our *between* validity model presented in 2.7.3, the preposition is considered valid provided the referent appears anywhere in the convex hull of the landmark objects. However, use of the preposition would only be informative if the referent was moderately central relative to the landmark objects. Such specific salience models are discussed for each of the specific prepositional models in Section 2.7 as they arise.

We combine these 3 salience measures by taking their harmonic mean. The harmonic mean is often employed when we wish the lowest of a number of scores to dominate the mean; in particular, if any of the scores are 0, the overall score will be 0. This is computed as $H = 3/(\frac{1}{s_1} + \frac{1}{s_2} + \frac{1}{s_3})$.

2.6.2.1 Models based on the landmark

A number of existing approaches examine the visual salience of objects.
[Kelleher and van Genabith, 2004] presents an approach in which bitmap images of the observer's view are examined to determine each object's size and centrality are examined, where the pixels associated with a particular object are assumed to be identified. Each of these pixels are assigned a weighting which linearly decreases with distance from the centre of the image. The visual salience score is then the sum of these weightings for each pixel in the object. Scores are normalised such that the object with the highest salience score has an associated value of 1. The drawback of this approach is the large computational expense associated with identifying the objects in the bitmap, or if a 3D

Figure 2.4: Instead of [Kelleher and van Genabith, 2004]'s inefficient approach of computing salience by computing a 2D image based on what the robot can see, we can compute the angular range visible to the observer both horizontally and vertically, but directly using the 3D data of the landmark object.



environment is provided, the expense of generating a labelled bitmap based on the pose of the observer.

Recall that in Section 2.4 for our visibility algorithm, visible objects were modelled as strips of a cylinder, with some visible angular range on the horizontal range and some height. If we converted the height to a vertical angular range (where 0 degrees is the observer looking horizontally), then instead of summing over pixel weights as before, we could (double) integrate the weight function across the two angular ranges. By simple trigonometry, we can determine the vertical range to be between $\theta_{v_1} = -arctan(\frac{h_o}{d})$ and $\theta_{v_2} = arctan(\frac{h_l - h_o}{r})$ where h_o is the height of the observer's eye-level above ground, h_l is the height of the landmark and r is the distance to the arc representation of the landmark. This is illustrated in Figure 2.4. We can represent the possible angles in this dual range as $\theta = (\theta_{yaw}, \theta_{pitch})$ where the yaw (most common in aviation terminology) is the rotation on the horizontal plane, and pitch is the angle in the up-down direction. We can assume the observer has a pitch of 0, and thus looking horizontally, i.e. $\theta_{obs} = (y, 0)$ where y is the bearing/yaw of the observer. Thus our visual salience score V is as follows:

$$V = \int_{\theta_{v_1}}^{\theta_{v_2}} \int_{\theta_{h_1}}^{\theta_{h_2}} -|\theta_{obs} - \theta| d\theta \tag{2.3}$$

The minus is so that larger angular deviations of a dual angle from the pitch/yaw of the observer are penalised, with salience once again deteriorating linearly with this distance. These values V are calculated for each visible object (a set we can obtain from the visibility algorithm), and then scaled so that the highest score is 1. This integration turns out to be inelegant and convoluted due to the use of the Euclidean distance function (even if in closed form). Thus a suitable approximation is to calculate the weighting just for the centre of the visible region of the landmark, i.e. for pitch $(\theta_{v_1} + \theta_{v_2})/2$ and yaw $(\theta_{h_1} + \theta_{h_2})/2$ (forming an angle pair θ_{centre}), and assume the weighting is constant for the entire region. This approximation is sufficiently accurate, particularly when the deviation of the landmark's position from the observer's gaze is large, and thus the curvature of the

Euclidean distance function is suitably small such that function is close to linear (where the average of the weightings in the region will occur in the centre of the landmark region). Then the visual salience is:

$$V = -|\theta_{obs} - \theta_{centre}|(\theta_{v_2} - \theta_{v_1})(\theta_{h_2} - \theta_{h_1}) \tag{2.4}$$

2.6.2.2 Models based on the relationship between referent and landmark

We saw in Section 2.5 that R-Trees could be utilised to model the enclosure of objects inside each other. It would seem natural to assume that objects would be compared only if comparable in their scope; chairs in a lab for example could be compared because their scope is only their enclosing room. Chairs in separate rooms are less related (visibility issues aside) since we have to cross more enclosures (i.e. the labs of each, and their enclosing building) in comparing their scope. To quantify the relative relatedness of the referent and landmark object, we can simply find the number of edges that require traversing between the two objects in the hierarchy. While we did not precisely establish the mapping from enclosure distance to relevance measure, we found that in practice a distance less or equal to 2 was fully sufficient for a relation to be considered informative (e.g. two objects enclosed by a room have two edges between them in the R-Tree), with relatedness rapidly deteriorating thereafter.

2.7 Models

We have explored a number of factors which determine validity or associated measures, such as the visibility, salience and enclosure of objects. These factors applied generically to all spatial prepositions. We now therefore propose specific models for a number of spatial prepositions motivated by experimental data. Suppose we represent our spatial relation (coupled with the environment it is evaluated over) as a variable z, and the pose of the observer as a variable x which represents its position and orientation[1]. Our aim is then to define a probability mass function:

$$p(\top_z | z, x) \tag{2.5}$$

$\top_z$ represents the 'truthfulness' of the relation in the context of the environment, and thus can take the values *true* or *false*. The reason $p(z|x)$ is not used is to avoid the implication that we're interested in the probability of the observation being made, rather than how a given observation is judged to be valid. It would also mean that the distribution would be over all possible spatial prepositions and referent/landmark objects, which is clearly not desired. How such *truthfulness* can be interpreted with respect to *human confidence* is philosophically nebulous. According to a *subjectivist* approach, rational agents can have *degrees of belief* in a proposition, and that probabilities are an adequate model to represent such subjectivity. [Finetti, 1937], describes the degree of a belief as *"the degree of probability attributed by an individual to a given event [when]*

[1]Consideration is often given to the agent used as the observer, whether the system or the user in discourse. We ignore this ambiguity and assume the observer is the robot, given the availability of odometry data.

revealed by the conditions under which he would be disposed to bet on that event". It is possible to formalise the relationship between our dispositions towards uncertainty and the axioms governing probability, such that these probabilities can legitimately be incorporated into, for example, Bayesian manipulation, as will be seen in Chapter 3. An example of such an axiom that must hold is *finite additivity*, i.e. $\forall A, B : (P(A) \geq 0) \wedge (A \cap B = \emptyset) \rightarrow P(A \cup B) = P(A) + P(B)$. Justifying these axioms is based on these hypothesised betting games, in which the agent acts based on their confidence in the propositions presented. More details can be found in [Fishburn, 1986] which explores the various attempts to axiomatise such subjectivism.

We frequently wish to compare values of $p(\top_z|z, x)$ for different positions of the referent object, where all other variables (including landmark objects and other potentially distracting objects in the environment) are kept constant. For notational convenience, we often refer to this as the confidence/probability mass function:

$$\psi(i, j) = p(\top_{z_{i,j}}|z_{i,j}, x) \tag{2.6}$$

where $z_{i,j}$ is the relation in which the referent object is positioned at the coordinate (i, j) in space. This is the same manner in which prepositional models presented in 2.2 consider how the validity of the preposition changes as the referent object moves relative to the landmark object.

2.7.1 Data Collection

In order to determine the optimum parameters for various parameters that constituted our spatial models, we conducted an online experiment that required participants to rate the acceptability of various prepositions, given a variety of scenes. An example scene can be seen in Figure 2.5. The questions were clustered into groups, either based on a single preposition (for example, questions regarding exclusively the preposition 'near'), or a group of similar prepositions. In each group, questions were randomised to try and prevent previous answers influencing later ones, especially given that many pictures involved subtle movements of either the referent, the pose of the observer, or shifts to a similar preposition, e.g. from *by* to *on*. The experiment was also restricted to native English speakers only, due to cross-linguistic variations in spatial coding. In particular, there is a lack of one-to-one mapping of spatial relations across languages, with some languages lacking a distinction between different frames of reference (that is, distinguishing between say the deictic interpretation of "in front of the tree" based on the position of the observer, and the intrinsic interpretation based on the salient side of an object, as in "in front of the shop") [Levinson and Wilkins, 2006a].

We normalised participants' answers from the 1 to 7 scale to the range $[0, 1]$ taking the average across answers to a given question, in light of our discussed treatment of subjectivity.

The scenes in questions were constructed so as to use real objects (such as trees, houses, roads, etc.) where possible, and estimating the base-width and height of said objects where appropriate. [Kelleher and Costello, 2009] and [Logan and Sadler, 1996] for example use very artificial experimental set ups, with the former considering only the relative positioning of small equally sized shapes within the confines of the user's screen. It is presumptive to claim that models obtained via this method can be extrapolated to

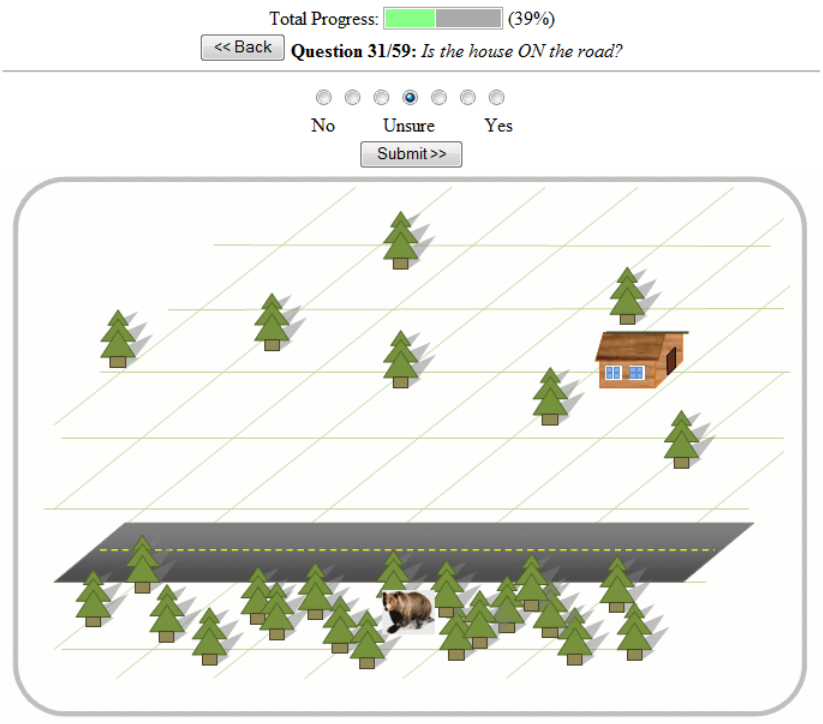


Figure 2.5: One of the questions for the 'on/next to/by' section in an online experiment. There were 132 questions in total across the 3 sections, with 40 participants.

the way in which humans reason about space in a real-world environment. However, it is difficult to accommodate all contextual considerations that influence our interpretation of prepositions; for example, there was no consideration for environment density on the effect of deterioriating of validity with distance. Due to the practical constraints of obtaining experimental data to accommodate a large number of model factors, coupled with the practical necessity for concrete implementations of the various spatial models working robustly on a robot platform, some of these model factors (particularly qualitative ones) are founded upon general reasoning rather than experimental data.

More generally, we place a greater emphasis on accommodating as large a feature set as possible for models useful on real-world platforms, and less emphasis on obtaining precise fine-tuned parameters that adjust the behaviour of these features. As [Regier and Carlson, 2001]'s overview of prepositional models seems to suggest, research has historically been more preoccupied with yielding parameters for a small feature set, such as distance and angular deviation. Eschewing more diverse qualitative considerations limits such models' applicability to practical systems.

2.7.2 By

Of the main topological prepositions: *near*, *at* and *by*, the last of these has received the least research attention. [Kelleher and Costello, 2008] presents a generic model for topological prepositions, but does not discuss how degradation of distance quantitatively varies between different prepositions. [Herskovits, 1986]'s discussion is limited to identifying that validity for *by* decreases more rapidly with distance than *near*, i.e. that *by*

implies greater proximity of the referent object to the landmark than *near*. Our contribution here is the production of a model which takes into account multiple properties of the landmark object in determining the tolerated distance from the landmark, and the use of experimental data to define a precise quantitative relationship between distance and validity for this preposition.

In our experiments, we identified three salient features that can influence the validity score; the base width (w) and height (h) of the landmark object (both available from our environment representation), and the distance (d) between the referent and landmark objects (which we measure from the perimeter of the shapes rather than their centres, as per the Proximal Model). For polygonal objects, users were presented with 8 different landmark objects in their scenarios, of a variety of different widths and heights. We made the following three hypotheses for our model:

- That validity has some linear relationship with the height of the landmark object. However, they are not directly proportional (i.e. the linear function has a non-zero intercept), as flat landmark objects such as lakes still permit some tolerance of distance.

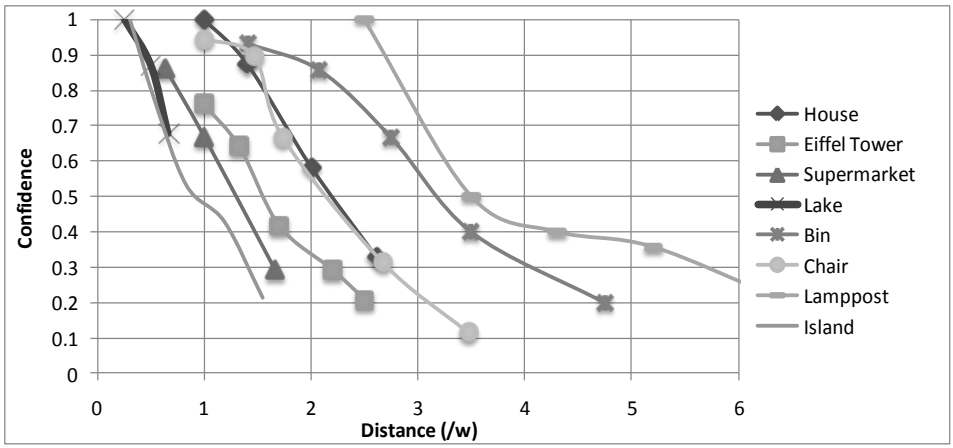
- That validity decreases to some extent linearly with distance in units of the base width of the landmark object. That is, we might expect validity to deteriorate half as much with absolute distance if the landmark object is twice as large in width. Figure 2.6a demonstrates how validity decreased with distance when the distance was measured in units of base width, for a variety of different landmark objects. While this assumption is alone not sufficient (since clearly from the graph validity is still not consistent across different landmarks), this yields greater consistency when raw distance is used alone.

- However, raw 'absolute distance' of the referent from the landmark still has some impact on validity. That is, while larger objects tolerate a larger distance for *by*, this tolerance reduces to some extent with larger object. For example, the tolerated distance is likely to be larger for a mountain as the landmark than say for a vehicle, but this tolerated distance relative to the size of the landmark is less for the mountain, due to the absolute tolerated distance being larger.

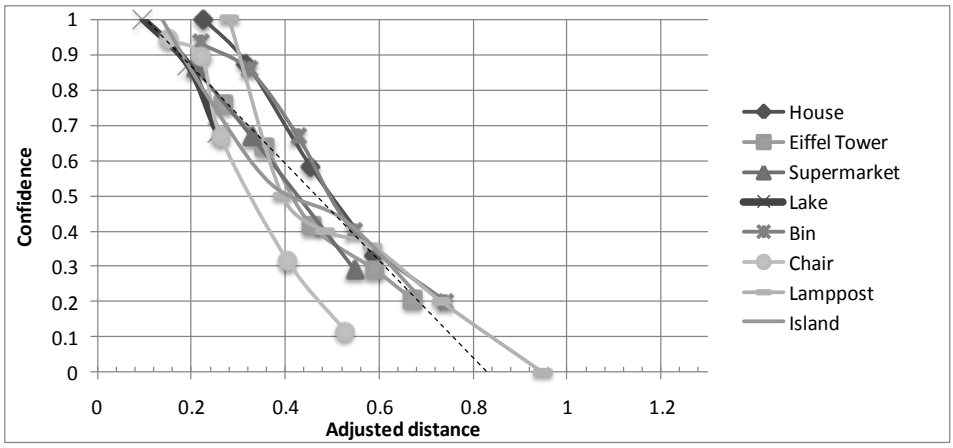
We used the following model to combine these assumptions:

$$\psi(i,j) = clamp(k_c - k_m d \frac{\log(w + k_w)}{h + k_h w}) \tag{2.7}$$

where k_c, k_m, k_w and k_h are appropriate constants and *clamp* ensures the outputted value is in the range $[0, 1]$ (increasing or decreasing the value if it is outside this range). The effect of scaling the distance by $1/w$ is to incorporate the second of our hypotheses, and scaling by $log(w)$ incorporates the third, where larger distances are penalised more with respect to the base width of the landmark if this base width is large. Figure 2.6b shows the effect of these using these transformations, using $k_w = 14$ and $k_h = 2$, resulting in values for k_m and k_c of 1.38 and 1.15 respectively, and yielding a close fit to the experimental data. Ultimately it is impossible to base any model of 'by' on physical metrics alone; the *use case* of objects, i.e. the set of contexts in which an object is used,



(a)



(b)

Figure 2.6: Experimental results for the preposition 'by' for a number of different examples. Each series indicates the *landmark object* used in the locative expression, e.g. 'lake' in "The house is by the lake". Graph (a) shows distance (in terms of width) plotted against confidence. Graph (b) shows adjustments as per equation 2.7.

is likely to have an effect. In Figure 2.6b for the case where the landmark object was a chair, it is apparent confidence deteriorated with distance much faster than expected. But if one considers that a chair is intended *to be sat on*, and therefore adjust the recorded height h to the more salient *seat-level*, we obtain confidence values very close to the model for this example.

For the *relevance* score, there are no evident salient spatial factors that would seem to enact an influence. We therefore use the default measure of the relatedness of the referent and landmark objects using the environment B-Tree.

2.7.3 Between

Unlike topological prepositions such as 'near' and 'by', 'between' is less contingent on the underlying properties of the landmark objects (such as height), given by definition it concerns the space enclosed by the objects. Responses from participants showed a

significant degree of agreement, perhaps explained by these fewer number of quantitative/qualitative features determining judgements. As expected, we determined that any point within the *convex hull* of the two landmark objects (excluding the area of the two objects themselves) was deemed to be fully valid. The convex hull is defined to be the convex region which minimally encloses a number of points, in this case, encloses the points on the surface of each of the landmark objects, such that the convex hull yields the area directly between the two landmarks.

Outside of this area, the degredation of validity becomes more complex. Data indicated that validity degraded inversely proportional to the centrality of the referent between the two landmarks, and degraded with distance from the convex hull. Equivalently, the tolerance of distance permitted from the convex hull increased with centrality, although the model best matched the data when this increase was polynomial, not linear.

We present our model below:

$$\psi(i,j) = \begin{cases} max(0, 1 - \frac{|(i,j)-p'|}{tol}) & \text{if } (i,j) \notin Hull(l_1 \cup l_2) \\ 1 & \text{otherwise} \end{cases}$$

$$\text{where } p' = \arg\max_{p''} \{|(i,j) - p''| \big| p'' \text{ is on line } l\}$$

$$tol = |v_1 - v_2| k_1 (1 - |1 - \frac{|p' - v_1|}{|v_1 - v_2|}|)^{k_2}$$

$$l : (v_1, v_2) = \arg\max_{l'} \{|p'' - (i,j)| \big| p'' \text{ is on line } l', l' \in conn\}$$

$$conn = \{(\bar{v}_1, \bar{v}_2) | \bar{v}_1 \in l_1, \bar{v}_2 \in l_2, (\bar{v}_1, \bar{v}_2) \in e(C(ref_1 \cup ref_2))\}$$

(i,j) is the central point of the referent in question, l_1 and l_2 are the vertices of the two landmark objects, $Hull(V)$ gives the convex hull of the set of vertices V (thus p' is the nearest point on the convex hull to (i,j)) and tol gives the maximum allowed distance from the convex hull before the confidence score is 0. $conn$ is the set of two edges on the convex hull which connect the shapes corresponding to l_1 and l_2 (that is, the straight dotted lines in Figure 2.7(a). Lastly the function e gives the edges of a polygon.

We establish p' in order to determine the centrality of the referent between the landmarks. Once computed, we can determine the percentage of the distance between the two landmarks it is using $1 - |1 - \frac{|p'-v_1|}{|v_1-v_2|}|$, which gives us a number in the range $[0, 0.5]$ (we clearly can't be greater than half way between). This centrality is then used to compute the tolerated distance tol from the convex hull (i.e. one of the edges in $conn$), where it is maximal if halfway between the landmarks. The distance $(i,j) - p'$ gives us the distance from the convex hull. If the distance is 0, we have full confidence. Once the distance exceeds the calculated tolerance, the relation is invalid. In between this distance range, confidence linearly degrades.

k_1 controls the maximum tolerance permitted, a specified proportion of the distance between the two objects, and k_2 controls the curvature of this ambiguous region; if $k_2 = 1$ then the tolerance of distance would increase linearly with centrality. Via model fitting we found values of $k_1 = 0.55$ and $k_2 = 2.5$ yielded the best results; diagram 2.8 illustrates the extent to which the model matches the experimental data. An image of the model in action can be found in Figure 2.7(b).

Figure 2.7: (a) The variation of confidence for the preposition 'between', for two landmark objects. (b) The model implemented and deployed in a real-world environment. The intensity of red indicates the confidence in the between relation for different positions of the referent.

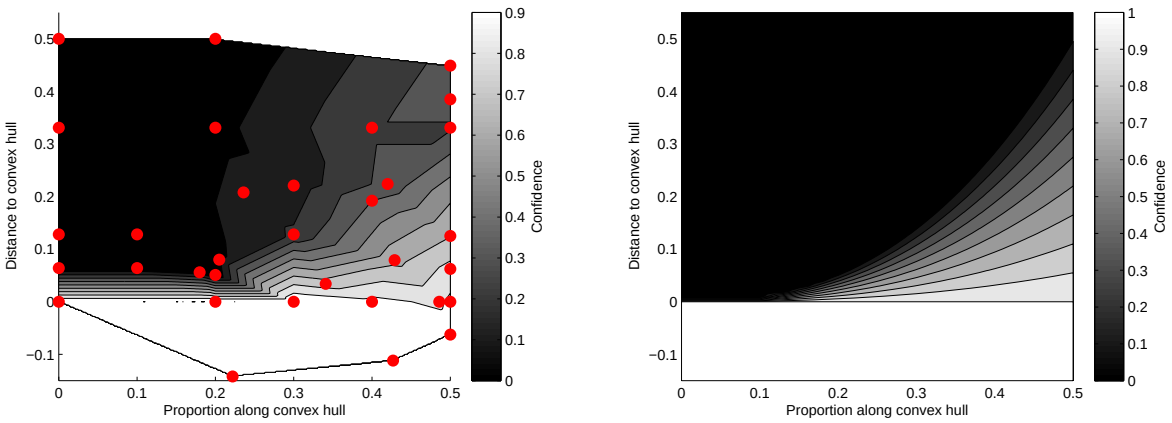


Figure 2.8: A comparison of experimental results (left), using linear interpolation, against the inferred model for 'between' (right). Both the x and y axis are in terms of the length of the convex hull edge.

From an implementation perspective, the convex hull of the two landmark objects can be found using the *gift wrapping algorithm*, also known in the two-dimensional case as the *Jarvis March* [Jarvis, 1973]. As the name implies, the algorithm can be visualised by a string being wrapped around nails on a board. This can be achieved by starting with the left-most vertex (guaranteed to be on the convex hull), and maintaining a 'wrap direction', initially pointing upwards. We find the node that has the smallest angular deviation clockwise, the continue with this node using the angle between the old and new current node. The process repeats until we reach the original node, and output the sequence of nodes that were visited. The connecting edges *conn* can be determined simply by observing adjacent nodes on the convex hull where one belongs to the one landmark object, and the other node to the other object.

2.7.4 Near

[Minsky, 1974] highlights that our perception of distance is often qualitative, and that continuous numerical data is often of limited value when considering the contextual factors of the scene. [Denofsky, 1976] is the first work that attempts to embody the high context-sensitivity coded into our interpretation of *near*, which we will explore. Despite the age of the work, a context-based model has been widely shunned by instead using the numerical distance data directly to quantify validity, as in [Kelleher and van Genabith, 2006] and [Kollar et al., 2010]. The latter assumption is justified if we fix the type of environment; for example, for *indoor* environments, our notion of near based on distance may not be affected by the properties of the landmark or referent objects. However, our desire to adapt to a variety of both indoor and outdoor environments prevents such a simplistic model.

[Denofsky, 1976] offers a rich model for *near* which incorporates a large number of both qualitative and quantitative factors. At its heart is the *near threshold*, the distance between two objects at which there is a 0.5 probability of them being considered 'near' each other. One can then use some suitable smooth function to model the region between

gradations such as *definitely near* and *definitely not near*. The memo suggests a normal distribution whose mean is at 0 distance[2].

The nearness threshold is based on three properties of the referent and landmark objects:

1. **Range**: The 'nearness' of one book to another in a bookcase is based in part upon the width of the shelf, given that if one were to extend the length of the shelf to double the size, our perception of nearness is likely to be altered.

2. **Object size**: Larger objects tend to raise the nearness threshold. A house might be considered near a mountain even if a gap of 1 kilometre separated them, although this distance might be considered to be far if say between two people.

3. **Standard**: If $\frac{3}{4}$ of the books on a shelf were to be removed, two books which were considered to be far may now be considered to be near. Pairs of adjacent books define a 'standard' for their locality, which is larger for an empty shelf than for a full shelf. Similar scenarios include city blocks, or cars on a road (affected by the level of traffic). We do not consider this factor in our model.

Both the range and standard attributes come in *local* and *global* variants; while extending the length of a particular bookshelf increases the near threshold between two books to some degree, there is a point at which our perception of the distance between two books on a *typical* shelf becomes more dominant. [Denofsky, 1976] suggests the near threshold can be calculated based on two thresholds that correspond to the metrics above (if for model simplicity we disregard the influence of 'standards'):

$$NearThreshold = \sqrt{ObjectSizeThreshold \times RangeThreshold} \tag{2.8}$$

This is the *geometric mean* of the two thresholds, dominated by the larger of the two figures. The *range threshold* is "one eighth of the way from the minimum possible distance to the maximum possible distance". That is, we must determine the *range* over which the objects may appear. The *object size threshold* is the diameter of the larger of the objects, although the paper offers refinements that takes into account the relative sizes of all 6 dimensions (i.e. the 3 dimensions of each of the two objects).

The object size is already immediately available by our choice of entity representation. The range required for its corresponding threshold must be determined however; [Denofsky, 1976] does not provide details on this. We have seen, though, that our use of B-Trees in Section 2.5 models the enclosure of objects. To compute the range using this data structure, we propose a notion which we refer to as the *Lowest Common Enclosure*, defined for two objects as such:

$$LCE(x, y) = z \ s.t. \ x < z \wedge y < z \wedge \nexists z\prime(z\prime < z \wedge x < z\prime \wedge y < z\prime) \tag{2.9}$$

[2]Use of a Gaussian function however yields a large region of 'vagueness' given the smoothness of the function shape, that is, a large range of distance for which we cannot confidently assert whether near does or does not apply. A *sigmoid function* would be more suitable, since the shape is such that two tails with low and high probability respectively (i.e. when our confidence in nearness is unambiguous) and a region of adjustable size with the validity of near is ambiguous.

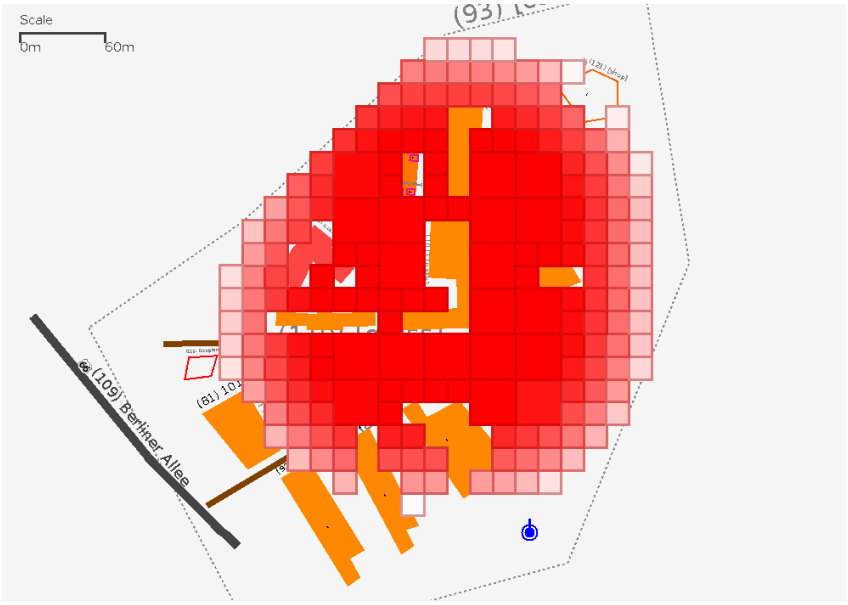


Figure 2.9: The validity of the preposition *near* when the landmark is the 'comma shaped' building in the middle of the image, and the referent is the observer (the circle at the bottom of the image). The *lowest common enclosure* of these two entities is the campus boundary denoted by the dotted line. This determines the *range threshold*, which in combination with the *size threshold* calculated from the size of the landmark object, determines the tolerated distance.

where $<$ is the antisymmetric and transitive relation representing enclosure. This constitutes finding the smallest enclosure which encloses the two objects x and y. This is analogous to finding the lowest common multiple of two integers, and indeed has the same definition as above if $<$ is redefined to mean *is a factor of*. We argue that such an entity can be used as the required range. This seems intuitive; the nearness of a chair and table is in the context of the enclosing room, whereas the nearness of a student to a laboratory is in the context of the enclosing town or city. To determine the LCE efficiently, we ascend the tree from each object in parallel, one ancestor at a time. As soon as an ancestor we added to one list is contained in the other, we return this object. In practice, this typically requires no more than a few steps.

An example of this model can be found in Figure 2.9.

2.7.5 Projective Prepositions

As identified, *projective prepositions* are those where the validity is determined by some axis in addition to some distance between the landmark and referent objects. Here we specifically address the prepositions *in front of*, *behind*, *left of*, *right of*, *north of*, and so on. Validity diminishes as we deviate from some axis. For example, suppose (as one possible model) we were to choose a single point and axis on the landmark object (often referred to as the *frame of reference*). If we were evaluating the preposition *right of*, we would choose an axis 90 degrees clockwise from the direction the observer is facing. Suppose we were to choose a point at the centre of the point as the *origin* of the axis. Then in considering the validity of our preposition for some referent, we would consider

the angle between the origin and the referent object, and compare this to the axis angle we chose.

Different models choose different axes. [Gapp, 1994] and [Olivier and Tsujii, 2004] uses the centre of the the *'bounding box'* for a landmark object as their axis, where the bounding box is the minimal rectangle that encloses the points of an object.
[Kelleher and van Genabith, 2006] identifies the flaw in the use of this approach, that choosing a point in the centre can lead to cases where an object may be considered *in front of* another despite the fact it may be *inside* it. [Kelleher and van Genabith, 2006] then proceeds to present an approach of projecting a three-dimensional object into the viewer's two-dimensional plane, finding the centre of this silhouette, and determining the point on the landmark object's 3-D surface that corresponds to this point. The advantage is that the origin of the frame is chosen at a point on the surface, which arguably serves as a better point of reference from a perceptual level. However, the paper is vague on the choice of frame for any preposition other than *in front of*; for *behind* for example, choosing a point on the object's surface visible to the observer would result in points being considered valid inside the landmark object, and distances would be measured from the front of the object, not from the back of the object. For similar reasons, the model does not extend to prepositions such as *left of* and *right of*.

We argue all such choices of frame are poor if we wish to consider the mass rather than merely the location of landmark objects; limiting our model to one frame of reference per landmark object/preposition results in numerous problems that we will explore. For example, Figure 2.10 illustrates the validity across space for the preposition *in front of* for 3 different choices of frame: the centre of the landmark object as per [Gapp, 1994], the 'silhouette approach' as per [Kelleher and van Genabith, 2006], and our approach which chooses the frame of reference dynamically depending on the position of the referent. We now explore four questions. The first considers the type of frame of reference to employ, as per the discussion in Section 2.1. These consisted of the *intrinsic*, *extrinsic* and *deictic* frame of reference. Secondly, we determine how we can use this choice (coupled with the position of the referent object) to determine the origin and axis of the frame in order to calculate the distance and angulation deviation. Thirdly, we consider how this angular deviation and distance is used to determine the validity. Lastly, we consider other qualitative considerations that may effect the validity, such as visibility considerations.

2.7.5.1 Which frame of reference?

To determine the effect of a referent's location on the frame of reference, we incorporated questions into our survey which varied the position of the observer relative to a landmark with a salient front face. For example, when the observer was to the left or right side of the landmark, we wished to establish whether *in front* was valid in the cases where the referent was between the landmark and observer (i.e. true according to the *deictic* frame of reference), and when adjacent to the front face of the landmark (i.e. true according to the *intrinsic* frame of reference). As demonstrated by Table 2.1, it was found the deictic frame of reference was more dominant than the intrinsic one (although this may be a reflection of the magnitude of the salience for the natural front side of the landmark object), although both were generally considered to be valid. In addition, it was found that confidence decreased as the observer moved from in front of the object to behind

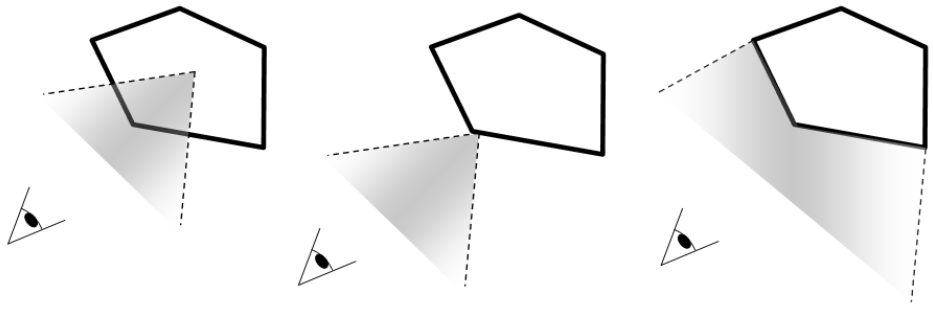


Figure 2.10: How the model for 'in front of' varies depending on the frame of reference chosen. In the first image, the frame of reference (from which distance and angular deviation metrics are computed) is chosen to be the centre of the landmark object, as per the COM model. In the second, the point is chosen on the surface of the object at the centre of the observer's perceived 'silhouette' of the object, as per [Kelleher and van Genabith, 2006]. In the last, the geometry of the landmark is better utilised by adapting the frame of reference depending on the position of the referent.

Table 2.1: The level of confidence for the preposition *behind* as the position of the observer moves around the landmark object. The *deictic* frame of reference indicates that *behind* is relative to the position of the observer, whereas the *intrinsic* frame indicates that *behind* is relative to the salient front face of the landmark object.

Confidence		Pos of observer (relative to landmark object)		
		Front	Side	Back
Frame of reference	Deictic	1	0.905	0.773
	Intrinsic	1	0.545	0.318

the object.

2.7.5.2 Which point of reference?

We proposed to adapt the choice of the frame of reference according to the position of the referent object. To achieve this, we consider the projection of the referent object onto the surface of the landmark according to some angle. That is, we can envisage firing a ray from some representative point of the referent object (we use the centre) at a given angle, and determine the point of collision on the landmark's surface. We determine the angle based on the context, as illustrated in Figure 2.11:

1. If the intrinsic or extrinsic frame is used, we fire the ray adjusted from the salient axis of the referent object. If for example the object was facing in a Northerly direction, we would fire the ray towards the South. The result is that if the referent object is in a region that is indeed North of the landmark, the ray will collide with the North surface of the landmark. The adjustment depends on the preposition. For prepositions which result in use of the extrinsic frame, we would use 180° for 'north of', 90° for 'west of', etc (where the angle is a bearing working clockwise

from North). For the intrinsic frame, where the landmark has a salient axis θ, we would use $\theta + 90°$ for 'left of', and so on.

2. For the deictic case, the position of the observer now influences this angle. We use the angle between the observer and the centre of the landmark, then adjust according to the preposition. For 'left of' for example, we would deduct 90° from this angle.

3. In the special case when the landmark object is the observer (as in 'in front of you'), the left and right directions must be flipped.

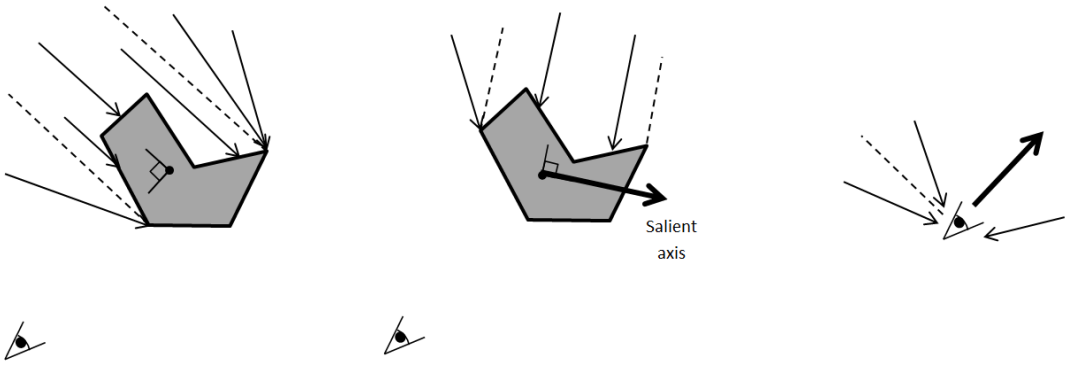
In cases where this ray collides with the surface of the object, we use the point of collision as the origin of our frame, and the angle in the opposite direction to this ray as the axis. If the ray did not collide with the landmark object, we choose the point on the surface which minimises the angular deviation from the ray, since by minimising this angle, we maximise the eventual validity based on this deviation. It is only for these rays that we have angular deviation; if the ray collides with the object, then both the angle from referent to point of collision and the chosen axis are the same, hence no angular deviation is seen. In the case where we don't collide, the angular deviation is the difference between the original direction of the ray projection, and the angle between the ray source and the point on the landmark surface that we chose. For the case when the landmark object is the observer, we can additionally leverage the size of the referent object. If we convert it to an arc representation as per the visibility algorithm in Section 2.4, then if the ray falls within this angular range, the angular deviation is 0, and if outside, the deviation is the extent it deviates from these bounds. Thus the angular deviation is based on the point on the object which deviates least from the adjusted orientation of the observer, rather than based on the centre of the object as before.

2.7.5.3 Determining the Validity

We have so far established that for given positions of the landmark and referent objects (as well as the preposition, position of the observer and the chosen frame of reference) the point on the landmark's surface from which we measure our distance and the axis by which the angular deviation can be measured. How then can we yield the validity of the relation given these two features?

The complication is that, under certain circumstances validity does not often depreciate with distance. For example, in *'It's behind that lab over there'*, there are two interpretations: one that the referent is in close proximity to the laboratory, and the other in which the relation was used merely to narrow down the angular range of the referent. In the latter case, the bound of distance separating the landmark and referent is theoretically infinite. In some cases, certain factors can be indicative of the case; for example, if using the intrinsic frame of reference (as in *in front of the shop*), one would expect the tolerance of distance to be finite based on convention. Similarly, the tolerance is finite if we use adverbial modifiers such as *just*. In cases where the landmark object is the observer, the tolerance of distance is finite, since the restriction is only necessarily imposed on the angular range.

When the deterioration with distance is finite (which we presume for simplicity is always the case), we once again use a sigmoid function to yield the desired relationship



(a) Figure (a) (b) Figure (b) (c) Figure (c)

Figure 2.11: The ray projections from the referent to the landmark object (indicated by arrows) in a variety of different scenarios. In (a), ray projection is based on the deictic frame and the preposition *left of*. Rays are fired perpendicular to the angle between the observer and the landmark, although if not colliding with the landmark (i.e. if the referent is positioned outside the dotted line), the point of smallest angular deviation is found. (b) demonstrates ray projection for *right of* when the landmark has a salient orientation and the intrinsic frame is used. Finally, in (c) the observer is itself the landmark object, with the axis (the dotted line) obtained according the preposition *left of*.

between distance and validity. $1 - \sigma(z) = e^{-z}/(1 + e^{-z})$ gives a function which tends towards 1 for $z < 0$, tends towards 0 for $z > 0$ and is 0.5 for $z = 0$. When using the distance d, if we specify some distance $d_{0.5}$ at which the confidence becomes 50%, then we can use $1 - \sigma(d - d_{0.5})$. Similarly to the model for *by*, the tolerated distance is based largely on the size of the landmark. If we take the base width w of the landmark object, we can instead define $d_{0.5}$ to be how many widths away from the landmark the referent is when the validity becomes 0.5; for example, if the landmark is 10m in size and $d_{0.5} = 1$, then at 10m from this landmark the confidence would be 50%. We can control the rate of deterioriation of the sigmoid/confidence by introducing an additional constant, i.e. $1 - \sigma(\alpha(d - w \cdot d_{0.5}))$. The larger the value of α, the steeper the slope of the sigmoid and thus the faster the deterioriation. If we again assert that the mapping from angular deviation to validity can be obtained using the sigmoid function, and that validity based on distance and angular deviation are independent, then we can obtain our overall validity by taking the product of the two sigmoids:

$$\psi = \frac{e^{-(\alpha(d - w \cdot d_{0.5}) + \beta(\theta - \theta_{0.5}))}}{(1 + e^{-\alpha(d - w \cdot d_{0.5})})(1 + e^{-\beta(\theta - \theta_{0.5})})} \tag{2.10}$$

Our survey did not encompass this particular quantitative data, thus values for α, β, $d_{0.5}$ and $\theta_{0.5}$ were chosen that worked well in practice.

2.7.5.4 Non-Polygonal Landmark Objects

If the landmark object was modelled as a line (e.g. a path or road), the tolerance of distance is typically lower than that of polygonal objects relative to the width of the line. Again, validity may or may not deterioriate with distance. *'Left of the road'* would

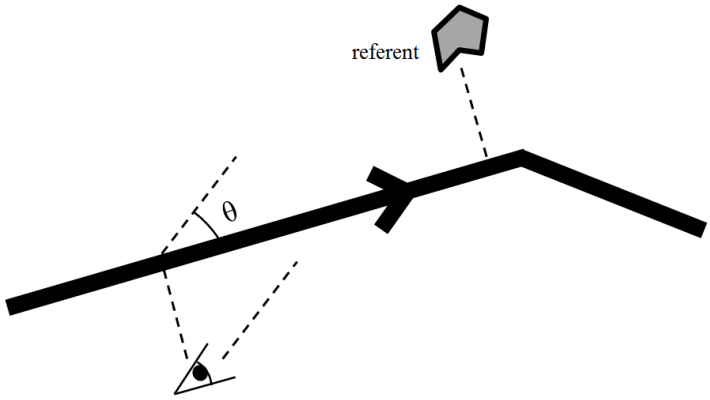


Figure 2.12: If we find the nearest point on the line (the landmark) to the observer, we can find the positive direction of the trajectory (indicated by the bold arrow) since we pick the direction which minimises θ. This then allows us to establish whether the referent is *left of* or *right of* the line, by similarly finding the point on the line nearest to the referent.

typically imply some proximity to the road, although *'North of the river'* and particularly *'North of the equator'* would not necessarily do so.

For *left of* and *right of*, the axis for our frame of reference is dependent on the axis of the line, not the angle between the observer and the centre of the line as before. To determine the origin, we simply find the nearest point on the line to the surface of the referent. The axis is then perpendicular to the trajectory of the line at this point; whether clockwise or anticlockwise is determined by the direction along the line the observer is facing. To determine this, we find the nearest point on the line to the observer, and use the direction of the line (given the line flows in two directions from this point) as the *'positive'* direction which has the smallest angular deviation to the direction the observer is facing. This is illustrated in Figure 2.12. If the axis of the observer deviates too far from the axis of the line, these prepositions are not valid, just as *in front of* and *behind* are not valid if the difference deviates too far from the perpendicular (we used 45 degrees). For example, it would not be valid to say *'left of the road'* if we were facing the road, and the positive direction of the road was not otherwise made clear.

2.7.5.5 Other Qualitative Considerations

To refine our model, we consider a number of qualitative factors that could render a projective preposition invalid:

1. If the referent is the observer, and the landmark object has no salient angle. One would not usually say *"You are in front of the building"* if the building had no salient front face (unless we treated the addressee and addressor as separate observers, such that this might be interpreted as 'you are obscuring the building'!).

2. If the path in the enclosure R-Tree between either the landmark object and the observer (in the case of the deictic frame), referent and observer (again in the deitic frame), or landmark object and referent (in any frame) are blocked by an object with

an exterior shell. For example, it would be invalid to compare an object within the confines of a room with another outside it. Similarly, it would be invalid to compare two objects in the deictic frame if both objects were outside the observer's closed enclosure.

3. If the referent object is inside the landmark object or vice versa, or if the observer is inside either object. This is implicitly suggested by the model, as the assumption is that rays from the referent object are projected from outside the landmark.

4. If in the deictic frame, the landmark object occupies a large viewing angle from the perspective of the observer (i.e. they are very close to the object with respect to its size) and the preposition is either *left of* or *right of*. Typically, projective prepositions would not be used, except in the case of *in front of* and *behind*, unless the landmark is at a suitable distance from the observer. To determine whether this violation is made, we simply convert the landmark object into an arc as in the visibility algorithm to yield a visible range of the object, and have a suitable threshold for this angular range. A value of 90° was used.

5. If the landmark object is the observer, and there is no direct line of sight of the referent given the relevant angle (e.g. for *behind you*, a ray fired directly backwards does not come into contact with the object), but there is contact for one of the other prepositions. This handles cases where the observer is close to an object and it runs along one side of the observer (e.g. to their left); while usually we would allow some angular deviation such that for *in front* there need not be a point on the object that is directly in front, in this particular scenario there is some projective preposition that is more dominant. This is illustrated in Figure 2.13.

6. *Behind* is generally only used if the objects have non-zero height. In the case of flat objects, such as lakes, roads, paths, rivers and grass, *beyond* is a more appropriate preposition. *Beyond* may also be valid for non-flat objects, such as *'beyond the trees'* or *'beyond the traffic lights'*.

2.7.5.6 Summary

In summary, our model for projective prepositions required us to initially establish the frame of reference type for the projective preposition: either deictic, intrinsic or extrinsic. Based on this choice, we then determined the most suitable point of origin and axis as the frame of reference, used to determine the angular deviation of the referent from this axis, and the distance of the referent from this point. Finally, we used these two features metrics to determine a suitable measure of validity. A summary of the choices of origin and axis can be found in Table 2.2.

Figures 2.14a and 2.14b illustrate this model in action.

2.7.6 In/On/Through

Given that *in* implies binary values of validity rather than some gradation, it is considerably simpler than other prepositional models. For simplicity, our definition requires

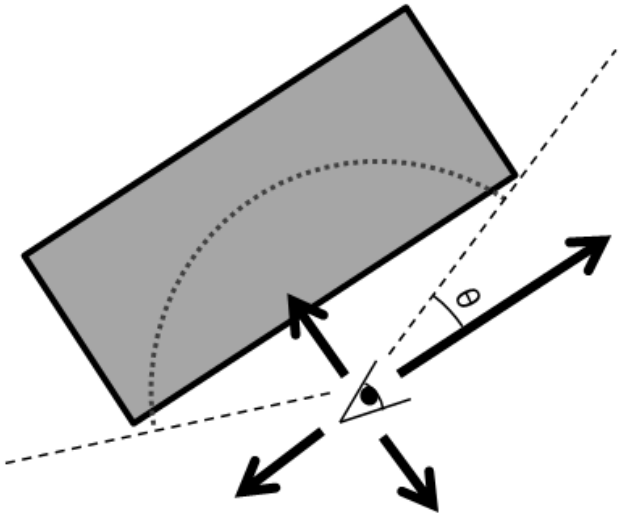


Figure 2.13: Considering the validity of *in front of you* for large close by objects: If we compare the deviation θ of the observer direction from the angular range of the arc representation of the referent object, the value is sufficiently small that *in front of you* would be deemed valid. But clearly the relation is false. Using the *Centre Of Mass* model would resolve this complication (since the centre of mass of the referent is clearly to the left of the observer), but the model would be subject to the numerous limitations that using COM brings. Instead, we consider whether another direction (*left of*, *right of* and *behind*) is contained strictly within the angular range of the referent (i.e. the ray fired collides directly with the landmark). Since *left of* does, we permit no angular deviation for *in front of*, and thus the preposition is not considered valid.

Table 2.2: A summary of the choice of axis and origin by which to determine the angular deviation and distance metrics.

Scenario	Origin	Axis
Deictic frame	Collision on surface of landmark object of ray projected from the referent, based on angle between observer and landmark object.	Opposite direction to this ray.
Intrinsic frame	Same, but angle of projection based on salient orientation (i.e. 'front face') of the object.	As above.
Extrinsic frame	Same, but angle of projection opposite to the cardinal orientation (e.g. 180 degrees in the case of *north of*)	As above.
Observer is landmark	The position of observer.	Direction of observer. Note that left and right are swapped from the deictic case.
Landmark is a line	Nearest point on line to the referent.	Perpendicular to line trajectory at the origin, based on alignment of the observer to the line.

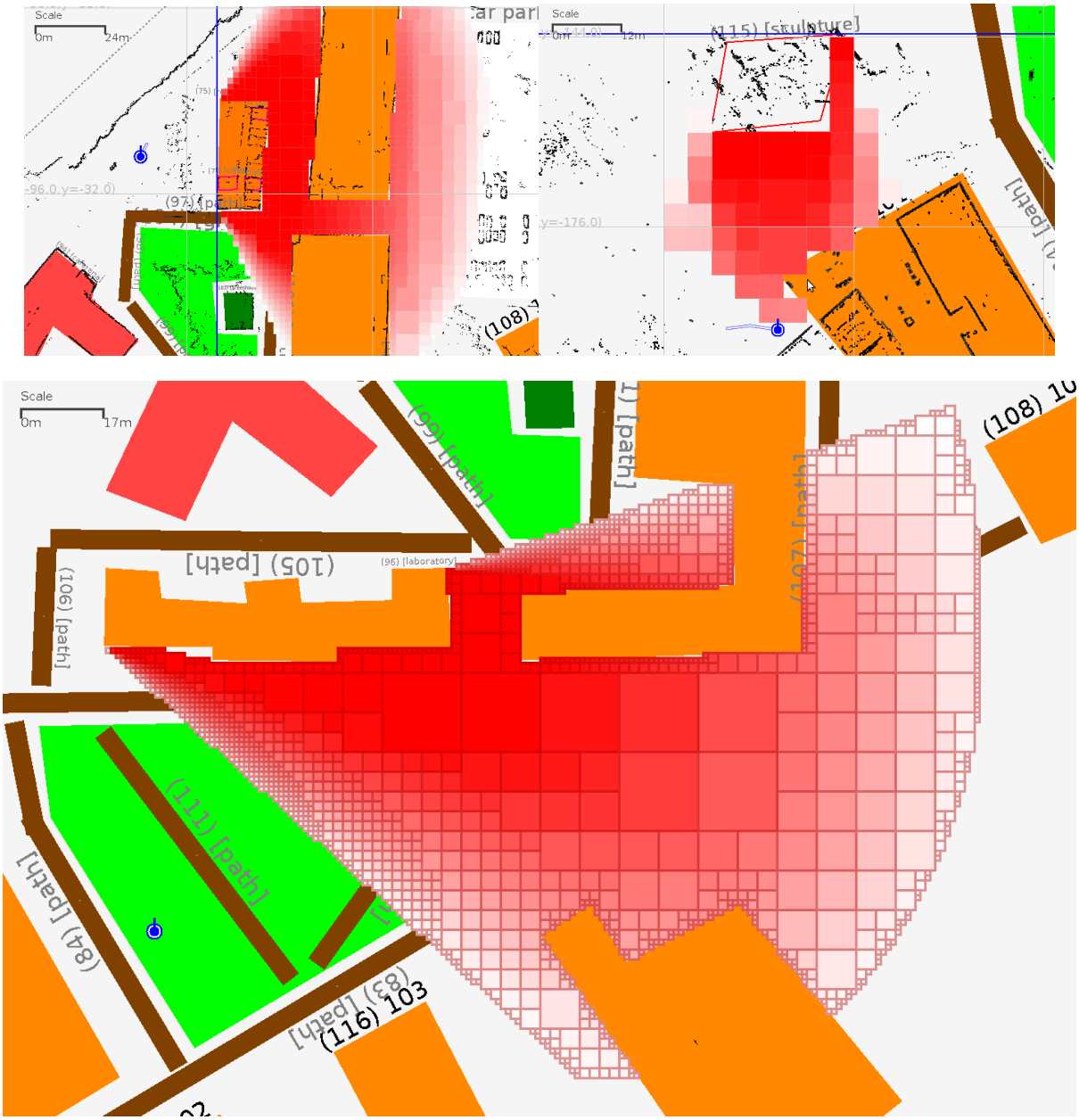


Figure 2.14: The validity for *behind*, *in front of* and *right of* as the landmark object moves in position, indicated by the intensity of red. The observer position and direction is indicated by the blue circle and its protruding line. The landmark is in the centre, top and left of each of the images respectively.

all vertices of the referent object to be within the span of the landmark object, with no overlap permitted. To determine whether a point occurs within a polygon, we can compute it geometrically by firing a ray horizontally from infinity, level with the point. We simply count the number of collisions with an edge of the landmark object on its journey to the point in question, and observe the parity. Since each collision switches the ray from inside the polygon to outside, and vice versa, an odd parity indicates the point is inside the polygon.

In is usually only applicable for landmark objects with non-zero height; it would be erroneous for example to say *"The tree is in the grass"*. For flat polygonal landmark objects, *on* is a more suitable preposition, and similarly, the preposition *through* is more suitable when the referent is a line (e.g. 'the path through the grass'). For non-flat point objects (as previously defined, objects with no coordinate information other than placements, and an assigned radius), we can reuse the same model, and compute containment simply by finding if all the points of the referent have a Euclidean distance from the centre of the landmark which is less than its radius.

When the landmark is modelled as a line, the preposition *on* behaves like the preposition *by* (an example of a *sense shift* as described in [Herskovits, 1986]). Since line objects (e.g. roads) are typically flat, then the model in Equation 2.7 can be simplified to:

$$\psi(i,j) = clamp(k_c - k_m d \frac{\log(w + k_w)}{w}) \tag{2.11}$$

2.7.7 Enclosed By

Our model for *enclosed by* offers a variant of the preposition *in* when the referent is not strictly contained within the landmark object, but is enveloped by it. We can again employ our arc representation of the landmark object used in Section 2.4, but using the centre of the referent as the reference point rather than the observer. This gives a measure of the angular range by which the referent is surrounded by the landmark. We can then once again employ the sigmoid function so that the validity increases as the angular range becomes larger. This model is illustrated in Figure 2.15.

2.7.8 Down

References are often made to objects that are further up the trajectory of the current line the observer is bounded by, such as a road or path; the *down* preposition embodies this notion. Like *between*, the *down* preposition is a tertiary relation. One landmark serves as a *bound* which identifies the trajectory for which *down* is defined, usually a path or road. A second landmark identifies the start of the region along the trajectory for which *down* starts to apply, which is the observer if not specified (i.e. *'down the road [from you]'*). Our validity model is based on two features, the projected position along the trajectory (i.e. the nearest point on the trajectory to the referent, which we refer to a p_{ref}), and the distance of the referent from this point, as indicated by Figure 2.16.

The former allows us to determine whether the preposition is valid, based simply on whether the point p_{ref} is on the correct side of the point p_{lan} obtained by similarly finding the nearest point on the trajectory to the landmark object. As with our *left/right of* model when the landmark was a line entity, we determine the 'up' direction of the

Figure 2.15: The validity of the preposition *surrounded/enclosed by* for the landmark in the centre of the image.

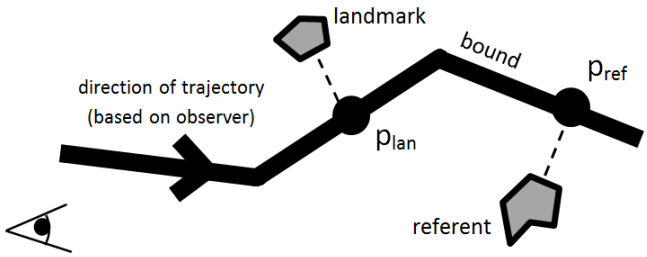


Figure 2.16: The features used for the *down* model: The projection of the landmark onto the bound p_{lan}, the projection of the referent onto the bound p_{ref} and the 'up' direction of the trajectory based on the angle of the observer.

road based on the orientation of the observer. By following the trajectory of the line it is then easy to see if the referent is at a valid position along the trajectory. The latter of the two features can be calculated by finding the distance between the referent object and p_ref. Since *'X is down Y'* implies that *'X is on Y'*, we can use the *on* model for lines from Section 2.7.6 to determine how validity deterioriates with distance from the line.

2.8 Summary

In this chapter, we explored the various considerations involved in producing models for spatial prepositions evaluated over potentially complex environments. These models give some notion of validity of the preposition given some referent (the object being referred to) and one or more landmark objects. Some of these considerations universally affect all models, in particular visibility, for which an efficient algorithm was presented by making reasonable simplifying assumptions about the observer's perception of objects.

We discussed how objects in the environment may be represented, and how the use of R-Tree indexing to organise them could both improve efficiency (e.g. to reduce the set of objects to consider in the visibility algorithm) and have computational uses (e.g. to calculate the 'range' of two objects for use in the *near* model).

We presented specific models for a variety of spatial prepositions, most of which had parameters tuned by experimental data. This included prepositions often referred to as topological (i.e. those based on merely distance, such as *by* and *near*), projective prepositions such as *left of* and *behind*, and other prepositions which fall into neither category, such as *between*. These accommodate the polygonal information associated with objects often neglected by many existing models. We also identified the distinction between validity models and *relevance* models, often coalesced into one in existing models, and explored some of the features that could help establish such a measure.

These models could in some respect be considered as *atomic*, in that they evaluate 'units' of spatial language which are not decomposable into smaller parts. In the next chapter, we will explore how these prepositional models can be employed in a number of higher level algorithms that either interpret or generate longer spatial expressions composed of these units.

Chapter 3

Spatial Algorithms

3.1 Overview

In the previous chapter, we explored a variety of probabilistic models that can be used for a variety of different spatial prepositions, both in modelling their validity and relevance. We defined a function $\psi_z = p(\top_z|x)$ for each model, where z is the relation, $\top_z$ is the event that z is considered to be true, and x is the pose of the observer/robot. Thus ψ_z represents the varying degree of validity of the relation, interpreted as a probability between 0 and 1. There are certain contexts in which these low-level models may be employed directly, such as in discourse where the user asks if a particular spatial relation holds (as in *'is it in front of me?'*). However, these models are more typically employed in higher-level algorithms. This chapter concerns how these models for these units of spatial language can be further used in a variety of algorithms and settings.

Such applications broadly fall into two categories, those which involve the *generation* of spatial language, and those which involve the *interpretation* of spatial language. The former of these tasks has received a considerable amount of research attention, where the task is generating a spatial expression which uniquely identifies a particular entity amongst others in the environment. Issues involved in such a task are the conciseness of the description generated, efficiency considerations and whether we can accommodate binary/tertiary relations rather than just unary properties such as colour and size. We explore this task in Section 3.2. These algorithms disambiguate on *identity* rather than location, which is problematic when considering questions such as "Where am I?" (where the identity of the observer is not in question, but their *location*). We propose an algorithm in 3.3 which disambiguates instead on the location of the entity.

The converse task is to interpret existing spatial expressions. *Geographic search* is the well-established task of retrieving a set of entities given a spatial query (perhaps with some degree of confidence to matching the expression). This differs from the discussion in the previous chapter given that spatial expressions might involve multiple relations and relations embedded within each other. An associated task is to treat these expressions as assertions rather than queries, where the user is providing a description of the environment for the system to learn from its environment. While many approaches involve the task of learning *semantic* information about the environment [Anguelov et al., 2004], we instead analyse how spatial descriptions of an object can be used to gradually map the space in which the object may occupy. We explore this in Section 3.5.

Name	Theoretical Complexity	Typical Runtime
Full Brevity	NP-hard	$n_a^{n_l}$
Greedy Heuristic	Polynomial	$n_a n_d n_l$
Incremental Algorithm	Polynomial	$n_d n_l$

Table 3.1: The different variants to [Dale, 1989]'s algorithm to generate referring expressions (involving only simple attributes) as outlined in [Dale and Reiter, 1995], where n_a = the number of properties true of the variant, n_d = the number of distractors and n_l = the number of attributes used in the final referring expression.

One particular key issues common to all these algorithms is that of efficiency. Environments may consist of thousands or millions of objects, and thus suitable data structures and optimisations must be used to ensure that these algorithms are tractable for real-world settings. All but the last of these tasks are actively used by the EUROPA platform, and thus our presentation of these algorithms are not a theoretical excursion, but with the view of producing tractable workable implementations that can be used within the context of spatially motivated dialogue.

3.2 Generating Referential Expressions

3.2.1 Existing Research

Of these tasks, that of generating referential expressions constitutes the largest body of research. Given a set of objects contained in the environment, and a referent we wish to describe, the task is to generate a description which can uniquely and unambiguously identify this object amongst all others. These other objects are commonly known as *distractors*. Work in this area falls broadly into two categories, those which describe objects based on unary predicates restricted to the object itself, such as colour and size, and those which describe objects using relations, such as the spatial prepositions we explored in the previous chapter. The origins of such research stem mostly from [Dale, 1989], although earlier focusing more on the cognitive aspects of referring expressions was conducted by [Appelt and Kronfeld, 1987]. This work focused solely on unary predicates to disambiguate the referent.

For both types of descriptions, we can subcategorise approaches into their computational complexity and the optimality with respect to the length of generated descriptions, the notion being that, in light of conversational axioms which we explore Chapter 4, shorter descriptions are preferred over longer ones (provided of course both are fully disambiguating). [Dale and Reiter, 1995] describes this taxonomy, a summary of which can be found in Table 3.1. This perhaps constituted the first work which transcended mere linguistic factors and analysed a variety of approaches in terms of their computational complexity

These approaches pertain to the optimality with respect to the length of the final referring expression. If we wished to generate an expression of the optimum brevity, we could use an exhaustive brute force approach to generate the optimum expression: we start with the set of all (distracting) objects, and attempt to apply each possible attribute

of the referent, filtering the set according to objects for which the attribute is also true. If this leads to the distracting set being reduced to a single object, i.e. the referent, then we successfully return the expression (i.e. the attribute). Otherwise we try all pairs of possible attributes, then triples and so on. Clearly this gives rise to an exponential number of steps; if the length of our final description is n_l,for each attribute there was n_a possibilities and there are n_d distractors, then we assessed $n_a n_d n_l$ descriptions. This approach is shunned due to its intractibility for any realistic domain, which significantly outweighs the necessity to ensure the description is as short as theoretically plausible.

The *Greedy Heuristic* differs in that it doesn't backtrack, so that at each iteration, it finds the attribute which reduces the number of distractors by the largest extent, and retains it. The iterations continue, adding more attributes, until there are no distractors. We have n_l iterations, where for each we tested each of the n_a attributes on the n_d distractors, thus the runtime is $n_a n_d n_l$.

The final *Incremental Approach* is motivated by psychological factors; i.e. that human speakers often use redundant modifiers in the expressions they construct, and that partial descriptions are often uttered before even the environment is scanned [Pechmann, 1989]. This approach therefore leverages a *taxonomy* of all possible attributes in terms of importance or salience, such that for example colour might be preferred over size. The Incremental Approach scans down the taxonomy, applying attributes that are true of the referent. If we have no distractors given the attributes chosen so far, we terminate with these accumlated attributes. This significantly reduces the complexity over the Greedy approach. We discuss the merits of this approach later in terms of the quality of descriptions generated in the next section.

This taxonomy can also be applied to algorithms which generate descriptions involving relations also. [Dale and Haddock, 1991] was to first to propose such an algorithm, which can be summarised as follows: We maintain a property set P_x which contains all the spatial or semantic constraints of some object x (e.g. where we to have some predicate *Between(v,x,y)*, this would be an element of P_x). We also have a *referent stack*; a stack of referents we wish to describe (since we recursively need to uniquely identify each object in the expression, e.g. *'the cup on the table on the floor by the bin...'*). Lastly, we have a constraint network *N*, consisting of the predicates accumulated in the description, and the sets of objects which these predicates C_x for each variable x used for which these predicates are satisfiable, e.g. $\langle cup(x), in(x,y), [C_x = \{c_1, c_2\}, C_y = \{b_1, b_2\}\rangle$ would correspond to the description *'the cup in something'*, where two different cups could fit the description. We start with *Stack = [Describe(r,v)]* to embody the goal of describing some object r using some variable v. P_r is the set of all facts true about r, and $N = \langle\{\}, |C_v = \langle allentities\rangle\rangle$ indicates that we currently have no description, and there are no constraints on what the variable v can be. We then iteratively perform these 3 simple steps:

1. We first check whether the description we have constructed so far uniquely identifies the object (i.e. $|C_r| = 1$ for the object r on the bottom of the referent stack), and if so, remove this describe goal from the referent stack. If the stack is now empty, we have finished.

2. If not, we choose the most 'useful' fact and add it to the description, represented by the $\oplus$ operator. This amounts to picking a constraint that minimises the con-

straint set C_x for the variable x representing the object being described, as we did previously for the attributes-only algorithm.

3. The description in N is updated with this fact, and we add a *Describe(z,v_z)* goal to our referent stack to describe any objects introduced into our expression, if any.

The full algorithm is presented in Algorithm 4.

There are a number of other extensions to these algorithms. [Stone, 2000] allows the generation of expressions involving *sets* of entities, for example 'the squares clustered on the lower left'. [Van Deemter, 2002] extends the set of logical connectives/operators by permitting negation, e.g. 'the car not by the tree'. Work has also been conducted on the extent to which descriptions generated by GRE algorithms corroborate with human descriptions. [Viethen and Dale, 2008] identified that humans used relations even when not strictly necessary to disambiguate an object, and that the salience of landmarks in the environment encourages the use of such redundant relations.

3.2.2 Implementation and Modifications

An inherent assumption with the algorithm and its variants so far is that we were able to enumerate the predicates and relations true of our referent object. For large dynamic environments, this would be entirely intractable, given that we would have to calculate the validity of $O(n^2m)$ spatial prepositions given n objects and m different prepositions (and more in the case of tertiary relations such as *between* and *down*). Extensions of the algorithm therefore assess the validity of prepositions dynamically only as they are needed.

We specifically make a distinction between the *distractor set* and the *environment set*. As previously stated, the distractor set represents the set of objects the generated description must distinguish the referent from. In contrast, we introduce the notion of the *environment set* to be the set of objects available for use in relations. This specifically relates to our consideration of prepositions being dynamically tested as the need arises; at the point in the algorithm where we have a relation involving variables (and potentially specific objects), we wish to know what instantiations of the involved variables are such that the relation will be true. We can use the environment set to restrict the domain of possible instantiations we test; if our environment set is of size e and we have n variables, then there are e^n relations to test. In some cases it is possible to use spatial considerations based on the particular preposition to restrict the set of relations to validate further; if the referent object of the relation *in front of* is instantiated and the landmark object a variable, we could consider only the intersection of the environment set with objects in the region behind the referent. We have however not implemented this optimisation, and use the environment set instead to constrict our set of valid relations. We choose the environment set to be the set of *visible* objects (using the visibility algorithm we described in Section 2.4) as well as any other salient objects in the environment. This is advantageous, since the description will not include references to objects that either can't be seen by the user or are not likely to be already known about. It is also occasionally possible to restrict the distractor set to the set of visible objects also; if the referent object is visible, we would only typically need to distinguish it from other visible objects.

Algorithm 4 The *Greedy* version of the description generation algorithm for expressions involving relations, as per [Dale and Haddock, 1991]. Note that in Steps 1, 2 and 3, r and v relate to the current $Describe(r, v)$ on the top of the stack.

1. Check Success

if Stack is empty **then**
    return L as a DD
else if $|C_v| = 1$ **then**
    pop Stack & goto Step 1
else if $P_r = \emptyset$ **then**
    fail
else
    goto Step 2
end if

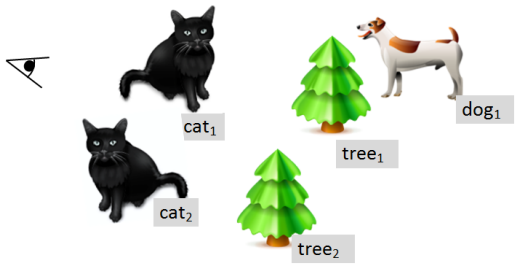
2. Choose Property

for each property $p_i \in P_r$ **do**
    $p'_i \leftarrow [r \backslash v]p_i$
    $N_i \leftarrow N \oplus p'_i$
end for
Chosen predication is p_j, where N_j contains the smallest set C_v for v.
goto Step 3

3. Extend Description (w.r.t. the chosen p)

$P_r \leftarrow P_r - \{p\}$
$p \leftarrow [r \backslash v]p$
for every other constant r' in p **do**
    associate r' with a new, unique variable v'
    $p \leftarrow [r' \backslash v']p$
    push $Describe(r', v')$ onto Stack
    initialise a set $P_{r'}$ of facts true of r'
end for
$N \leftarrow N \oplus p$
goto Step 1

These variants and extensions all are motivated by principles known as the *Gricean maxims* [Grice, 1975] (axioms associated with 'effective' conversation, which we explore in Section 4.1) as a heuristic in order to motivate its choice of descriptions: the *principal of adequacy* (a conversational manifestation of the logical principle of 'soundness'), which ensures the generated expression can identify the intended reference, and the *principal of efficiency*, which implies that the generated expression should be no longer than necessary. The latter clearly relates to the variable (i.e. description length) we optimise over, while the former is honoured providing the soundness of the algorithm in its terminating conditions, i.e. that the referent is compliant with the accumulated constraints and there is no distractor object for which the constraints are valid, thus ensuring global uniqueness.



x	y
cat_1	$tree_1$
cat_2	$tree_2$

Figure 3.1: A simple demonstration of the erroneousness in using constraint sets to represent constraints. The table shows the constraint tuples at the point in the algorithm where the description is $cat(x) \wedge inFrontOf(x, y) \wedge tree(y)$. In the original algorithm, the constraints sets would be $x = \{cat_1, cat_2\}$ and $y = \{tree_1, tree_2\}$, i.e. as independent sets. When introducing the dog, we would have to check all combinations of dogs and cats to check the new set of constraints are met (i.e. 4 combinations) rather than just each tuple (2 comparisons).

A note should be made about implementation of the description algorithm and its variants. Despite the extensive research from a variety of sources, no evidence was encountered that the algorithm had been implemented at all, with 'evaluation' merely revisiting the variety of considerations that motivated the algorithm or its extension. Especially given that a large emphasis of much research is efficiency and decreasing the runtime complexity, it seems somewhat surprising that the algorithm never surpasses the theoretical. One perhaps misleading element of [Dale and Haddock, 1991]'s original algorithm (for descriptions involving relations) is the way the constraint sets for each variable in the description so far are erroneously presented as being independent, suggesting that valid instantiations of variables can be taken from the cross product of these constraint sets. This doesn't affect the soundness of the algorithm's output since the

Table 3.2: The taxonomy of relations. For the incremental variant of the description generation algorithm, these relations are tested in order of their rank.

Rank	Example	Name
1	$oakTree(ref)$	Object subtype if applicable.
2	$tree(ref)$	Object type (e.g. 'tree').
3	$belongsTo\_Bob(ref)$	Other key attributes integral to the description.
4	$behind(ref, \langle you \rangle)$	Two argument relations where the landmark object is the observer such as 'behind you'. The priority of relations are: projective prepositions, by, near, in, on, through.
5	$behind(\langle you \rangle, ref)$	Similarly, two argument relations where the referent is the observer, as in 'the object you are behind'.
6	$behind(ref, e57)$	General two arguments relations not involving the observer.
7	$between(ref, \langle you \rangle, e51)$	Three argument relations such as 'between' and 'down'.
8	$behind(e57, ref)$	Binary relations in which the object we are generating relations for appears as the landmark rather than as the referent, i.e. *"The grass the car is on"*.
9	$blue(ref)$	All other attributes.

condition for popping elements from the referent stack is based only on the size of the constraint set. However, from an implementation perspective it results in solving the *Boolean Satisfiability* (or SAT) Problem at each iteration, given we must test the validity of our accumulated relations for each element from the cross product of the constraint sets, when in reality the size of the overall constraint space may be much smaller. In order to avoid this problem, our constraint space is instead represented as a set of tuples, where each tuple is a valid instantiation of all variables, as illustrated in Figure 3.1. It means when applying a new constraint (corresponding to the $\oplus$ symbol in the algorithm), whenever the relation contains variable(s) already in the constraint table, we use the method previously explored to yield valid instantiations of the relation, resulting in a constraint table just for this relation, and use it to filter our current constraint table for the overall description (with the effect of discarding tuples that aren't satisfied by the additional constraints imposed by the added relation). For any new variables introduced in the relation, we take the cross product of the two tables. In both these cases, the effect is the same as that of the join operator $\bowtie$ found in database theory.

Our implementation allows switching between the greedy and incremental variants of the algorithm, in addition to optionally allowing the *head-category-first strategy* found in [Dale and Haddock, 1991], which immediately applies the category (i.e. object type) of a new object added to the referent stack, regardless of whether it reduces the size of the constraint sets. For the incremental algorithm, we used a taxonomy of relations as outlined in Table 3.2, which is a slightly expanded form of that in [Dale and Reiter, 1995]. Tertiary relations such as *between* appear with lower rank merely for efficiency reasons, since the number of relations to test for each preposition is squared in number.

A Gricean axiom perhaps neglected however is the *principal of manner*, which states that lack of perpescuity and obfuscation should be avoided. As identified by [Horacek, 1997],

the algorithm (and variants) places emphasis only on the soundness and length of generated descriptions, with lack of regard of the structural complexity and communicative adequacy for human interpretation. These problems might be that of organisation (*"The bottle which is on the table on which there is a cup which is besides the bottle..."*), complexity (*"The large, red, speedy, comfortable car"*) or scoping (*"the cup which is besides a bottle which is on a table which is left of another table and which is empty"*). It solves these problems in part by making checks for these scoping problems and rejecting descriptions that suffer from them. Our modification to the original algorithm is much simpler: we relax the restriction of considering relations only concerning the object at the top of the referent stack, and require only global uniqueness rather than local uniqueness. For example, in the expression *"The table on which there is a glass and cup"*, the glass and cup may not be uniquely identifiable (given there may be other glasses and cups on tables in the environment), but by allowing addition of the relation $on(cup_1, table_1)$ before uniquely identifying the glass (as the original algorithm would have done), global uniqueness is maintained for the table, and a much simpler expression is yielded.

To allow the resulting output to be linguistically realisable, we convert it into a tree structure, with the nodes representing entities and the edges representing relations. To achieve this, we initially create (initially unconnected) nodes for each variable, with the node corresponding to the variable assigned to the referent object used as the root of the tree. It is then simply a case of moving through the relations and creating edges between variables involved in the relation. The edges can be reversed (using a special flag), and occurs when the lexicographic ordering of the variable letters in the relation is reversed, providing the letters for new variable names were generated in lexicographic order. This is illustrated in Figure 3.2.

Finally, we note that the algorithm requires relations to be in boolean form, whereas the prepositional models we have presented are probabilistic. We simply take values above 0.5 to be true and false otherwise. We leave the potential variant of a description algorithm that encapsulates the continuous variation in validity as an area of future research.

3.2.3 Examples

We now present a few examples of the algorithm in operation, using some of the variants proposed, and identify any problems encountered. All the examples use the environment in Figure 3.3, with the referent indicated as the numbered object, and the pose of the observer indicated as the numbered arrow.

Example 1: The steps of the first case (using the Greedy variant of the algorithm) can be found in Table 3.3. The outputted description is *class_lab(x), LeftOf(x,y), observer(y), Near(x,z), observer(z)*, which when realised in textual form by the natural language generator, produces:

'the nearby lab left of you'

The algorithm makes use of the *head-category-first strategy*. For most objects, this header predicate is the class of object (e.g. tree). Since we have introduced the observer as an object, its corresponding header predicate is simply *observer(x)* to identify the object as such. The description is both correct and natural, and the intended object

1. $table(x) \wedge on(y, x) \wedge bowl(y) \wedge by(y, z) \wedge cup(z)$

2. $table \xleftarrow{on} bowl \xrightarrow{by} cup$

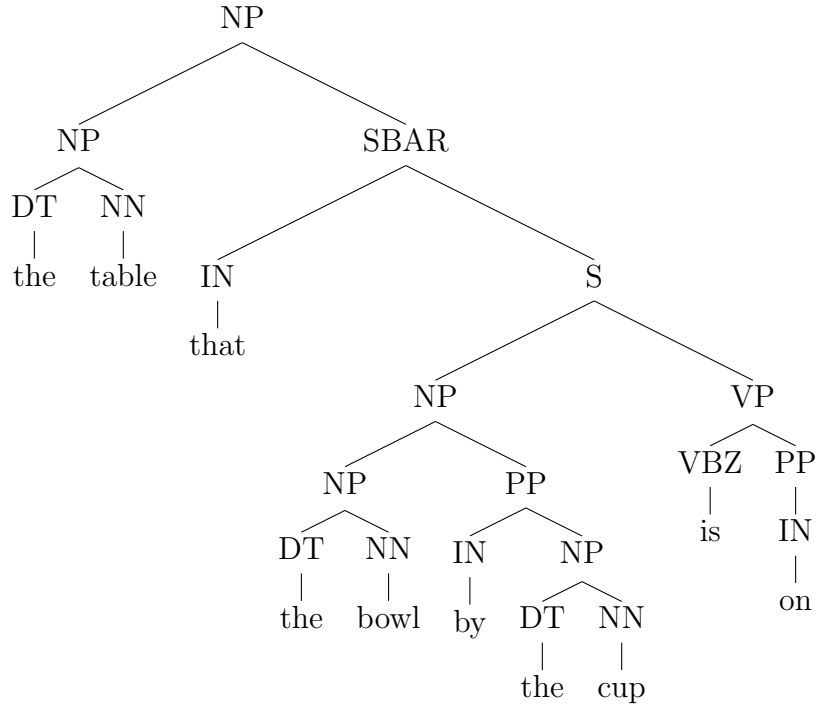
3. 

Figure 3.2: The various stages involved in realising a description using natural language. We start with (1) as an outputted description, where the variable x corresponds the referent. (2) is a tree where the root node is on the left. While the arrows would typically point rightwards from parent to child, the arrow between *table* and *bowl* is reversed due to reversed lexical ordering of variables in the relation *on(y,x)*. It would not be valid to make the bowl the root of the tree, since the overall description is referring to the table. This leads to the more complex syntactic structure observed in (3).

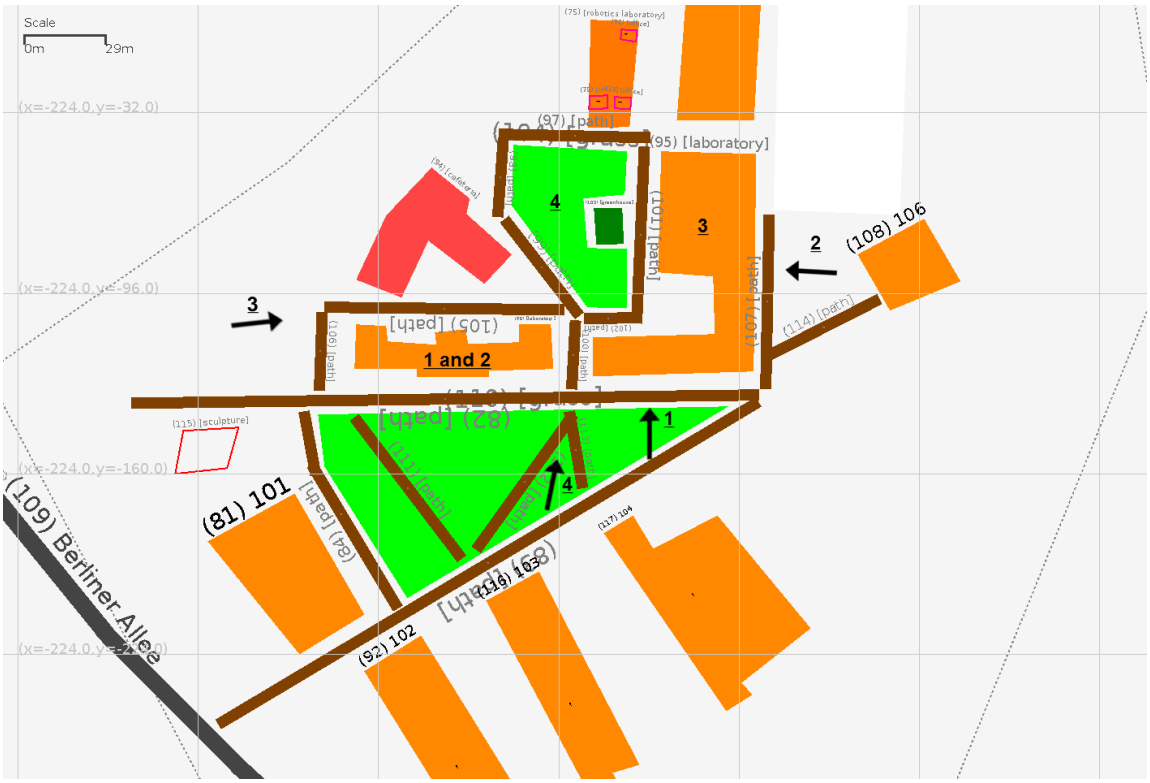


Figure 3.3: The environment used for the examples in Section 3.2.3.

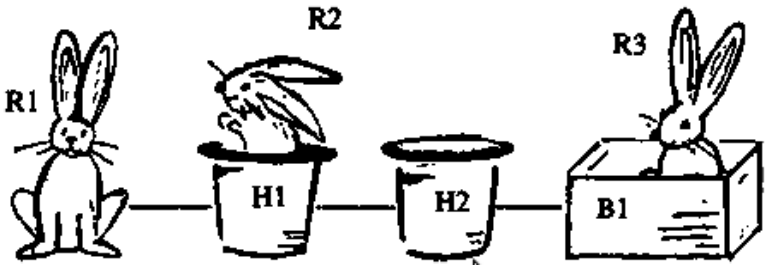


Figure 3.4: An environment used in the second example of Section 3.2.3, used to demonstrate an alternative strategy to identifying landmark objects.

Table 3.3: The steps carried out in generating the description for Example 1.

Referent Stack	Constraints	Comments
Describe(e96,x)	N = { class_lab(x) }, C_x = {e75, e79, e81, e92, e95, e96, e108, e116, e117}	The head category is applied, in this case the class of the object.
Describe(e96,x)	N = { class_lab(x), LeftOf(x,y) }, C_x = {e81, e96, e116, e117 }, C_y = { you, e92, e116}	The relation which minimised the constraint set was LeftOf(e96,you).
Describe(you,y), Describe(e96,x)	N = { class_lab(x), LeftOf(x,y), observer(y) }, C_x = {e81, e96 }, C_y = { you }	Head category is again applied, which for the observer is just *observer*. Constraint set for y is now of size 1, so top of referent stack can be popped.
Describe(e96,x)	N = { class_lab(x), LeftOf(x,y), observer(y), Near(x,z) }, C_x = {e81, e96 }, C_y = { you }, C_z = { you, e81, e92, e96, e115, e116 }	Best relation was Near(e96, you)
Describe(you,z), Describe(e96, x)	N = { class_lab(x), LeftOf(x,y), observer(y), Near(x,z), observer(z) }, C_x = { e96 }, C_y = { you }, C_z = { you }	At this point, all constraint sets are of size 1, so we can empty the referent stack and output the description.

can immediately be identified. In this particular case, both the greedy and incremental variants yield the same description.

Example 2: In this example, we describe a different object, but towards the right of the environment. Given that most objects are obscured from view by the laboratory in front of the observer, it significantly limits the number of relations available for use in the description, given our previous requirement that relations must involve only visible objects. For the greedy variant of the algorithm, the description yielded is *'the laboratory near 104 and by another lab'*. The incremental variant however performs very poorly, generating *'the lab behind another lab and by that lab and near 104'*. The reason for the longer description is the method in which the first relation is chosen after the head category. The incremental method uses the *behind* preposition initially because of its higher priority than *near* in Table 3.2, despite its poorer disambiguating power. This demonstrates that the incremental method frequently chooses poorer descriptions when the relations available to build the description are limited (and consequently, the description produced is more complex), even if the time complexity is significantly lower.

Both descriptions however suffer a significant deficiency, that 'the other lab' in the description is not individually identified in isolation. This laboratory is theoretically identifiable when considering all the constraints in the description simultaneously, but not when considering the noun phrase encompassing this lab alone. For the example *'the lab behind the lab'* in an environment of numerous labs, only one pair of labs may match this relation, thus technically satisfying our requirement of global uniqueness. But the cognitive complexity for such a description may be too high, due to the time cost (from a human perspective) of resolving multiple objects without permitting independent resolution of each object in turn. Even worse, the referent may be a nonobservable object, where despite the landmark being observable, the preposition relating them will not be. For example, where we to generate the description *'the car park behind the lab'* with the car park not visible, and multiple labs observable, it is theoretically impossible to interpret the expression, since we can't see the car park, and thus do not know which of the labs it is behind. The original algorithm does not tackle this deficiency because it assumes simple environments where all objects are observable, and thus all relations between objects can also be observed. To illustrate, consider Figure 3.4, as found in [Haddock, 1987]. With the current algorithm, to identify the rabbit $R2$, *'the rabbit in the hat'* would have been been generated. If however we assumed rabbit R2 is obscured by the hat, and we modified the algorithm to ensure that the hat it is in is uniquely identifiable, then a description such as *'the rabbit in the hat right of another rabbit'* might be generated.

The solution involves changing the way in which our constraint sets are determined based on the relations in our description. If for example we wish to determine the constraint set C_y (i.e. the possible object instantiations of variable y in the description), we could consider only the relations in the description which involved only the variable y and variables created for the referent stack *after* y. The corresponding syntactical consequence is that the constraints on the identity of an object embodied by a noun phrase is restricted to its child prepositional phrase, rather than any prepositional phrases that occur in its ancestors. With the rabbit example, the hat landmark in the description is not disambiguated by its corresponding referent, the rabbit, since this rabbit is not

visible. As such the *in* relation is ignored with respect to narrowing the constraint set of the hat, i.e. the fact a rabbit is in it does not provide any disambiguating information. However, the latter rabbit R1 used in the description *is* visible, as is the hat it is left of. Consequently, we are able to identify the other rabbit by the constraints that appear to the left of it in the sentence.

To implement this change, we used a simple approach of using the original algorithm to generate a description, before determining if the referent is visible. If so, no change is required. If not, we recursively replace the landmark objects by generating their respective descriptions using the same algorithm. *'The rabbit in the hat'* would be initially generated. Since the rabbit is not visible, we would remove 'the hat' and apply the algorithm again to the hat to generate a new description for it. *'The hat right of the rabbit'* would be generated. Since in this case the referent (the hat) is visible, we need not apply the algorithm again, thus yielding our desired description. Examples of this can be observed throughout the next section. When combining these descriptions to form *'The rabbit in the hat right of another rabbit'*, the 'another' determiner can be inserted to improve the description by explicitly distinguishing the second rabbit from the first.

Example 3: The description generated is *'the lab behind another nearby lab'*. In the greedy variant of the algorithm, while the head category is applied to any new variables, we note that the algorithm does not introduce it until Step 3 (after the best relation has been introduced to the constraint network). As a result, considering the effect of each candidate relation without considering the introduction of head categories for new variables immediately after does not give an accurate reflection of the effect of the disambiguating power of the relation. For example suppose we were to consider a relation *Behind(a,b)*, where *a* is a car and *b* is a tree. Then after replacing the constants with variables, our measure of its disambiguating power is the number of generic *things* that are behind a tree, rather than *a car* behind a tree. Since the predicate *tree(y)* would have been added anyway in step 3 were our behind relation chosen (assuming *Describe(b,y)* was put on the referent stack), then it seems appropriate to consider the head category in our decision on the relation to introduce to the description. For this particular example, the resulting alternate description is *'the lab right of me and behind the cafeteria'*. While in this case our modification did not yield a shorter description, it more accurately reflects the interplay between the greedy strategy and the use of head categories.

Example 4: The description generated is *'the grass in front of me'*. While not a poor description, the deficiency lies in the ambiguity in the identity of the grass. Since the observer is on one patch of grass, 'in front' would be a poor preposition given the grass surrounds the observer in all directions; in general projective prepositions are invalid if the observer is within the confines of its bounds. However, when introducing a second patch, the ambiguity arises because the relation applied to the first patch is merely *inappropriate*, not entirely invalid (since some portion of the first patch is still in front of the observer). One simple solution would be to make all projective prepositions valid for flat landmark objects when the observer is on the landmark, and set the score from the *relevance* model to 0. This would require some modification of the Distinguishing Description algorithm; while constraints sets would still be based on the validity models of prepositions, and thus we still disambiguate over objects for which the relations are technically valid rather than necessarily inappropriate, we would modify Step 2 and 3 such that relations can only be

added to the description if they are both valid *and* relevant. This further emphasises the need to distinguish between validity and salience/relevance models as we identified in Section 2.6.

3.3 Generating Locative Expressions

3.3.1 Introduction

So far, we have been able to generate descriptions which uniquely identify an object amongst a set of distractors (which we refer to subsequently as the *Distinguishing Description* algorithm, or DD algorithm). The underlying assumption however is that the identity and location of all objects are known; typically research uses simple household environments where all objects can be clearly observed. If we wish to describe the *location* of the object and the object is observable, then the output of the previous algorithm is sufficient, since by yielding the *identity* of the object, the location is trivially available by way of sight. But in many scenarios, this is clearly not the case; descriptions to answer the question *"Where is it?"* must disambiguate based on location, not on identity. An example description we wish our system to generate can be found in Figure 3.5. Surprisingly, no research was encountered in trying to answer this pertinent question (except some qualitative considerations in the context of generating route directions, e.g. [Tomko and Winter, 2009]), despite the fact that the constraint that knowledge of the locations of objects are presumed to be known is a somewhat encumbering one.

Figure 3.5: An example description generated by our system was *"[The cafeteria] is in the nearby building in front of me and behind the wall"*, as found in Section 6.6. The picture shows the view of the robot at the time.

We consider the following linguistic motivations when devising such an algorithm:

1. Hearers are content with descriptions which disambiguate location to a sufficient degree; i.e. an approximate area is sufficient. This depends both on the context and the referent object. For example, the required accuracy for *"Where is Chicago?"* (for which an adequate answer might be *"In Illinois, just west of Michigan City"*) is

considerably less than that for say *"Where shall we meet?"* (for which an adequate answer might be *"In front of the restaurant"*).

2. Disambiguating within the confines of the area of the referent object does not improve the quality of the description. For example, if two descriptions yielded areas contained entirely within the target area, and one description resolved to a single point, while the other to say half the target area, these are considered to be of identical merit. The metric used to assess the quality of the description should reflect this.

The algorithm can be viewed as a *continuous* complement of the discrete algorithm in the previous section. Instead of disambiguating over objects, we instead disambiguate over points across space. While the former used the size of the constraint set as a metric for the value of a relation under consideration, we instead minimise the number of points, or some appropriate metric of the area the accumulated description disambiguates to. The analogy only deviates slightly when we consider the terminating conditions. The former algorithm terminates upon global uniqueness; that the object has been uniquely identified. However in our locative variant, we need not disambiguate to a single point, as discussed, since a degree of tolerance is permitted.

3.3.2 The Algorithm

Given these considerations, we present a conceptually simple algorithm that achieves these aims, outlined in Algorithm 5. Before we provide a more detailed analysis of its operation, we present the various variables and operators that constitute the algorithm:

- The algorithm maintains a single probabilistic field F across the span of the entire environment. This represents the validity (using the appropriate spatial models) of the description accumulated so far, where the function's argument is the positioning of the referent object at some point in the environment. Suppose that each preposition p_i in our description has an associated function $\psi_i(\mathbf{x})$, which gives us the probability that a human would consider this preposition to be true given a particular position (in 2D or 3D) $\mathbf{x}$ of the referent object. Then if we make the independence assumption that the validity of each relation is considered independently, then $F(\mathbf{x}) = \prod_{i=1}^{n} \psi_i(\mathbf{x})$.

  To illustrate, suppose in our description we were to say the referent was *in front of you, by the tree*. Then if a given point $\mathbf{x}$ was entirely valid for both the first and second relations of this description, then each relation's corresponding spatial model would yield 1, and the overall validity for this point is simply the product of these, $1 \times 1 = 1$. F is initially set to $\underline{\mathbf{1}}$, where $\underline{\mathbf{1}}$ is the function $\underline{\mathbf{1}}(\mathbf{x}) = 1$ for all x. This represents the notion that initially, when we have not yet constrained the location of our referent, all points are vacuously valid for its positioning.[1]

[1]Note that F is not a probabilistic *distribution* over $\mathbf{x}$; the sum of probabilities across space does not necessarily equal to 1, instead each point in space has a separate distribution over the overall description being valid or not valid given the hypothetical positioning of the referent at this point.

- We define an operator $\otimes$ that combines two probabilistic fields, simply defined as $[F_1 \otimes F_2](\mathbf{x}) = F_1(\mathbf{x}) \times F_2(\mathbf{x})$. This allows the definition of F to be maintained as we accumulate more relations for our description. Clearly, $\underline{\mathbf{1}} \otimes F = F$, thus when we introduce our first relation, the overall field is simply that associated with this relation.

- In the DD algorithm, N is defined to be the constraint network, i.e. a pair $\langle L, C \rangle$, where L is the set of predicates constituting the description (involving some variables), and C the constraint sets for each variable. Now, we analogously define N to be a pair $\langle L, F \rangle$, where L is again a set of predicates, but now initially involving only constants, and F is the probabilistic field as defined above. F is comparable to C given that it still defines the constraints of our description, although now it provides constraints over points instead of variables/objects, (since if $F(\mathbf{x}) = 0$ for example, then $\mathbf{x}$ has been eliminated as a candidate position for the referent object), and we provide a variable notion of validity rather than a boolean notion as before.

- The $\oplus$ operator previously introduced a new predicate to the constraint network: its action was to add a relation to L after renaming its constants to suitable variables, and updating the constraints sets to reflect the objects the conjunction of these relations were now applicable to. Again, we use an analogous definition over $\langle L, F \rangle$. We add the relation to L, except now without renaming, and use the $\otimes$ operator to introduce its corresponding field to the existing field representing the constraints of the previous relations. Thus its definition is $\langle L, F \rangle \oplus p = \langle L \cup \{p\}, F \otimes F_p \rangle$.

- In the DD algorithm, the $|..|$ operator is defined over sets, and consequently gives the cardinality of the set. We analogously define it over our probabilistic field to indicate use of a metric that conceptually represents the 'size' of the field, i.e. reflects some property regarding the probability mass or the 'total ambiguity' of the relation across all points. While more complex alternative definitions could be formulated, we define it to be the sum across the range of the field function, i.e. $\int_{\mathbf{x} \in span(ENV) - span(r)} F(\mathbf{r})$, where r is the referent, $span(r)$ gives the area that the referent spans, and ENV is the entire environment. Recall that we wish to exclude the area inside the referent object, since such points don't represent *distractor regions*, in the same way that we had distractor objects in the DD algorithm; hence its exclusion in the integral. If we were to restrict our spatial models to boolean values, i.e. 0 or 1, then this metric would have been equal to the area outside the referent object that is valid for *all* the relations in the description (i.e. the intersection of the areas).

Given these components, the algorithm is then as such:

Step 1: Check Success We earlier described our condition for successful termination as having accumulated a description that has reduced the size of the constraint set to 1 (or in the relaxed version, the size of referent's constraint set to 1, i.e. a globally unique object). Now, our terminating condition is that our continuous metric on the size of the distractor *region* (i.e. $|F|$) is below some suitable threshold. $threshold(r)$ can either be a constant, or based on the size of the referent, i.e.

Algorithm 5 The *Greedy* version of our locative description generation algorithm.

L is the accumulated description, F is the field across space and $N \leftarrow \langle L, F \rangle$

0. Initialise

$L \leftarrow \{\}$
$F \leftarrow \underline{\mathbf{1}}$, where $\underline{\mathbf{1}}$ is the function $\underline{\mathbf{1}}(\mathbf{x}) = 1$.
$P_r \leftarrow$ the set of spatial relations true about the referent r, such that r appears as the referent (i.e. first argument) of each relation.

1. Check Success

if $|F| < threshold(r)$ **then**
    goto Step 4
else if $P_r = \emptyset$ **then**
    fail or goto Step 4 as a compromise
else
    goto Step 2
end if

2. Choose Property

for each property $p_i \in P_r$ **do**
    $N_i \leftarrow N \oplus p_i$ where
end for
Chosen predicate is p_j, where $|F_j|$ has the smallest size, provided $|F_j| < \lambda|F|$ (or goto Step 4 if no predicate met this criteria).
goto Step 3

3. Extend Description (w.r.t. the chosen p)

$P_r \leftarrow P_r - \{p\}$
$N \leftarrow N \oplus p$
goto Step 1

4. Resolve landmark objects to produce final description.

Associate referent r with a unique variable v.
$C \leftarrow \langle C_v = r \rangle$
$L \leftarrow [r'\backslash v']L$
$R \leftarrow$ set of all landmark objects used in description L.
for $r \in R$ **do**
    associate r' with a new, unique variable v'
    create a new stack, add $Describe(r', v')$ and apply DD algorithm (Algorithm 4) to produce description $N' = (L', C')$
    $L \leftarrow [r'\backslash v']L \cup L'$
    $C \leftarrow C +\!\!+ C'$
end for
return (L, C)

$threshold(r) = \alpha|span(r)|$ for some suitable constant α, the idea being that larger referent objects allow a larger tolerance of ambiguity.
Another terminating condition is upon exhausting the list of predicates true of the referent object. In such a case, we can compromise by outputting the description despite not reaching the required threshold, the logic being that a 'loose' description is better than no description at all. In the DD algorithm, the equivalent scenario would be compromising with a "one of the ..." type description, where our description constrains the distractor set to some degree, but fails to uniquely identify the referent.

Step 2: Choose Property As with the greedy variant of the DD algorithm previously, we see the effect of applying each remaining property involving the referent in terms of how it constrains the area. We use the $\oplus$ operator as defined above. For each relation p_i we add to our constraints, the resulting field is F_i. We choose the relation which minimises the 'size' of the field according to our discussed metric, given it locally disambiguates location by the largest degree. A new relation may result in very small disambiguating power, or yield no information at all if it is implied by a previous one. To address this, we introduce a *discount threshold* λ where $0 < \lambda < 1$. This specifies the factor by which the value of $|F|$ must be reduced between iterations in order to warrant adding an additional relation. Thus we choose a predicate only provided that $|F_j| < \lambda|F|$. This also protects us when a suitable approximation is used for $|F|$ (as will be seen in the next section): even if we expect $|F|$ to remain constant because a candidate relation is implied by a previous one, error in the approximation may lead to an $|F|$ value which is just slightly smaller.

Step 3: Extend Description As before, we add the locally optimum relation from Step 2 to our description and update the field accordingly. However, since the algorithm is initially 'flat' in the sense that we don't recursively apply our algorithm to landmark objects, no variable renaming is required. We remove the added relation from the available set as before, so that it is not applied again.

Step 4: Resolve relation objects to produce the final description Our description has at this stage disambiguated the location of the referent to a sufficient degree. However, the description is not yet linguistically realisable as the landmark objects in the accumulated relations have no descriptions associated with them. Since each of these must be uniquely identified, we apply the previous DD algorithm. Doing so is simply a matter of the appropriate variable renaming: After associating the constant (i.e. entity) we wish to describe with a unique variable, we initiate the DD algorithm with this variable and entity initially on the referent stack. Upon obtaining this description we add the relations to the existing description, i.e. $L \leftarrow [r'\backslash v']L \cup L'$. We create a constraint set (in the DD sense) initially with a single variable and the referent object (the constant r must be renamed to this variable to reflect this). We append more constraint sets to this list as we describe each of the landmark objects. The final description is a pair $\langle L, C\rangle$, the same type as that outputted by the DD algorithm.

,

3.3.3 Doing It Tractably

While the complexity of the algorithm is $O(n_a n_l)$, compared to $O(n_a n_d n_l)$ for the greedy variant of the DD algorithm, determining the probabilistic field of a relation p, i.e. F_p, is potentially hugely computationally expensive; the source of the problem is the necessity to compute values of our spatial model across all of space. Since $|F| = \int_{-\infty}^{+\infty} \prod_i \psi_i(\mathbf{x}) d\mathbf{x}$[2] (if we treat the referent as a point object, so that we need not exclude the area inside the correct region), then given a suitable parametric function involving $\mathbf{x}$, we can use calculus to find an exact solution for $|F|$ without having to ever evaluate $\psi_i(\mathbf{x})$ for any relations or for any specific points.

For example, suppose that descriptions could consist only of the *near* and *by* relations, and we used Gaussian functions to model the deterioration of validity with distance, such that the validity right by the landmark object is 1[3]. We can therefore define our hypothetical model as:

$$\psi(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)} \tag{3.1}$$

where $\mathbf{x}$ is in general an n-dimensional vector representing the point in space of the referent, μ is the location of the landmark object and Σ is a positive-definite matrix that defines the covariance of the Gaussian distribution. For this model the matrix is $\sigma^2 I$ (where I is the identity matrix), since we wish to have a circular function shape such that the validity deteriorates at the same rate in all directions. Note that this function contrasts from the Gaussian distribution $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$, in that we don't require the normalisation constant usually found in its probability density function. Our coefficient at the front of the function (i.e. the *amplitude* of the Gaussian) is instead 1, since we wish by definition of the model for $\psi(\mu) = 1e^0 = 1$, i.e. our validity for *near* is full when at 0 distance away from the landmark object. Therefore $\psi(\mathbf{x}) = (2\pi)^{\frac{n}{2}} \sqrt{det(\Sigma)} \mathcal{N}(\mathbf{x}|\mu, \Sigma)$ where $det(..)$ is the determinant function, i.e. offsetting for the normalisation constant of the Gaussian distribution to yield a coefficient of 1. When finding the product of Gaussians, we can compute the new means and covariance matrices using the following equations. For the product of m Gaussians distributions:

$$\Sigma = (\sum_{i=1}^{m} \Sigma_i^{-1})^{-1} \quad \mu = \Sigma \sum_{i=1}^{m} \Sigma_i^{-1} \mu_i \tag{3.2}$$

Consequently, the integral of the product of our non-normalised Gaussian functions can be calculated as such:

$$|F| = \int_{-\infty}^{+\infty} \prod_{i=1}^{m} \psi_m(\mathbf{x}) \tag{3.3}$$

$$= \int_{-\infty}^{+\infty} \prod_{i=1}^{m} (2\pi)^{\frac{n}{2}} \sqrt{det(\Sigma_i)} \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{3.4}$$

[2] $d\mathbf{x}$ is shorthand for $dx_1 ... dx_n$ where $\mathbf{x} = (x_1, ..., x_n)$

[3] Although there are many other prepositions we could approximate with Gaussians if we allow less restricted covariance matrices. e.g. *On* with respect to a line (e.g. 'on the road') could be represented as a flattened and rotated Gaussian, provided the road is straight.

$$= (2\pi)^{\frac{mn}{2}} \prod_{i=1}^{m} \sqrt{det(\Sigma_i)} \int_{-\infty}^{+\infty} \prod_{i=1}^{m} \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{3.5}$$

$$= (2\pi)^{\frac{mn}{2}} \prod_{i=1}^{m} \sqrt{det(\Sigma_i)} \int_{-\infty}^{+\infty} \mathcal{N}(\mathbf{x}|\mu, \Sigma) \tag{3.6}$$

$$= (2\pi)^{\frac{mn}{2}} \sqrt{\prod_{i=1}^{m} det(\Sigma_i)} \tag{3.7}$$

As discussed, this value $|F|$ gives us a metric of the 'size' of the area the complete description gives us. If we assume our individual covariance matrices have the form $\Sigma_i = \sigma_i^2 I$, then we can simplify further:

$$|F| = (2\pi)^{\frac{mn}{2}} \sqrt{\prod_{i=1}^{m} det(\sigma_i^2 I)} \tag{3.8}$$

$$= (2\pi)^{\frac{mn}{2}} \sqrt{\prod_{i=1}^{m} \sigma_i^{2n}} \tag{3.9}$$

$$= (2\pi)^{\frac{mn}{2}} \sqrt{(\prod_{i=1}^{m} \sigma_i)^{2n}} \tag{3.10}$$

$$= (2\pi)^{\frac{mn}{2}} (\prod_{i=1}^{m} \sigma_i)^{n} \tag{3.11}$$

In Step 2/3 of Algorithm 5, picking the most disambiguating relation is simply a case of picking the one with smallest associated σ value in its Gaussian. The exact value of $|F|$ would still however be required to determine whether the threshold has been met in Step 1.

While such use of Gaussians leads to clean analytic result in the definite integral, we previously saw that our spatial models were influenced by factors that prevented a parametric definition (e.g. by considering an object's shape or visibility relative to other objects, which leads to discontinuities in the validity function). Therefore, such an approach of calculating $|F|$ directly is alas not feasible in the general case.

In the separate task of computing the effect of distractor objects on prepositional validity, [Kelleher and Costello, 2009] uses an array of points, computing a 'proximity score' for each of these. It gives no regard however to the granularity of these points, and the span of the grid is simply a known small finite area, encompassing the visible screen bounds to the user of an experiment the paper conducted. This introduces us to a number of challenges:

1. How to adapt to the varying scope of different prepositional models? Our "in front of the restaurant" and "west of Maine City" examples had valid regions of noncomparable size; fixing the grid size would either result in an infeasible number of points to compute, given the grid spans too wide an area (or conversely, with

a grid size too large to accommodate a small scale observation), or restricting our domain to observations we expect to encompass regions of similar size. This latter restriction is not in the spirit of producing generic multi-purpose implementations.

2. How to we combine the fields for the conjunction of multiple observations when the granularity of the data varies across these observations?

3. How do we account for the large regions of space in which the validity is likely to be 0? For example, for the "in front of the restaurant" observation, for most of the environment (say if the entire scope was the town, or larger) the validity will be 0. Would a sparse matrix say be sufficient, or is there a more suitable data structure?

An established data structure which deals with all these problems is the *quad tree*. It is a tree structure where each node represents a square region; if it is an internal node it has exactly four children, representing the four quadrants the region decomposes into, and the leaves represent a square region which decomposes no further, containing the data associated with that region. This is illustrated in Figure 3.7. The quad tree's most pertinent property for our task is the compact representation of areas that have the same value associated with them; notably, if we define the value of each leaf to be a validity associated with that region (i.e. all points within that grid's region has this validity), it addresses the third problem of representing the 0-regions of our environment when the valid region potentially occupies a small amount of it. If we restrict the granularity of the validity (not to be confused with the granularity with respect to grid size), such that say we round the validity to the nearest 0.05, it also allows us to compactly represent regions where the validity plateaus or remains approximately equal. This effect can be seen nicely in Figure 3.6.

Quad trees can easily be merged, thus solving the second of these problems. This can easily be achieved by comparing nodes of each tree synchronously. If both nodes are leaves, we take the product of the values associated with these leaves (recall that combining the validity across observations involves taking the product). If both nodes are internal, we recursively merge each of the four pairs of quadrants. If one node is the leaf and the other an internal node, we split the leaf into 4 (copying the value of the leaf into the 4 quadrants) then again recursively merge the pairs of quadrants. For efficiency, we can also compress the tree by working upwards from the leaves, merging quadrants if its children are leaves and their values are the same according to the given level of validity granularity (i.e. when rounded say to the nearest 0.05).

To compute the size metric $|F| = \int_{(i,j) \in span(ENV) - span(r)} F(i,j)$ for quad trees, we can use $|F| = \sum_l l^2_{width} l_{value}$ (where l is each leaf of the quad tree), where we simply ignore leaves if their centre point is contained within the area of the referent, i.e. we add only if $l_{centre} \notin span(r)$.

Two questions remain however: how do we determine the grid size to use for each relation, and how to determine the region to calculate these points? Our solution to the latter question is to approximate a rectangular region, one such that we can guarantee that all points outside of the region are 0 in validity. If we assume we want to limit computation to k different points per relation, then the former question can be answered by the latter, since if the area of this rectangular region is A, then we can use $W = \sqrt{A/k}$ to yield the grid width. With a quad tree we require any grid width to be a factor of the

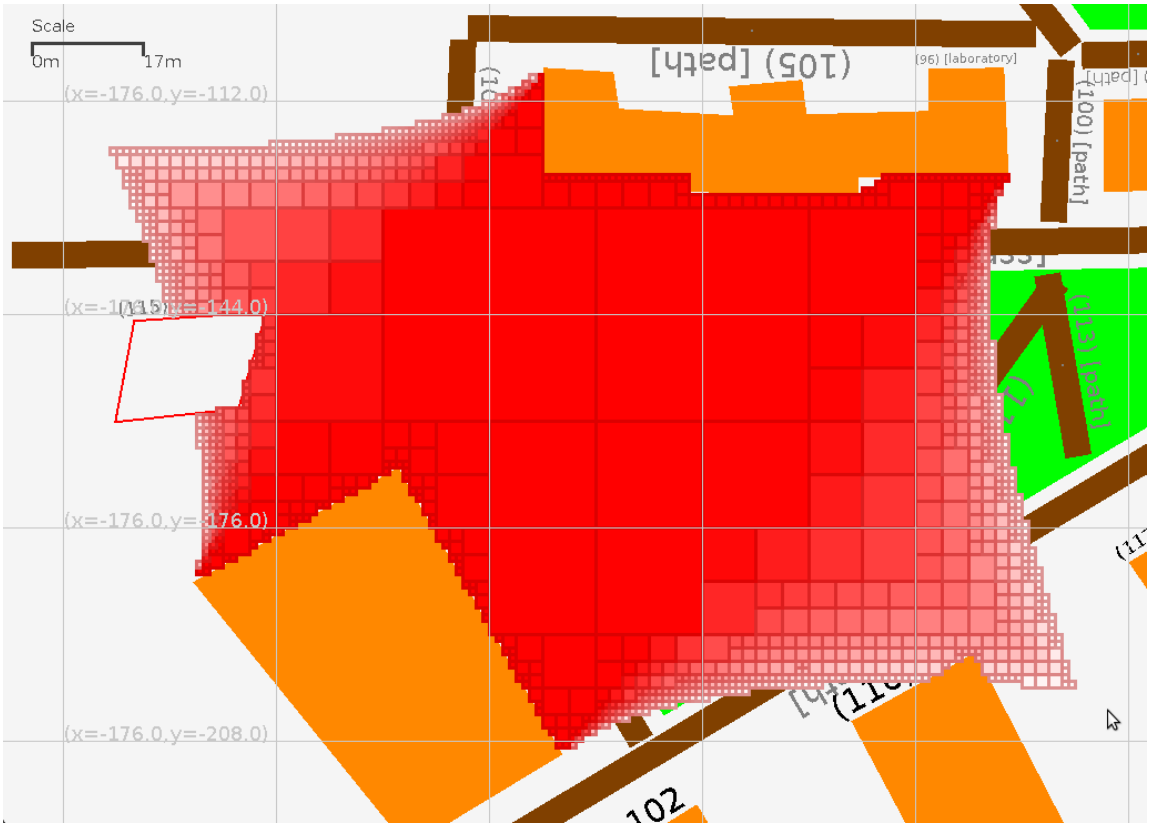


Figure 3.6: An illustration of the quad tree in practice, stemming from an observation involving the preposition *between*. The intensity of red is associated with the value of each leaf of the tree. The leaves are visibly of varying grid size, and the tree has been compressed so that areas of similarly or identical intensity (e.g. the region directly between the two buildings, where the validity is 1) are represented more compactly. The chosen grid size (i.e. before compression) has been artificially lowered for illustration purposes only; such fine granularity is not required given the quad tree is used merely as an intermediate form to compute $|F|$.

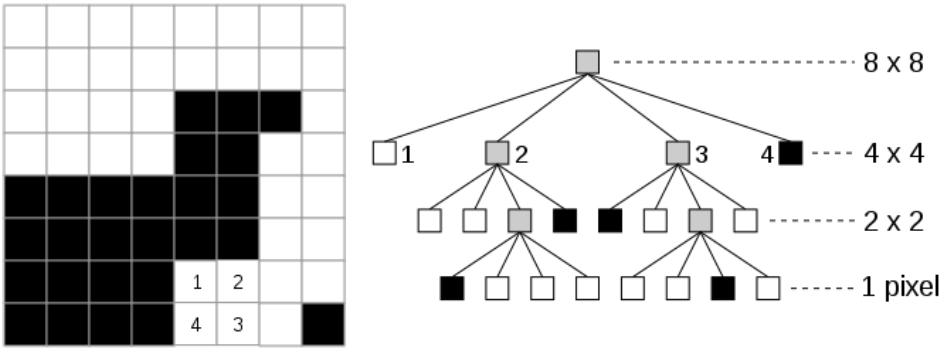


Figure 3.7: A demonstration of the quad tree structure (figure by Wojciech Muła). It allows a compact representation of a 8×8 bitmap image (i.e. where the value of each leaf is binary to indicate occupancy) by splitting nodes only when the data within their associated area varies. The root of the tree represents the entire image.

unit grid size (say g) that is a power of 2 (for example, if g is 1.5 metres, then $3g$, $6g$, etc. would be valid grid widths). We can use the formula:

$$W' = g \cdot 2^{\lfloor \log_2(\frac{W}{g}) + \frac{1}{2} \rfloor} \tag{3.12}$$

to do this conversion. Given the region and the grid size we obtain an appropriate lattice of points for which we can calculate our validity scores across space. The corners of this region require aligning to the nearest multiple of the grid size relative to the centre of the root node in the quad tree, to ensure that different quad trees can be merged.

How then do we calculate this rectangular region? This depends on the spatial preposition used. For topological prepositions such as *by* for example, we find the solution in terms of distance that yields validity 0. By using the rectangular region that minimally encloses the span of the landmark object, we simply expand it by deducting this distance from the minimal x and y coordinates, and adding it to the maximal x and y coordinates. This region guarantees that no point outside it will have validity greater than 0.

Projective prepositions are slightly harder; since we need to find the extrema of the circle segment shape the field forms. We first find the distance r at which the validity becomes 0 as before, in addition to finding the angular deviation θ from the preposition's axis at which similarly the validity becomes 0. We expect a circle segment shape we see in Figure 3.8. It is sufficient to observe this shape for each of the vertices of the landmark object, since at any point on the edge the field emanating from it will fall within the extrema of the overall field. We could use the equation of the circle, of radius r, and some simple calculus to find the minimum and maximum value in the x and y axis. Its centre would be the vertex of the landmark object under consideration, and the middle of the angular range of the arc determined by the reverse of the ray projection angle we previously observed in the projective model. However, this is difficult given we wish to restrict the solution to the angular range of the arc. A quick and adequately accurate approximation is to form two isosceles triangles, where the centre of each outer edge touches the arc. By basic trigonometry, the length of these sides is $r / \cos\theta$. By computing all these points and determining the minimum/maximum x/y values, we again obtain a

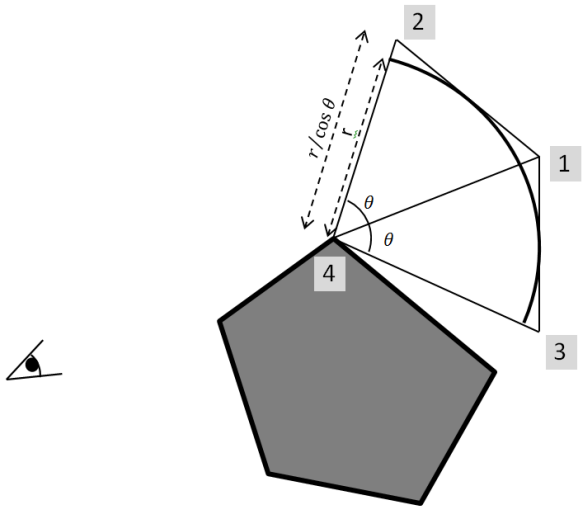


Figure 3.8: An illustration of the 4 points we compute to accommodate the 'arc' shape of the projective preposition field, specifically the 'behind' preposition. θ is the angular deviation at which the validity becomes 0, and r is the distance at which the validity becomes 0. By computing these 4 points for each vertex of the landmark object, and taking the minimum/maximum x/y coordinates, we obtain a rectangular region that is guaranteed to contain all the non-0 points.

region which is guaranteed to contain all non-0 regions of the field, and one that is not too much larger than necessary.

There are however some difficulties involving the calculation of bounds that are often hard to overcome. For the *behind* preposition for example, there may be no deterioration of validity with distance at all, thus resulting in an infinitely large field. We therefore unavoidably and inelegantly impose a limit on this distance so that the field is finite. A similar problem with the *near* preposition is that its spatial function is neither monotonically decreasing nor increasing; as we get further from the landmark object within an enclosure, the validity decreases, but upon moving from one enclosure to its parent, the validity jumps back to 1 due to the definition of the model. Just as with *behind* where the distance could extend infinitely, with *near* we can move far up the enclosure hierarchy, yielding a potentially huge area. We can again inelegantly address this problem by restricting the number of levels we can move up the hierarchy, and use the bounds of this outer enclosure to determine the non-0 region.

Using the grid of points, using the bounds and grid size we discussed, we form the appropriate quad tree. We start with a single node, encompassing the entire environment, and with a validity of 0. We then apply a simple recursive algorithm as such: whenever the current node is a internal one, we select the correct quadrant depending on the grid cell we wish to insert, then continue. If it is a leaf, but the size is larger than that of the grid cell we're inserting, we split the leaf into 4 leaves then again descend into the correct quadrant. Otherwise, with a leaf the same size as the insertion cell, we set the value accordingly. After inserting all points in this fashion, we can compress the quad tree as previously described. Since we started with a validity of 0 spanning the entire

Iteration	Relation	$\|F\|$
1	LeftOf[just](e115,you)	6,550
	LeftOf(e115,you)	9,640
	RightOf(e115,e81)	**4,699**
	By(e115,e81)	10,345
	Near(e115,e81)	24,580
	Near(e115,e96)	29,336
2	**LeftOf[just](e115,you)**	**2,687**
	LeftOf(e115,you)	3,561
	Near(e115,e96)	3,813
3	**Near(e115,e96)**	**2,004**

Table 3.4: The steps carried out in the Locative Description algorithm for the sculpture in 3.9 (with the identifier `e115`). The building named '101' has identifier `e81` and the orange lab in front of the observer `e96`. The **bold text** indicates the chosen relation in each iteration.

environment, any region outside the cells we inserted will be 0, as required.

The rectangular bounds we calculated also serve another invaluable efficiency purpose. Suppose upon introducing the first relation to our description, we maintain its associated rectangular region R. Upon computing $|F|$ when introducing a second relation, say with region R', we need only test points within the region $R \cap R'$; even though there may well be points within $R' - R$ that have validity greater than 0, we know that when taking the product of the values, the 0 from the field associated with R will yield 0. Only in $R \cap R'$ can points from both R and R' have values greater than 0. Consequently, we can also instantly ignore a candidate relation if $R \cap R' = \emptyset$. Once a new relation has been accepted, we replace R with $R \cap R'$ for the R' associated with the chosen relation, and continue the algorithm. The efficiency increase from these two modifications is very considerable.

3.3.4 Example

Consider the question 'where is the sculpture', with the environment and pose of the robot depicted in Figure 3.9. The steps of the algorithm are detailed in Table 3.4.

On the first iteration in Step 2, there are 6 valid prepositions, and for each we calculate $|F|$. `RightOf(e115,e81)` (corresponding to "right of the building named 101") yields the lowest value, and thus is added to the description. Note that from iteration 2, the speed up in processing time to compute each $|F|$ value is roughly a factor of 10, using the optimisation of taking the intersection with the rectangular region associated with the first relation we chose. Observe that in iteration 2, `By(e115,e81)` and `Near(e115,e81)` have been removed. This is an amendment to the algorithm to yield more natural descriptions, where in addition to removing relations from the set P_r that we have added to the description, we can also remove any relations which shares a landmark object to the one added. Thus by referencing building 101 in our first added relation, we have consequently removed 'near 101' and 'by 101'.

Table 3.5: A summary of results for human assessment of the algorithm's generated descriptions..

Quality	Raw Text	Via Pointing
Poor	24%	8%
Satisfactory	23%	24%
Good	53%	68%
Correctness	Raw Text	Via Pointing
Correct	85%	96%

Whether we reach iteration 3 is dependent on the terminating threshold for $|F|$ we have set. If we set it to 2500 for example, we would reach it, and thus before Step 4 have the description `RightOf(e115,e81)` $\wedge$ `LeftOf[just](e115,you)` $\wedge$ `Near(e115,e96)`. We apply Step 4 to form a distinguishing description of all landmark objects used, and rename our referent to a variable `MAIN`. This yields the final description:

```
   RightOf(MAIN,x0) ∧ LeftOf[just](MAIN,x1) ∧ Near(MAIN,x2)
∧ attr_name_101(x0) ∧ observer(x1) ∧ class_laboratory(x2)
∧ InFrontOf(x2,y2) ∧ observer(y2)
```

corresponding to *"The sculpture is right of 101, just left of you and near the laboratory in front of me."*. We can raise the threshold to shorten the descriptions produced.

One deficiency in the algorithm so far is that repeated application of the same relation would hypothetically lead to the value of $|F|$ being depreciated, despite the fact that repeating an observation provides no disambiguating power. While the algorithm prevents application of precisely the same relation since it is removed from P_r after being added to the description, then without our augmentation of the algorithm to remove relations involving the same landmark object, "left of" may well be added in the third iteration to our description after "just left of", despite the fact that the latter is more specific than the former (i.e. the latter adds no new information: $\forall x, y : \psi_{JustLeftOf}(x, y) \leq \psi_{LeftOf}(x, y)$). The problem is the result of using multiplication in place of logical conjunction; we could resolve the problem by binarising our algorithm by rounding up to 1 or rounding down to 0 as appropriate for our validities. Since the field of "left of" would then be a subset of "just left of", taking the conjunction would yield just the "just left of" field again, and thus no improvement to $|F|$ would have been made, thus preventing it from being added. But such an alteration would be at the expense of losing the gradation in validity.

3.4 Evaluation

To evaluate our algorithm, we presented 10 human assessors with 25 descriptions generated by the algorithm for 250 assessments in total, for example, *'That lab is right by the car park, just right of you and right by another lab in front of me'*), given an indicated pose of the observer and with a graphical representation of the environment. More specifically we used the ALU Freiburg science campus discussed in Section 6.6. Assessors were told that their knowledge of the environment was restricted only to the objects that were

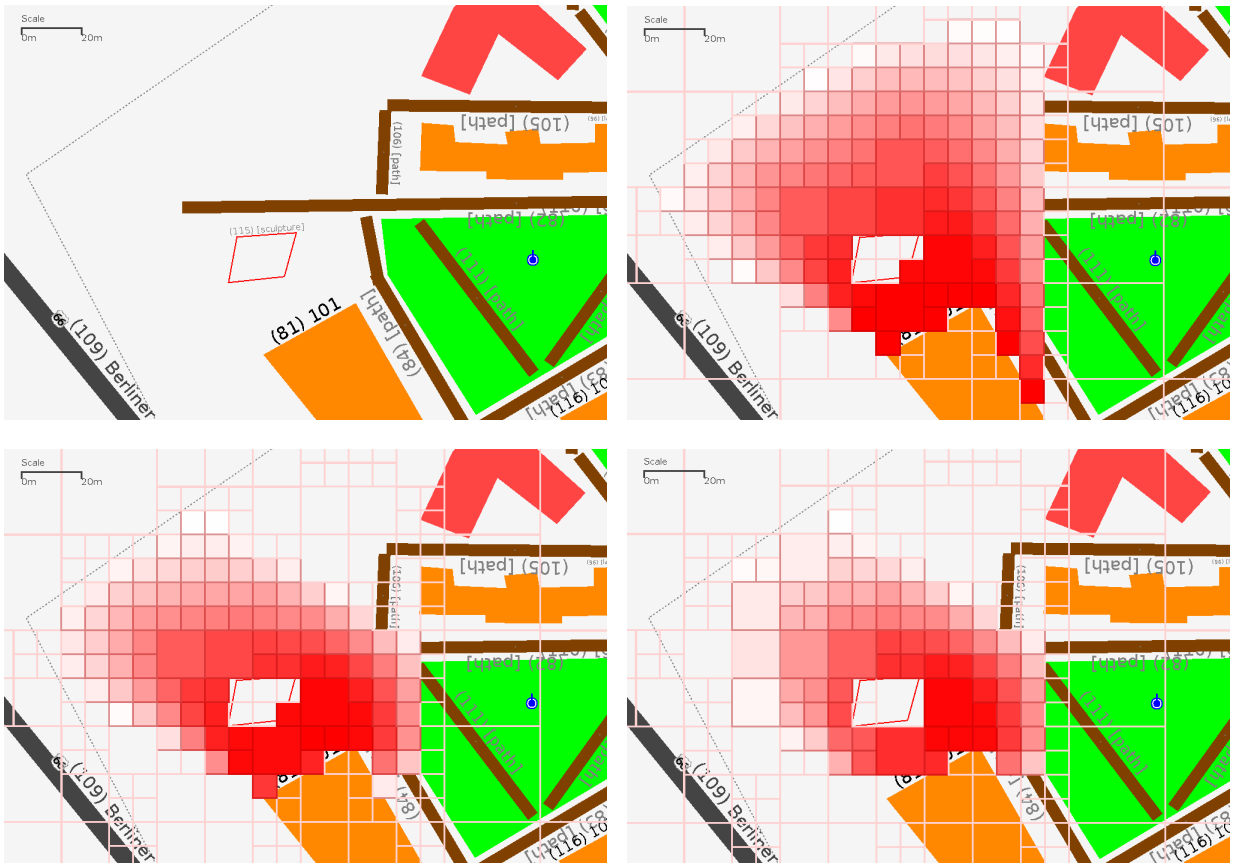


Figure 3.9: The field/quad tree $|F|$ after Step 3 in each iteration of the algorithm, for the example provided in Table 3.4. The circle towards the right of each image indicates the position and orientation of the observer. The 4-sided object towards the left of the image shows the region of the object in the question (the sculpture). The resulting description was *'The sculpture is right of 101, just left of you and near the laboratory in front of me'* when the terminating threshold was set to a low value (i.e. generating a long description). The last relation (*'and near...'*) would not typically be generated provided the discount threshold λ threshold was set appropriately, since the reduction in ambiguity is low.

visible (except the object being described). The scenarios involved a number of different observer positions in the environment, and described objects in short, medium and long range from the observer. Descriptions were assessed by two different methods; by using the raw linguistic description generated by the algorithm, and by pointing to landmark objects in the environment, as if physical pointing was permitted by the describing agent in real life (i.e. *'it's by [that object over there] and just left of [that object]'*). The purpose of the latter is to isolate the performance of the location description algorithm from that of the DD algorithm used in Step 4 to generate distinguishing descriptions for landmark objects. That is, identification of landmarks in the description can be achieved via pointing instead of describing them, and thus we can more accurately assess the extent to which our algorithm picks suitable spatial relations between objects that disambiguates the area, not how well it identifies landmarks.

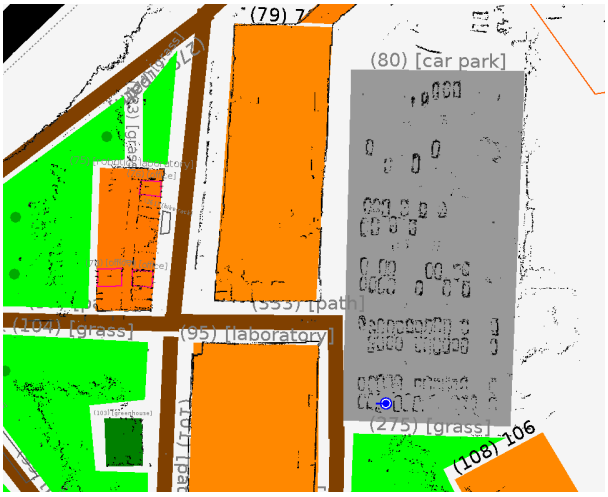


Figure 3.10: An example context for which the expression *"The robotics lab is just left of the laboratory right of me and right by the laboratory in front of me"* was evaluated, where the robotics lab is towards the right of image and the robot towards the bottom of the car park facing left. The description was deemed to be less natural due to the multiple axis used for different prepositions.

Descriptions were assessed both on their correctness, and by their quality or *naturalness*. Regardless of the assessment, participants were invited to describe the deficiencies in the description and how they might be improved. A summary of results can be found in Table 3.5. Agreement between assessors was high, although slightly lower for the assessment of quality for raw text descriptions, as one would expect given the more subjective nature of 'naturalness'. While the Fleiss Kappa coefficient κ was only 0.57 amd 0.31 for the correctness of raw text and pointing respectively, and 0.36 and 0.32 for the quality of raw text and pointing descriptions (when conflating judgements to 'good' and 'bad'), the metric is entirely unsuitable to assess participant agreement in this circumstance, since the calculation of the probability of assessors agreeing by chance is based on the empirical distribution over the answers 'good' and 'bad', coupled with the greater prevalence of 'good' answers. For example, if hypothetically assessors were to both agree on every question but answer 'good' for every question, κ would be undefined, and just a single disagreement can lead to a rapid depreciation in score. There was in fact full agreement for 84% of questions on average across the four types of judgements. The following comments were common amongst assessors:

1. In a few cases, the description was slightly longer than necessary. This issue could potentially be resolved by raising the terminating threshold, although the experiment did not establish an appropriate threshold to use or whether using a termination threshold proportional to the size of the referent is sufficient. This is an area of further investigation.

2. In the majority cases where the description was not judged to be good, this was due to landmark objects being described in a technically correct but obfuscated way. For example for *'...right by the lab behind the lab'*, there may have only been one pair

of visible labs that matched this description, but it was difficult to identify each lab separately. Since the data was collected, such problems would have been resolved by our improvement to the Distinguishing Description algorithm, discussed in Section 3.2.3, such that all further landmarks nested within other landmark descriptions would themselves be uniquely identifiable.

3. Users gravitated towards using the projective prepositions 'behind' and 'in front of' when available, either recommending inclusion if not already in the description, or placing these relations at the start of the description if already present. This is perhaps because such relations restrict the angular range of the target object (even if still leading to a large possible area of ambiguity) while being low in cognitive complexity. This corroborates with [Dale and Haddock, 1991]'s discussion of head categories, where predicates/relations that were not strictly necessarily to identify the object were included given the added clarity to the overall description.

4. One minor problem encountered was that the axis in the frame of reference across different prepositions in the same description could change, given that each prepositional model is evaluated independently. This is illustrated in Figure 3.10, for which the description *"The robotics lab is just left of the laboratory right of me and right by the laboratory in front of me"* was generated. There are three different axes that are required for users to correctly interpret the expression: the robot's axis when interpreting "in front of me", its axis again when interpreting "right of me", and after identifying the laboratory right of the robot, the axis formed from the angle between the observer and the object in order to interpret 'left of [this lab]'. While entirely correct, the description was judged to be less natural due to this complexity. In addition, one evaluator commented on the multiple uses of the word *right*, both as a adverb in 'right by', and as a preposition in 'right of'.

The algorithm is further evaluated in Section 6.6, in the context of testing the entire robot platform, and where the evaluators were immersed in a real environment.

3.5 Modelling Spatial Observations as a Sensor Model

In the previous section, we considered the possibility of using an environment and a referent object to generate a description concerning the location of the object. So is the converse possible; i.e. using a description (or multiple descriptions) that use spatial properties of an object to learn something about where it is, and more specifically, the space which it occupies? On a broader level, can we therefore learn about our environment over time?

We therefore shift our focus from the probability of a given position of the referent being considered valid, to the probability of a given point (or more accurately, a small region of space), being occupied by some part of this referent, i.e. that the cell occurs within the region of the object. Fortunately, there is a data structure in robotics that tackles this very problem. Occupancy Grid Mapping is a technique employed to generate maps of an environment via noisy sensor measurements. The data structure is robust as it can subsequently be fused with other maps obtained from physical sensors for example, such as laser scans that can detect directly whether a point is occupied by matter (even

if we don't know which object this matter may belong to). The goal is to produce a posterior probability $p(m|z_{1:t}, x_{1:t})$ using the standard notation, where $m = \{m_i\}$ is a partitioning of space into a finite grid of cells m_i, $z_{1:t}$ are the observations made up to time t, and $x_{1:t}$ are the poses of the robot at each observation. m_i is the event that cell i is occupied, thus $p(m_i)$ describes the probability that cell i is occupied.

The converse task, using Occupancy Grid Maps to generate descriptions, has received a small amount of research attention. [Skubic et al., 2004] for example segments the map into discrete objects, and uses a local 360° sweep to generate simple projective and topological relations such as *"The object is mostly in front of me but somewhat to the right"* and *"The object is very close"*. Descriptions are restricted to objects in the immediate vicinity. However, we again are unaware of any research which has attempted to tackle the opposite problem we are exploring.

If we make the common simplifying assumption that the occupancy of each cell is independent of neighbouring cells, we can approximate the full posterior as the product of the probabilities of the constituent cells. Computing the posterior involves two models, the *prior distribution* $p(m)$, and the *likelihood* $p(z_t|m)$. Application of Bayes rule leads to a recursive implementation requiring us to provide $p(m_i|z_t, x_t)$, which is known as the *inverse sensor model* ('inverse' because we endeavour to infer the occupancy from the sensor reading z, rather than sense based on the occupancy). The overall effect of these update equations is to compute $p(m|z_{1:t}, x_{1:t})$ for our accumulated observations over time. More details can be found in [Thrun, 2003], but we present here the *update equations* required to incorporate the evidence of new observations.

$$p(m|z_{1:t}, x_{1:t}) = 1 - \frac{1}{1 + exp\{l_{t,i}\}} \tag{3.13}$$

$$l_{t,i} = \begin{cases} l_{t-1,i} + h^{-1}(m_i, x_t, z_t) - l_0 & m_i \in \text{field}(z_t) \\ l_{t-1,i} & \text{otherwise} \end{cases} \tag{3.14}$$

$$h^{-1}(m_i, x_t, z_t) = \log \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)}, \quad l_0 = \log \frac{p(m_i)}{1 - p(m_i)}$$

where $\text{field}(z_t)$ is the perceptual field of z_t (i.e. the range of the sensor) and h^{-1} is the inverse sensor model.

Our aim is to produce this inverse sensor model, by establishing a parallel between this generic sensing framework and the 'linguistic' sensing of processing the textual observations concerning an object, and employing our spatial probability density functions $p(\top_z|z_t, x_t)$ from Chapter 2. An important simplifying assumption we make is that locative expressions refer to a *specific point in space*, within the boundaries of the object in question. This seems intuitive; were we to describe a town as being '10km away', it would clearly be fallacious to assume that the entirety of the town is precisely 10km away (which would suggest the town's span is restricted to an infinitesimally thin arc). We define a probability $p(r_{i,j}|x_t, z_t)$, where $r_{i,j}$ represents the event that the observer was referring to a point (i, j) in their observation, and $z_t^{\ominus}$ is the locative expression such that the position of the referent is not specified (since we consider such a position in $r_{i,j}$). We can then calculate the desired probability easily by simply normalising our confidence function across the space: $p(r_{i,j}|x_t, z_t^{\ominus}) = \alpha\psi_{z_t^{\ominus}}(i, j)$ where $\alpha = 1/\int_{-\infty}^{+\infty} \psi_{z_t^{\ominus}}(i, j)$ (and as

usual $\psi_{z_t^{\ominus}}$ is the spatial function associated with the relation/observation $z_t^{\ominus}$. For example, suppose the validity for some $2m \times 2m$ region was 1, and 0 everywhere else. Then the probability that a particular point in the region was intended by the observation is 1/4, and 0 elsewhere.

Before we determine how to calculate $p(m_{i,j}|x_t, z_t)$, we analyse the conceptual parallelism between traditional Occupancy Grid Mapping and that employed in our linguistic context.

A Comparison of Sensor Models There are some immediate clear similarities that can be drawn between the traditional occupancy grid map and our linguistic variant. Both involve the pose of some observer x_t (although depending on the spatial model this is sometimes irrelevant) and some manifestation of an observation z_t; a physical sensor reading with respect to the traditional approach and a locative expression for the linguistic approach. Upon closer analysis more similarities can be drawn. With a physical sensor, we expect a measurement of distance to a point being sensed to be noisy, and thus maintain a probability distribution with regards to the precise position of this point. This corresponds to our distribution $p(r_{i,j}|x_t, z_t)$. For a locative expression of a town being "10km away", human error or rounding is likely to lead to uncertainty in the judged distance, and additionally the direction of the town is unspecified, leading to a 'blurred doughnut' type distribution.

There are however a number of conceptual differences. With traditional Occupancy Grid Mapping the posterior for a cell is only updated if it was part of the sensor range (i.e. we make no assumption with regards to space outside the limited range of our sensor). With locative expressions however, we can infer data outside that explicitly conveyed. Suppose for our town example, the town was 1km in diameter, and that the distance judgement of 10km (to some point within the town) was entirely accurate. If the centre of the town was actually 10.5km away, our observation would still hold, but a point any further could not possibly be occupied by 'town'.

Computing the Inverse Sensor Model We can use the above fact to compute $p(m_{i,j}|x_t, z_t)$ from our previously calculated values of $p(r_{i,j}|x_t, z_t)$. Let $\mathcal{Q}$ be the set of possible 'poses' for the referent object such that the point (i, j) is within the object's boundary, and a pose is the position and orientation of the object. Given our assumption that the observer referred to a point within the confines of their perceived position of the object, $\mathcal{Q}$ represents all valid poses of the object given such a point. It follows that $p(m_{i,j}|x_t, z_t) = \int_{q\in\mathcal{Q}} p(q|x_t, z_t)\,\mathrm{d}q$. For each pose $q \in \mathcal{Q}$ there is an associated frame (i_q^*, j_q^*, θ_q) where (i_q^*, j_q^*) is the *nominal centre* of the shape in pose q (say the centre of mass) and θ_q is the rotation of the object about this point. It is then possible to use $p(r_{i_q^*,j_q^*}|x_t, z_t)$ to refer to the probability of the object being positioned at (i_q^*, j_q^*) (see Fig. 3.11). The pose also has a probability $p(\theta_q)$ associated with its orientation; for simplicity we assume this is independent of x_t and z_t (although the use of $p(\theta_q|z_t)$ would allow us to model for example observations such as "The boat is in front of you, *facing East*"). Putting this together, this gives us the following equation to compute the occupancy probability:

$$p(m_{i,j}|x_t, z_t) = \int_{q\in\mathcal{Q}} p(r_{i_q^*,j_q^*}|x_t, z_t)p(\theta_q)\,\mathrm{d}q \quad \text{s.t. } \mathcal{Q} = \{q \mid (i, j) \in R(q)\} \tag{3.15}$$

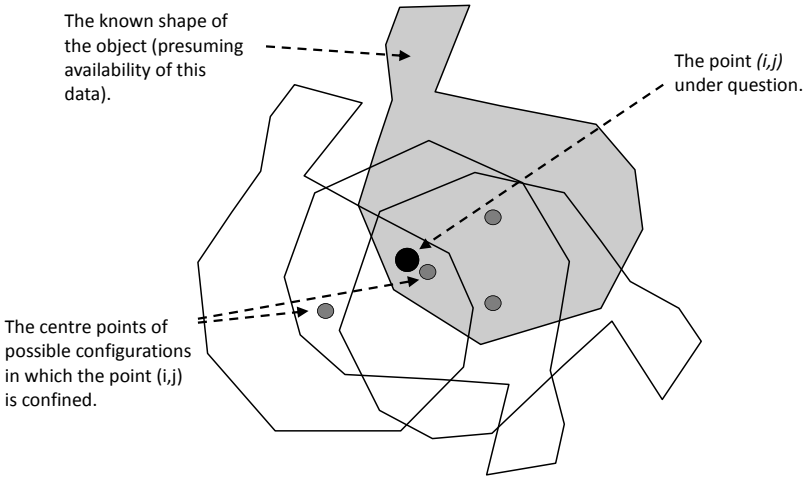


Figure 3.11: In calculating the occupancy probability $p(m_{i,j}|x_t, z_t)$, we consider all possible poses of the located object in which the point (i, j) is confined. The probability of each pose q is $p(r_{i_q^*,j_q^*}|x_t, z_t)p(\theta_q)$.

where $R(q)$ is the region of the located object in pose q. Considering the pose of the referent object has useful consequences; it allows us to model for example that vehicles are aligned to the direction of a road, or as (tentative) scientific research suggests, cows are more likely to be aligned in a North-South direction[4]. Given a lack of prior shape information with regards to the referent object, and given the above integral is somewhat intractable, a suitable approximation is to use the approximate width W of the object (which can be obtained via knowledge of the class of the referent object, say the usual width of a town). If we infer as little about the shape as possible, the resulting approximation of the shape is a circle of diameter W. Equation 3.15 then reduces to the following:

$$p(m_{i,j}|x_t, z_t) = \int_{(i',j')\in R(\frac{W}{2},i,j)} p(r_{i',j'}|x_t, z_t)\,\mathrm{d}i'\,\mathrm{d}j' \tag{3.16}$$

s.t. $R(\frac{W}{2}, i, j)$ is a set of points in a circular region of centre (i, j) and radius $\frac{W}{2}$.

For cases where the object is small but the grid size is large (e.g. "the ant is in England"), we find the insufficient granularity of grid size leads to an overestimation of the occupancy probability. Given that the probability $p(r_{i,j}|x_t, z_t)$ is uniform within a given cell (i, j), we can estimate this probability by the following formula, as a consequence of equation 3.16:

$$p(m_{i,j}|x_t, z_t) = \frac{\pi W^2 p(r_{i,j}|x_t, z_t)}{4g^2} \tag{3.17}$$

where g is the grid size of the grid the point is located in and W is the estimate width of the object as before.

[4]http://news.nationalgeographic.com/news/2008/08/080825-magnetic-cows.html

Given the inverse sensor model, it is then a case of determining the prior model $p(m)$. This is however where the analogy of linguistic sensing with occupancy grid mapping unfortunately starts to break down. We typically maintain only one occupancy grid map for the entire environment, whereas in our case we have one just for the referent object. Given that most of the map will be unoccupied (since our referent would usually only occupy a small region of the entire environment space), the prior probability of occupancy would consequently be very low, and thus the posterior probability also low. However, we see this preliminary research as an important first step towards the task of learning object occupancy from mere linguistic observations alone.

It is possible to simplify the task if we concern ourselves only with the most likely *position* of the referent, rather than more detailed information on its occupancy in space. In this case, we can use a more direct complement of our Locative Description generation algorithm. If we let $r_{t,(i,j)}$ be the event of referring to point (i, j) as before, for the observation at time t, then the most likely point (i_{ref}, j_{ref}) our referent object is centred at is:

$$(i_{ref}, j_{ref}) = \arg\max_{(i,j)} p(r_{1,(x,y)}, ..., r_{t,(x,y)} | x_{1:t}, z^{\ominus}_{1:t}) \tag{3.18}$$

$$= \arg\max_{(i,j)} \prod_{k=0}^{m} p(r_{k,(i,j)} | x_k, z^{\ominus}_k) \tag{3.19}$$

$$= \arg\max_{(i,j)} \prod_{k=0}^{m} \alpha_k \psi_k(i, j) \tag{3.20}$$

$$= \arg\max_{(i,j)} \prod_{k=0}^{m} \psi_k(i, j) \tag{3.21}$$

Simply put, the most likely position of the referent is just the point which maximises the probability of the entire description (i.e. the product of the validity functions for each preposition in the description), when the models are evaluated considering the referent at that particular point.

3.6 Searching

Entity Retrieval is the act of performing a search on a collection of entities/locations according to some user specification. Whereas a document search would yield a set of documents given an appropriate representation of a user's query, we analogously retrieve a set of entities, each accompanied by a suitable metric of its validity, by use of a description using spatial language and a database of entities. While spatial search is not the emphasis of our research, we explore some of the associated challenges and propose a system that is fit for purpose in the context of the overall robot architecture.

Spatial search is by no means a novel concept. Commercial engines such as *Bing Maps* and *Google Maps* provide search for locations, businesses and services, accommodating queries such as "5 stars restaurants in London" or 'museums near OX1'. [Jones et al., 2004], [Vaid et al., 2005] and [Jones et al., 2003] similarly use a set of permitted spatial relations in its query language, such as distance based (within/near),

topological (in) and directional (north of). [Jones et al., 2003] introduces the notion of a *geometric footprint*, where each object may be represented by a representative point (i.e. a centroid), a minimum bounding box, a polygon or a line, just as we explored in Section 2.3. Spatial models are evaluated with respect to these entities.

The drawback of all these approaches is the limited scope of spatial queries that are permitted. [Jones et al., 2003] for example permits only a single spatial relation in a query with the query terms for the landmark and referent objects restricted to either a place name or an entity class (e.g. restaurant). Other engines such as *Bing Maps* and *Google Maps* restrict the landmark expression further, such that they must be uniquely identifiable; they would not for example be able to interpret queries such as 'museums near parks'. We augment the query language to accommodate the following:

- Non-spatial semantic information, e.g. *A shop that sells jewellery*, including the class of the entity and any adjuncts.
- Nested spatial expressions, e.g. *The bench between the two trees in front of me.* More generally, we recursively allow each of the landmark and reference expressions to themselves be spatial expressions, allowing an unbounded number of spatial relations and other information.
- Ordinals and superlatives, e.g. *The second road on the left* or *The nearest restaurant to the lake.*
- Parts of objects, e.g. *The house near the end of the road.*

If spatial relations are present in the expression, our approach is to recursively find matching entities for the landmark and referent expressions (where there will be multiple landmarks for tertiary relations such as *between*), before testing the validity of the relation for each choice of landmark/referent pair from the cross product of the two sets. If we treat spatial prepositions used in an expression as independent, the overall validity of the expression is simply the product of the various spatial models we presented in Chapter 2 For expressions involving only non-spatial properties, we can find entities which satisfy the conjunction of these properties via an appropriate indexing method.

This can be expressed by the following simple algorithm:

where each spatial relation $q_r P(q_{l^1}, ..., q_{l^n})$ consists of a referent object query q_r, a number of landmark object queries q_{l^i} (usually one in the case of binary relations) and a spatial preposition P. ψ_P applies the respective spatial model to return a validity probability in the range [0,1], as per Chapter 2. To illustrate, recall the scene in Figure 3.1, and suppose we now wish to interpret (rather than generate as previously) *'The cat in front of the tree by the dog'.* Then the following steps would be taken to yield a match:

1. The overall query is one which involves a spatial relation. We therefore recursively find matches for the landmark query *the tree by the dog* first. Its own landmark query is *the dog.*
2. Since this subexpression contains only non-spatial attributes, we use an index of the environment's entities to retrieve all entities of class *dog.* As per the second condition of Algorithm 6, this returns a set $\{(dog_1, 1)\}$, where the 1 indicates the full validity of the match.

Algorithm 6 A recursive algorithm for determining matches for a locative expression with an arbitrary number of spatial relations.

function MATCHES(q)
  if q's identity is already known **then return** $\{(q, 1)\}$
  else if q contains only non-spatial attributes **then**
    $S \leftarrow$ the set of entities satisfying all these attributes **return** $\{(s, 1) | s \in S\}$
  else
    suppose $q = q_r P(q_{l^1}, ..., q_{l^n})$ for a spatial preposition P
    $L_1 \leftarrow matches(q_{l^1})$
    ...
    $L_n \leftarrow matches(q_{l^n})$
    $R \leftarrow matches(q_r)$
    $S \leftarrow R \times L_1 \times .... \times L_n$ **return** $\{(e_r, \psi_P v_r \Sigma_i v_{l^i}) | \langle (e_r, v_r), (e_{l^1}, v_{l^1}), ... \rangle \in S\}$
  end if
end function

3. The referent query *the tree* similarly yields the two trees $\{(tree_1, 1), (tree_2, 1)\}$.

4. We can now evaluate higher level landmark queries. We apply the spatial model *by* to each pair of entities in the cross product of our two sets, i.e. $\langle tree_1, dog_1 \rangle$ and $\langle tree_2, dog_1 \rangle$. The first yields a validity of say 0.9 (since the first tree is indeed by the dog), and the latter 0. We can discard the latter as a future possibility (and in general prune intermediate matches if their validity is below some threshold, since we know that multiplying by further probabilities can only reduce the value).

5. As per the last step in the algorithm, for the landmark query *the tree by the dog* we can return a result set with a single result $\{(tree_1, 0.9 \times 1 \times 1)\}$.

6. At the top level, we can then repeat the process, finding matches for the referent query *the cat* and using our result set just obtained for the landmark query. The algorithm overall would yield the match $\{(cat_1, 0.9)\}$.

In this particular example, the first condition of the algorithm was not used. The condition holds if an entity was previously grounded, or if the query is an explicit reference to the robot/user's current location. Evaluating our spatial models over all elements in the cross product can be very expensive if the sets comprising the cross product are large. Fortunately there are various optimisations that can be made. Recall from Section 3.3.3 that all spatial models with a fixed landmark object (or objects) could generate a rectangular region restricting the possible locations of the referent object where the validity of the preposition may be non-zero. Consequently, if we resolve the matches for the landmark queries first, we can restrict our search of entities for the referent query to be those in the region permitted by each matching landmark. This seems conceptually sensible; were we to evaluate *the tree by the dog* and had a candidate match for the dog landmark, we could visually restrict our search of trees to the nearby region surrounding the dog, since any objects outside this region could not possibly satisfy the preposition *by*. By maintaining a suitable index on the region occupied by entities in the environment

(as all geographic search engines do, using for example R-Tree data structure we explored in Section 2.5), search becomes tractable for large environments.

To handle superlatives such as *nearest* or more generally *'nth' nearest*, we simply rank the current matches in order of distance from the given reference point (usually the observer), and preserve only the nth item. For parts of an entity, e.g. *the end of* or the *the centre of*, then in the results for a query involving a part, we replace each match with its corresponding parts. For example, with *the end of the road*, e.g. matching road is replaced with two points entities for each end.

Chapter 4

Dialogue Manager

4.1 Introduction

We have so far explored how spatial models can be defined for the various aspects of spatial language, and how they can be employed in a number of higher level algorithms for both the generation and interpretation of spatial expressions. Our task now is to leverage these models to produce a spatially motivated dialogue system. In this section, we specifically examine the production of a 'Dialogue Manager' that governs the discourse behaviour of the system. In Section 5, our focus shifts to translation techniques, to produce a system that can convert natural language utterances into a form that can be interpreted by this Dialogue Manager. Finally in Section 6.4, we examine some of the more practical details in producing the compiler that interprets our specification language that defines the behaviour of the Dialogue Manager.

Dialogue is the most fundamental and specially privileged arena of language, developed from when a child and imperative to the interactions that occur between humans, whether it be ordering lunch or participating in meetings. Dialogues tend to differ from other forms of discourse (such as prose) in its greater use of ellipsis (e.g. 'Where?'), discourse structure and coherence [Jurafsky et al., 2000]. Fully autonomous conversation agents have captured the public imagination for some time, as with *HAL* in Stanley Kubrick's *2001: A Space Odyssey*, and Weizenbaum's famed *ELIZA* program, a parody of the responses of a nondirectional psychotherapist that intentionally "sidestepped the problem of giving the program a data base of real-world knowledge" [Weizenbaum, 1976]. Perhaps due to the emergence of speech recognition technology, as well as the development of richer representations of dialogue context, conversational agents have experienced a surge in research interest in the past couple of decades, and now form a ubiquitous part of modern industrial systems, most notably in the form of automated phone systems (even if, as traditionalists would emphatically note, do not closely emulate discourse with a real human agent).

There are three important features which distinguishes dialogue from monologue. Firstly, it involves interaction on a turn-by-turn basis and in which spoken natural language is often used [Fraser, 1997]. A *turn* can be considered to be a single utterance or multiple adjacent utterances made by a particular agent participating in the discourse, whereas an *utterance* is a single, and often elliptical, sentence. *Conversational analysis* [Sacks et al., 1974] concerns the means by which these turns are made at a *transition-*

relevance place, that is, places where the structure of the language allows a shift in speaker to occur. A significant part of dialogue research involves the identification of these turn boundaries when the speakers are known, usually relying on *cue words* (such as 'well', 'so'), particular N-gram or POS sequences (which can be acquired by training) and prosody effects (such as pitch, accent and pauses).

A second significant feature that distinguishes dialogue from monologue is that of *grounding*. The set of facts mutually believed by both speakers is known as *common ground*, and any assertions must be implicitly or explicitly acknowledged by the other agent. A *continuer* (also commonly known as a *backchannel*) such as "Mm hmmm" is a method of such grounding, providing assurance that an utterance has reached the addressee.

Finally, *conversational implicature* concerns the meaning of utterances beyond what is explicitly conveyed. Consider the following dialogue segment:

> A: And, what day in May did you want to travel?
> B: OK uh I need to be there for a meeting that's from the 12th to the 15th.

The presumption that A has the capacity to draw inferences from B's utterance permits an indirect answer. Clearly, the contextual information provided by B allows A to deduce that the time of arrival needs to be sufficiently before B's meeting. Such an example was identified in the seminal paper by Paul Grice [Grice, 1975], who proposed four maxims dictating cooperative discourse:

1. *Maxim of Quantity*: Be exactly as informative as required, that is, avoid superfluous information.
2. *Maxim of Quality*: Ensure your contribution is one that is true and does not lack substantiating evidence.
3. *Maxim of Relevance*: Ensure your contribution is relevant.
4. *Maxim of Manner*: Avoid perspicuity, obscurity and ambiguity, while attempting to be as laconic as possible.

It is the Maxim of Relevance for example that allowed A to infer that B wished to travel by the 12th. In B assuming A's compliance with the maxims, the provision of dates not directly concerning the dates of flight was to provide details relevant to the dialogue.

Utterances can be interpreted as *actions* [Austin, 1975], whether it be *locutionary acts* which communicate an utterance with a particular meaning, *illocutionary acts* which 'make it clear to some other person that the act is being performed' (known as 'securing of uptake' by [Austin, 1975], and incorporates acts such as asking and promising), or *perlocutionary acts* which instigate effects upon the feelings, thoughts or actions of the addressee. The distinction between illocutionary and perlocutionary acts can be exemplified with the example:

> "You can't do that"

This may have the illocutionary force of protesting (i.e. communicating some state of mind of the speaker to the addressee), or the perlocutionary effect of ceasing the addressee's current activity (i.e. with the focus on the physical response of the addressee).

4.1.1 Dialogue Act Taxonomies

Such illocutionary acts have formed part of a number of taxonomies, which categorise the acts by the differing functional roles in dialogue. All dialogue systems need an appropriate representation of these acts, subsequently used by a dialogue model in its control logic. Our particular choice of taxonomy can be observed later in Section 4.4.2, as well as throughout Chapter 5. [Searle, 1976] provided a coarse categorisation that provided the foundation for a number of finer grained taxonomies. Their 5 major classes proposed are as follows:

1. *Assertives*: committing the speaker to something being the case (suggesting, putting forward, etc). For example *'The prisoner was clearly guilty'*.

2. *Directives*: attempts by the speaker to get the addressee to do something (asking, inviting). For example *'Please give your verdict'*.

3. *Commissives*: committing the speaker to some future course of action (promising, opposing). For example *'I will carry out the sentence'*.

4. *Expressives*: expressing the psychological state of the speaker about a state of affairs (thanking, welcoming).

5. *Declarations*: bringing about a different state of the world via the utterance. For example *'I hereby declare you guilty'*.

These have been further enriched to more effectively capture the conversational functions dialogue acts can play, as well as to incorporate additional data encoded in the utterance. *Annotation schemes* provide a means by which the acts of existing dialogues can be tagged to conform to such a taxonomy, and provide an indispensible component of machine learning techniques as well as evaluation of conversational agents. They have been applied to a number of preexisting large dialogue corpora, such as *COMMUNICATOR* (consisting of flight-booking conversations). The *DAMSL* set [Core and Allen, 1997] for example has tags for statements, signals of non-understanding (i.e. modelling *meta-communication*, as will be revisited) and acceptance. These are divided into two categories; *forward-looking acts* represent *initiating* moves, usually in the form of questions or assertions not directly in response to another initiating move. In contrast, *backward-looking acts* represent some kind of *response* move, usually in the form of answers and acknowledgements. Such a distinction allows systems to effectively model *adjacency pairs*, pairs of dialogue acts that commonly occur together, such as question/answer, greeting/counter-greeting, etc. Other annotation schemes exist, such as the *DATE* scheme [Walker and Passonneau, 2001]. This is composed of three dimensions: the speech act (in a similar vein to the *DAMSL* set) embodying the type of act, the *task* the act relates to, and the *conversational domain*. The last of these categorises the context of the act, whether it pertains to a task, or is a metacommunicative phenomena, or is a 'situation frame', which relates to the goal of managing the culturally relevant framing expectations (such as instructions, apologies and presenting information regarding the system's capabilities, etc).

4.2 Dialogue Models

Given such an analysis of this concept of a dialogue act, as well as appropriate ways to model such acts, we now shift our focus to the task of determing *dialogue policy*, i.e. determining the system acts to generate given an appropriate representation of the discourse context. The task of the dialogue manager can be defined simply: given some state of the discourse context, what utterance, or utterances do we generate next? There are a number of considerations that must be made:

1. **Number of agents**: While a conversational agent would by definition not be developed for *monologue* (although certain discourse semantic representations can encapsulate the evolution of context throughout the discourse), the decision remains whether to permit *multilogue*: discourse involving more than one agent (excluding the system agent that act generation is required for). [Ginzburg and Fernández, 2005] provides the principles and benchmarks required to scale up to multiple agents, but research and framework implementations is predominantly centred on dialogue only.

2. **Choice of context representation**: What features and level of abstraction are chosen to appropriately model the dialogue context? For Finite State Machines for example, the state by definition can be stored by merely a reference to the particular state in the data structure, which needs only identity rather than associated data. For frame-based approaches, the state may be represented by 'slots' we require filling. For Partially Observable Markov Decision Processes, we might choose to represent the state as some constant dimensional feature vector. We will explore each of these.

3. **System initiative versus mixed initiative**: System initiative dialogues are those which permit only the system (as opposed to the human agent) to make initiating (or forward-looking) moves. Typically such systems are task-based and have a fixed agenda. Mixed initiative dialogues however allow complete freedom for all agents, thus allowing the human agent to shift the topic of conversation or specify the information it requires.

4. **Task-based versus social**: Dialogues can be categorised into two classes; task-based dialogues engineered to solve some particular problem or carry out some particular tasks (particularly systems where some domain of information is expected to either by imparted or required). Social dialogues however are for more aesthetic purposes, intended to comfort, entertain and operate in a social context. Often for such systems, there is no objective agenda that the system is required to satisfy.

5. **Discourse flexibility**: [Ginzburg, 2012] provides a number of benchmarks that dictates the requirements for a conversational agent to emulate human conversation. Such benchmarks include assertoric ones (accommodation of proposition content update and disagreement), allowing metacommunitative interaction (such as grounding, feedback and repair) and short answers (including *sluices*; utterances that consist of a single wh-pronoun, such as 'who' or 'where'), as well as high-level multimodal benchmarks such as accommodating gestural acknowledgements (such as nodding). The versatility of the system agent required dictates to a large extent the representation of the context needed.

These distinctions motivate a number of existing discourse models used as dialogue managers:

4.2.0.1 Finite State Machines

Finite State Machines are by far the simplest model. In such a system, there are a fixed number of predefined states, each of which can have transitions between them, representing dialogue actions made by the user. The system is clearly system-initiative, maintaining control of the dialogue, since the user is restricted to a narrow conversational path. At each state, the user is prompted for input, typically from a fixed set of forms (such as answers to specific questions or choice from a selection), using the response to determine the next state. The advantages and disadvantages are clear: the restrictions on the user input equally restricts the grammar and vocabulary required, both leading to higher speech recognition accuracy, as well as simpler and more predictable processing of the input. However, the lack of flexibility restricts behaviour associated with the user taking initiative, and any flexibility (e.g. backtracking if the user realises they have provided incorrect information) must be explicitly encoded into the state machine. While some user initiated behaviour can be incorporated by adding extra states and transitions, this leads to multiple states having the same discourse behaviour; it would not be possible to have a single state for 'clarification' for example, since it is impossible to determine the transition history to a state if there were multiple routes to it. Such systems are typically employed on automated phone systems, where a rigid inflexible task structure is desirable.

4.2.0.2 Frame-Based Systems

Frame-Based systems (also known as *Template-Based* systems) are a minor extension of this behaviour, where the task is to fill a number of pre-defined 'data slots' (a process which is known as *slot-filling*). This typically entails the user being requested to supply a desired set of data in order to satisfy some system goal. The distinction from finite state machine based systems is not immediately obvious in terms of visible characteristics to the user, until one considers that the terminating condition is the successful filling of all slots, rather than reaching a final state from a set path. As a result, the data can be retrieved in any order, as well as being able to cater for over-informative answers (one of the benchmarks as defined in [Ginzburg, 2012]). A conversation controlled by this system may play out as follows (from [H Aust, 1995]):

> System: What is your destination?
> User: London.
> System: What day do you want to travel?
> User: Friday.
> System: What is your destination?
> User: London *on Friday around 10 in the morning.*
> System: I have the following connections...

Clearly, in order to accommodate these more informative answers, more advanced language processing is required. The disadvantages remain the same; frame-based systems

only marginally higher flexibility over finite state machines (the resulting dialogue is still firmly system-initiative), juxtaposed against the higher complexity of the user input.

4.2.0.3 Information State Update

Dialogue has two important properties that largely motivate the choice of discourse contextual representation:

1. *Discursive potential*: At each point in the discussion, the discursive potential is the set of topics available for discussion or utterances available.

2. *Ellipsis*: The particular context restricts the resolution possibilities for uses of ellipsis.

[Ginzburg, 1996] proposes a structured view of dialogue context that incorporates two elements: *questions under discussion* and the *'latest move'*. This builds upon [Stalnaker, 1978]'s proposition that context merely comes from *assertions* made throughout the discourse (if presuming an absence of objections), that is, a set of facts representing the true statements grounded by the participants by means of assertoric utterances. As a set, this representation naturally does not encapsulate what statements were made when; this dichotomous view however restricts the discursive potential preventing a notion of *discussion*. But an important component of dialogue is *locality*; as we have seen, much work in conversational analysis is predicated on the existence of *adjacency pairs*. As soon as the illocutionary info of an utterance is accepted, this serves the value of a variable named *'LATESTMOVE'*. We leverage this variable to provide a range of reactions to queries and assertions as follows:

1. If *LATESTMOVE* was assertion p, either:

   (a) Make a move that accepts p and updates the contextual repository.

   (b) Raise the issue of *whether* p as the current topic of discussion.

2. If a query q, either :

   (a) Accept q as a topic for discussion and provide information specific to p.

   (b) Reject q as a topic for discussion.

However, it is not the case that moves react immediately the preceding ones, as this example demonstrates:

A: Why did the Pacers lose?
B: Miller wasn't playing well.
A: Well, he scored 30 points.
B: But no offensive rebounds.
A: Hmmm, go on.
B: The refs were biased.

Here, B's last utterance continues to provide an answer for the question originally posed by A in the first utterance. This motivates keeping track of the *questions under discussion* (referred to hereon as *QUDs*), a partial order of queries currently being discussed. The maximal element of this order is the current topic of the discussion, and the partial order reflects the fact that more than one question can be under discussion simultaneously. Raising or terminating such questions is known as *updating* and *downdating* respectively. We can *update* a QUD if for some query q, there is an utterance specific to q which either conveys information *ABOUT* q, or conveys some other question q' on which q depends (e.g. *A: Who committed the murder? B: Who was in town at the time?*). Similarly, QUDs can be downdated if information is accepted that either decides q or indicates that no information about q can be provided.

Such theory has led to the *Information State Update* approach which has been adopted by frameworks such as *TrindiKit* [Larsson and Traum, 2001] and *DIPPER* [Bos et al., 2003a] as discussed in Section 4.3. The information state theory consists of a description of *informational components* of the contextual modelling (including aspects of the modelling introduced by [Ginzburg, 1996], but extending it with beliefs of agents, intentions, etc), formal representations of these components (whether lists, sets or discourse relation structures), a set of dialogue moves which trigger updates in the information state, a set of *update rules* which govern the updating of the information state, and finally an *update strategy* that determines which rules to select from the list of applicable ones. The aim of the ISU is to combine the advantages of *structured* approaches to dialogue modelling, where transitions between predefined states are specified (such as with Finite State Machines), and plan-based approaches (discussed in 4.2.0.5). The main advantage of such an approach is that dialogue rules can be conditioned upon *partial* specifications of the state, for example the triggers may be based on the values of a subset of variables in the information state. This results in full flexibility in dialogue behaviour, and consequently, full support for mixed initiative dialogue. The consequent disadvantage is that the execution of dialogue rules becomes more unpredictable [Bohus and Rudnicky, 2009] and unwieldly, given that the effects of one rule may have unexpected consequences, particularly given that the requirement only to partially specify partial state in the triggering conditions may lead to multiple rules being fired.

4.2.0.4 Reinforcement Learning

Dialogue systems that can be determined solely from annotated sample dialogues (via the use of machine learning techniques) are making an emergence in recent research. While the approaches analysed so far rely on large degrees of fine tuning, systems using this approach have the natural advantage of requiring little or no training, with the additional benefit of being able to cope with noise in the input. A *Markov Decision Process* provides a mathematical framework for decision making, where the outcome is determined by a combination of a user's actions and an element of randomness. Similarly to a Finite State Machine, there is some existing state s and some set of actions A_s available at that state. In the non-deterministic variant of the MDP, there is some probability $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ associated with a transition to some possible states s' given a particular former state and action. Additionally, there is a *reward function* $R_a(s, s')$ representing some notion of reward or *utility* in making the transition a

between states. In the context of discourse, the reward function amounts to the strength of a particular choice of dialogue act, whether it be on a global level of stepping closer to a successful terminating dialogue (in which say some required information is obtained or the user's demands are met) or more local phenomena, such as forward-looking moves being followed with a backward-looking move that forms an established adjacency pair as discussed. [Henderson et al., 2008] for example award +100 for actual or perceived task completion, and a reward of -1 if the state follows a system action, to encode that idea that short dialogues are preferable to longer ones.

As per our original aim, our intention is to a devise a policy, denoted $\pi(s)$, that chooses a system action given the current state of the dialogue s. The strategy is to choose the action with the best *expected future reward*, that is, the reward that we expect over all possible future states given that the optimal action is always chosen. This expected future reward can be calculated as $\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})$ where γ is the discount rate, and a_t is the optimal action at state s, i.e. $\pi(s)$. The discount factor allows us to represent the notion that longer-term consequences have less prevalence than local ones, although is usually set close to 1 to avoid 'greedy' decisions based on immediate reward.

Determining this policy $\pi(s)$ is known as *reinforcement learning*, and there are a number of algorithms to compute this policy. One is to simply use a *brute force* strategy; enumerating all possible policies and choosing the one which maximises the future expected reward. A major drawback of this approach is that the policy space may be extremely large, even infinite, and thus is typically not used. A much more efficient method is to use *value functions*, which involves storage of two arrays indexed by state, the value V containing real values, and π containing actions. They are each defined as follows:

$$\pi(s) := argmax_a\{\sum_{s'} P_a(s, s')(R_a(s, s') + \gamma V(s'))\} \tag{4.1}$$

$$V(s) := \sum_{s'} P_{\pi(s)}(s, s')(R_{\pi(s)}(s, s') + \gamma V(s')) \tag{4.2}$$

The first line states that for each state s' that the candidate action leads to, we take the local reward directly from the reward function, along with the discounted expected future reward from the state s'. These are weighted by the probability of each state to get the expected value of the future reward relative to possible states. However, learning V is typically also intractable for large state spaces.

Partially Observable Markov Decision Processes (or POMDPs for short) generalises MDPs by making the state hidden, that is, it allows a distribution $b(s)$ (known as the *belief distribution*) over possible states, and incorporates a set of observations O, as well as associated probabilities $P(o|s', a)$ that these observations were made. For the sake of brevity we do not give further detail, but the conceptual heart is that the states are *partially observed* in the sense that their identity is not known; instead the use of observations yields merely clues to the state. [Young et al., 2006] utilises POMDPs by defining the observations as the N-best outputs from the speech recognition module, each with their associated probability. [Henderson et al., 2008] attempts to apply a reinforcement learning approach to the Information State Update approach we saw in 4.2.0.3. States are represented as a real-valued feature vectors consisting of agent whose turn it is, the conversational domain, filled and unfilled slots (as per the frame-based approach), and

the history of speech acts. In order to represent textual data numerically, all such fields are represented as attribute-value pairs for all possible values. For fields where the text is entirely arbitrary, we simply use 1 to represent the presence of text and 0 otherwise. Since the states are vector based rather than discrete, *linear function approximation* must be used to calculate the future expected reward, having the effect that two states are similar if they share the same features. A pure reinforcement learning strategy performs badly due to the limited fixed-training set and intractable state space; most state-action pairs are unlikely to appear. It therefore presents a hybrid reinforcement learning/supervised learning approach, which reduces the reward for portions of the state space where data is not available.

One might argue that reinforcement learning is not a suitable method for learning dialogue policy, due to a number of reasons:

1. Evaluation metrics are generally based on very limited dialogue behaviour. In the case of the flight booking domain data, there are four slots for which simple information is required, and no feedback or processing of these inputs is required.

2. Current systems are incapable of processing input. [Henderson et al., 2008] for example represents arbitrary textual fields as a boolean value for their mere presence. This is only sufficient for the particular evaluation metric given, merely attempting to retrieve 4 arbitrary pieces of information. The systems are incapable of question answering, unless the specific instantiated question-answer pair explicitly appear in the data.

3. They are difficult to integrate with other systems. Determining the policy function concerns only the production of dialogue acts, not the intermediate actions (such as perhaps issuing database requests to a physical system) that result in such acts.

4. It is not clear whether nondeterministic Markov Decision Processes are conceptually necessary to model dialogue. For a given state and act, most *task*-based systems would expect deterministic behaviour. While introducing nondeterminism permits behaviour such as choosing an action randomly from the discursive potential (for example, if the system had multiple questions it could ask at a given point in the dialogue, it could choose any), the additional learning complexity and more unpredictable policies learnt arguably does not offset the advantages that the nondeterminism provides.

5. Identifying problems in the dialogue or fine tuning it are very difficult due to the quantitative parameterisation, and behaviour would be highly unpredictable for portions of the state space that have not been extensively observed, making the systems hard to manage.

4.2.0.5 Plan-Based and Task-Based Systems

In plan-based systems, the system models the goals of the conversation, using a planner to guide the dialogue along a path towards the satisfication of these goals. These employ the use of *inference rules*, where some preconditions (or sub-goals) must be met in order to assert the success of a goal. This is coupled with one of two methods of reasoning: *forward*

chaining starts with the available data (i.e. things which are already true or goals that are satisfied), and repeatedly applies the inference rules to satisfy more goals, until the target goal is reached. *Backward chaining* starts with the target goal, and works backwards from the consequent to the antecedent, finding clauses that lead to postconditions known to be true. [Boye, 2007] and [Bohus and Rudnicky, 2009] both have hierarchical task trees, where some task is not considered to be complete until its associated subtasks are complete. [Boye, 2007] applies this to the automatic troubleshooting domain, with rules such as the following:

```
satisfy(locate-modem) {
   perform requestAction(locate-telephone-plug);
   perform groundStatus(locate-modem)
}
```

Such rules can be parameterised with variables to allow richer behaviour. Utterances can be associated with tree events, with say a 'topic intro' event if a node is expanded, representing the introduction of a new goal. [Bohus and Rudnicky, 2009]'s system is called *Ravenclaw*, and provides a richer hierarchical rule language that allows user inputs to be mapped onto the execution of particular actions or inserting a new goal into the task tree. It incorporates task-independent error-handling behaviour (such as asking for a short answer to be repeated or a longer answer to be rephrased), with an associated error handling decision process.

In general, plan-based systems often exhibit more complex problem-solving behaviour. The *Circuit-Fix-It Shop system* [Smith and Hipp, 1994] for example uses complex logic-based reasoning to diagnose and rectify problems with electric circuits based on data supplied by the user. The dialogue evolves in the form of a proof.

The advantages and disadvantages are similar to that of finite state machine-based systems: the system has comparatively predictable behaviour, with the additional advantage that routines of subtasks, such as say acquiring login details, may be reused in different contexts (depending on the nodes placement in the tree) without the need to cumbersomely repeat states. However, the inflexibility characteristic of Finite State Machines is still present, permitting little user initiative.

4.3 Review of Existing Frameworks

4.3.1 TrindiKit

TrindiKit [Larsson and Traum, 2001] is a dialogue framework based on the Information State Update model. Its central component is the *Dialogue Move Engine*, whose function is to update the Information State based on the observance of dialogue moves and subsequently selecting moves to perform. The DME consists of one or more *update modules* containing the dialogue rules, where each update module can individually have an *update strategy* specified. The update strategy specifies how the dialogue rules are executed. This might involve taking the first rule that is supplied, applying all matching rules in a given sequence, or choosing amongst rules with probabilistic information. The framework distinguishes between *update rules* and *selection rules*. The former uses inputs (typically

dialogue acts from the user) to update the information state. *Selection rules* then subsequently use this modified state to plan the next course of action, usually involving the production of output dialogue acts back to the user and possibly interfacing with an external system.

$[PRIVATE = [AGENDA = \langle RAISE(X(TO = X)\rangle]]$
`Sys: Where do you want to go?`
(triggers **integrateSysAsk** rule)

$$\left[\begin{array}{l} PRIVATE = [AGENDA = \langle\rangle] \\ SHARED = \left[\begin{array}{l} QUD = \langle X(TO = X)\rangle \\ LM = ASK(X(TO = X)) \end{array}\right] \end{array}\right]$$

`Usr: Malvern`
(triggers **integrateUsrAnswer** rule)

$$\left[\begin{array}{l} SHARED = \left[\begin{array}{l} BEL = \{(TO = MALVERN)\} \\ QUD = \langle\rangle \\ LM = ANSWER(MAVERN) \end{array}\right] \end{array}\right]$$

Figure 4.1: An example update rule for *TrindiKit*'s dialogue move engine toolkit. QUD represents the Question Under Discussion, LM the latest move, and before the first exchange, the system has an agenda item to raise the question about the user's destination.

There have been a number of implementations using the *TrindiKit* framework. *GoDis* [Larsson et al., 2000] is an initial experimental system providing simple algorithms for update and selection rule modules, and a keyword spotting module to parse user input. An example execution of update rules based on new input can be found in Figure 4.1, with the rule used found in Figure 4.2. This demonstrates how the information state may be represented, consisting of records for the private agenda of the system, and shared information between the system and user, such as the questions under discussion and latest move (both motivated by Ginzburg's model of Information State).

U-RULE: **accommodateQuestion**(Q,A)

$$\text{PRE: } \left\{\begin{array}{l} val(\text{SHARED.LM}, answer(usr, A)), \\ in(PRIVATE.PLAN, raise(Q)), \\ answer - to(A, Q) \end{array}\right\}$$

$$\text{EFF: } \left\{\begin{array}{l} del(PRIVATE.PLAN, raise(Q)), \\ push(SHARED.QUD, Q) \end{array}\right\}$$

Figure 4.2: The update rule for accommodating questions, as supplied in the *GoDiS* implementation of *TrindiKit*. The precondition is triggered if the latest move is the user answering a question in the system's private agenda (where the answer A is assumed to be a **reply** move). The effect is to prevent the system from raising the question at a later point by removing it from the agenda, and for the question to then be raised as one under discussion.

TrindiKit has been criticised however for its flexibility being at the expense of obfuscation. Even when using *GoDis* to provide simple concrete implementations for some of the components (such as the control strategy for selecting from and ordering matching rules), even building systems for simple discourse is highly complex and laborious. [Burke et al., 2002] identifies a number of other problems with *TrindiKit*; in its long lag between user utterance and system response due to the inherent inefficiency of Prolog, inconsistent semantics where constructs of the GoDis plan language were interpreted differently depending on the depth of the plan (a problem partly due to Prolog's backtracking behaviour), the difficulty in finding bugs given Prolog's lack of feedback for incorrect goals, and the lack of multimodal support, given that *TrindiKit* does not distinguish different inputs by source (i.e. between user and say a robot system).

[Bos et al., 2003a]'s toolkit *DIPPER* builds upon *TrindiKit* by stripping away some of the complexity and offering a unique update language. It also reduces flexibility in the update strategy, instead adopting a fixed strategy of repeatedly applying the effects of rules whose conditions are satisfied. The framework however is still difficult to use, partly in its strong coupling with a text-to-speech engine (rendering even a simple task of 'echoing' the user's input convoluted, with numerous lines of code required) and its intentional absence of *variables*, which avoids the back-tracking problems associated with Prolog, but requires a *functional* form for rules which is unnecessarily restrictive and in the particular context not entirely intuitive. *DIPPER* attempts to address the problem of connectivity with other systems by offering *Open Agent Architecture (OAA)* support, designed to be a generic platform for integrating different software agents (possibly coded in different programming languages). However, this framework is not widely adopted.

TrindiKit gives some regard to typing of arguments, in that variables within the Information State can have assigned types, and new datatypes can be created based on Prolog constructs such as relations, functions and selectors. There is additionally a type checking facility to ensure that operations applied to variables are consistent with their type. Typing is not explicitly applied to the dialogue system input; type problems are only detected once the system runs and some component of the input has an invalid operation applied to it.

4.3.2 Ravenclaw

Ravenclaw [Bohus and Rudnicky, 2009] meanwhile adopts a task-based approach to dialogue modelling. It sets out a number of objectives; the first of these is task-independence, in the sense that generic conversational skills (such as error-handling, timing and turn-taking, etc.) are abstracted away from domain specific elements such as dialogue control logic. Other objectives are that of flexibility (for the framework to accommodate a wide range of application domains), modularity and reusability (by separating components for input processing and executing control logic for example) and scalability (i.e. having sufficient efficiency so that larger scale applications can be accommodated).

There is a two-tier dialogue manager structure, distinguishing between *Dialogue Task Specification* and the *Dialogue Engine* which captures domain-independent aspects of dialogue control. The former is specified via task trees representing a hierarchical plan; the root node represents the overall goal of the system, with children representing subgoals needed to be carried out in order to satisfy this goal. The nodes of the tree are categorised

into two types. The first, *dialogue agents*, appear as the leaves of the tree, representing some atomic action. This might be informing the user of something, expecting some information input from the user, or performing some domain-specific system operation (e.g. accessing a database).The second, *dialogue-agencies* are non-terminal nodes whose role is to control the execution of its constituent subagents (for example, the task of 'logging in'). Each of these agents can have defined preconditions, triggers, and success/failure criteria. It is these that termine the execution order of the various agents in the tree, rather than the ordering of nodes in the tree. *Concepts* are Ravenclaw's equivalent of variables. For example, a success criteria for the 'logging in' dialogue-agency may be that a 'username' concept is set.

The dialogue engine during runtime has two data-structures: a *dialogue stack* capturing the discourse structure at runtime, and an *expectation agenda* which captures what the system expects to hear from the user at each turn. Error-handling is moderately thorough, motivated by the unreliability of speech recognition systems. *Non-understandings* occur for example when the speech recognition fails to give a parse at all or one from which the semantic content cannot be surmised, while in contrast, *misunderstandings* occur when the input can be parsed, but yields incorrect data. The developer can specify a number of different error handling strategies depending on the type of error detected. These may involve asking the user to repeat information, or requesting confirmation of data provided.

The Ravenclaw system is be used in conjunction with a number of systems developed by the same institution, including *SPHINX* for speech recognition, *PHOENIX* [Ward and Issar, 1994] for parsing of input, and *ROSETTA* for language generation. In addition, *HELIOS* is a 'confidence annotation' component that uses different information sources (such as the recognition, understanding and dialogue) to determine a confidence score for each parsed hypothesis. This allows the dialogue manager to distinguish between multiple semantic interpretations, so that only the one with the highest score is used. The parser uses chart parsing to achieve basic slot filling, and both domain-independent constructs (such as *[Yes]* and *[Repeat]*) representing parameterless dialogue act types, as well as domain-specific grammar rules specified by the developer, so that the semantic output consists of the concepts (such as 'username') identified from the input to fill the slots in the grammar rules.

The advantages of *Ravenclaw* over say *TrindiKit* is that the task hierarchy provides more robust and predictable execution of rules, and error handling is more robust. However, as previously identified, the discourse is still rigidly system-initiative, and the system's goal known in advance. Another disadvantage is the input representation is too simplistic; the representation is limited to the dialogue act type with some appropriate attribute-value slots.

4.3.3 TALK

TALK [Young et al., 2006] is an example of a POMDP approach to dialogue modelling, specifically adopting an approach known as the *Hidden Information State* model, that attempts to overcome the difficulties inherent in POMDPs with exponential explosion of state and observation space size. The model is composed of a number of components: $b(s)$ is a distribution over states representing the *belief* of being in each state (contrasting

from before where the state was observed/known). It is due to this distribution our Information State is considered 'hidden'. A_m and A_u are the set of actions the system and user can take respectively (corresponding to dialogue acts), transition probabilities $P(s'|s, a_m)$ of moving between states given a particular action (i.e. the state resulting from a new dialogue act input is non-deterministic), and O is a set of observations. As previously identified, an observation model is required because we can't be certain of the precise dialogue act from the user; we can for example exploit that speech recognisers can often yield a probabilistic distribution over sentence hypotheses.

States are triples combining the models of the user goal s_u, the last user action a_u (corresponding to *LATESTMOVE* from before) and the dialogue history s_d. The system state is then the belief state distribution $b(s_u, a_u, s_d)$, whose probability mass function is obtained by taking the product of the output of the probabilistic models for each of these, along with the observation model for the speech recogniser input. Variables are marginalised as appropriate.

The overall model assumes that the state can be *partitioned* into groups or equivalent classes, where the members of each class are indistinguishable from each other. There is initially just one partition p_0, so that $b(p_0) = 1$. The *User Act Model* $P(a_u|p, a_m)$ for a partition p can then be composed of two parts: bigram probability of the dialogue act type given the previous system dialogue act type (i.e. modelling adjacency pairs), and some measure of how the act is consistent with the partition.

Lastly the model consists of probabilistic ontology rules with probabilities manually assigned according to the prior belief in such rules. For example, a rule `type -> restaurant(+food,music,-decor)` might have a probability of 0.35 assigned, represented the fact that 35% of the time in which a user wishes to locate a venue. Rules consist of *class nodes* (non-terminal nodes representing an abstract type which typically consists of child components, e.g. 'location', 'entity', etc.), *lexical nodes* (non-terminal nodes with representing attributes of types, e.g. 'music', 'decor', 'food type') and *atoms* the values of lexical nodes (e.g. 'Italian'). Using the rules and these nodes, we can build a tree representing the user's goal, for example:
`find(venue(restaurant(food(Italian),music,decor)))`
Dialogue acts consist of a type along with attribute-value pairs, for example
`inform(music=jazz)` to represent an answer 'jazz music'. Candidate system acts that may be generated from the goal tree use a number of rules. For example, if a partition is bound to an entity, then for every attribute value pair $a = v$ in that entity that has never been mentioned before, the act `inform(a=v)` is considered. To determine the act from these candidate hypotheses, we take the act which maximises the expected utility (reward) across all possible states, using the belief distribution.

Other than problems intrinsic to reinforcement learning type approaches, there are numerous deficiencies of this model. Firstly, despite the complex probabilistic formulation of the problem, the grammar rules which partition the goal states must be manually crafted and assigned probabilistic priors. The framework also only permits simple system-initiative dialogue for requesting and supplying attributes of objects and carrying out actions involving these objects, and rules for generating candidate system acts are again ad hoc. There are also limitations in the dialogue act representation which we examine in Chapter 5.

4.4 The HURDLE Language/Framework

4.4.1 Motivations

In the previous section we explored a number of models for dialogue context, as well as multiple frameworks which support these. These suffered however from a number of deficiencies, most predominantly in the areas of usability for developers, or lack of discourse flexibility. There are a number of characteristics we would like to incorporate into a dialogue system that such frameworks lack, in an effort to satisfy these two aims:

1. **A decoupling of syntax and semantics**: Systems such as *TrindiKit* and *DIPPER* incorporate the processing of raw textual utterance input within the framework of the Dialogue Manager. From a linguistic perspective one may initially deem this to be a sensible decision; incremental (i.e. word by word) processing may be required of utterances in which *disfluencies* such as self-correction or interruptions occur [Ginzburg et al., 2007], e.g. *A: There are subsistence farmers that... B: There are what?*. Meta-communicative interactions may also involve clarification requests of syntactic rather than semantic components of an utterance, if say particular words were not heard. However, the majority of such disfluencies can be addressed by a pre-processing stage to correct them (converting for example *Peter was, [pause] well he was fired* to *Peter was fired*). The resolution of sluices or anaphoric usage is dependent not on the latent representation of the discourse context, but of the observed utterances, and thus can be performed independently of dialogue rules and state. Lastly, some meta-communitative interaction such as *misunderstandings* can be addressed by simply embodying this information in the semantic content of the input sent to the dialogue manager, or even some partial representation of the semantics of the utterance with missing components marked. The disadvantages of limiting the dialogue manager to only the semantic content of the input is outweighed by, as we will see, the practical advantages of cleaner rules and simpler code.

2. **Events from multiple agents**: Similarly, many dialogue manager frameworks lack support for receiving event-based input from non-human agents or from additional human agents. In the case of a robot, this may involve a signalling of arrival at a location.

3. **Ease of use**: While perspicuity may seem like an obvious requirement of a discourse framework, in the context of the *EUROPA* project for which our dialogue framework will be deployed, it is of particular paramount importance. Wide-coverage dialogue systems typically employ hundreds of rules using potentially a complex representation of discourse context, and thus facilitating concise and transparent code is critical. In addition, the framework should include development of a powerful debugging tool to analyse the evolving state of the discourse as dialogue moves and other events transpire.

4. **Strongly type-based**: The majority of dialogue systems treat data in a primitive textual or primitive form. While frameworks such as *DIPPER* permit records based on a composition of other structures such as stacks and queues, there is a lack of

any system of type enforcement which bounds types to particular semantic forms of utterances and other inputs. As a result, code written by developers is much more liable to bugs. While *TrindiKit* has a type-checking facility, this is not performed at run-time. In addition, while *TrindiKit* has some limited support for user-defined types, these are moderately restrictive. Such a system of associating types with semantic content is inspired by *Type Theory with Records* [Ginzburg, 2012]. On one level, this embodies the semantic role of data in either our input or the dialogue state; 'inform' acts for example expect an argument with a 'proposition' type, as does a yes-no question. At a higher level, inputs from human agents, if permitting flexible input, would be of type 'list of dialogue act', where the subtypes of a dialogue act are types associated with the chosen dialogue act taxonomy as per Section 4.1.1 (e.g. 'inform'). As a final example, our search algorithm in Section 3.6 for example expects a 'locative expression' type, and produces a list of grounded locative expressions. While ultimately the motivation for weaving such typing into the discourse input, state and external interactions is one of code cohesion, this is strongly predicated on linguistic motives.

5. **Connectivity with external systems**: *TrindiKit* does provide some interfacing to external systems; 'resources' are normal objects within the Information State, whose methods and data can be accessed to provide connectivity with databases and other resources [Larsson et al., 1999]. This implicitly allows some decoupling of the discourse logic with the implementation details of these external interactions, since such detail is abstracted away via use of interfaces that are used within dialogue rules. Given the complex interactions required for example on the EUROPA platform, we must incorporate this ease of interaction. Coupled with the typing system, we can improve on TrindiKit's approach by providing more predictable handling of input and output data with these systems.

6. **Decoupling of generic discourse control and domain specific behaviour**: Modularity both improves code reuse and reduces the risk of introducing bugs. By making the framework suitably generic, we can abstract away dialogue rules pertaining to generic discourse behaviour from domain specific behaviour, as we observed of the *TrindiKit* and *Ravenclaw* frameworks of Sections 4.3.1 and 4.3.2. We explore this in Section 4.4.5.

7. **Incorporate multiple dialogue models**: To further enhance modularity, the framework should allow a number of different dialogue models as outlined in Section 4.2, such the Information State Update model and Finite State Machines. Pre-existing code for each of these models minimises the amount of code required for a domain specific implementation.

Given these concerns, we have developed a framework called *HURDLE*[1], whose core tasks are the handling of dialogue and other multimodal input, maintaining the dialogue state, and deciding on the course of action to take. At its heart is a compact scripting

[1] *HURDLE* is an acronym for *HUman Robot DiaLogue Engine*. An early incarnation of the framework was specific to discourse interactions with a robotic platform, hence the name, although upon further development all robot-specific elements of the framework were abstracted away.

language that is interpreted as *Java* code, verbose enough to allow efficient handling of inputs and data, but only intended to embody the 'logic' of dialogue rules/state, with hooks provided to delegate lower-level implementation to external (Java) code.

We analyse the various components of *HURDLE* in the following sections. Section 4.4.2 provides an introduction to how input to the system (such as human input) is controlled in the context of *HURDLE*'s type system. This underpins Chapter 5 when we examine the problem of translating natural language input to this representation. Section 4.4.3 gives an overview of how dialogue rules which govern the behaviour of the dialogue system operates. Section 4.4.4 examines in more depth the problems associated with the execution order of matching dialogue rules, and the distinction made between *update rules* and *selection rules*. We also examine how such a distinction can be exploited to process multiple speech acts within the same utterance. In Section 4.4.5 we examine how we can implement generic discourse control that are independent from task specific considerations. Section 4.4.6 examines how to handle use of anaphora and how our use of typing can improve resolution to antecedents. We then proceed to discuss an example implementation using the *HURDLE* framework for spatial dialogue in Section 4.5.

4.4.2 Input Representation and Typing

To introduce the scripting language that forms part of the *HURDLE* framework, we first examine how we can construct types to describe our input language for a particular input. As noted, our representation encapsulates predominantly the semantic content of the utterance, and no natural language is seen by the central dialogue control module. For example, we might ascribe a type `DIALOGUEACT` to input from a human user, which can subsequently break down into further types to embody a dialogue act taxonomy and lower-level semantic content. Achieving a dialogue act taxonomy involves simply allowing subtypes to be enumerated. For example:

```
enum DIALOGUEACT {
    informYes,
    inform(PROP),
    requestInfo(QUESTION),
    requestInfoYN(PROP),
    confirm(PROP),
    ...
}
```

Predicative/relational types allow types with a header and optionally a number of arguments each with a given type, allowing us to provide lower level semantic content. For example, the predicate `inform(PROP)` represents a dialogue act where the agent is informing of some given proposition (of type `PROP`). We can then provide type information for `PROP` by specifying its subtypes, which may be domain-independent (e.g. a type `not(PROP)`, representing negation of a proposition) or domain-specific (e.g. `nameOf(AGENT,STR)`, such that an instantiation `nameOf(user,'Bob')` indicates the user's name is Bob). These composite types (among others we will see across the course of this chapter) eventually decompose into primitive types such as strings, integer, reals and boolean values.

Forming the input in such a structured Context Free Grammar-like way (in the sense that types used within other type expressions can be broken down further just like non-terminals), as opposed for example to a predicate calculus based representation, yields a number of advantages. Firstly, the resulting tree structures for instantiations of these types means that extraction of subcomponents of data (i.e. subtrees) is immediate with no need for calculation or theorem-proving; a feature particularly useful when constructing the triggering conditions of dialogue rules (as will be seen in the next section). Secondly, it bounds our input to a constrained language, leading to simpler and more predictable dialogue behaviour. Lastly and relatedly, it lends to the automated generation of Context Free Grammars which can be used to aid parsing from natural language text, as will be seen in Chapter 5.

Concise grammars that reuse predicate constructions and other types are encouraged. For example, we could use our `nameIs` predicate in a variety of different dialogue acts representations:

`inform(nameIs(user,'Bob'))`	"My name is Bob"
`requestInfoYN(nameIs(user,'Bob'))`	"Is your name Bob?"
`requestInfo(?$x.nameIs(user,$x))`	"What is your name?"

The last of these uses a lambda expression to represent the question (as frequently used in discourse and question answering systems), where `?$x.f($x)` represents the expression $\lambda x.f(x)$. In addition to allowing us to reuse predicate constructions in questions, we can construct generic rules to execute beta-reduction, resolving questions with supplied answers to form a proposition which can be asserted to be true (see Section 4.4.5). For type-safety, we can constrain our type `QUESTION` to lambda-expressions which yield values of type *PROP*, i.e: `enum QUESTION { LAMBDA<?,PROP>, ... }`, where the `?` indicates that the lambda expression can take any type as its argument.

4.4.3 Summary of Language Features

The main dialogue control is embodied by a file using *HURDLE* syntax. This is composed of a number of elements:

Input/Output assignments: We saw in the previous section how the input language for an agent, say a human agent, could be specified by an appropriate type system. With each input, we assign therefore an expected type for this input, in addition to a symbol for which this input can be accessed in dialogue rules, the source of the input, and a translator to use which converts the raw source input to an expression of the required semantic type. For example

```
use input CONSOLE , SCFG('my-scfg-rules.txt') , U , DIALOGUEACT;
```

would read input from a console, use a parser for Synchronous Context Free grammars as described in Chapter 5, and values of inputs (which must be of the specified type `DIALOGUEACT`) are assigned to a global variable `U`. Multimodal input is supported, so additional inputs could be defined for say a robotic platform. Output channels can similarly be defined, specifying a translation method to produce natural language text

(for example swapping the left and right hand side of a SCFG if this translation method is used, yielding the reverse translation) and an output sink for the natural language text.

Dialogue State: State is maintained by the use of global variables. The state depends on the particular dialogue model. For example, for a Finite State Machine, the identity of the state is all that is required, for example `global INT State`. For the Information State Update model, we may maintain some set of facts known to be true by agents in the dialogue, e.g. `global SET<PROP> FACTS = {}`. For more complex models such as a POMDP (where we have some distribution over states), we might use `global MAP<INT,REAL> StateDistribution` as the probability mass function from states to their associated probability.

Dialogue Rules: As per many other dialogue frameworks, the central component that governs dialogue behaviour are the dialogue rules, each consisting of a number of conditions and a number of effects. The condition again depends on the particular model in use; for Finite State Machine based systems, the condition would merely be the identity of the current state. For the Information State model, the condition is based on some partial specification of a more complex dialogue state as well the current or historical inputs. For example:

```
U = greet    ->    say(greet);
```

would define a rule representing the act of responding to a greeting with a counter-greeting (*HURDLE*'s equivalent of the ubiquitous 'Hello World!' program), presuming that `greet` has been defined as a subtype of expected type of the input `U`. In a similar manner to Prolog, we can instantiate local variables such that the context they're used in is satisfied. For example:

```
U = inform($p) & $p = nameIs(_,$n)
   ->  FACTS+= $p,
       say( inform(niceToMeet($n));
```

While Prolog uses backtracking while iterating through clauses to satisfy a goal, we limit such instantiation to those which satisfy equalities or matching items contained within a collection of items; it is therefore considerably more efficient.[2] Like Prolog, conjunction is not commutative; the ordering of clauses is crucial given the order in which variables are instantiated.[3] While a lack of backtracking prevents multiple constraints on the same variable in a condition, it prevents unpredictable behaviour when clauses have side-effects (e.g. function calls which modify the information state), and renders error

[2]The underscore allows us to match anything without assigning its value to a variable.

[3]Consider for example:

`$nm = 'Bob' & ¬($nm - NAMES)`

where the `-` operator behaves as $\in$ for sets and lists, and `NAMES` is a global variable containing a collection of names. This has the effect of making the assignment `$nm = 'Bob'`, before checking that this is not contained in `NAMES`. Suppose this is true. If we were to reverse this however, the behaviour is considerably different:

`¬($nm - NAMES) & $nm = 'Bob'`

On the presumption `NAMES` is not empty, `$nm` is instantiated to the first name in the set (since as no restrictions are placed on `$nm`,any arbitrary element is chosen) and the 'element of' relation yields true. This is negated by the negation operator, yielding false.

Action Type	Example
Outputting to the user/output sinks.	`say( inform(answerIs(42)) )`
Updating the dialogue state/global variables.	`LASTRESULT:= $result` or `FACTS+= $p`
Calling a method.	`setRobotTarget(42, 33)`
Direct assignment to a local variable.	`$age = lookupProperty('age',$person)`
Indirect assignment to a local variable.	`('age',$age) ~ $person` determines if some pair `('age',_)` is an element of the collection, instantiating `$age` to satisfy it if possible.

Table 4.1: A summary of the types of statements that can be executed in dialogue rules.

handling difficult when a problem is encountered evaluating a clause. Another language feature shared with Prolog is a distinction between instantiated and uninstantiated variables; in many cases, such as for the arguments of a function call, instantiated variables are required. The *HURDLE* interpreter notifies of such violations.

If a dialogue rule condition matches, there are a number of actions that can be carried out, illustrated in Table 4.1. There is appropriate syntax for set and list construction in the spirit of set builder notation. For example:

```
[2*$i | $i - [1 TO 100]]
```

would give the even numbers between 1 and 100. There is also support for the usual `if`, `while` and `for` constructions found in procedural languages.

In summary, the *HURDLE* language is not intended as a fully furnished programming language, but more a collection of features and syntactic constructs to enable the discourse policy and event handling to be defined. Were it implemented as a Java library rather than as a separate language interpreted in Java, it would be more difficult to see the control logic abstracted away from implementation details. Were it implemented in Prolog in a similar manner to *TrindiKit*, there would be a significant depreciation in efficiency and be more difficult to integrate with existing systems conventionally implemented in procedural languages.

4.4.4 Control Flow, Classifying Rules & Handling Multiple Dialogue Acts

We have so far not specified how dialogue rules with matching conditions are ordered or their execution over time. The ordering of rules is somewhat of a systemic problem particularly associated with plan-based discourse models. Some of such problems are associated with development problems, where unexpected side-effects can easily arise from a different ordering of rules when there are very precise restrictions underpinning updates to the information state and triggers for particular actions. Other justifications for due consideration to rule ordering are more linguistic; we might for example wish to prioritise answering a question before we accommodate longer term goals. *TrindiKit* allows a number of execution strategies, such as taking the first rule that applies (applying

rules iteratively until all rules' conditions are false), executing applicable rules in a given order, or choosing rules according to probabilistic information. *DIPPER* permits only the first of these strategies.

In its existing state, the *HURDLE* framework adopts *DIPPER*'s simple approach in terms of executing rules in a given order, but permits optional ordering of rules by assigning priority weights; those with higher weights are selected first, and those with the same weight are executed in an arbitrary order (resulting in a partial order of rule execution). In addition, we can specify whether a rule can be executed multiple times or once only in a particular execution cycle, where the start of a cycle is when some new input is received, and ending when there are no longer any applicable rules. For rule triggers directly involving input data (like an utterance), clearly we wish the rule to only be executed once, to avoid processing the input multiple times. However, consider the scenario in which we wish to ascertain the identity of multiple entities (e.g. locations) under discussion. If there was a rule that removed this ground goal when the identity of the object was ascertained, the same rule may be legitimately executed more than once in an execution cycle.

Related to execution ordering, *TrindiKit* categorises dialogue rules into two classes. *Update rules* encapsulate changes to the information state, possibly in response to new input. *Selection rules* meanwhile use this updated state to determine the next course of action. Neither [Larsson and Traum, 2001] nor other work by this author illuminate on such a distinction further. One might argue that partitioning rules into these two distinct states is not always possible; *selection rules* under such a definition may make modifications to the information state also, since in the system selecting a particular agenda item, further goals may be created or the `FACTS` set under the ISU model augmented in response to interactions with external systems. Nevertheless, if we refine the definition of an update rule to require that updates are triggered directly in response to new stimuli, with all other updates to information state incorporated as part of selection rules, then such a distinction allows us to exploit various linguistic phenomena.

One such application is the ability to handle multiple utterances within a single dialogue move, as evidenced in this example from the *TownInfo* data-set [Lemon et al., 2006]:

> "Hi I'm looking for a bar but I don't have much money on me and the other thing is I'd like it to be in the south of town because I've a train to catch at the station. Is there anywhere suitable?"

This single move can be segmented into 5 dialogue acts: a greeting, three inform acts and a question. The majority of work on action planning in discourse makes the assumption that each utterance is encoded as a single dialogue act [Allen and Perrault, 1980]; indeed in some research it is a necessary constraint in the model [Traum and Hinkelman, 1992]. [Schmidt et al., 1978] for example permits multiple moves before a plan is carried out, but such acts are restricted to 'inform' ones, and the plan of action more generically pertaining to the performance of tasks rather than necessarily discourse response. There is however a considerable body of research in the process of segmenting dialogue utterances, which we briefly explore in Section 5.5.

How then do we accommodate these multiple acts? One simple approach could be as presented in Algorithm 7. That is, we instigate the update rules only for each of these acts in turn, before executing the selection rules for the move as a whole. Were we to execute

Algorithm 7 A simple algorithm for handling multiple dialogue acts within one utterance.

```
for all dialogue acts U in the utterance do
    execute update rules (using U as input)
end for
execute selection rules
```

all rules for each act in turn rather than constricted to update rules, responses would be generated for each before the rest of the input has been processed. This would cause problems for example if a latter act augmented the information provided in an earlier act, since a plan of action would be carried out after the initial incomplete information. Our algorithm assumes that update rules can be executed independently. While under most circumstances this modelling assumption is adequate, there are a few intra-move adjacency pairs that this violates:

> System: 'By X, did you mean Y'
> User: 'No, I meant Z'.

If we represent the user's utterance as two acts: a no answer and a correction, then the effect of the rule for answer accommodation (see Section 4.4.5) upon processing the first act is to downdate the system's clarification question. Thus upon processing the the second act, the question is no longer in context, and therefore the information provided (i.e. the correction) could no longer be interpreted in light of it. Such a complication can be accommodated by more verbose update rules; by maintaining questions downdated in the current move (e.g. by a `JUSTANSWERED` set in the information state which is emptied at the end of each execution cycle), we can preserve this context.

By analysing the *TownInfo* corpus, we identified the following common combinations of acts that frequently co-occur within utterances. Each is analysed in terms of the strategy we encountered in Algorithm 7. This is presented in Table 4.2.

While most of these can be handled without any special treatment, the last adjacency-pair presents a number of challenges. If we execute update rules independently for each question, each question will be added in order to the QUD stack. Upon executing the selection rules, the latter question would be at the top of the stack, and thus handled first. But we'd typically expect parallelism in discourse of questions with answers, where questions are answered in the order they're asked, not addressed in a 'last in first out' manner. This could inelegantly be handled by collecting questions raised in update rules, deferring their addition to the QUD stack until a suitable selection rule so that their order is preserved. A second challenge is related to the specific example: that of scoping. The second utterance 'does it serve wine' contains use of anaphora that is scoped to the existential quantifier in the first utterance. But since our act representation excludes existential quantifiers in order to simplify rule matching, then to accommodate the scoping we'd require a special reference for the 'it' to indicate that it refers to any result provided for 'suitable nearby restaurants'. The spanning of quantifiers across multiple dialogue acts is also exhibited in Question-Inform and Inform-Inform pairs. The last problem is that of inference. The system could provide the nearest suitable restaurant according to previous requirements in the discourse, before perhaps approaching the second question

Intra-move adjacency pair	Example	Rule handling
Answer - Inform	"No, I need a hotel that serves beer"	See discussion in Section 4.4.4.
Answer - Question	"That sounds great, could I have the address?"	We have seen the effect of the update rule on an answer downdates the existing question under discussion. While the proceeding question issued immediately after may be related, this can be handled independently.
Greet - Any act	"Hello, I'm looking for some chocolates for my wife."	Since the illocutionary force of each act is independent, our strategy is sufficient.
Inform - Inform	"My wife is giving birth, we need to get to a hospital"	Typically when informing acts occur together, latter acts augment the information of previous ones. Although the acts present related information, the overall effect of applying update rules on the acts to add the conjunction of the informed propositions to the information state. Applying update rules on each in turn independently yields this overall change.
Inform - Question	"I'm going to the north of town and I'm looking for a bar there what have you got."	Again, while the two acts are related, informational acts can be processed independently.
Question - Question	"Is there a suitable restaurant nearby, and does it serve wine?"	See discussion in Section 4.4.4.

Table 4.2: A sample taxonomy of commonly occurring adjacency pairs encountered within the same move, as observed in the *TownInfo* dataset. In each, we examine the adequacy of processing the update rules for each act independently (with selection rules subsequently performed for the whole move), in light of potential dependencies between acts.

and declaring the lack of wine provision. However, the second question inferentially modifies the first, requiring that a 'nearby restaurant selling wine' is provided if one exists (even if a nearer restaurant not selling wine is present), and preserving the original unmodified query if one does not exist. Such complex interplay between utterances is very difficult to account for.

In summary, we've explored a variety of issues concerning the ordering and execution of the dialogue rules that determine the behaviour of the discourse system. Rule ordering provided a mechanism to prioritise certain discourse behaviour (such as question answering over satisfying longer term goals). We examined the distinction between update and selection rules, and leveraged this to suggest a method by which we could process the segmented dialogue acts of a single utterance. This strategy was examined against common intra-move adjacency pairs found in a corpus of spatial dialogues, suggesting means by which we could resolve some of the difficulties encountered.

4.4.5 Rules for Task-Independent Discourse Behaviour

As discussed, it is possible to decouple general dialogue control logic from task-specific behaviour. We present some of the update and selection rules found in the *Information*

State Update model. These rely on a number of global variables representing the dialogue state: `SYSAGENDA` is the system's agenda, i.e. the goals the system must carry out, `QUDs` are the questions under discussion, and `LATESTMOVE` is the particular dialogue act being examined in the last utterance. Methods such as `raiseQUD(QUESTION)` and `removeQUD()` push and pop the `QUDs` stack respectively.

Integrating Questions

```
LATESTMOVE = requestInfo($q) & ¬($q ~ QUDs)
        ->      raiseQUD($q);
```

Provided a question is not already under discussion, any question raised by an agent will be added to the Questions Under Discussion.

Integrating Yes/No Questions

```
LATESTMOVE = requestInfoYN($p) & ¬(isTrue($p) ~ QUDs)
        ->      raiseQUD(isTrue($p));
```

The `QUESTION` subtype `isTrue(PROP)` is the question of whether a given proposition holds.

Integrating Requests

```
LATESTMOVE = requestAction($t) & currentQUD() ¬= choiceOfAction($ls)
        ->      abandonAnyConflictingTasks($t),
                SYSAGENDA+= $t;

LATESTMOVE = requestAction($a) & currentQUD() = choiceOfAction($ls)
        ->      if($a ~ $ls) removeQUD()
                else say( inform(invalidSelection($a)) );
```

In some cases, a new request may replace a previous goal rather than augment it. Thus by overriding the empty `abandonAnyConflictingTasks` method, we can specify if and when older system goals can be removed in light of new ones. In others cases, a request is not an initiating move but a response move, answering to an offer of a selection of actions by the system (e.g. *'Do you wish to go there or do you wish me to describe where it is?'*). If the action was a valid option from this selection, we remove the question of choice from the QUDs. The condition of the first rule will then be satisfied, so the task can be added to the agenda.

Accommodating New Information

```
 LATESTMOVE = inform($p) & $p!=answer(_)
        ->      FACTS+= $p,
                FACTSJUSTADDED+= $p;
```

This rule applies only to initiating moves; we therefore exclude *inform* acts when providing an answer to a question.

Accommodating Yes Answers

```
LATESTMOVE ~ {informYes,acknowledge} & currentQUD() = isTrue($p)
        ->      FACTS+= $p,
                FACTSJUSTADDED+= $p,
                removeQUD();
```

If the latest move is either a yes answer or an acknowledgement (since acknowledgement often implies a yes answer rather than merely acknowledging that the question was asked), and by extracting the proposition `$p` for which the current question was asserting the validity of, we add the proposition to our list of facts, and downdate the question.

Integrating Answers to λ-Expressions

```
 currentQUD() = ?_._ & LATESTMOVE = inform(answer($a))
        ->      $q = (LAMBDA<?,PROP>) currentQUD(),
                if(wasValidAnswer($q,$a))FACTS+= $q($a)
                else ...
```

If a question was in the form of a λ-expression, we accommodate the answer by beta-reducing the expression supplied with the answer as its argument, `$q($a)`. Again, we provide an empty method `wasValidAnswer` which we expect to be overridden (it always returns true by default); this determines whether an answer was appropriate to a given question.

Integrating Indirect Requests

```
currentQUD() = isTrue( actionIsPerformable($t) )
        ->      if(isPerformable($t) {
                     say( informYes ),
                     abandonAnyConflictingTasks($t),
                     SYSAGENDA+= $t
                } else {
                     say( informNo ),
                     say( inform(not(actionIsPerformable($t))) )
                },
                removeQUD();
```

We previously observed from the Gricean Axioms that the Maxim of Relevance permits indirect means of having an illocutionary effect, often not strictly following the adjacency pairs that typically relate initiating and response moves. In this case for example, a question *'Are you able to take me there?'* (in its parsed form `requestInfoYN( actionIsPerformable(go(@LOCEXPR) )`) is functioning more as a request than a yes/no question, and thus an yes/no answer in isolation would be considered uncooperative. [Clark, 1979] categorises indirect acts into those which are interpreted in its literal sense, or *pro forma*, where the intended meaning is responded to. For example, in answering *'Do you know the time?'*. an answer could be a two-part mixture of literal response and accommodating the intended meaning, i.e. *'Yes I do. It's six'*, or a one-part response to the intended meaning only, i.e. *'It's six.'* There are a number of properties which determine

this response, including conventionality within a language, purposefulness (considering the intended effect on the addressee) and rationality (reasoning based on background facts which are assumed to be common knowledge).

[Wilske and Kruijff, 2006] handles indirect speech acts in its dialogue system according to a process flow diagram, with the transitions contextually dependent on the information state. For example, given the declaration *'I need a cup of tea'*, a node decides whether to produce an utterance to resolve the ambiguity (*'Would you like me to get you a cup?'*) or to immediately carry out the action.

Our approach bears some similarities. For example, other selection rules observe recently added facts (resulting from declarations by the user) to determine if a particular goal should be added to the agenda. However, for the sake of simplicity, we make the assumption that the *pro forma* meaning is intended, thus taking the implied meaning of any utterances as the user's intent.

4.4.6 Reference/Anaphoric Resolution

Anaphoric resolution, or more generally reference resolution, concerns the task of resolving references in natural language to previous objects mentioned in discourse, such as pronouns. The expression used to reference another is typically known as the *referring expression*, while the entity being referenced is the *antecedent* or *referent*. [Webber, 1978] describes the hearer's mental model of ongoing discourse as the *discourse model*, containing the various entities encountered and the references both between them (where such references are said to *corefer* to the same object) and to tangible entities in the hearer's perception of the world. New entities can be introduced by indefinite and definite noun phrases. With the latter, references can also be to previous entities introduced in discourse. Pronouns cannot refer to new entities, but are able to refer back to both previous entities (anaphora) or to entities yet to be encountered (*cataphora*), such as in "Before *he* instructed the robot, Bob moved it to a suitable starting position". Other types of anaphora include *demonstratives* (i.e. 'this' and 'that', either alone or functioning as determiners) and *one anaphora* (e.g. "I want one").

There are various factors which complicate resolution. Pronouns may be bounded within quantifiers, as for example in "Every robot planned *their* route to the destination". Clearly, the route is specific to each robot rather than one used by all; it thus behaves as a variable bound to the quantifier. *Inferrables* [Haviland and Clark, 1974] are entities where some inference can be made in terms of their relationship to another entity, for example in "The robot was ready to commence its journey, but *a wheel* had broken".

There are a number of factors that determine how references may be resolved. These can broadly be divided into *constraints*, that impose restrictions on valid resolutions, and *preferences*, which provide heuristics to best choose between multiple valid referents. Constraints include *number agreement* (e.g. plural pronouns must refer to plural objects), *gender agreement* (i.e. 'he' versus 'her') and syntactic constraints (e.g. 'Bob bought *him* a robot' versus 'Bob bought *himself* a robot'). While preferences are moderately subjective in their relative weighting, *recency* is arguably the most dominant, where references to entities occurring closer to the referring expression in discourse are more likely. Other preferences include the grammatical role of the referent (entities in the subject position of a verb are more likely than the object position), the frequency of mention of the entity

(those which dominate the discourse at a particular point are more likely to be referred to) and verb semantics. In the last of these, the default preference for the subject position of a verb for a referent can be overridden by the particular function for the verb. For example, with the verb *telephoned*, the subject is dominant, whereas with *criticised*, the object is dominant, as in 'Bob criticised the robot. *It* lowered its head dejectedly'.

The task then is to provide at least some support for such resolution in our discourse framework. There have been a large variety of existing approaches that account for some of the constraints and preferences. [Lappin and Leass, 1994] tackles the problem specifically of pronoun interpretation. It uses a weighting scheme for many of the selectional preferences discussed, so that a candidate referent in the subject position obtains a score of 80 for example, an a maximum score of 100 for recency, which halves which each subsequent sentence. The values from these various salience factors are added to produce an initial salience score. The 'discourse model' of [Webber, 1978] is explicitly generated, since each of the entities in the discourse so far are enumerated and assigned a weighting. These scores are initially independent of the pronoun; we therefore calculate the final score for each entity in the discourse model for the particular pronoun we wish to resolve, taking into account additional salience weights specific to the positioning of the pronoun, with a negative weight to penalise cataphora and a positive weight for the occurrence of pronoun *parallelism*, where pronouns occurring in a non-subject position of a verb are more likely to resolve to referents also in a non-subject position (e.g. 'Bob fixed *the broken robot*. Jim then took *it* to the repair shop'). The entity in the discourse model with the highest overall salience score is chosen as the referent, provided it adheres to the referential constraints previously described.

An alternative approach of [Hobbs, 1978] does not explicitly use the discourse model, instead searching through syntactic representation in tree form of the previous sentences in discourse, searching in particular for subtrees either rooted by a noun phrase of a sentence (since demonstratives can refer to an entire sentence constituent (e.g. 'It was *this* mistake that the robot reflected upon'). Selectional preferences are not explicitly represented either, although these are approximated by the order in which the syntactic trees are traversed by using appropriate traversal rules (for example, subjects of verbs are favoured over objects due to left-to-right traversal order of nodes in trees.

Resolution of definite and indefinite noun phrases (when first introduced) to objects in the real world however is often intrinsically linked to appropriate grounding dialogue, which we cover in Section 4.5.3. As for anaphoric resolution, we previously noted it can occur independently of the dialogue manager, since their resolution to antecedents is independent of the internal state of the dialogue. However, we can exploit the semantic representation of our natural language utterances to yield extra selectional constraints that would not be available using grammatical structure alone as in [Hobbs, 1978]. Suppose we were to use the syntax `@TYPE` to indicate an anaphoric item of type `TYPE`. Suppose also for example we were to use the semantic representation `requestAction(go(@LOCEXPR))` to represent the utterance "Go to it/Go there" (since our predicate `go` expects an argument of type `LOCEXPR`, representing a location). Then consequently we know that any antecedent we find in the semantically parsed previous utterances must have the same semantic type, i.e. `LOCEXPR`. That is, while the pure grammar syntactic tree could theoretically match any Noun Phrase for our pronoun 'it', we can use the semantic context of the pronoun to infer it specifically refers to locations (in this particular case, because

it is associated with the verb 'go'), and thus constrain our search considerably.

Given this, a simple algorithm for resolving anaphora, given a list *HISTORY* can a list of dialogue acts in parsed form, could be as such:

1. Find each occurrence of `@TYPE` in the semantic construction, i.e. each anaphoric usage.

2. Attempt to locate a item of type `TYPE` in the last item of `HISTORY`. For predicates or other constructs with multiple arguments, the search order is determined by the specific construct.

3. If an item is found of this type, we return it immediately as the resolved item and replace the `@TYPE` with this instance.

4. Otherwise, we search in the dialogue act occurring next in `HISTORY`, up to a maximum of 3 utterances from the end of the discourse.

5. If no item is found, the anaphoric usage is left as an unresolved item (which can be accommodated in the dialogue rules by suitable verbal feedback to resolve the item).

There a number of observations to make about this algorithm. Firstly, in a similar vein to [Hobbs, 1978], recency dominates, with the first matching candidate antecedent encountered chosen for our resolution. The difference, as discussed, is the tighter constraint of ensuring semantic suitability of the antecedent. Secondly, we can enforce subject/object preferences of particular verbs or other grammatical preferences by the order in which arguments of semantic constructs are traversed. For example, take the semantic representation of "The car is by the tree":

```
inform(
         objDesc(
              LOCEXPR{ dA: true, class: 'car' } ,
              LOCPROPS{ relations:
                   [ By( LOCEXPR{dA:true, class: 'tree'})
                   ]}
         )
      )
```

where the predicate of `objDesc` indicates that a location/object (its first argument) is described by the second argument. If resolving "How far is it?" (where the 'it' in the semantic representation has type `LOCEXPR`), then although there are two items of matching type (i.e. the car and the tree), we can choose to traverse the first argument of `objDesc` first as a general policy for this predicate, thus yielding the intended expression "How far is the car?". Synctatically, this correponds to favouring the head noun phrase over noun phrases contained in the adjunct prepositional phrase.

The other resolution process is to account for indefinite/definite noun phrases that refer to previous entities in discourse rather than introducing them. Consider for example:

System: "There's the restaurant just over there with the blue door, and another by the river."
User: "Great, let's go to the one over there then."

Since the two options are both grounded from the perspective of the system (that is, their identity is known to the system), resolving the user's reference to one of these is integral to avoiding additional disambiguating conversation (since the user's utterance in isolation may yield multiple matches). To match, we employ a simple process to compare equality specifically of location/object description pairs, or more specifically, whether the information provided in one is a subset of the other, so that we permit the loss of information in the referring expression (i.e. the loss of the 'with the blue door' adjunct and the 'just' prepositional modifier), although do not permit any new information being introduced. If no object in the recent discourse history matches, the expression is not resolved and treated as an expression instead referring to some object in the environment. If an object does match, we copy the identifier of the antecedent into the referring expression, indicating that both expressions *corefer* to the same real-life object.

One final anaphoric phenomenon we handle are *discontinuous* sets, where plural pronouns can refer to sets of entities evoked together. The antecedents are 'discontinuous' in the sense that they occur separately, potentially embedded within different subtrees in the syntactic structure of the sentence. Consider for example:

System: 'You're by X and right in front of Y.'
User: 'How far are *they*?'

The *'they'* clearer refers to the entities X and Y. This causes problems with our type matching approach, since the type of 'they' in context is `LIST<LOCEXPR>`, while X and Y individual `LOCEXPR` types embedded within different prepositional phrases. Since such grouping is dependent on the particular syntactic construction, we require additional rules dependent on the task domain. In this particular instance, when searching for a `LIST<LOCEXPR>` and encountering a list of prepositional phrases (i.e. `LIST<PREPOSITION>` in the semantic *HURDLE* form), it is sufficient to collect the landmark objects from each to produce a suitable list as desired.

4.4.7 Other Framework Features

Other core components of the *HURDLE* language include:

- **External Functions** allow a developer to hook external code to the dialogue logic while maintaining type coherence. By specifying within a file that type signature and name of the function (for example, a method `robotGo` may have the type signature `FUNCTION<INT,STR>`, which takes the identifier of an object for a robot to travel to, and returns a response string), the compiled class can be dynamically loaded by *HURDLE* and subsequently used. We use such functions for all communication with the robot and use of the various spatial algorithms of Chapter 3. As discussed, this reduces contingency on lower-level implementation details in the dialogue control logic, with the requirement of type specification ensuring greater predictability of the data being passed in and out of these functions.

- Function signatures support generic typing. For example, the function `function A first(PAIR<A,B> $pair)` might return the first item of a pair, ensuring the type of the first item of the pair matches the output type.

- **Namespaces** improve modularity by grouping constants, global variables and functions (within a particular file) by a given prefix. For example, accessing such items from the *ISU* discourse model would be done so by prefixing them with `'ISU.'`. This syntax will be seen frequently in Section 4.5. This is equivalent to static class members/functions found in most Objected Oriented Programming languages.

4.4.8 Debugging

As [Bohus and Rudnicky, 2009] identifies, the use of an Information State Update model or other less constricted model can make unexpected behaviour difficult to track. We can deal with runtime errors (usually when external functions experience some problem rather than within the dialogue rule logic) in the HURDLE by specifying a 'error strategy' function, which allows the system to recover appropriately, for example, verbally informing the user an error occurred, logging the error and/or refreshing the dialogue state so any existing questions or agenda items are abandoned. However, problems are more likely to arise by problems in dialogue logic, whether it be rules being fired in the wrong order, rules that are triggered when they shouldn't (or aren't when they should), or changes in state having unexpected side effects. Given the dialogue logic is a separate syntax interpreted in Java, it allows us to easily track when rules are fired, changes to state are made and input/output events occur.

Figure 4.3 shows the debugging tool developed to achieve this aim. 'Log events' are created on the fly as HURDLE runs; each event has the value of all global variables at the time the event occurred, and in the case of notification of rules being triggered, the instantiation of local variables is given that resulted in the rule's clause being satisfied.

4.5 Spatial Conversations

4.5.1 Introduction

We turn our attention now to the task-specific elements of a dialogue system implementation, specifically, a system which operates on the *EUROPA* platform and facilitates conversations about space. There are a number of aspects of the task we need to account for, centred around the discussion of and interaction with objects/locations in the environment. This includes making requests for the robot to move involving landmarks in the environment, discussing objects, and providing disambiguation conversation to identify the user's intended object when they have a particular one in mind, or recommend one when a specification is given.

There has been much research in developing dialogue systems operating on a robot platform, with varying degrees of discourse flexibility. [Kruijff et al., 2007] presents a robot named "PeopleBot", capable of augmenting autonomously acquired metric maps with quantitative information, by interpreting user's descriptions about objects in the environment. It also handles commands, such as requests to go to a destination, turn,

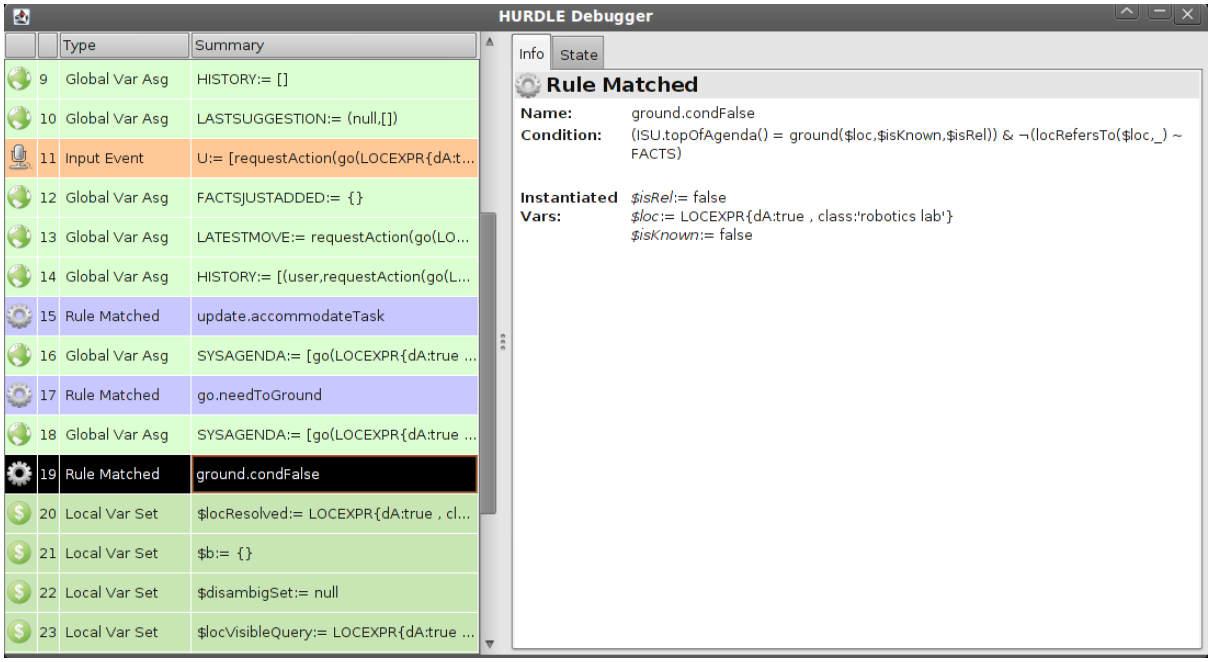


Figure 4.3: The HURDLE debugging tool.

and for the robot to follow the user (using the appropriate visual tracking systems). The contrasting roles of user and system are described as a 'mixed control strategy', which combines the robot's autonomous capabilities with an appropriate level of human control. This is not to be confused with mixed-initiative dialogue, but instead delegates control to the robot when the need arises, for example requesting the robot find an object without the need for further human intervention. This is known in other work as *sliding autonomy* [Heger et al., 2005] or *adjustable autonomy* [Dorais et al., 1999]. [Bos et al., 2003b] meanwhile presents *'Godot'*, which the user can direct to specific locations, ask for information about its status, and supply information about its environment. It uses an internal map for navigation, communicating its current orientation and accessible locations (e.g. doorways). Using the Dialogue Move Engine of the *DIPPER* framework we explored, it uses various update rules to accommodate simple dialogue, including establishing contact with the user, clarification dialogue if the speech recogniser fails and answering questions. [Matsui et al., 1999] similarly presents an indoor robot specifically designed for office use, such as answering queries about people's location, route guidance and delivery tasks.

The most similar dialogue system to our own in terms of task-domain is the *TALK* system we saw in Section 4.3.3, based on the *TownInfo* dataset [Lemon et al., 2006]. This concerns the identification of locations in an urban environment and requesting properties about these (e.g. locating restaurants which serve Indian cuisine, asking whether it is five star, etc.). However, the reinforcement learning approach limited versatility of the dialogue system, and it did not operate on a robotic platform.

We now explore how we develop our own system for accommodating spatial conversations. In Section 4.5.2, we explore the data structures we use to represent locative expressions and their associated semantic/spatial properties. In Section 4.5.3, we explore the task of disambiguating objects in the environment. In 4.5.4, we propose how we can

adapt the behaviour of the system depending on whether definite articles (i.e. 'the') and indefinite articles (i.e. 'a') are used.

4.5.2 Data Structures

In HURDLE, we can form records in a similar manner to other dialogue frameworks. Our most fundamental type is a *locative expression* (`LOCEXPR`), representing any reference to an object or location in the environment, whether a particular object or a specification concerning an object required. It is defined as such:

```
structure LOCEXPR {
        INT[?@] id,
        PART[?@] part,
        STR[?] class,
        LIST<PREPOSITION>[?] relations,
        STR[?] name,
        LIST<ATTRVAL>[?] attrs,
        BOOL[?] isVisible,
        BOOL[?] dA,
        BOOL[?] multiple
};
```

The `[?]` field modifier indicates that the field is optional. The `@` modifier indicates that the fields with this marker combined are sufficient to indicate equality of two locative expressions. This is particularly useful for reference/anaphoric resolution, since if two locative expressions have an identifier set to indicate reference to the same object, then by definition they are equal, even if in one expression more information may have been provided than in the other.

A summary of the field is as such: `id` is a identifier for a specific grounded object. `part` is the part of an object (e.g. entrance, centre), `class` is the semantic category (e.g. laboratory). `relations` is a list of spatial relations. `name` is the name of the object, if applicable. `attrs` is a list of semantic attributes of the object, excluding spatial prepositions (e.g. what style of food it serves). `isVisible` specifies whether we require the object to be seen or unseen, used for example in the parse of 'What can I see?'. `dA` refers to whether a definite article was used, i.e. 'a car' compared to 'the car'. Finally `multiple`, if true, refers to a collection of objects matching the rest of the specification, i.e. 'the trees'.

This definition uses the `PREPOSITION` type representing a spatial preposition.

```
enum PREPOSITION {
        prep(ONEARGPREPTYPE, LOCEXPR, BOOL, BOOL),
        atDist(LOCEXPR,REAL),
        nearOrdinal(LOCEXPR,INT),
        ...
};
```

The relation `prep` represents a preposition with associated landmark (e.g. 'by' and 'near'). The two booleans control whether the preposition is negated (e.g. 'not by the lab') or if the head noun phrase and noun phrase in the prepositional phrase attachment is reversed (e.g. 'the grass the tree is on' versus 'the tree on the grass'). The latter was useful in generating descriptions in Section 3.3. For convenience, we can use `prep(ONEARGPREPTYPE, LOCEXPR)` where the two boolean parameters are presumed to be false. Some spatial relations require different information, so have separate predicate types. `nearOrdinal` for example represents the superlative 'nearest' and allows us the specify the ordinal, e.g. 'the third nearest'.

Perhaps the most used predicate in our system is `objDesc(LOCEXPR,LOCPROPS)`, which indicates that the object/location specified by the first argument can be described by the second. `LOCPROPS` is very similar to the record type `LOCEXPR`, except that there is no id attribute (since it represents a description of an object, not the object itself). For example:

"What can I see?" `requestInfo( ?$x.objDesc($x,LOCPROPS{isVisible:true}) )`

"Is it far?"
```
requestInfoYN( objDesc(
.  @LOCEXPR,
.  LOCPROPS{relations:{prep(Far, CURRENTLOC)}}) )
```

The second of these examples uses the constant `CURRENTLOC`, an instance of `LOCEXPR` representing the current position of the observer.

The last construction key to spatial conversations is the predicate `ground(LOCEXPR)`, which is a subtype of `TASK`. This represents the task of grounding/establishing an entity's identity.

4.5.3 Grounding Objects

In general, *grounding* provides a crucial function in dialogue of ensuring that the content of an utterance is understood and acknowledged by the addressee.
[Traum and Hinkelman, 1992] introduces the notion of 'conversation acts', extending the traditional notion of speech acts to encompass behaviour specifically relating to higher-level discourse behaviour such as turn-taking and grounding. Grounding acts can be grouped together to form a single *discourse unit*, representing some information being successfully communicated or aborted. They categorise grounding acts into classes such as initiation (where some assertion or question is made), repair (where content in the current discourse unit is modified), continuation (where additional information is provided in a separate speech act), acknowledgements and requests for acknowledgement/repair (e.g. 'is that OK?'). [Kruijff et al., 2006] examines clarification for its Human-Augmented Mapping robot in [Kruijff et al., 2007], where this additional dialogue is required for grounding when uncertainty arises in automatic classification of objects (e.g. doorways), there is inconsistency between an object's semantic information and that which the user provides or there are faults in perception caused by the robot hardware (e.g. localisation problems). This grounding is accommodated by appropriate update rules. In [Bos et al., 2003b], clarification is due to low confidence scores from the speech recognition module, asking the user to repeat information.

We specifically examine here the task of grounding locative expressions to either their intended object by the addressee, or by providing an arbitrary one that matches their specification (a distinction we explore further in Section 4.5.4). Such identification can be thought of as a higher-level variant of grounding, since multiple discourse units may be spanned in order to achieve it, involving multiple question and declarations which may have been grounded in the traditional sense.

We now present a worked example of such a grounding process, incorporating many of the discourse units for lower-level grounding we have observed. Consider the following conversation:

> U: I'm looking for the Indian restaurant. It's near the park. S: Did you mean St James' Park? U: Yes. S: There are 3 Indian restaurants by St James' Park. U: Which is the closest? S: Jamal's. U: OK. I'll go there then.

We analyse the steps that were carried out in order to provide this mixed-initiative grounding conversation.

- **Step 1a: Perform the update rules on the first two utterances in turn.** The semantic parse of the user's first utterance is as follows:

```
 inform( wantsAction(find(LOCEXPR{dA:true, class:'restaurant',
.  attrs:[(cuisine,'Indian')]})) ),
inform( objDesc(@LOCEXPR, LOCPROPS{
.  relations:[prep(Near,LOCEXPR{class:'park'})]}) )
```

  By using the 'accommodate new information rule' in Section 4.4.5, the `wantsAction` proposition is added to the information state, specifically to `FACTS` and `FACTSJUSTADDED`. The second dialogue act is an example of the *grounding continuation* we have seen, which must modify the proposition added from the first act to reflect this new information. Presuming we have resolved the anaphoric usage and again applied the 'accommodate information' rule, we can achieve this using the following rule:

```
objDesc($loc,$lprops) ~ FACTSJUSTADDED
      -> $augmentedLoc = $loc + LOCATIVE.convertToLocExpr($lprops),
         FACTS:= [replaceLocExpr($f, $loc, $augmentedLoc)
                                  | $f ~ FACTS]
         FACTSJUSTADDED:= [replaceLocExpr($f, $loc, $augmentedLoc)
                                  | $f ~ FACTSJUSTADDED];

function T replaceLocExpr(T $f, LOCEXPR $l1, LOCEXPR $l2) {
  if($f = find($l1))return find($l2),
  if($f = wantsAction($a))return wantsAction(
          replaceLocExpr($a, $l1, $l2)),
  ...
}
```

The result is that the extra described properties `$lprops` of the locative expression `$loc` are appended together (using the append operator + on records), before replacing each fact `$f` with its variant containing the augmented information. Consequently, we now have `wantsAction(find(LOCEXPR{class:'restaurant', attrs:[(cuisine,'Indian')], relations:[prep(Near,LOCEXPR{class:'park'})]}))` on our list of recently added facts.

- **Step 1b: Use the model of the user's desires/intentions to generate the appropriate system goal.** For our first *selection rule*, we can simply accommodate any action the user wishes the system to carry out, adding it to the system agenda.

```
wantsAction($a) ~ FACTSJUSTADDED  & $a !~ SYSAGENDA
     ->  SYSAGENDA+= $a;
```

- **Step 2: In dealing with the 'find' goal, we create 'ground' goals for all locative expressions**. Before we can successfully complete our 'find' goal, the associated locative expression must be fully grounded. We separate the tasks of grounding landmarks and grounding the referent itself. Thus for example, grounding "the Indian restaurant by the park" would involve the creation of two ground goals: the Indian restaurant by *[the grounded park]*, and the park. The latter grounding must clearly occur prior to the previous one, since grounded landmarks are required prior to the referent which we ground relative to these.

```
ISU.topOfAgenda() = find($loc) & needsGrounding($loc)
   ->  createGroundGoals($loc, false);

function BOOL needsGrounding(LOCEXPR $loc) {
        if($loc['id']!=null)return false,
        if(locRefersTo($loc,_) ~ FACTS)return false,
        if(($loc['dA']=null | $loc['dA'] | $loc['name']!=null)
               & ground($loc,_,_) !~ SYSAGENDA)return true,
        if($loc['part']!=null & $loc['part']['parent']!=null) {
                if(getGroundedItem($loc['part']['parent'])!=null)
                      return true
        },
        if($loc['relations'] = null)return false,
        foreach($r ~ $loc['relations']) {
                if($r = prep($p,$l1,_,_) & needsGrounding($l1)) {
                        return true
                }
                ...

function VOID createGroundGoals(LOCEXPR $loc, BOOL $isLandmark) {
        if($loc['id']=null & ($loc['dA'] | $loc['name']!=null))
                SYSAGENDA+= ground($loc, isKnown($loc), $isLandmark),
```

```
        if($loc['relations'] != null) {
                foreach($r ~ $loc['relations']) {
                        if($r = prep($p,$l1,_,_)) {
                                createGroundGoals($l1,true)
                        }
                        ...
                }
        },
        ...
};
```

The function `needsGrounding` determines if either the referent or any of its landmarks requires grounding. If the entity has an identifier, then by definition the object is grounded. If in addition the object has not been previously grounded, then we recursively descend into the landmark objects to determine whether these have been grounded. The lack of an identifier does not necessarily means an object requires grounding; if for example an indefinite article is used, then grounding is not required, since an arbitrary match will suffice (see Section 4.5.4). `createGroundGoals` similarly adds the ground goals to the system agenda.

- **Step 3: Accommodating a ground goal**. Grounding of locative expressions is divided into a number of stages. We first determine if any landmark objects of the referent in question have been grounded, and if so, replace the ungrounded landmarks in the locative expression with these. The second step is to use this (potentially resolved) locative expression to search the environment for matching objects, to produce a disambiguation set of candidate objects.

  The set we use depends on the mode of grounding. Suppose for example that the user is taking the initiative in describing an object known both in identity/location to them but not yet grounded by the system. If the object is visible, the set to disambiguate to is typically restricted to only other matching objects that are similarly visible. For example, were the user to have said 'go up to that Indian restaurant', then the presence of a visible match excludes occluded restaurants, since an agent can presume that other dialogue participants will make such an assumption on the dominance of visibility. This operates in reverse: when the system generated a description of a visible object as per sections 3.2 and 3.3, we limited our distractor set to visible objects since we presumed the interpreting agent would implicitly assume this distractor set. This however only applies if the addressee is aware of the location of the object in question. For our particular scenario, the user wishes to *find* the object, thus by definition, the user is unaware of its location, and thus the distractor set cannot be restricted in such a way.

  The next step is disambiguating from the resulting candidate objects/distractor set. If there is one match only, we ask the user to confirm their intended object (except under certain conditions, as will be seen). If there are 2 or 3 matches, there is a sufficiently low number to enumerate the candidate objects and ask the user to clarify which they intended. Any more, and it would be cumbersome to list all the choices to the user. We instead inform the user of the number of matching

objects, and rely on their initiative to disambiguate further, for example, additional declarations about the object in question so that the 'accommodate information' rule we have seen augments the locative expression's level of detail.

The steps can therefore be implemented as such:

- **Step 3a: Update our landmark objects (if required).**

```
ISU.topOfAgenda() = ground($loc,$isKnown,$isLandmark)
          & getGroundedItem($loc)=null
   -> $locResolved = resolveGroundedLocExprs($loc),
      ...
```

The `getGroundedItem` function determines the item a locative expression has been grounded to. If the object has an identifier, it is already explicitly grounded and thus the method returns the expression itself. If a locative expression was previously grounded, then a `locRefersTo` predicate will be contained within the information state, and thus we wish to avoid grounding the object again. This condition has slightly different behaviour to the `needsGrounding` method we encountered, since here we are simply testing whether the referent itself needs grounding, rather than any landmarks it refers to.

- **Step 3b: Determine the distractor set.**

Given that the `$isKnown` variable determines whether the location of the entity is known to the addressor, as previously discussed[4]:

```
...
if($isKnown) {
   $locVisibleQuery = $locResolved + LOCEXPR{isVisible:true},
   $visibleResults = robSearch($locVisibleQuery),
   ($rs,$status) = $visibleResults,
   if(#$rs>=1) {
      $disambigSet:= getResultsOverThreshold($rs,THRESHOLD),
   }
   else {
      ($allResults,_) = robSearch($loc),
      $disambigSet := getResultsOverThreshold($allResults,THRESHOLD),
   }
}
else {
   ($allResults,_) = robSearch($loc),
   $disambigSet := getResultsOverThreshold($allResults,THRESHOLD),
},
LASTSEARCH:= ($loc, $disambigSet)
...
```

[4]The # function gives the size of the set or list following it.

- **Step 3c: Use the distractor set to determine the disambiguation strategy.**

  If the distractor set has just one object, then the decision is whether to request confirmation from the user, or whether to assume that they would be comfortable with the selection. If for example the description generated for the object happens to be the same as that used by the addressee, we don't request confirmation (avoiding scenarios such as *'Go to the restaurant in front of you', 'Did you mean the restaurant in front of you?'*). Similarly, to avoid cumbersome conversation (in aid of Grice's axiom of conciseness), we avoid confirmation if grounding a landmark object (indicated by the `$isLandmark` variable). This is not entirely arbitrary; since landmarks are by definition to aid in disambiguation of the referent. Therefore, landmarks are likely to be chosen such that they are easy to disambiguate, and thus there is less of a need to confirm their identity if the landmark expression does indeed disambiguate the distractor set to just one object.

  Notice that in the `generateDescripDis` method (to generate a distinguishing description using the algorithm in Section 3.2) we again use information on whether the location of the entity is known to the user, embodied by the `$isKnown` variable. If the user does not know the location of the object (although by virtue of the fact we are grounding the object, can assume they know its identity), then disambigating the object based on spatial properties would be counterproductive.

```
% (a) No results. ----------------
if(#$disambigSet = 0){
   ANAPHORA.say( inform(dontHaveKnowledge($loc)) ),
   ...
}

% (b) 1 result. -------------------
else if(#$disambigSet = 1){
   $r = UTILS.head($disambigSet),
   $vis = robVerify($r,LOCPROPS{isVisible:true}),
   $descrip = generateDescripDis($r,$isKnown),
   $descripWithoutId = LOCATIVE.stripLocExprOfIds($descrip),
   $locWithoutId = LOCATIVE.stripLocExprOfIds($loc),
   if($locWithoutId = $descripWithoutId) {
      FACTS+= locRefersTo($loc, $r)
   } else if($isLandmark) {
      FACTS+= locRefersTo($loc,$r)
   }

   ...

   else {
      ISU.raiseQUD( isTrue( locRefersTo($loc,$r) ) ),
      if($isRel)ANAPHORA.say( requestInfoYN(locRefersTo($loc,$descrip)) )
      else ANAPHORA.say( requestInfoYN(locRefersTo($descrip)) ),
      LASTSUGGESTION:= ($loc,[$r]),
```

```
      stopMatching
   }
}
```

In the case of 2/3 objects in our distractor set, we enumerate the possibilities and ask the user to make a selection to disambiguate between them.

```
else if(#$disambigSet = 2 | #$disambigSet = 3){
   $descrips = [ generateDescripMutDis(
         $r, $loc,
         $isKnown | UTILS.firstOfPair(robVerify($r,LOCPROPS{isVisible:true}))
        ) | $r~$disambigSet ],
   ISU.raiseQUD(whichMeantBy($loc,$descrips)),
   ANAPHORA.say( requestInfo(whichMeant( $descrips )) ),
   stopMatching
}
```

If there are too many objects to enumerate, we inform the user of the number of matches, as discussed.

```
else {
   ANAPHORA.say( inform(thereExistsM(
          LOCMULTMATCHES{quantity:#$disambigSet,loc:$loc})) ),
   stopMatching
};
```

- **Step 4: Integrate user response** In our initial conversation, we searched for matches of the landmark object, and yielded one result. The user was asked to confirm their intention of *St James' Park* as the landmark. We can use the 'integrate yes answer' update rule from Section 4.4.5 to process the user's response. This in turn adds `locRefersTo(X,Y)` to the information state, where `X` was the original landmark expression and `Y` was its proposed (and accepted) referent. We can use a 'ground completion' rule to accommodate this new information and remove the grounding of the landmark object as a system goal. This is in essence the goal equivalent of downdating for Questions Under Discussion, where the successful answering of a question provided sufficient grounds for it to be removed from the information state.

```
ISU.topOfAgenda() = ground($loc,_,$isLandmark)
         & getGroundedItem($loc)!=null
    ->  SYSAGENDA-= ground($loc,_,$isLandmark);
```

- **Step 5: Disambiguating the referent.** We now proceed to ground the referent object (the restaurant) in a similar manner. This time, there are three results. The user then takes initiative by trying to round down the selection, asking which is the

closest. Such a question implicitly refers to a set of objects to which to apply our 'nearest' operation. This can be handled with an anaphoric entity in our semantic representation of the question, and use our anaphora resolution process to resolve this to the '3 Indian restaurants'.

- **Step 6: Grounding completion.** With the user's *go* request at the end of the dialogue, the grounding process is implicitly complete. The adoption of the *go* goal abandons any conflicting or redundant goals, and thus the *ground* and *find* goals are removed from the system agenda.

4.5.4 Grounding Identity versus Specification Satisfaction

We have previously seen that whether the location of an object is known to the user or otherwise can affect discourse behaviour. This affected the choice of descriptive properties to use regarding an entity when disambiguating, as well as the distractor set to disambiguate from using visibility considerations. However, grounding an object provided by the user assumes the exchange of knowledge of the identity of an entity from the user to system.

This however may not be case; frequently the addressee merely formulates a *specification* of a desired entity, for which any arbitrary suggestion matching this specification is adequate, as for example in *'I need to find a toilet'*. One might think that this distinction naturally extends from *definite* articles and *indefinite* articles, which precisely intends to identity nouns identifiable to the listener, and those which are not. This is slightly complicated by occasional elliptical usage, for example "I'm trying to find a restaurant" may imply "I'm trying to find a *particular* restaurant". However, for the purposes of simplicity, and given the difficulty in disambiguating between the two cases without additional contextual information or further disambiguating dialogue, we use the class of the article (indicated by the `dA` field) to determine whether grounding is required or not required.

In the latter case where only an arbitrary entity suggestion is required, the most natural policy is to recommend the closest matching entity. For example, in the 'find' task, this can be accommodated by the selection rule:

```
ISU.topOfAgenda() = find($loc) & $loc['dA'] = false
    ->  $nearestOneQuery = $loc + LOCEXPR{
                relations:[nearOrdinal(LOCATIVE.CURRENTLOC,1)]},
        $origresults = robSearch($nearestOneQuery),
        ($rs,_) = $origresults,
        $results = getResultsOverThreshold($rs,THRESHOLD),
        if(#$results = 0){
            ANAPHORA.say( inform(dontHaveKnowledge($loc)) ),
            ...
        }
        else {
            $r = UTILS.head($results),
            $dist = robDistance($r,LOCATIVE.CURRENTLOC),
            $ll = $nearestOneQuery+LOCEXPR{id:$r['id']},
```

```
        ANAPHORA.say( inform(objDesc($ll, LOCPROPS{
              relations:[atDist(LOCATIVE.CURRENTLOC,$dist)]}))  ),
        LASTSUGGESTION:= ($loc,[$r]),
        $q = choiceOfAction([
            go($ll),
            giveAnswer(whereIs($ll))
        ]),
        FACTS-= wantsAction(find($loc)),
        ISU.removeTask( find($loc) ),
        ANAPHORA.say(requestInfo($q)),
        ISU.raiseQUD($q)
    };
```

The effect is to augment the original locative expression to prefix it with "the nearest". After searching for matches (of which there will either be 0 or 1), we inform the user of the distance to the nearest matching entity, before offering a choice of actions between describing its location, or being escorted there.

4.6 Conclusion

In this Chapter, we presented a new generic dialogue framework that builds upon the strengths and weaknesses of similar frameworks. The framework's main features were the handling of multimodal input, a Prolog like syntax that allowed easy matching and extraction from user input and dialogue state, and flexible enough to accommodate a variety of discourse models, and a tight type system that reduces errors, couples naturally with natural language parsing and improves anaphoric resolution. We also examined a strategy in which multiple dialogue acts in one utterance, while avoiding many of the complications that arise.

A specific implementation for spatial conversations using this framework is used on the EUROPA platform, and we evaluate this dialogue system in conjunction with the robot's other functionality in Section 6.6.

Chapter 5

Parsing

5.1 Introduction

Semantic parsing is the process by which natural language text can be converted to a suitable representation of its *meaning*. This representation may be subsequently processed in a variety of systems, whether for the purposes of discourse, the semantic web [Guha et al., 2003] or question answering systems [Voorhees and Dang, 2003]. The motivations for such a representation is to utilise the information content within text, while discarding superfluous properties of the text such as syntactic representation. One advantage is that we unify sentences that embody the same meaning but are expressed in a variety of different ways. Another is the potential use of theorem provers to combine a meaning representation with contextual sources to make further inferences. In this chapter, we specifically explore the issues surrounding parsing for discourse, and consider the issues and challenges that arise when wishing to implement a robust parser for task-oriented dialogue.

We have already observed in Chapter 4 that the expected input type of our system is a structured representation of the utterance which is subsequently processed in the dialogue rules. We motivate our exploration by revisiting our previous example from the TownInfo corpus, found in Figure 5.1. The figure however shows the semantic representation produced by the methods we present in this chapter, rather than the shallow (and inaccurate) representation used in the original annotated corpus.

In order to accommodate such complex input, there are a number of associated challenges to consider, as well as issues more generally pertaining to semantic parsing of input in a discourse context:

5.1.1 Expressiveness

Semantic representations (frequently referred to as *Meaning Representation Languages* or *MRLs*) can vary significantly in their verboseness. Shallow representations reduce the computational complexity of processing input and leads to simpler handling, but at the expense of limited representation of meaning. Representations such as *predicate calculus* more fully represent meaning and phenomena such as scoping, but can be fairly unconstrained, potentially requiring theorem provers to reason about input. We have previously seen that systems such as *GoDis* [Larsson et al., 2000], and indeed a large

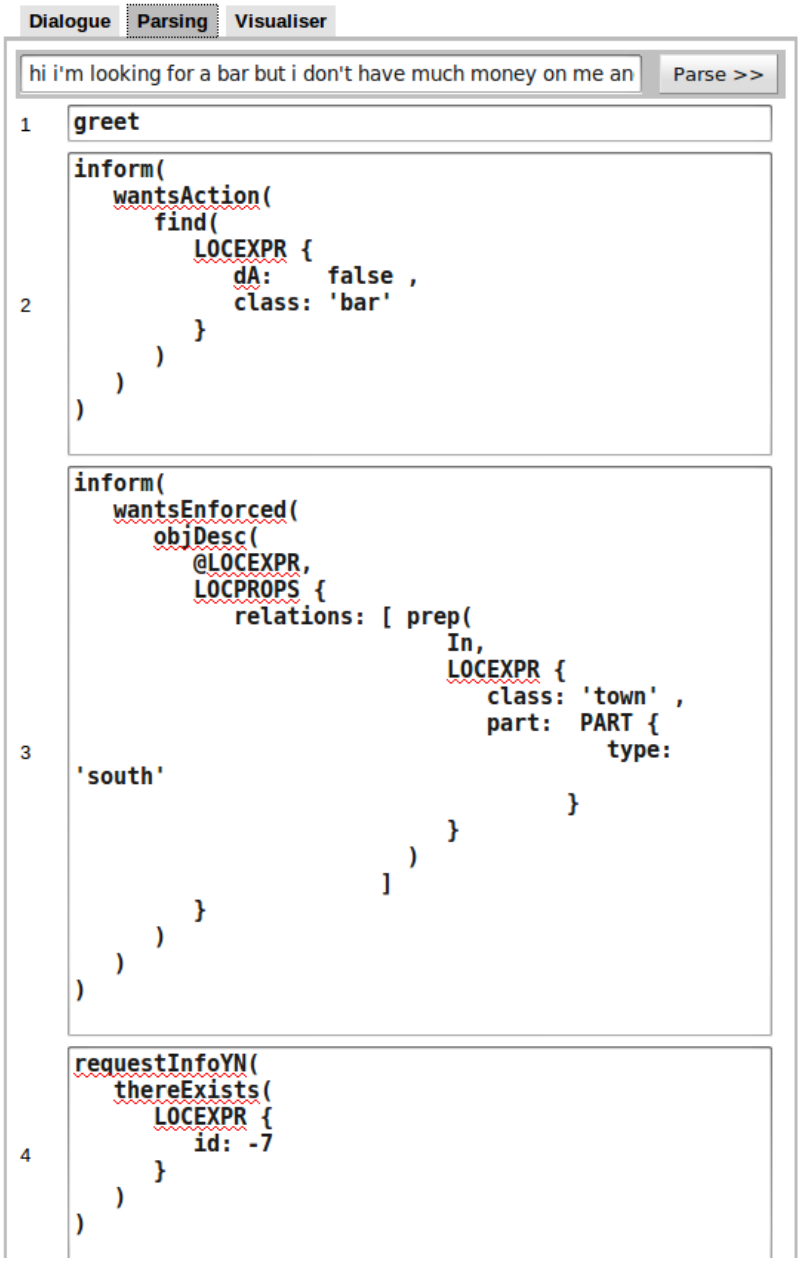


Figure 5.1: The outputted semantic parse from HURDLE's parsing system for the utterance: *"Hi I'm looking for a bar but I don't have much money on me and the other thing is I'd like it to be in the south of town because I've a train to catch at the station. Is there anywhere suitable?"*. The *-7* identifier is a reference to a locative expression that matches some previous requirement, in the same way for example that *-2* is a special identifier designating the current location.

proportion of dialogue systems, use a simple keyword spotting approach. Others such as the *TALK* system used the type of dialogue act coupled with attribute-value pairs (such as *cuisine=Indian*).

Outside of dialogue contexts where concerns of representation are not necessarily fettered by practical constraints, more explicit attention is given to the encapsulation of meaning. [Alshawi, 1992] uses an intermediate meaning representation for language known as *'quasi logical form'*. In this form, variables (whether unscoped or scoped) and representations of anaphoric and other referential usage are in an initially unresolved state. By fixing the scope of variables and resolving references based on the context of the sentence (e.g. the preceding sentences), a fully logical form can be obtained. The purpose of the intermediate form is to separate the issue of the compositionality of semantics from scoping/reference resolution, since we can compose the meaning of this intermediate form and defer the latter tasks to avoid multiple interpretations of the utterance before the fully logical form is required. This makes use of rules to parse text that consist of syntactic grammar rules coupled with semantic rules that dictate how the meaning of the constituents are composed, as described in [Allen, 1987]. For example, if using a pure logical form[1]:

$$\begin{aligned} S \rightarrow NP\ VP &: NP(VP) \\ Det \rightarrow every &: \lambda P.(\lambda Q.(\forall x.(P(x) \rightarrow Q(x)))) \end{aligned}$$

[Alshawi, 1992] extends first order logic to allow more complex constructs not directly encapsulated by this formalism, for example, the inclusion of more generalised quantifiers that accommodate expressions such as *'most A are B'*. [Van Noord et al., 1999] points out that such a full representation of meaning leads to utterances having multiple meanings, and identifies the associated expensive in processing them all. It instead opts for a variant of *Definite Clause Grammar* [Pereira and Warren, 1980], providing a better balance between computational efficiency and linguistic expressiveness. Their resulting grammar formalism, *OVIS2*, caters for various clausal types found in discourse, such as declaratives and yes/no questions.

An increasingly popular representation for MRLs is the use of Context Free Grammars, frequently employed in machine translation methods where we'd typically wish to translate a natural language sentence according to the source language grammar to a sentence in another natural language according to another. We will explore a formalism known as *Synchronous Context Free Grammars* [Lewis II and Stearns, 1968], which associates each Context Free Grammar rule in the source language to one in the target language, allowing us to generate sentences in each language in parallel. This formalism has been extended by some learning approaches to *λ-SCFGs* [Nguyen et al., 2008] [Wong and Mooney, 2007], which extends rules with lambda variables to enable questions to be better represented. In this extension, non-terminals can be parameterised with variables, so that further rules involving variables use this correct bounded variable.

The type of representation used for our own purposes must adhere to the expected input type of the dialogue system, notably, conforming to the structured representation with some underlying grammar. The fullness of meaning captured is dependent on the

[1]These rules are higher order, but reduce to first order.

particular task domain, but should be arbitrarily rich. We explore this in Sections 5.2 and 5.6.

5.1.2 Quality of Input

Another consideration is the quality of data. In cases where the natural language text is outputted by a speech recognition engine, errors such as misinterpreted words are likely to arise. But as previously identified, discourse in general may contain disfluencies, contractions and ellipsis, interjections (*uh*, *erm*), ungrammaticisms (e.g. lack of number agreement) or colloquialisms. In the case of written dialogue, we may also need to account for spelling errors. Existing semantic parsers have a variety of strategies for coping with these errors.

We saw in Section 4.3.3 that some dialogue systems involving *partially-observable* Markov Decision Processes can incorporate the uncertainty in the speech recogniser's output into the belief distribution over possible states. The approach therefore accounts for all possible hypotheses of the user's utterance rather than making a single choice of the most likely. [Zettlemoyer and Collins, 2007] meanwhile extends its set of rules in the particular grammatical formalism used, *Combinatory Categorial Grammars*, or *CCGs* (which we explore in Section 5.1.3), to better accommodate the grammatical irregularities often found in spoken language, for example: *"flights one way"*, *"flights boston to new york"*, *"dallas to washington the latest on flights"*. Other approaches [Kate et al., 2005] incorporates optional words into rule patterns, so that non-terminals/terminals in CFG rules can be interspersed with a sequence of arbitrary words up to a specified maximum number. This allows words not integral to the content to occur, e.g. adverbs, without requiring additional rules to be specified.

Such accommodation of superfluous words is implicitly supported in approaches known as *concept spotting*, where the emphasis is to spot *meaning units* rather than sentences [Ward, 1989]. These work by identifying maximum sequences of words that conform to given patterns, which are subsequently inserted into a semantic frame to provide an amalgamated structure. Concept spotting provides more flexibility than keyword spotting by allowing larger substrings of the input to be detected than mere words, and account for a variety of problematic issues with input associated with stricter forms of parsing, such as missing words, spurious words or phrases, out of order constituents and elliptical usage. However, such an approach is still hampered by the lack of a mechanism to unify these matched fragments, preventing more complex structures which require specific orderings of these matched constituents to provide higher-level structure.

We will see that our parser that satisfies the aims of this section strikes a balance between approaches such as concept spotting, which offers greater robustness against ungrammatical/colloquial input but a shallow representation of the input, and grammar-based approaches which typically offer richer representations but make strict requirements in terms of fully grammatical input.

5.1.3 Manually Configured vs Learned Parsers

While semantic parsing has traditionally been achieved by manual specification of rules that determine patterns to identify in input, more recent approaches have moved to-

wards automating the process of determining these rules. Most of these approaches have varying degrees of supervision. [Zettlemoyer and Collins, 2007] for example trains based on natural language sentences coupled with their corresponding lambda calculus form parse, but requires CCG rules governing the source language. Other approaches such as [Wong and Mooney, 2007], [Blunsom et al., 2008] and [Kate et al., 2005] require less supervision by requiring the source and target strings as training data only, with no other prior knowledge or rules required. Other approaches are more unsupervised; [Poon and Domingos, 2009] does not require the target MRL strings in training, although does however require syntactic parses of the source string, as well as the target predicates and object constants pre-specified via supervised semantic parsing. However, all these particular approaches require a degree of inference, due to the lack of exact alignments in the training data between components of the source and target representation (e.g. correspondences between particular source words and particular predicates in the MRL).

Approaches to learn this correspondence from source language to MRL differ greatly in their approach. Those grounded in more traditional machine translation techniques employ a data structure known as a *Synchronous Grammar*. While usual grammar rules define the permissible strings in a particular language, synchronous grammars simulateneously generates a string both in the source and target language. For translations between two natural languages, these string pairs represent the same sentence in each language respectively. We can define Synchronous Context Free Grammar rules for example in a similar manner to CFG rules, but associate each non-terminal on the left hand side with a pair $\langle \alpha, \beta \rangle$, where each represents a list of non-terminals and terminals. Non-terminals on the right-hand-side are assigned alignment numbers, such that when a non-terminal is expanded in the first rule of the pair, the corresponding non-terminal in the second rule with the same alignment number is expanded synchronously. For example:

$$NP \rightarrow \langle ADJ_{\boxed{1}} N_{\boxed{2}}, N_{\boxed{2}} ADJ_{\boxed{1}} \rangle \tag{5.1}$$

$$N \rightarrow \langle cat, chat \rangle \rangle \tag{5.2}$$

$$ADJ \rightarrow \langle black, noir \rangle \tag{5.3}$$

might represent a rule that encapsulates how adjectives and nouns swap between English and French. We subsequently expand each aligned pairs of non-terminals, until we yield two strings of terminal symbols.

[Blunsom et al., 2008] is one such method which attempts to induce SCFGs from parallel sentence pairs. It makes use of a *Hierarchical Dirichlet Process*, in this case a generative model yielding a distribution over possible SCFG rules. These rules are restricted to three types: *emissions* which generate a single terminal, *binary monotone* productions which generate two pairs of non-terminals that preserve the order (for example, the ordering of noun phrase and verb phrase are preserved between English and French), and *binary reoordering* productions which swap the two non-terminals, as with the first rule of the above example. By use of Gibbs Sampling, we can then determine the most likely parse of a given source sentence by taking the full Bayesian posterior over all possible sets of SCFG rules which may generate the training data. The advantage of such an approach is the rigorous use of probabilistic inference to yield an exact distribution over possible parses, and no grammar is required for the target language. However, the

restrictions imposed on the grammar rules in order to make learning tractable also limit the versatility of translations, which we discuss in 5.1.4.

Other approaches involving SCFGs *do* take advantage of a provided grammar for the MRL. [Wong and Mooney, 2006]'s approach first parses the training target sentence according to the associated grammar to produce a parse tree, making the assumption that the grammar is unambiguous in order that only one parse is obtained. This is then linearised to produce a listing of production rules used in the parse. A standard word alignment model can then be used to align specific words (or multiple words) in the source sentence to these production rules. This is subsequently used to generate a lexicon of possible couplings of source string patterns to produce SCFG rules, increasingly making the source patterns more generic as we refer to MRL production rules further up the parse tree. A variety of feature functions (such as the number of times each MRL rule is used in the training data) are then used combined with a suitable learning method to yield a distribution over derivations for a given test instance. This is a semi-supervised approached, since candidate derivations are generated without the prior availability in training data of synchronous grammar rules used in each instance, or the CFG rules used in the source language. We earlier saw the system could be extended to accommodate λ-SCFG rules in [Wong and Mooney, 2007], by permitting non-terminals in MRL rules to be replaced with variables bound by the associated lambda expression in the rule, while in the process of constructing candidate synchronous grammar rules. [Nguyen et al., 2008] adopts a similar approach, although uses an online method of learning to deduce the weights of the feature vectors, yielding greater accuracy.

Learning for parsing to logical form is intrinsically harder due to the less structured form of the MRL. The aim is to produce predicate calculus expressions, potentially involving lambda calculus expression. For example:

Sentence:	"list flights to boston"
Logical Form:	$\lambda x.flight(x) \wedge to(x, boston)$

[Zettlemoyer and Collins, 2007] makes use of CCG rules. Each word in a sentence can be associated with a CCG expressions that governs the manner in which is can be combined with other constituents in the sentence. It makes use of forward and back slashes: we might associate a rule *(N\N)/NP* with the word *to*, to express that the word combines with a noun phrase to its right, before expecting a noun to its left to yield a noun overall. [Zettlemoyer and Collins, 2007] then couples the rule for each word with a lambda expression corresponding to its semantic interpretation. This is similar to the *Semantically Augmented Parse Tree* of [Ge and Mooney, 2005] for example that combines CFG grammar rules with lambda expressions, but provides the advantages of the CCG formalism that handles a wide range of linguistic phenomena such as coordination and scoping [Steedman and of Techn, 1996]. The learning approach operates by generating a lexicon of candidate CCG-lambda calculus pairs that may be used for parsing, comparable to the way we saw [Wong and Mooney, 2006] generate a lexicon of syntactic CFG-MRL pairs. Such generation uses a small predefined set of rules that use each training example to generate lexical items of a particular form. For example, if a training instance contains a predicate p with a single argument, we'd add an item $N : \lambda x.p(x)$ to the lexicon. These units are then used to form all possible derivations of a training sentence by considering all applications of subsets of these rules to form to full sentence. By applying the same

process to test instances to yield all possible derivations, the approach subsequently uses a perceptron classifier to determine the optimum of these derivations, with weights in the classifier learned from the training data. An appropriate feature function is formed based on each candidate derivation coupled with the test instance.

In summary, there are a variety of methods that can be used to learn different types of semantic representation, whether logical or using a more structured grammatical formalism such as a CFG. These differ in the level of supervision needed. However, all such approaches require large amounts of training data, and only achieve good accuracy by making certain assumptions regarding the data. [Blunsom et al., 2008] and [Kate et al., 2005] for example limit the changes in structure between source and meaning representation language, rendering the representation of longer-range dependencies difficult without excessive numbers of SCFG rules. We examine this issue in the next section.

5.1.4 Capturing long-range dependencies

We saw above that approaches such as [Blunsom et al., 2008] assume that the correspondence between source and target language exhibits *isomorphism*. Strict isomorphism would imply that the derivation trees in source and target languages would be structurally identical except for renamings in leaves. However, this definition is relaxed to permit reordering of two non-terminals between the right-hand-sides of a SCFG rule. We saw for example that this accommodated translations such as English-French, where certain single terminals can be relabelled (e.g. *cat* into *chat*), and minor structural changes can occur, such as the reversal of the ordering of adjective and noun. The translation between natural and semantic grammars however is considerably more complicated. Consider the example below, using our target representation from Section 4.5:

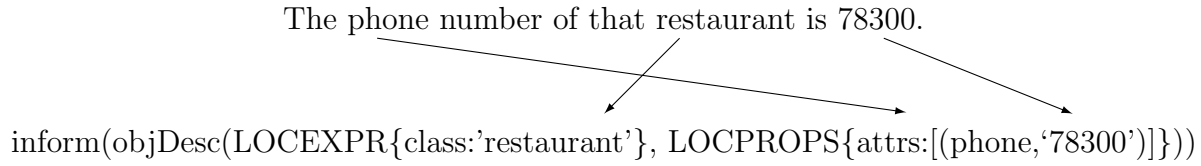


Figure 5.2

Suppose we were to represent both source and target strings are derivations according to their respective grammars. Due to the presence of long-range dependencies, i.e. words from both the start and end of the source sentence become adjacent at the end of the target string, compact SCFG rules become impossible to define. We might wish to generate a sub-derivation in the target language associated with the noun phrases 'the phone number of that restaurant' and '78300', and combine the resulting trees in some way. However in the target language, it is evident from the bracketing that we group the semantic units *phone* and *78300* into a subtree that is separate from the *restaurant* unit. The resolution is to either require more verbose and less general SCFG rules (leading to flatter derivation trees), or require extending the SCFG formalism to *Synchronous Tree Substitution Grammars*, accommodating more general transformations between tree fragments. Such rules however are more unwieldly and cumbersome to define, and more

difficult to learn than SCFGs without simplifying assumptions, given the exponentially greater number of possible STSG rules [Cohn et al., 2010].

[Wong and Mooney, 2007] alleviates such problems by transforming the form of the training instance to one which is equivalent but is more isomorphic in its parsed tree form to the source tree. However, this is based ad hoc on *conjunctions* being represented in the target grammar (by use of a CFG rule $CONJ \rightarrow ARG_1 ARG_2$), and exploiting the commutatively of conjunction to permit the associated CFG rules to have their non-terminals reordered and moved between rules. We can see with the example above that such a technique is not applicable[2]. Our method must accommodate the large degree of non-isomorphism intrinsic between the source grammar and meaning representation language for our particular task domain.

5.1.5 Recognising Sentential Fragments

[Fern'andez and Ginzburg, 2002] describes a number of *non-sentential utterances* that exist in discourse. These are 'fragments' that prototypically consist of short answers to questions (including yes/no answers), but also include acknowledgements (*'OK'*), fillers that complete gaps left by the previous utterer, propositional modals (e.g. *'probably'*) and *clarification ellipsis*, where the content of a question is omitted and usually resolvable from the previous utterance (e.g. *'Where?'*).

[Van Noord et al., 1999] accommodates some of these fragment types by allowing certain constituents which would usually be matched as part of a complete sentence to function as a top level constituent. Noun phrases or prepositional phrases may therefore be offered as a match in the absence of a fully formed sentence. This can be achieved by CFG grammar rules which generate such constituents (e.g. *NP*, *PP*) from the root node.

Given that sentential fragment usage is abundant in discourse, our system must adequately accommodate it.

5.1.6 Utterance Segmentation

We previously explored in Section 4.4.4 how we might process multiple dialogue acts within a single utterance, in terms of the appropriate discourse behaviour. A challenge from the perspective of parsing is to partition an utterance into substrings corresponding to each act. There has been a substantial amount of work in this task of *dialogue act segmentation* [Ang et al., 2005] [Granell et al., 2010]. This typically uses features such as n-gram statistics (e.g. words or phrases that commonly occur at dialogue act boundaries) or prosodic features such as intonation and pauses. Many dialogue corpuses however either assume the utterance consists of a single dialogue act (e.g. the *TownInfo* corpus [Lemon et al., 2006], which poorly represents utterances with multiple acts when they occur in the data), or assumes the utterances have already been segmented.

A parsing system which accounts for multiple acts may either opt for a strategy of pre-segmenting utterances before parsing each segment, or we might consider a means by which we can combine the segmentation and parsing process.

[2]The only HURDLE construction in which ordering is irrelevant is the fields within structures/records. e.g. *STRUC{a:1, b:2}* is equivalent to *STRUC{b:2, a:1}*

5.1.7 Reversibility

We have so far examined the task of converting a natural language sentence into a MRL. Dialogue however is a two way process, requiring both interpretation and generation of natural language utterances. We therefore require the reverse process: to take the output of a discourse system in MRL form and realise it using natural language. A number of discourse systems approach the task as a separate one: [Larsson and Traum, 2001] for example provides *template generation*, where a template sentence is first generated based on the semantic form of the utterance (e.g. "Go to X"), and data within the utterance is used to fill the gaps. However, we'd ideally wish to choose a translation method which yields both directions without requiring implementation of a separate system for each.

The methods we explored involving the conversion to a MRL in a logical form, in both the cases when the conversion process was learned and manually specified, do not provide this reverse process. Synchronous Grammar rules meanwhile intrinsically yield both, since the source-target rule pairings do not prescribe any directionality. It is therefore sufficient to swap source and target components of each SCFG rule to yield a string in the source language.

5.2 A Summary of our Approach

We evaluate the success of our parsing system as one which successfully accommodates as many of these aims and challenges as possible, working robustly on our robotic platform.

Our approach uses a Synchronous Grammar. There are a number reasons that motivate this choice of translation technique. The first is the choice of representation for the Meaning Representation Language. We previously argued in Section 4.4.2 for a tree structured representation, conforming to an implicit grammar indicated by the type constructs. This had a number of advantages. Firstly, it both simplified dialogue rules in terms of extracting subtrees for use in rule conditions or further data manipulation. Secondly, as identified by [Wong and Mooney, 2007], it constrains the MRL to a predictable and structured representation. Thirdly, it provides sufficient expressibility to represent large syntactic constructs with potentially nested information (such as locative expressions). And lastly, the use of tree representations across instances can naturally be described according to a particular Context Free Grammar. As was identified in Section 5.1.3, generating the CFG for the MRL is required for many learning approaches, when forming part of a SCFG used for parsing. We will examine in Section 5.6 how our dialogue framework can be used to generate CFGs to describe the permissible values of permissible semantic inputs (whether for utterances or other inputs).

There are many other advantages of the use of synchronous grammars. They provide the process of language generation without the need for any additional implementation, and sentential fragments as well as segmentation can be accommodated via appropriate CFG rules, the latter of which we examine in Section 5.5. While CFGs intrinsically expect fully grammatical input, we can accommodate some noise in data (such as superfluous words or information) by suitable extension of the CFG formalism, which we explore in Section 5.4.

We have opted not to learn the translation process. This is in part due to large degree of non-isomorphism we have discussed, where the SCFG grammar rules are sim-

ply too complex without the flexibility assumed in [Wong and Mooney, 2007] to be able to transform the semantic forms in the training data. Accounting for this would require annotating an intractable quantity of data, since the number of the possible SCFG rules increases exponentially with the number of non-terminals and possible alignments permitted in each rule. If we consider our original motivating example in Figure 5.1, it is evident with dialogue act segmentation, large amounts of noise, and high levels of non-isomorphism, learning is infeasible without suitable prior assumptions (such as segmentation being provided).

The other motivating factor is the high accuracy required given the context of the parser's use in conjunction with a discourse system. While failed parses do not result in major repurcussions (given we can ask an agent to repeat their utterance if appropriate), errors in parsing may result in incorrect information state or plans action which can be conversationally cumbersome to rectify. One final disadvantage of learning grammars is that the resulting grammar more not necessarily be human readable. In [Blunsom et al., 2008]'s approach, since the induced grammatical structure of source and target languages is latent (i.e. never seen in the training data), the non-terminals in the SCFG rules have arbitrary labelling. These may naturally correspond to actual constituents such as noun phrases, but this may not be the case. As such, it is difficult to augment a set of learned rules when new sentence constructions need to be quickly added.

The input to the system is expected to be directly entered in text form rather than the output of a speech recognition module. This was due to an early decision in the EUROPA project to not use such a module; in part due to the exit of a research company specialising in speech recognition, and the testing of the robot in environments with non-English/American users where the large array of accents would likely cause issues with the recognition accuracy. Input comes the mobile interface described in Chapter 6.5.

The remainder of this chapter is as follows: In Section 5.3, we consider the means by which we can parse Synchronous Context Free Grammars. In Section 5.4, we examine how we can represent our SCFG rule set more compactly by providing more flexible specification of SCFG rules without changing the grammatical formalism, and how this can be accommodated in the parser. In Section 5.5, we explore how we can extend the rule set to provide dialogue act segmentation, as well as pruning information irrelevant to the MRL. Finally, in Section 5.6, we demonstrate how we can generate a Context Free Grammar autonomously from the dialogue manager, and explore the benefits this brings.

5.3 Initial Parsing

We have advocated Synchronous Context Free Grammars as our choice of translation mechanism. As we have seen, each SCFG rule consists of two CFG rule, the result being that two corresponding strings can be generated from the grammar instead of one. There are multiple existing algorithms for parsing CFGs. The most popular of these is the *CYK Parser* [Kasami, 1965] and the *Earley Parser* [Earley, 1970]. The former requires the grammar to be in *Chomsky Normal Form*, where all rules must be in the form $A \rightarrow BC$ or $A \rightarrow a$ for some non-terminals A, B, and C and terminal symbol a. While all CFGs can be converted to CNF, we note that our particular Meaning Representation Language contains constructions involving multiple terminal symbols (for example, a 'predicate'

construction consists of the predicate name, opening and closing brackets and potential internal commas), leading to a significant increase in rules required to represent the MRL. Further, suppose we define the rank of a SCFG to be the maximum number of non-terminals on the right-hand-side of each of its rules. Then any SCFG of rank 4 or above cannot be expressed in such a way that each of its source and target rules can be expressed in Chomsky Normal Form (i.e. rank 2). Consequently, the CYK Algorithm is eliminated as a potential choice of parser[3].

In contrast, the Earley Parser operates on arbitrary grammars, and uses a dynamic programming top-down approach that uses 'dot' indicators to maintain the portion of the right-hand-side of a CFG rule that has currently been parsed against an input string. The derivation trees can be retrieved by following back-pointers where non-terminals were successfully parsed. If the grammar is *unambiguous*, one derivation tree will be produced for strings conforming to the grammar.

Extending parsing to Synchronous Grammars is as trivial as parsing the source sentence according to the source component of each synchronous grammar rule. If we maintain the CFG rules used in the parse of the source sentence, we can use the correspond target grammar rules to reconstruct the derivation in the target language, making use of the alignments between non-terminals in source and target rules.

5.4 Enhancing Grammar Expressibility

CFGs define possible strings that may be generated by a language. For many task domains, our lexicon may be large or theoretically unbounded, given the potential list of place names or nouns. Enumerating SCFG rules for each instance of these would be cumbersome. Instead, it would be useful to be able to conflate all such instances into one rule embodying the string pattern to match, often known in parsing literature as a *rule/grammar schema*. With regular expressions (over characters rather than words), matching arbitrary characters would traditionally be accomplished using pattern matching. We extend this to SCFGs by permitting alignments on terminal symbols and introducing a special symbol $*$ to indicate that any word may be matched. The alignment ensures that both of these symbols in source and target rule are instantiated to the same concrete symbol. The below example indicates the SCFG rules that might be used to allow conversions such as *'the restaurant'* to **LOCEXPR{class:'restaurant', dA: true}**.

$$\text{C} \rightarrow \langle \text{the [1](CLASS)}, \textbf{LOCEXPR \{ [1](CLASS) , dA : true \}} \rangle$$
$$\text{CLASS [1]*} \rightarrow \langle \text{[1]*}, \textbf{class : [1]*} \rangle$$

It should be noted that this extension is equivalent to the SCFG formalism provided that the permitted range of $*$ is finite, since such rules can be replaced with their concrete instantiations. The implication is that the properties of the language do not change, such as the complexity of parsing or the decidability of various language tasks. We also permit many-to-one mappings for alignments on terminal symbols. This has the effect of

[3]Although the CYK+ algorithm [Chappelier and Rajman, 1998] would resolve this particular difficulty.

appending multiple terminal symbols together into one (space separated) terminal, and is useful in capturing compound proper nouns. For example:

$$\text{N} \rightarrow \langle \text{[1]* [2]*}, \textbf{LOCEXPR \{ name : [1,2]* \}} \rangle$$

Permitting any arbitrary symbol for * symbols would lead to a high number of derivations upon parsing. We therefore allow *filters* within the source rule that constrain the permitted matches according to some associated function. In addition, *transformation functions* allow some arbitrary transformational process on the corresponding terminal in the target rule. For example, we could reformulate our rule for matching nouns with determiners like such:

$$\text{CLASS} \rightarrow \langle \text{[1]*noun}, \textbf{ class : [1]*str} \rangle$$

where `*noun` is a filter which matches nouns only, and ***str** is a function that encases a symbol in quotes[4]. We use the *Stanford Parser* [Klein and Manning, 2003] to obtain the Part of Speech tags associated with each word. Tagging a word in isolation would lose information, and the filter however acts upon a single terminal. To resolve this problem, our framework allows a full string to receive an arbitrary annotation across all symbols at once (in this case, the POS tagging), such that filters can subsequently access this annotation for each individual symbol.

Transformational functions may or may not be reversible. In cases where they are not, i.e. no inverse function is provided, the translation from source to target language can only be performed one way for this particular rule; the rule is consequently ignored if translating back to the original language. If the inverse function is provided, then the reverse translation can occur by performing the inverse function on the terminal in the MRL form, and subsequently checking if the filter holds in the natural language. For example in the rule:

$$\text{C} \rightarrow \langle \text{the [1]*plural}, \textbf{class : [1]*singularise , multiple : true} \rangle$$

in the right-hand-rule the symbol in the MRL following the colon symbol would be converted to plural form (i.e. the reverse of finding the singular form). This would naturally satisfy the filter on the left-hand-side rule.

Implementing a parser to handle these extended rules involved augmenting the means by which the Earley parser carries out *scan* rules. The filter function is applied to the current symbol in the string being parsed to check it can be scanned. If it is, we store the matched value according to the alignment number accompanying the terminal in the source rule. When reconstructing the target derivation tree as before, this matched value can be retrieved, with the transformational function applied if given.

5.5 Dialogue Act Segmentation and Pruning Superfluous Information

Multiple dialogue acts can be accounted for by appropriate SCFG rules. As identified, the advantage of incorporating this segmentation explicitly into the rule set is that we

[4]This function is particularly useful in conjunction with many-to-one terminal alignments, since we ensure that the words comprising the symbol are treated as one when tokenising a string in the target language.

consider only segmentations which yield valid parses for each individual dialogue act.

$$\begin{aligned}
\text{ROOT} &\rightarrow \langle SCOMP\,,\ [\ \textbf{SCOMP}\ ]\ \rangle \\
\text{SCOMP} &\rightarrow \langle [1]S\,[2]SCOMP\,,\ \textbf{[1]S}\quad\textbf{[2]SCOMP}\ \rangle \\
\text{SCOMP} &\rightarrow \langle [1]S\,,\ \textbf{[1]S}\ \rangle
\end{aligned}$$

The resulting derivation trees are right-branching, where `ROOT` indicates the top of the tree, `S` a single dialogue act, and `SCOMP` allows a composition of dialogue acts. In our particular MRL, the dialogue acts in semantic form are collected in a HURDLE list construction (hence the use of the square bracket terminals in the first rule), since we wish our output form to be of type `LIST<DIALOGUEACT>`.

However, in our original motivating example, dialogue acts were interspersed with contextual information not used in the translation process, that is, there were portions of text whose semantic content was irrelevant to the particular task domain. Ideally, our parser should identify the portions of the input string according to that encompassed by our grammar, while discarding unnecessary information that would otherwise yield our input unparsable. We can achieve this by incorporating additional rules into our grammar. These match an arbitrary number of `*` terminals surrounding each full sentence.

$$\begin{aligned}
\text{ROOT} &\rightarrow \langle [1](WILDCARDS)\,[2](SCOMP)\,[3](WILDCARDS) \\
&\qquad ,\ \textbf{[1](WILDCARDS)}\ \textbf{[2](SCOMP)}\ \textbf{[3](WILDCARDS)}\ \rangle \\
\text{ROOT} &\rightarrow \langle [1](SCOMP)\,[2](WILDCARDS) \\
&\qquad ,\ \textbf{[1](SCOMP)}\ \textbf{[2](WILDCARDS)}\ \rangle \\
\text{ROOT} &\rightarrow \langle [1](WILDCARDS)[2](SCOMP) \\
&\qquad ,\ \textbf{[1](WILDCARDS)}\ \textbf{[2](SCOMP)}\ \rangle \\
\text{WILDCARDS} &\rightarrow \langle *\,,\ \textbf{ignore}\ \rangle \\
\text{WILDCARDS} &\rightarrow \langle *[1](WILDCARDS)\,,\ \textbf{ignore [1](WILDCARDS)}\ \rangle
\end{aligned}$$

The *ignore* terminals in the target tree (corresponding to symbols in the source tree which are superfluous information) are stripped from the resulting derivation trees. Matching portions of the input is akin to the notion of *concept spotting* we previously examined. However, use of these rules alone would produce large numbers of valid parses, since it might deem all but a single small segment of the input string to be superfluous, even if in an alternative parse these segments are parsable and included in the derivation tree. To counter this, we can simply favour parses with the minimum number of *ignore* symbols, which in turn maximises the portion of the input that can be parsed by the remainder of the rule set.

5.6 Generating the Target CFG Autonomously

We previously saw that numerous approaches to learning grammars involving SCFGs were contingent on a CFG for the target Meaning Representation Language being made available. All such approaches encountered assumed that the MRL grammar was *unambiguous*, which allowed a single derivation tree to be produced for the target form in

each training instance. In Section 4.4.2, we proposed a rigid type structure for the *HURDLE* framework that governed the permissible forms of dialogue input (including, but not restricted to, parsed utterances from a human user). The task therefore is to analyse the type constructs for a particular dialogue implementation to generate a suitable CFG representation.

We analyse some of these type constructs and state how they naturally yield corresponding formal CFG rules.

- **Input**: Each input channel to the dialogue system was assigned a particular type, for which concrete inputs had to be valid instantiations of, for example, an input from a human agent was of the type $LIST\langle DIALOGUEACT\rangle$. We therefore add a single CFG rule of the following form, where T is a non-terminal representing this particular type.

  $$\text{ROOT} \rightarrow \text{T}$$

  where ROOT is the starting non-terminal.

- **enum**: These constructs allowed types to be declared as subtypes of another. For example `inform(PROP)` and `requestInfo(QUESTION)` may be subtypes of `DIALOGUEACT`. Rule generation is trivial: for each subtype of the parent type, we generate a rule where the latter can generate the former. If P is the parent type:

  $$\forall C \in P : P \rightarrow C$$

- **Predicate Types**: Predicate types consisted of a header, followed by 0 or more arguments according to the particular definition of the type, for example `informYes` and `nameOf(AGENT,STR)`. Rule generation is again simple: on the right-hand-side we simply have a mixture of terminals and non-terminals for the header/parenthesis/commas and arguments respectively. For example:

  $$\text{inform(PROP)} \rightarrow \text{inform} \quad (\quad \text{PROP} \quad)$$

  where the left-hand-side is a single terminal, `PROP` is a non-terminal and all other right-hand-side symbols are terminals. For predicates with no arguments, we distinguish between the non-terminal representing the type, and the terminal representing an instantiation of the type. Thus:

  $$\text{informYes} \rightarrow \text{informYes}$$

  This may seem odd, but provides consistency given for example we would consider `inform(nameIs(user,'Bob'))` to be an instantiation of `inform(PROP)`. Even with no arguments, the distinction between expected types and their instantiations is still made.

- **Lists**: Lists can be arbitrary long (i.e. `[item1, item2, ... ]`, but can be defined right-recursively. And appropriate definition is:

  $$\begin{aligned}
  \text{LIST}\langle\text{T}\rangle &\rightarrow [\quad] \\
  \text{LIST}\langle\text{T}\rangle &\rightarrow [\quad \text{LIST}\langle\text{T}\rangle' \quad] \\
  \text{LIST}\langle\text{T}\rangle' &\rightarrow \text{T} \\
  \text{LIST}\langle\text{T}\rangle' &\rightarrow \text{T} \quad , \quad \text{LIST}\langle\text{T}\rangle'
  \end{aligned}$$

A list however is parameterised by the type of its elements; non-terminals do not allow such parameterisation. If we were to generate concrete rules for all possible non-terminals T representing types, there would be an infinite number (consider for example a list of lists, and so on), leading to an infinite number of rules. To resolve this, we scan the *HURDLE* type hierarchy for all uses of lists and their parameterised types, subsequently generating rules for each possible instantiation of T.

- **Records**: Records consist of some identifying name, followed by a number of field values, some of which may be optional, for example
`LOCEXPR{class:STR, relations:LIST⟨PREP⟩}`. If F_m is the set of mandatory fields and F_o the set of optional fields, then we generate a rule for each possible set of fields in the instantiation:

$$f \in \{F_m \cup f_o | f_o \in \mathcal{P}(F_o)\}$$

i.e. using the power-set of F_o. This allows any selection of optional fields to be specified as an instantiation of this record type. For each, we generate a rule in a similar manner to predicate types, with terminals for symbols such as curly braces, the structure name and field names, and non-terminals for the field values. For example:

$$\begin{aligned}
\text{LOCEXPR} &\rightarrow \text{LOCEXPR } \{ \text{ FIELDS-LOCEXPR } \} \\
\text{FIELDS-LOCEXPR} &\rightarrow \text{FIELD-CLASS} \\
\text{FIELDS-LOCEXPR} &\rightarrow \text{FIELD-RELATIONS} \\
\text{FIELDS-LOCEPXR} &\rightarrow \text{FIELD-CLASS , FIELD-RELATIONS} \\
&\ldots \\
\text{FIELD-RELATIONS} &\rightarrow \text{relations : LIST}\langle\text{RELATION}\rangle \\
&\ldots
\end{aligned}$$

- **Primitive Types**: The primitive types `INT`, `REAL` and `STR` all hold arbitrary integer/real/string values. We can use the *filters* introduced from the previous section to constrain their value, e.g.:

$$\begin{aligned}
\text{STR} &\rightarrow \text{*str} \\
\text{INT} &\rightarrow \text{*int}
\end{aligned}$$

5.7 Accounting for Non-Isomorphism

We previously saw that non-isomorphism related to the difference in structure between source and target languages. SCFGs were said to be isomorphic if the right-hand-side of each production rule pair consisted either of a single terminal, or two non-terminals which retained their order or swapped. By permitting SCFGs with higher rank, we can accommodate any arbitrary non-isomorphic grammar. However, compact rules involving

smaller numbers of non-terminals/terminals are preferable. In some cases where long-range dependencies are exhibited, as in Figure 5.2, rules of high rank may be unavoidable, given the lack of natural decomposition into constituents in source and target sentences that can be aligned. Other difficult alignments may sometimes be accommodated by exploiting the fact that some data is *additive*. In logical expressions, conjunctions yield an expression that combines the two sources of information. In SCFG approaches such as [Wong and Mooney, 2007], a conjunction $X \wedge Y$ could be represented explicitly using a rule $CONJ \rightarrow X\,Y$, and permitting higher level constructions to accept these conjunctions as arguments. Consider the following example:

"What is that nearby restaurant I can see in front of me?"

```
requestInfo(?$x.identity(
    LOCEXPR{
        class:'restaurant',
         isVisible:true,
        relations:[Near(CURRENTLOC), InFrontOf(CURRENTLOC) ]
     }
, $ x ) )
```

We can see from the resulting structure that we would be unable to parse *that nearby restaurant* independently since its spatial information must be combined with the *in front of* relation at the end of the sentence. We can achieve sufficiently compact rules by observing that the locative expression embodied by the **LOCEXPR{..}** structure is simply an amalgamation of four different properties regarding the same object. In *HURDLE* expressions, the append operator can be + used on most constructions, such as lists, sets and records. For records, the effect is to combine the two sets of fields, recursively applying the append operator to fields that occur in both records. Thus appending two records **LOCEXPR{relations:**[r_1]**}** + **LOCEXPR{relations:**[r_2]**}** yields a new record **LOCEXPR{relations:**[r_1,r_2]**}** after evaluation. Consequently, the above example can be accommodated by two simple SCFG rules:

```
Q → ⟨ what is [1](WHATQ) ,
    requestInfo ( ? $ x . [1](WHATQ) )  ⟩
WHATQ → ⟨ [1](L) i can see [2](P) ,
     identityOf ( [1](L)
           + [2](P) + LOCEXPR { isVisible : true } , $ x )  ⟩
```

where **L** represents a full locative expression and **P** represents a spatial prepositional phrase. The two constituents embodied by **L** and **P** can then be subsequently parsed independently, with their information combined to yield the required structure. The + operator therefore functions similarly to the effect of [Wong and Mooney, 2007]'s *CONJ* non-terminal, but with the added benefit of combining multiple structures more flexibly.

Introducing the *append* operator required a further parsing stage of evaluating the expression in *HURDLE*, reducing it to a simpler form absent of such an operator. We therefore define a string as being valid according to a grammar if its reduced form is valid according to the same grammar. In addition, we can define two grammars as equivalent if the set of all reduced forms of strings generated by each grammar are the same. This

allows us to define a SCFG where the target rule of each synchronous rule may use the operator as a terminal symbol, even though it may not be contained in another CFG embodying the target language, say that generated by our approach in Section 5.6.

In summary, we can encourage simpler, reusable rules that yield a greater degree of isomorphism by introducing an operator which exhibits similar properties to conjunction, combining multiple data structures of the same type into one. By exploiting the commutativity/associativity of this operator, we can semantically parse separate segments of the source sentence and combine their parse, without requiring longer highly context-dependent rules that match larger segments of the sentence.

5.8 Examples/Evaluation

Evaluating the parser on a quantitative level would be vacuous; the SCFG rule set is not learned and thus can arbitrarily be extended to yield high accuracy; were it learnt we could then use approaches such as cross-validation and the use of held-out data. The parser does not present fundamental new parsing theory, but leverages existing parsing algorithm and data structures that accommodates the aims presented at the start of the chapter. We have already seen how appropriate rules can accommodate both dialogue segmentation and approximate a 'concept spotting' approach by matching maximal parsable portions of the text. We also presented a more flexible means of specifying SCFG rules to generify stipulations on terminal symbols (e.g. identifying any noun or converting a word in the source sentence to its singular form).

However, we can identify the qualitative strengths and weaknesses of this approach given a suitable existing SCFG rule set and examples again from the *TownInfo* corpus.

- **The lack of probabilistic information does not handle ambiguous segmentations.** Consider the example "Go to the nearest bus stop". This yields two parses that both utilise the entire string:

  [requestAction(go(LOCEXPR{
  class:'bus stop',
  relations:[NearOrdinal(CURRENTLOC,1)] }))]

  [requestAction(go(LOCEXPR{
  class:'bus',
  relations:[NearOrdinal(CURRENTLOC,1)] })),
  requestAction(stop)]

  The first parse arises of the SCFG rule that identifies compound nouns as designating the class of the object (the bus stop). The second arises when the rules consider a segmentation into two utterances, the latter of which involves a command for the robot to stop. Even if we were to use Probabilistic SCFGs, learning the weights for each rule using training data and the *Inside-Outside Algorithm*, probabilities associated with the rules presented in 5.5 don't effectively accommodate segmentation, since a single probability is used for any split in the sentence. However, there are two simple means by which the problem can be addressed. Firstly, note that

use of probabilistic SCFG would naturally favour shallower trees given the product of less probabilities. In addition to our heuristic that maximised usage of the source string, we can favour parses which minimise the depth of the derivation tree. This reflects the notion the 'simpler' interpretations of the sentence are favourable (known more generally as *Occam's Razor*), and results in the first translation, with a shallower tree in the target language, being chosen. We can additionally avoid some disambiguation by permitting use of the *filters* in Section 5.4 on concrete terminal symbols. Since we have access to Part-Of-Speech tag data of the source sentence, we can modify our stop rule to match only a 'stop' terminal if used as a verb. Since the Stanford Parser in this particular instance tags it as a noun, only the first parse then becomes valid, as desired.

Often ambiguity can occur on an intra-utterance level. For example, there are two possible interpretations of the following utterance:

> "I am looking for a chinese restaurant near the post office in the central part of town"

The ambiguity arises when considering whether the Chinese restaurant or the post office is in the centre of town. But given both parses use the entire source string, the former interpretation is favoured given it results in a shallower structure; the former effectively uses a source grammar rule $NP \rightarrow NP\ PP\ PP$ (i.e. with two non-nested prepositional phrase attachments) whereas the latter uses a rule $NP \rightarrow NP\ PP$ twice. In this particular case, the shallower parse yields the more likely interpretation.

- **The parser in general deals very effectively with multiple utterances with superfluous information**: As previously identified, integrating the utterance segmentation directly into the SCFG rule set favours segmentations which have more parsable content. Although valid derivation trees that arise may be the product of sub-optimal segmentations which still yield some partial information from each utterance, our heuristic of favouring derivations which minimise wildcard terminal usage ensures these are not chosen. The following example illustrates this, along with pruning of irrelevant semantic content:

  > "erm i've been recommended to go to a bar near the fountain as they're very nice can you tell me what music they play"
  > **requestAction(go(LOCEXPR{dA:false , relations:[**
  > **prep(Near,LOCEXPR{dA:true , class:'fountain'})] , class:'bar'}))**
  > **requestInfo(?$x. objProp(@LOCEXPR,(plays,$x)))**

- **It is difficult to combine fragmented information**: In concept spotting portions of the source text were matched and their information combined. This was useful if the source sentence was not well-formed grammatically but individual portions were. However, such an approach usually relies on shallow meaning representations due to the lack of formal mechanism to combine the data, except to consider the overall meaning as the conjunction of the parsed parts. With our approach we emulated this but on an utterance level; where individual but grammatically well-formed utterances could have superfluous information either before,

after or in between them. There are cases in the corpus however where this approach does not help:

> "ok but is there a bar that sells that i can listen to folk music folk music"
> [**acknowledge ,**
> **requestInfoYN(thereExists(LOCEXPR { dA:false , class: 'bar' }**
>))]

The utterance contains a disfluency, where the addressee has corrected their preference mid-utterance of bar choice from what it sells to music. The parser has matched the maximal segment for each utterance (i.e. 'ok' and 'is there a bar') but the disfluency fragments the latter utterance from the parsable adjunct involving music taste. There are two ways in which the problem can be resolved. The first is to pre-process the utterance to remove the disfluency. A second method is to intersperse on the right-hand-side of source grammar rules an additional non-terminal that scans a number of arbitrary terminals. For example:

> L → [1](C) [2](WILDCARDS) [3](P)

where L represents the locative expression, C would match the class phrase as before ('the bar'), P matches the prepositional phrase ('that I can listen...'), and WILDCARDS matches any number of arbitrary terminals, as per Section 5.5. This would generate a huge number of parses (just as when we considered all possible segmentations) and result in a higher processing time, although again derivation trees which maximised use of matched concrete terminals (i.e. non-wildcards) would be preferred, resulting in a correct parse:

> **requestInfoYN(thereExists(LOCEXPR { dA:false , class: 'bar',**
> **attrs:[(plays,'folk music')] }))**

The repeated phrase at the end of the utterance does not present a problem as it occurs outside a parsable dialogue act string segment.

- **It is difficult to acccommodate all ways of expressing the same information**: Capturing grammatical patterns rather the keyword spotting is problematic when the addressee expresses their information in a manner that has not been accounted for. For example, our system yielded this incomplete parse for the following source sentence:

> "hi could you recommend an indian restaurant where you could get a glass of wine with your food"
> [**greet , requestAction(recommend(LOCEXPR{dA:false ,**
> **attrs:[(foodVariety,'indian')] , class:'restaurant'}))]

The parser is unable to capture the information relating to wine provision because there is no pattern to account for it. While a rule could appropriately be added to accommodate for such a pattern, the multitude of ways in which information can be expressed may be difficult to preemptively cater for. A keyword-based system for parsing would have spot the 'recommend', 'indian restaurant' and 'wine'

keywords and combined appropriately, although at the expense of losing structural information.

We could again account for this by appropriate SCFG rules, using the strategy previously described of accounting for fragmented data, in addition to accommodating just a single keyword, e.g. 'wine'. With the lack of more structural information which would match a large portion of the source sentence (and thus give a derivation tree with greater weight), the system would fall back on using these smaller matched fragments. When for example we added the following rules:

L → ⟨ [1](C) [2](WILDCARDS) [3](LFRAG) ,
[1](C) + [3](LFRAG) [2](WILDCARDS) ⟩
LFRAG → ⟨ wine , **LOCEXPR{attrs:[(serves,'wine')]}** ⟩

(where LFRAG conceptually represents some fragmented information about an object while discarding the grammatical context, and C again matches the class of the object inclusive of adjectives), then the correct full parse was then yielded:

[greet , requestAction(recommend(LOCEXPR{dA:false,
attrs:[(foodVariety,'indian'), (serves,'wine')], class: 'restaurant'}))]

Our task was to build a robust, predictable semantic parser that accounted for a number of inherent challenges in our particular parsing task, such as the fullness of semantic representation required, the lack of isomorphic correspondence between the source and MRL grammars, dealing with short answers, the presence of superfluous information in utterances, and the task of dialogue act segmentation in cases where multiple acts were used in a single utterance. From an integration point of view, we wished the resulting strings from the translated input to be valid forms according to the implicit grammar of a dialogue system input, in particular, our *HURDLE* architecture.

Our resulting system was capable of processing complex and long utterances such as that in Figure 5.1, with a rich but highly structured representation that did not discard pertinent information. Our approach used Synchronous Context Free Grammars for the translation process between source natural language and target MRL. We accounted for dialogue segmentation and pruning of unnecessary information within the rules of the SCFG, and by extending the SCFG formalism we were able to represent our rule set much more compactly.

Our approach is advantageous relative to the keyword approach given the richer representation of meaning. However, by combining use of wildcard terminals, appropriate SCFG rules and heuristics for favouring particular derivation trees, we were able to produce a much more flexible robust parser than vanilla SCFGs would permit. We could better accommodate 'fragments' of information that concept spotting approaches could adequately handle, while being able to combine these fragments with segments of the utterance with deeper structure. The result is a parser that combines the advantages of concept spotting and those which give deeper representations but require fully-formed grammatical input.

Chapter 6

System Architecture and Platform Integration

6.1 Architectural Overview

We have so far focused on each of the components individually. However, a considerable degree of infrastructure is required in order that such components operate cohesively on a robotic platform. This architecture is illustrated in Figure 6.1. A summary of all components and the flow of data between them is as follows:

- A mobile device (pictured as an *iPad*) is used to input some utterance into the dialogue system. Via a web server that processes this data input, the utterance is put in a central database, *MOOS*, which we describe in the next section.

- The semantic parser listens to new dialogue input data in *MOOS*, and uses the process in Chapter 5 to convert it into the required form.

- This is then passed to the Dialogue Manager, *HURDLE*. When some dialogue rule requires use of a spatial model, the associated method builds a *job request*. Such job requests are processed by a module which manages these requests, sending them to a *robot intermediary* module, and handling replies (robustly dealing with replies that may be received in the wrong order, or requests that timeout). This job manager must send requests asychronously, since the same communication channel with the robot intermediary is used for receiving other alerts from the robot, such as notifications of arrival at a location. The appropriate concurrent code is used so that despite the asynchronous job handling, requests from the dialogue system appear synchronously as a simple method call with return value. This is achieved by thread blocking.

- The *Robot Intermediary* is a module that provides the dialogue system access to lower-level robot operations and to the various spatial algorithms/models. This module can operate on a separate system to the dialogue system; messages from the job handler can be sent over a network in XML form. Hypothetically, multiple dialogue systems could interact with the same robot intermediary, although this is not required on the EUROPA platform. The Robot Intermediary has direct

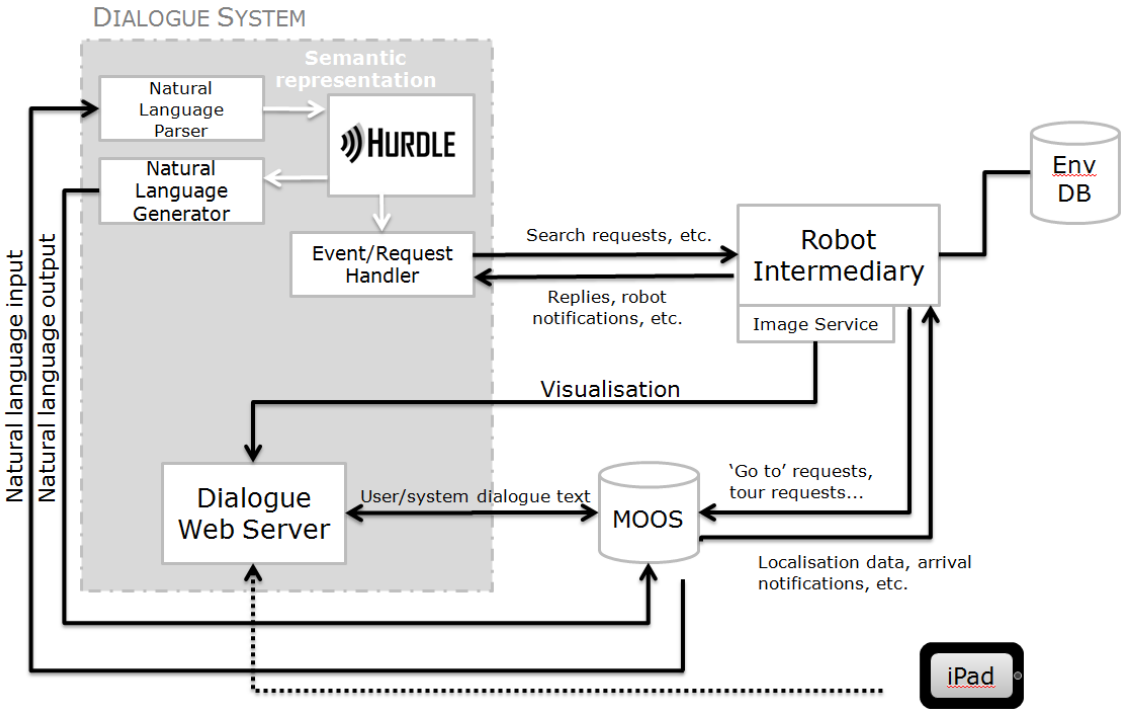


Figure 6.1: An architectural overview of all linguistic components on the robotic platform.

access to the database of spatial/environment data, used for job requests involving the interpretation, validation or generation of spatial descriptions. For requests involving a 'go to' request, the appropriate message is sent to the MOOS database to be picked up by the path planning module.

- Back in the dialogue system, any outputted utterances are fed through the natural language generator module, and passed to MOOS. The dialogue web service listens to dialogue outputs on this database, so that the mobile device can receive this response.

6.2 Interacting with External Components

MOOS[1] is a star-based topology which allows distribution of high volumes of data between different robotic modules on a single platform. It acts as a central data repository, with publishers and subscribers. Data is stored as attribute-value pairs; for example a 'GPS' attribute may store the current coordinate. Subscribers can register their interest in particular attributes, such that they are alerted when its value is changed, either receiving all changes to the value, or stating the frequency at which these updates are received.

The use of MOOS was prevalent in the previous section since multiple modules can subscribe to given data. For example, while the dialogue of the output system could be sent directly to the web server, publishing it instead to MOOS allows other modules to

[1]http://www.robots.ox.ac.uk/ mobile/MOOS/wiki/pmwiki.php/Main/HomePage

access this data (in this particular case, the *on screen* interface which also maintains an up-to-date dialogue history). A summary of all communication with external modules via MOOS is as follows:

6.3 Simulator

Given the linguistic components of the robot are numerous in number, they require rigorous testing in order to ensure they work cohesively together. Requiring the robot to be in active operation during these testing periods would be cumbersome. We therefore developed a simulator which could emulate the particular operations of the robot pertinent to the linguistic components, notably, simulating the path traversal when moving to a given destination, and giving a view of the robot's environment relative to its current pose.

The former is achieved by the appropriate use of the $A*$ algorithm, using the straight line distance as its appropriate heuristic. To generate a graph network for the environment that represents the traversable areas, we randomly place nodes across the environment; more specifically, we start with a lattice of nodes and add some random offset up to half a grid width. For each node, we consider whether each neighbouring node is obstructed by some obstacle. If not, we add an edge, since it is possible for the robot to travel from one node to the other. To accommodate assigned *entrances* of objects, we take the closest node in the graph both inside the object and outside the object, and add an additional edge. This effect can be seen in Figure 6.3. The robot moves by moving towards the next node on the optimal path, and switching goals to the next node when it is sufficiently close. The path planning is not meant to provide a full reflection of the actual robot path planning module, since for example our generated graph will happily include any flat regions which might not be traversable (e.g. grass). The pose of the robot can also be altered manually by right-clicking, and arbitrary points used as new target destinations by double clicking.

When a specific object is selected as a new destination, both the simulator and real robot must select an appropriate goal point to travel to. If there exists a specified *entrance* part of the object, we use the coordinate of the nearest entrance to the observer. Otherwise, we determine the nearest point on the surface of the object, and select a point a few metres away from this in the direction of the observer. Whenever the destination is represented as a point (e.g. *the end of the road*), we can use this point directly.

The view is used in a variety of other contexts. It for example shows an actively running robot's view of the world from the perspective of our data representation in Section 2.3. The developer can zoom in and out and pan the environment, and the position of the mouse cursor yields the corresponding coordinate in the robot's coordinate frame (using the appropriate coordinate translations). The view can also be outputted in image format, which is used by the Web Server in Section 6.5. The view is also capable of overlaying laser data for easier annotation of an environment. The robot laser module can output a directory containing files which split the map into regions, with a png image of laser points and a data file indicating the bounds of the region for each. The view can read this directory, merge the pngs into one and display it as an overlay.

There are a number of additional subsidiary tools. One allows objects to be rapidly added to the database by simply drawing it onto the environment, and selecting the

MOOS Variable	Description	Publishers	Subscribers
D_INPUT	The natural language input to the dialogue system.	Dialogue web server and robot's onscreen interface.	Semantic parser.
D_OUTPUT	The natural language output from the dialogue system.	Natural language generator.	Dialogue web server and robot's onscreen interface.
LOCALIZATION	The localiser module which determines the position (and orientation) of the robot.	The Localisation Module.	The Robot Intermediary.
START_TOUR	There is a tour module (independent from the systems presented in this thesis) which takes the user on prescripted tours of the city/town. The dialogue system must be put into hibernation while the tour is in action in order to avoid interference.	Tour Module.	Robot Intermediary.
STOP_TOUR	Used to indicate the tour has stopped. A separate variable is required from above because values persist in MOOS even when the Tour Module is not activated. By determining which of the values of START_TOUR and STOP_TOUR have the most recent timestamp, we can determine whether a tour is currently in action.	Tour Module.	Robot Intermediary.
PLANNER_TARGET	The current target coordinate for the path planning module, if any.	Robot Intermediary, Tour Module, Planner Debugging Tool, iPhone 'Find Me' Module	Robot Intermediary, Local Planner
PLANNER_STATUS	The response from the last planner target request. Either indicates its successful acknowledge, arrival at the location or an error code indicating the reason a goal was rejected.	Global Planner	All above publishers.

Table 6.1: The variety of different data on MOOS that the Robot Intermediary and Dialogue Web Server either publishes or subscribes to.

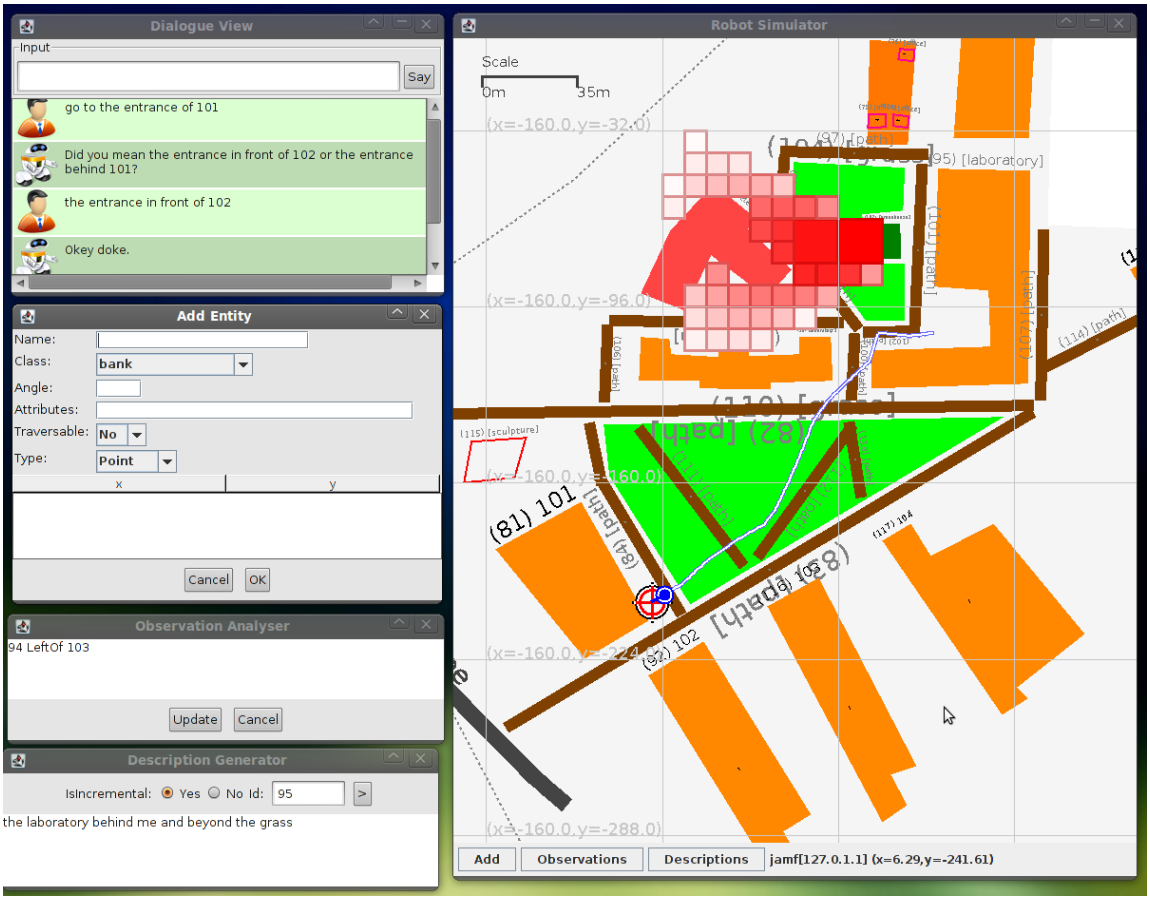


Figure 6.2: The simulator for the robot, allowing the linguistic components to be tested without need for a robot.

appropriate semantic properties on the accompanying dialog window. Another tool allows testing of prepositional models, allowing any preposition, referent and landmark objects to be specified (by their identifiers). The specified referent is only used if its semantic properties influence the probability associated with the overall relation. The relation is displayed on the view by generating the quad tree as per Section 3.3. Lastly, another tool allows generation of locative descriptions for a particular object (saving the need to generate it by means of discourse). All these tools are shown in Figure 6.2.

6.4 HURDLE Interpreter

We have previously presented our framework and programming language *HURDLE* for dialogue management and semantic parsing. We now briefly present some of the technical details in producing the interpreter able to read the *HURDLE* syntax.

Our parsing process requires four separate passes over the code. The first pass reads just the names of declared types (i.e. *enum* and *structure* types). By parsing these first, we avoid cyclicity problems where types might refer to each other, or types refer to another type not declared until later in the code. We next parse their full type information (e.g.

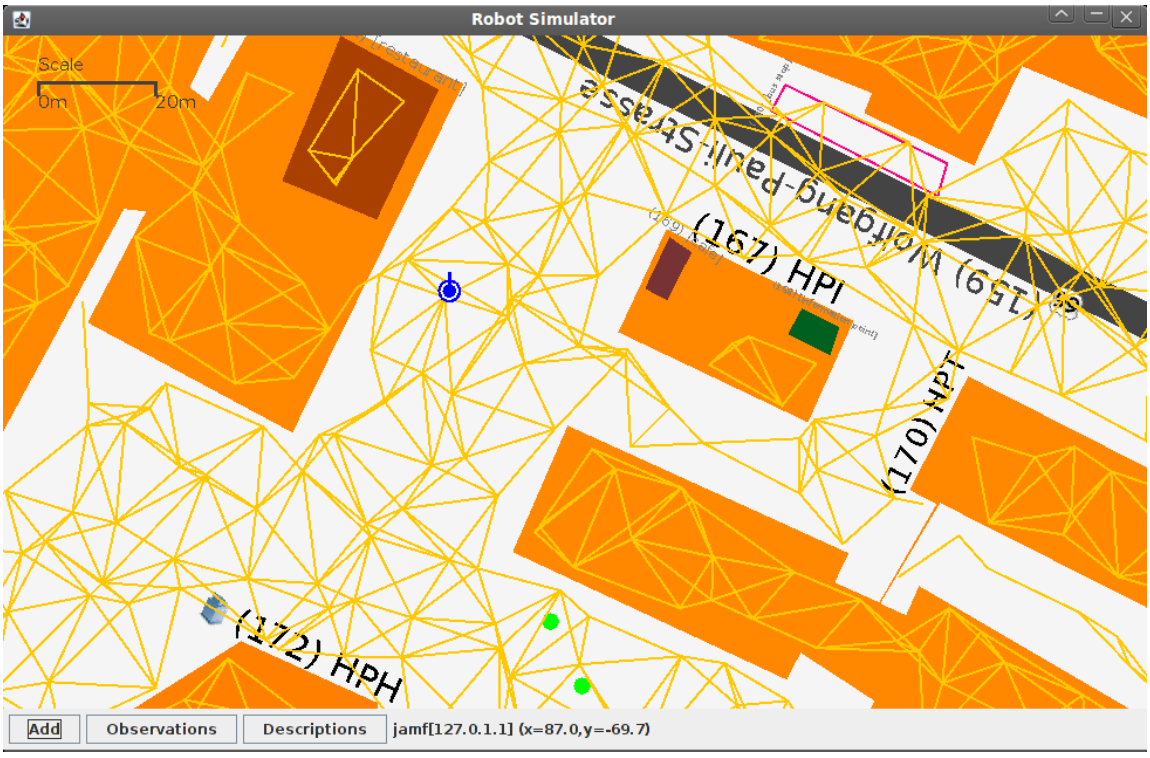


Figure 6.3: A randomly generated graph showing the traversable region of the robot, and allowing routes to be found via A* when the physical robot is offline. Note that due to a designated entrance for the leftmost object, an extra edge is added to the graph to ensure the interior of the building is accessible.

for the *enum* type, we parse its child types, and for *structure* types we parse the fields). This must be done before parsing constants and global variables, since these might refer to specific fields of a structure type for example. Thirdly, we parse for these constant and global variable declarations, since they may be used in rules before they are declared. Lastly, we have a fourth pass to parse the dialogue rules.

We use a custom-made tokeniser, which allows particular strings of characters to be treated as one token, for example `->`. Parsing usually takes place by parsing one token at a time, using a *Pushdown Automata* in order to incorporate syntax that can be represented as a Context Free Grammar. For convenience, we implemented a custom token stream which allows the parser to jump back tokens. For example in the construction:

```
[ 2*$x | $x <- [1..100]]
```

which gives the list of even numbers between 1 and 200. At the point at which the `$x` is first seen, we don't yet know its type, since it is determined by observing the expression after the `<-`, seeing it is of type `LIST<INT>` and only then inferring that `$x` is of type `INT`. Not having access to the type at this earlier point would be particularly problematic for example if the variable used in the enumeration was for example a structure type, since we would not be able to determine for example whether any access to its fields is according to the correct structure type. We therefore do a parse of the expression before the `|` without any type checks, then jump back after parsing the `[1..100]` to check `$x` has been used correctly in the expression given its inferred type.

When parsing local variables, we determine whether they have been previously seen by seeing whether they are contained in the current type environment. If so, the resulting atom is an instantiated variable with whatever type was previously inferred. If not, the resulting atom is an uninstantiated variable with a yet undetermined type. The parser can infer this type in a number of ways. If for example the expression was `U = inform($p)`, then the type of `$p` can be determined by the type within the declared `inform` construct. In equality, we resolve the types on either side of the equals. For the expression `4 = $x`, the left-hand side has type `INT`, while the right has indeterminate type. The result is that the type of the RHS is set to `INT`, which in turn allocates the variable `$x` this type in the type environment. This kind of resolution allows more flexible function arguments. We could for example implement a function `second` which returns the second argument of a pair:

```
function B second(PAIR<A,B> ($a,$b)) {  return $b  };
```

Since `($a,$b)` has the type `PAIR<A,B>`, we can infer each of the variables `$a` and `$b` have the generic types `A` and `B` respectively.

The type environment at a given point in the parse consists of two things: the types of any variables within scope, and any instantiations of generic types. The latter is used to infer the concrete type of a function call involving generic types. For example, with the expression `second((4,'foo'))`, we can determine after parsing the expression that the overall type is `STR`, since in the type environment, generic type `A` corresponds to the concrete type `INT`, `B` to `STR`, and the return type is `B`, resolving to the type `STR`.

Variable scope is implemented by allowing a type environment to be wrapped within another. For example in the code:

```
$x = 3,
for($y <- [1, 2, 3]) {
  ...
}
...
```

the variable `$y` only persists until the end of the for loop, whereas `$x` persists till the enclosing scope. By creating a new type environment within the for construction which references the previous environment containing `$x`, the latter is unaffected by new variable declarations. When the interpreter has finished parsing the for construction, the new environment can be discarded, thus also dropping all new variable types without losing the older ones.

6.5 Web Server

By developing a web server, it is possible to interface with the robot remotely, for example using a mobile or tablet device. We used *Google Web Toolkit*[2] to design the interface, in order to accommodate its dynamic elements, e.g. updating the conversation history, so that the page need not be refreshed in the web browser. The advantage of using such a framework is that the coding of tedious AJAX/JavaScript code for the dynamic elements of the interface behaviour, and ensuring cross-browser compatibility, is not necessary; instead GWT generates this code automatically by use of a Java API. Since the interface requires access to server side code, it is deployed as a *war* (Web Archive) file on a web server such as *Apache Tomcat*.

The interface consists of three tabs. The first shows the dialogue history and allows new natural language input from the user. The second shows the semantic parse of natural language test as per Chapter 5. The last shows the view of the robot as per Section 6.3, updated every few seconds. This requires a separate image server which yields *png* images according to the specified view bounds in the URL parameters. It is possible to switch between views which are centred on the observer/robot, and centred on the destination. The interface is pictured in Figure 6.4.

6.6 System Testing and Evaluation

For some of the components that constitute the overall linguistic system, we saw that evaluation was carried out or user data was collected to determine the parameters of our models. In Section 2.7.1, we conducted an online experiment to assess how humans assess various spatial prepositions. In Section 3.4, we evaluated the performance of our algorithm to generate locative expressions. And in Section 5.8, we provided some discussion of the extent to which the parser could accommodate more complex sample dialogue in the *TownInfo* corpus.

Clearly, on an integrated platform where these various components interact, as well as with non-linguistic components on the platform such as path planning, localisation and the general robot hardware, the whole must be at least the sum of its parts. In this section,

[2]http://code.google.com/webtoolkit

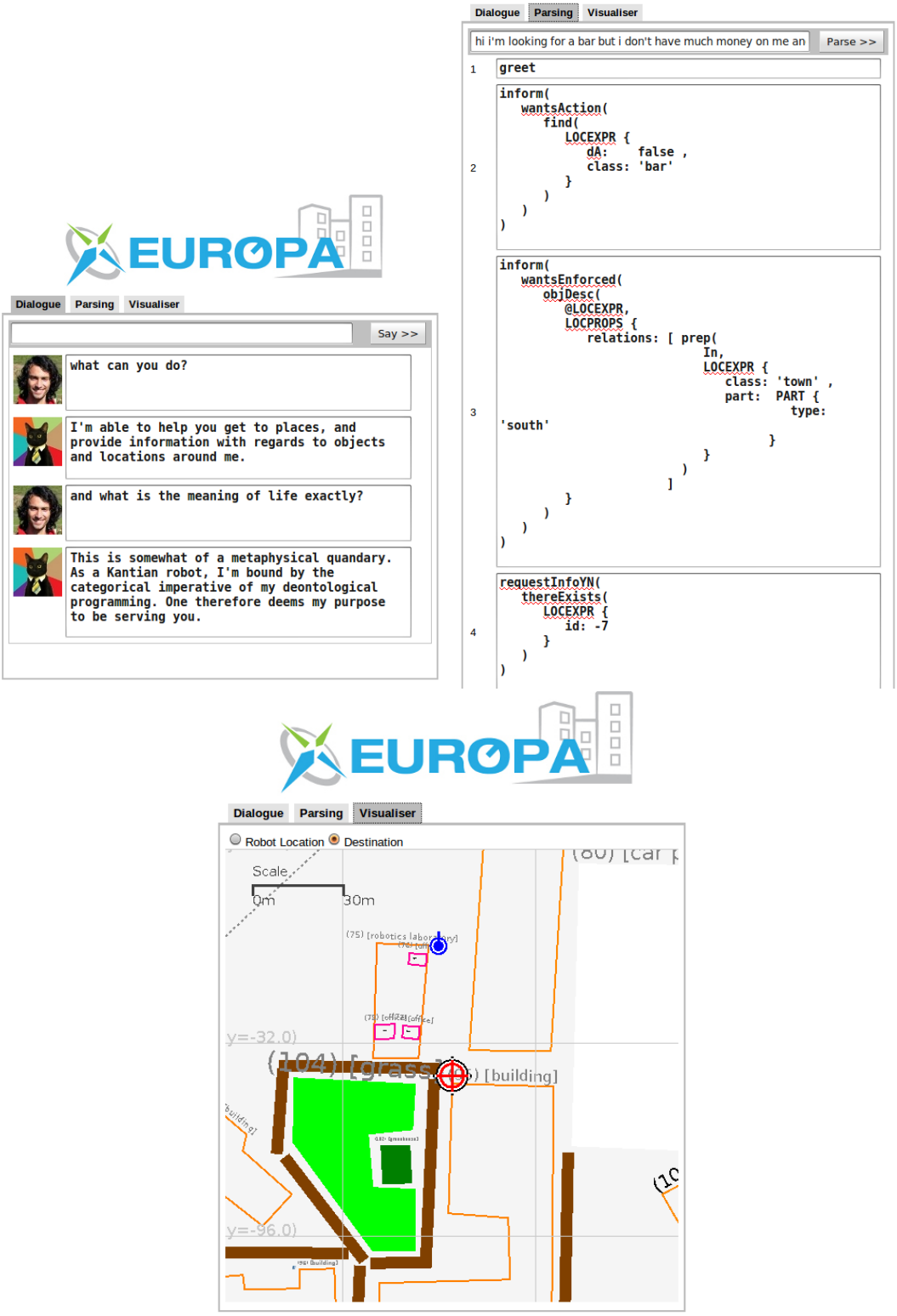


Figure 6.4: The web interface.

we provide evaluation of the system as a whole, including the specific implementation in our HURDLE dialogue framework for spatial dialogue. We firstly provide details of some part-prescripted tests aimed to showcase the capabilities of the system for the purposes of videos and project reviews. Secondly, we conducted experiments in which human participants with no prior familiarity with the system were presented with case scenarios, in addition to permitting more 'freestyle' interaction with the system.

6.6.1 Prescripted Demos

Three copies of the robot platform were located in Begbroke Science Park close to the city of Oxford, the science campus of ALU in Freiburg Germany, and the Honggerberg campus of ETH Zurich. For each site laser data was obtained and an annotated map produced as per the tools developed in Section 6.3. Interaction with the robot was via a tablet device with access to the web interface of Section 6.5. For the first two of these sites, demonstrations were produced for the purposes of videos and a project review, involving prescripted text from the user. In particular, the Freiburg transcript is from a live demo as part of the final EUROPA project review, prior to which the robot had undergone several hours of testing with the dialogue system in operation. The project received the highest 'excellent' rating.

The maps and route of the robot are pictured in Figure 6.5, and the conversations in each provided below. A video still from the Begbroke demo can be found in Figure 1.1.

6.6.1.1 Freiburg Demo

U: hi there i'm late for a meeting and need to find Cyrill's office, what building is it in?
S: Once again, nice to meet you. The building that Cyrill's office is in is the robotics laboratory. Cyrill's office is 51 metres away from you. Would you like to go to Cyrill's office or would you like me to tell you where Cyrill's office is?
U: go there please
S: Okey doke.
[Robot travels to Destination 1]
S: I've arrived at Cyrill's office. Is there anything else I can help you with?
U: now i need to lock my bike
S: There are 8 bike racks.
U: I meant the bike rack closest to the train track
. S: Okey doke.
[Robot travels to Destination 2]
S: I've arrived at the bike rack. Is there anything else I can help you with?
U: tell me about the sculpture to your left
S: I often see children playing here in the evening. Clearly they have no respect for art!
U: and can i see the car park from here?
S: No. The car park is not visible. The car park is obstructed by the building behind me and left of 51. U: finally, I have a lecture to catch so take me to building 101.

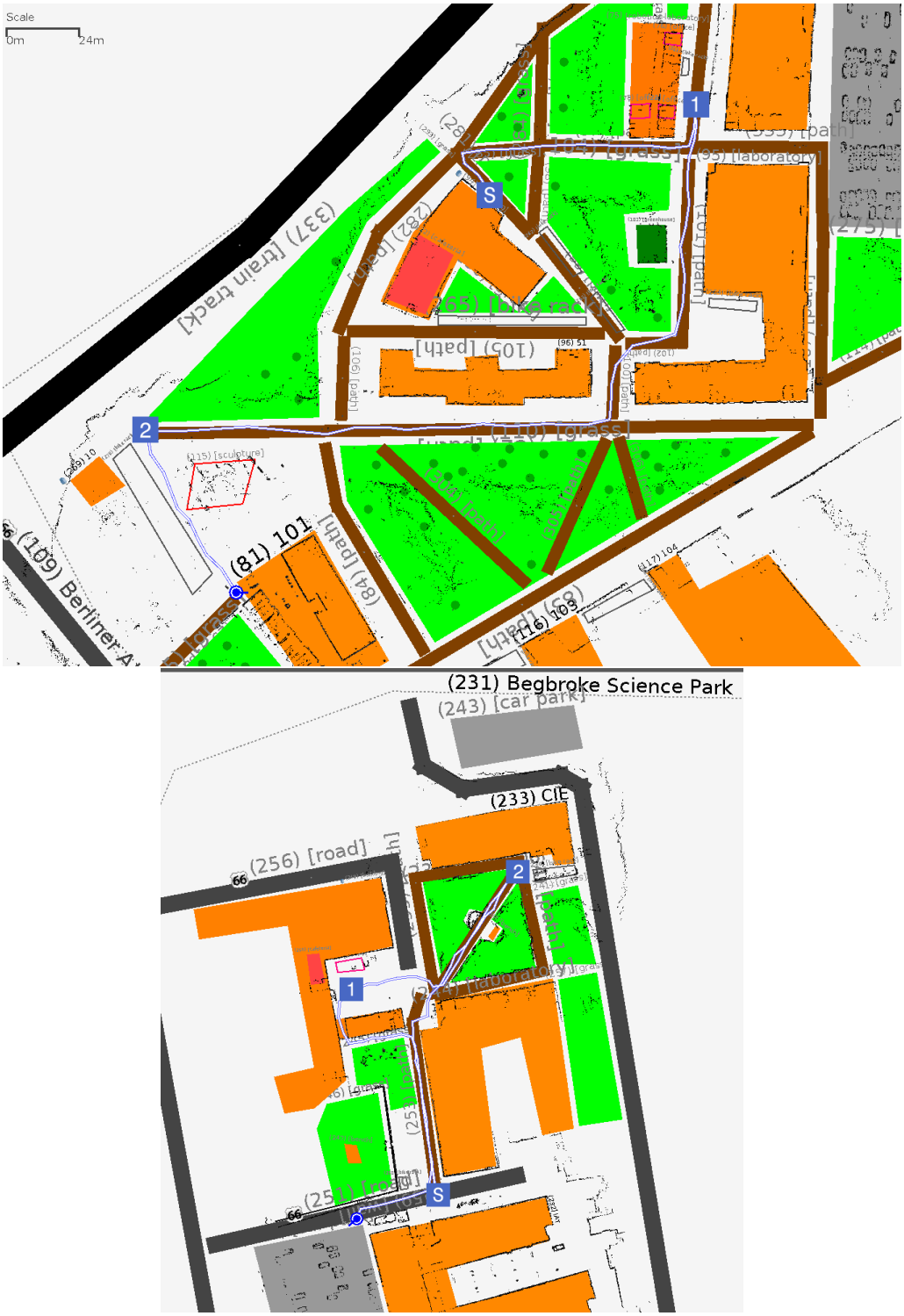


Figure 6.5: The two environments used for our demos, the science campus of ALU in Freiburg (top), and Begbroke Science Park of Oxford University (bottom). The white line with blue outline indicates the path the robot took in each of the scenarios, with the numbers indicating the intermediate stopping points (and S the start point).

S: Did you mean the entrance behind 101 or the entrance left of me?
U: the latter
S: Okey doke.
[Robot travels to final destination]
U: I've arrived at the entrance of 101 in front of me. Is there anything else I can help you with?
S: no
U: It was jolly nice to meet you. Goodbye.

6.6.1.2 Begbroke Demo

U: hi there what can you tell me about the campus i'm on
S: Nice to meet you. Did you mean Begbroke Science Park?
U: yes
S: Begbroke Science Park is home to research groups from 6 academic departments within the Mathematics, Physics and Life Sciences division of the University of Oxford...
U: i'm hungry so i want to find somewhere that sells sandwiches
S: The nearest place that sells sandwiches is 88 metres away from you. Would you like to go there or would you like me to tell you where it is?
U: tell me where it is
S: It is in the nearby building in front of me and behind the wall.
U: let's go there
S: Okey doke.
[Robot travels to Destination 1]
U: *[Enroute to destination]* how far is it
S: It is 42 metres away from you.
S: I've arrived at the cafeteria. Is there anything else I can help you with?
U: now i want to go to the bike rack
S: And did you mean the bike rack in front of CIE or bike rack right of the laboratory right of me?
U: the bike rack in front of cie
S: Okey doke. *[Robot travels to Destination 2]*
I've arrived at the bike rack left of me. Is there anthing else I can help you with?
U: what do you know about the building left of me
S: Did you mean CIE?
U: yes
S: The Centre for Innovation and Enterprise, a fully serviced and dedicated business incubator, was opened in the summer of 2006.
U: ok i'm done for the day take me to the car park by iat
S: Did you mean the car park...
U: yes
S: Okey doke. *[Robot travels to final destination]*
U: *[Enroute to destination]* what is in front of me
S: There's IAT and the bike rack.

Performance Metric	Value	
Utterances correctly parsed	83%	
Appropriate system responses	82%	
Avg delay before system reply	0.73s	(Response involving spatial language generation)
	2.16s	(Analysis involving spatial language interpretation)
	0.01s	(Response involving neither)
Number of user utterances	41	

Table 6.2: Dialogue performance metrics for the tests with participants not previously familiar with the system.

> S: I've arrived at the car park in front of me. Is there anything else I can help you with?
> U: no

6.6.2 Evaluation

Given that the robot is intended for use a pedestrian assistant, we devised a test using participants that had had no prior interaction with/knowledge of the robot. The 2 participants[3] were taken to the Begbroke Science Park site, and presented with a number of case scenarios, such as finding somewhere to sit, getting something to eat or locating a particular building. As before, their interaction was via a tablet device connected wirelessly to the robot. Beyond brief written guidance on the capabilities of the robot, they were free to carry out their dialogue related to their provided scenarios in any way they saw fit, and ask additional questions to the system not necessarily directly encompassed by the case scenario. They were also asked to write in a 'natural' manner, neither simplifying their language nor purposely obfuscating their language, in order to emulate spoken conversation.

Quantitatively evaluating the performance of a dialogue system is an inherently difficult task, partly due to the subjective nature of what merits a 'successful' dialogue and user satisfaction. There has been a small amount of research into frameworks to formulate and utilise appropriate metrics. The most prominent of these, the *PARADISE* framework [Walker et al., 1997], uses a weighted combination of *task success* measures, and *dialogue costs*, representing penalities associated with task-independent features such as efficiency (e.g. number of utterances, time delay before the system's response) and quality (e.g. the ratio of appropriate to inappropriate utterances). Task success is measured by the user achieving the information they require. This uses an 'Attribute-Value Matrix' of attribute-value information that must be exchanged between the agent and the user during the dialogue, and compared against the matrix for the current dialogue state.

There are a number of problems encountered when using such a method for evaluation. Firstly, as identified by [Beringer et al., 2002], the metrics assume a system without multimodal input and output. In the case of our robotic platform, this includes modelling whether the robot could adequately find a route to a destination via its local and global planners, and problems encountered with the environment, e.g. obstacles, travelling into

[3]Ideally the number of participants would have been much greater.

nontraversable terrain, etc. These can be incorporated into the quality measures of the PARADISE framework; [Beringer et al., 2002] for example uses metrics for the quality of gesture or face recognition for its particular multimodal discourse system 'SmartKom'. Multimodal input also results in Attribute-Value Matrix being an insufficient model to represent task completion, since a dialogue may not just involve the exchange of information, but superordinate goals such as successfully escorting the user to their desired destination. These however can be handled by appropriately modelling these concepts as units of information (for example *arrivedAtObj42 = true*).

Secondly, [Hajdinjak and Mihelic, 2006] and others have observed that certain factors, such as speech recognition accuracy, can be the dominant predictor of a dialogue system's performance. Given that in the PARADISE framework the overall user satisfaction score is modelled just as the sum of individual metrics, and that poor speech recognition can clearly depreciate other metrics such as 'appropriate to inappropriate utterance ratio', the result is that the score does not provide a fair portrayal of the dialogue system's performance. While our system did not use speech recognition, there is a similar problem that mobility issues or other issues with non-linguistic components on the robot platform would likely negatively reflect on users' perception of the dialogue system. While the problem could be circumvented by employing the simulator discussed in Section 6.3 to avoid use of the robot entirely, this would not adequately reflect on how the dialogue system operates in collaboration with other modules, nor would the user be immersed in a real environment (particularly detrimental given that the dialogue system largely concerns the interpretation and generation of spatial language based on an environment).

Lastly, the evaluation of the dialogue system is bound to a particular environment, making it difficult for a meaningful comparison to be made between quantitative measures of performance across sites. For a simple environment with a small number of objects, and with a simple task to, say, go to one of these objects, the resulting dialogue will likely be only a couple of utterances long without need for ambiguity resolution. For complex environments, establishing the user's intended location will result in lengthier dialogue, increasing the 'number of utterances' penalty. The dialogue quality metrics become less universally applicable when we consider for example that our dialogue is mixed-initiative, with the user being able to instigate subconversations, thus again undesirably inflating the 'number of utterances' penalty.

Despite these drawbacks, many of the metrics do offer good indicators of effective dialogue performance. We collected participant responses subsequent to testing, in addition to recorded system data, to provide an assessment of the system's performance. The data for the most pertinent performance metrics is presented in Table 6.2. For the metric of system response time, we calculated the average independently for different types of responses, in particular those which involved generation of spatial language in the response, those which involved interpretation of spatial language in the user's utterance, and those which involved neither (e.g. a request to go to an already grounded location, or a response to a greeting). We present our more qualitative findings below:

- **Spatial Descriptions:** Descriptions generated were usually clear to the user based on their/the robot's current position, given that object's used in the description were always visible as described in Section 3.2.2, and due to the modification of the GRE algorithm in Section 3.2.3 to ensure that landmark objects were uniquely identifiable without the need to consider their relationship with the referent. One

case where the description was deemed unnatural was due to a problem identified in Section 3.4, where the user's perceived axis for different spatial prepositions within the same description had to be shifted. When for example the user was looking for the closest bike rack, the robot's response was to offer an option: 'There's the bike rack 68 metres from me and just right of the laboratory right of me and near the building in front of me. Is that OK?'. In addition to being a slightly complex description, the user is required to identify 'the laboratory right of me' based on the current axis of the robot, then interpret 'right of the laboratory right of me' based then on a shifted axis where the robot/user is now facing this landmark.

- **Parser:** Participants avoided giving the kind of contextual information found for example in the *TownInfo* corpus or the demos in the previous section, and instead were more direct in their questions, requests and responses, perhaps because of assumptions made about the capabilities of the system, or because of the time taken to type on the tablet device given the absence of speech recognition. Most problems were associated with participants expressing a request/question in a way not expected by the parser; easily rectified by adding appropriate SCFG rules. For example, certain associations are made between user requests and the objects they wish to find, such as 'somewhere to sit' representing a desire to find a bench; these associations need to be explicitly stated. In this particular case 'where can I sit' was not recognised because there was a SCFG rule for this association appearing in a request, e.g. 'I want to sit', but not as a question. A more significant problem was an ambiguous parse for the first utterance of the following subdialogue:

  > User: "go to the bench behind the lab in front of you"
  > System: "There are 4 labs."
  > User: "i was referring to the closest laboratory"
  > System: "Sorry but I don't know where a bench behind that lab and in front of me is"

  Rather than the most natural interpretation of 'in front of you' applying to the lab, it was in instead interpreted as 'the bench that is both behind the lab and in front of you'. The result was to make the lab more ambiguous in the system's search algorithm. When the user disambiguated to the correct lab, this was spliced into the original request. However, because *'in front of you'* was incorrectly applying to the bench, the relation did not hold due to being obscured by the lab. The reason for multiple prepositional phrases being permitted to modify a single noun phrase, rather than embedded within each other, was due to this being a occasionally found in the *TownInfo* corpus, for example *'i'm looking to find a restaurant somewhere in the east somewhere near the castle'* where the conjunction ('and') is elliptically omitted. To resolve the problem, either the respective SCFG rule would have to be removed so that only the nested parse is obtained, or some additionally heuristics or probabilistic information would be required to distinguish between the two parses.

- **Dialogue Control Flow:** The system in most circumstances was able to correctly handle dialogue behaviour for object disambiguation, correction, requests for confirmations and responding to directives and questions. One problem encountered was when the user was asked to disambiguate between objects:

System: "Did you mean the bike rack in front of CIE or the bike rack behind the building in front of me?"
User: "the closest one"
System: "This wasn't one of the options I offered..."

When accepting an answer to a choice, the respective dialogue rule uses a comparison function to compare each option against the supplied answer (permitting for example some information to be omitted provided it distinguishes between the answers; 'the one behind the building' for example would be accepted for the latter option). However, *'the closest'* one requires some further spatial interpretation, which the choice selection update rule did not handle at the time. Had the user instead answered *"I meant the closest one"*, the dialogue rules for *correction* behaviour would have instead been used, yielding the correct response.

- **Integration of linguistic components with non-linguistic components:** The visualiser on the tablet device was well received by participants, in particular the feature to centre the view on the destination to allow users to make more informed decisions about making subsequent queries or requests before the robot reached this destination. No problems were encountered with the localisation of the robot (i.e. its perceived position and orientation, updated with subsequent odometry information from the platform) and the visualised environment was correctly aligned with the actual environment. One problem encountered was due to the local planner module (not encompassed by this thesis), which would cause the robot to occasionally 'cut corners' onto nontraversable regions, despite these regions (such as grass and gravel) being identified. The result was for the robot to become stuck, and as a consequence, the global planner would report that its current destination is no longer accessible since a planned path could not extend out of a nontraversable region. The dialogue system's response to such messages is to report the problem of reaching the destination ("It appears I'm having a problem reaching our destination."), and modifying the Information State so that the value of some 'destination entity' was removed. If the robot was manually moved out this region, it would replan a route without reannouncing the target, and start driving autonomously again. As a consequence, when the robot arrived at the target, and the dialogue system received this signal, it threw an error given the destination was now unexpected.

Chapter 7

Conclusion

The aim of this thesis was to produce a dialogue system for a robotic platform acting as a pedestrian assistant. This included the development of a number of spatial models and algorithms to allow spatial language to be both interpreted and generated within dialogue, a natural language parsing system to produce semantic representations of textual input, and architecture and interfaces to allow interaction of these components with non-linguistic modules on the robot required to operate with the physical platform. We analyse the work conducted in light of our original stated aims:

1. **Development of a dialogue framework:** We presented a generic dialogue manager *HURDLE* which supported a number of models of discourse, and was entirely decoupled from our particular task domain. This provided advantages over existing frameworks such as *TrindiKit*, by supporting multi-modal input, requiring the input language to be specified in a Context Free Grammar type structure (to both aid in natural language parsing, anaphoric resolution and allow easier manipulation of data in dialogue rules), and providing easier mechanisms to interact with external systems. Using the *Information State Update* model of dialogue, we built an implementation using this framework which supported the spatial conversations of our particular task domain. By further exploring previous distinctions made between update and selection rules, our dialogue system was successfully able to compensate for multiple acts within a single utterance, in addition to complex discourse behaviour involving grounding of locative expressions and correction, whilst using relatively compact rule definitions. Given the libraries provided and simple example implementations to progressively introduce features, we hope the framework will be used for further research and a variety of other task domains.

2. **Development of a semantic parser:** Using the synchronous grammar formalism, we built a semantic parser to be used in conjunction with the dialogue manager, capable of converting natural language text to a tree-structured semantic form according to the implied input grammar in *HURDLE*. The representation was chosen such that the output form had a tight predictable structure that could easily be processed by the dialogue manager, while providing a deeper level of semantic content than the shallow representations usually employed by dialogue systems. Our parser was capable of accommodating dialogue act segmentation, could prune superfluous information, and could account for the high levels of nonisomorphism

exhibited between the source and target grammars. While the rules were manually specified rather than learnt, suitable extensions to the grammar formalism allowed us to write more compact rules by suitable use of a Part-of-Speech parser. We discussed also how our approach could fall back on keyword based approaches when only fragments of the utterance could be parsed. While our parser was less flexible with mispellings, in practice this did not cause problems; either this would lead to a failed parse (in which case the user would be asked to repeat their utterance), or incorrect place names would lead to the system being unable to ground the user's desired location (where the user could reformulate the query or request). Overall, while the parser was not the primary focus of our research, it provided a considerably more powerful and flexible system than those usually used in conjunction with dialogue systems.

3. **The production of probabilistic models for various spatial prepositions, and algorithms which utilise these models:** Our models incorporated a large number of spatial prepositions. Unlike some prepositional models, these took into account the geometry of the referent and landmark objects, whilst handling a number of edge cases often neglected. We explored the differences between models for validity and those for relevance, and proposed means in which we could measure the latter. Our models also incorporated the visibility of objects, and we produced an algorithm which could efficiently compute this given certain simplifying assumptions. The effectiveness of these models were demonstrated when employed in higher-level algorithms that often computed hundreds of the models on a real environment. We produced a novel algorithm to produce natural language descriptions of objects that disambiguated their location in space. By appropriate selection of objects to use in description based on visibility, high quality descriptions were yielded. While computationally expensive, we provided an analytic approach to efficiently implement this algorithm for certain functions, and an approach using varying granularity and the quad tree data structure in the general case. We also modified an existing GRE algorithm to remove the assumption that all objects are known and visible to the user, yielding a significant improvement in the quality of expressions. We also proposed and implemented an algorithm for grounding spatial expressions that accounted for nested landmarks. We consider the greatest success of this particular research integrating models and algorithms in a broad domain into one unified system that operated successfully in a real-world environment and on a physical platform, where too often existing research is too parochial or inadequately caters for the complexities of outdoor environments.

4. **Integration:** Our final aim was to integrate all these components into a fully implemented system that could operate in conjunction with robotic modules on a physical platform. This motivated many of the features of the individual components; for example the choice of output representation of the semantic parser in light of the dialogue manager, and mechanisms in the dialogue manager to generically handle multi-modal input and hooks to interface with external modules that abstract implementation detail away from the dialogue logic. But this also involved the implementation of architecture to allow interaction with nonlinguistic modules such the odometry, local planner and global planner modules, and a remote inter-

face to allow fluid interaction between user and system in the absence of speech recognition. These components worked seamlessly together on all three platforms they were tested on (at the Oxford, Freiburg and Zurich campuses respectively).

In summary, we were successful in our aim in producing a fully operational existing system, but with novel contributions in many of the individual components that system comprised of. We believe this is a significant step towards a broad commercial implementation that operates on large databases and encompasses more open-ended dialogue with respect to finding and enquiring about objects/places in the environment.

7.1 Future Work

There are a number of possible future avenues in which this work could be extended:

1. Due mostly to legal restrictions, our robot platform was restricted to campus environments rather than urban ones. As such, our dialogue system was oriented more towards spatial properties of environments rather than non-spatial ones, given the lack of need to store large amounts of data for objects with regards to what the sell, serve, etc. While we supported yes-no and wh-questions with regards to these properties (for example, finding entities that sell sandwiches), the data representation was simple attribute-value pairs with no taxonomy of values (e.g. that a gold watch is a subtype of watch), and dialogue rules did not support more complex discussion of entities with regards to these non-spatial properties. For urban environments with large numbers of restaurants and shops, the capacity to account for this discussion would be paramount.

2. In Chapter 2 we distinguished between spatial models for validity and relevance. For the latter, our proposed measures were generic to all spatial prepositions. While we briefly discussed means by which the relevance of specific models might be determined (for example, a 'between' relation becomes more *informative* the more central within the convex hull of the landmarks it is), future research could obtain experimental data to more fully formulate these models and implement them for use.

3. In algorithms where multiple spatial prepositions were simultaneously used, in particular for generating descriptions, different axis for the frames of reference with these prepositions led to occasionally unnatural descriptions. Further research could remove our assumption that prepositions can be interpreted independently of each other.

4. Given the task domain, an algorithm for generating *route directions* would provide an alternative to either escorting the user or merely describing the location of the entity in question. Existing research could potentially be integrated into our system and utilise our prepositional models. This would require expanding our set of models to those which incorporate the motion, for example *along* and *around*.

Bibliography

[Allen, 1987] Allen, J. (1987). *Natural language understanding*. Benjamin/Cummings Publishing Company, Inc.

[Allen and Perrault, 1980] Allen, J. and Perrault, C. (1980). Analyzing intention in utterances. *Artificial intelligence*, 15(3):143–178.

[Alshawi, 1992] Alshawi, H. (1992). *The core language engine*. The MIT Press.

[Ang et al., 2005] Ang, J., Liu, Y., and Shriberg, E. (2005). Automatic dialog act segmentation and classification in multiparty meetings. In *Proc. ICASSP*, volume 1, pages 1061–1064.

[Anguelov et al., 2004] Anguelov, D., Koller, D., Parker, E., and Thrun, S. (2004). Detecting and modeling doors with mobile robots. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3777–3784. IEEE.

[Appelt and Kronfeld, 1987] Appelt, D. and Kronfeld, A. (1987). A computational model of referring. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 640–647.

[Austin, 1975] Austin, J. (1975). *How to do things with words*. Harvard Univ Pr.

[Bennewitz et al., 2009] Bennewitz, M., Stachniss, C., Behnke, S., and Burgard, W. (2009). Utilizing reflection properties of surfaces to improve mobile robot localization. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Kobe, Japan.

[Beringer et al., 2002] Beringer, N., Kartal, U., Louka, K., Schiel, F., Türk, U., et al. (2002). Promise- a procedure for multimodal interactive system evaluation. In *Multimodal Resources and Multimodal Systems Evaluation Workshop Program Saturday, June 1, 2002*, page 14.

[Blunsom et al., 2008] Blunsom, P., Cohn, T., and Osborne, M. (2008). Bayesian synchronous grammar induction. In *Proceedings of NIPS*, volume 21, page 41. Citeseer.

[Bohus and Rudnicky, 2009] Bohus, D. and Rudnicky, A. (2009). The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language*, 23(3):332–361.

[Bos et al., 2003a] Bos, J., Klein, E., Lemon, O., and Oka, T. (2003a). DIPPER: Description and formalisation of an information-state update dialogue system architecture. pages 115–124.

[Bos et al., 2003b] Bos, J., Klein, E., and Oka, T. (2003b). Meaningful conversation with a mobile robot. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 71–74. Association for Computational Linguistics.

[Boye, 2007] Boye, J. (2007). Dialogue management for automatic troubleshooting and other problem-solving applications. In *Proceedings of the 8th SIGdial Workshop on DIscourse and Dialogue*, pages 247–255.

[Burke et al., 2002] Burke, C., Harper, L., and Loehr, D. (2002). A dialogue architecture for multimodal control of robots. *Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems*, page 23.

[Chappelier and Rajman, 1998] Chappelier, J.-C. and Rajman, M. (1998). A generalized CYK algorithm for parsing stochastic CFG. In *Proc. of 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 133–137, Paris (France).

[Clark, 1979] Clark, H. (1979). Responding to indirect speech acts. *Cognitive psychology*, 11(4):430–477.

[Cohn et al., 2010] Cohn, T., Blunsom, P., and Goldwater, S. (2010). Inducing tree-substitution grammars. *The Journal of Machine Learning Research*, 9999:3053–3096.

[Core and Allen, 1997] Core, M. and Allen, J. (1997). Coding dialogs with the DAMSL annotation scheme. In *AAAI Fall Symposium on Communicative Action in Humans and Machines*, pages 28–35. Citeseer.

[Dale, 1989] Dale, R. (1989). *Generating Referring Expressions in a Domain of Objects and Processes.* PhD thesis, Centre for Cognitive Science, University of Edinburgh.

[Dale and Haddock, 1991] Dale, R. and Haddock, N. (1991). Generating referring expressions involving relations. In *Proceedings of the fifth conference on European chapter of the Association for Computational Linguistics*, pages 161–166, Morristown, NJ, USA. Association for Computational Linguistics.

[Dale and Reiter, 1995] Dale, R. and Reiter, E. (1995). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science: A Multidisciplinary Journal*, 19(2):233–263.

[Denofsky, 1976] Denofsky, M. (1976). How near is near? AI memo 344, MIT AI Lab, Cambridge, MA.

[Diosi et al., 2005] Diosi, A., Taylor, G., and Kleeman, L. (2005). Interactive slam using laser and advanced sonar. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1103–1108. IEEE.

[Dobnik, 2009] Dobnik, S. (2009). *Teaching mobile robots to use spatial words*. PhD thesis, Oxford University.

[Dorais et al., 1999] Dorais, G., Bonasso, R., Kortenkamp, D., Pell, B., and Schreckenghost, D. (1999). Adjustable autonomy for human-centered autonomous systems. In *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems*, pages 16–35.

[Earley, 1970] Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

[Ess et al., 2009a] Ess, A., Leibe, B., Schindler, K., and Van Gool, L. (2009a). Robust multiperson tracking from a mobile platform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1831–1846.

[Ess et al., 2009b] Ess, A., Müller, T., Grabner, H., and Van Gool, L. (2009b). Segmentation-based urban traffic scene understanding. In *Proceedings 20th British machine vision conference-BMVC 2009*.

[Fern'andez and Ginzburg, 2002] Fern'andez, R. and Ginzburg, J. (2002). Non-sentential utterances: A corpus-based study. *Traitement automatique des languages*, 43(2):13–42.

[Finetti, 1937] Finetti, B. (1937). La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'Institute Henri Poincaré*, volume 7, pages 1–68.

[Fishburn, 1986] Fishburn, P. (1986). The axioms of subjective probability. *Statistical Science*, pages 335–345.

[Fraser, 1997] Fraser, N. (1997). Assessment of interactive systems. *Handbook on Standards and Resources for Spoken Language Systems (D. Gibbon, R. Moore and R. Winski, eds.)*, pages 564–615.

[Galindo et al., 2005] Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernandez-Madrigal, J., and González, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2278–2283. IEEE.

[Gapp, 1995] Gapp (1995). Angle, distance, shape, and their relationship to projective relations. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 112–117.

[Gapp, 1994] Gapp, K.-P. (1994). Basic meanings of spatial relations: computation and evaluation in 3d space. In *AAAI'94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 1393–1398, Menlo Park, CA, USA. American Association for Artificial Intelligence.

[Ge and Mooney, 2005] Ge, R. and Mooney, R. (2005). A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics.

[Ginzburg, 1996] Ginzburg, J. (1996). Dynamics and the semantics of dialogue. *Logic, Language and Computation*, 1:221–237.

[Ginzburg, 2012] Ginzburg, J. (2012). *The interactive stance*. OUP Oxford.

[Ginzburg and Fernández, 2005] Ginzburg, J. and Fernández, R. (2005). Scaling up from Dialogue to Multilogue: some principles and benchmarks. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 238. Association for Computational Linguistics.

[Ginzburg et al., 2007] Ginzburg, J., Fernández, R., and Schlangen, D. (2007). Unifying self-and other-repair.

[Granell et al., 2010] Granell, R., Pulman, S., Martínez-Hinarejos, C., and Benedí, J. (2010). Dialogue act tagging and segmentation with a single perceptron. In *Eleventh Annual Conference of the International Speech Communication Association*.

[Grice, 1975] Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L., editors, *Syntax and semantics 3: Speech Acts*, volume 3, pages 41–58. New York: Academic Press.

[Guha et al., 2003] Guha, R., McCool, R., and Miller, E. (2003). Semantic search. In *Proceedings of the 12th international conference on World Wide Web*, pages 700–709. ACM.

[H Aust, 1995] H Aust, M Oerder, F. S. V. S. (1995). The philips automatic train timetable information system. *Speech Communication*.

[Haddock, 1987] Haddock, N. (1987). Incremental interpretation and combinatory categorial grammar.

[Hajdinjak and Mihelic, 2006] Hajdinjak, M. and Mihelic, F. (2006). The paradise evaluation framework: Issues and findings. *Computational Linguistics*, 32(2):263–272.

[Hajičová, 1993] Hajičová, E. (1993). *Issues of sentence structure and discourse patterns*, volume 2. Charles University.

[Haviland and Clark, 1974] Haviland, S. and Clark, H. (1974). What's new? acquiring new information as a process in comprehension1. *Journal of verbal learning and verbal behavior*, 13(5):512–521.

[Heger et al., 2005] Heger, F., Hiatt, L., Sellner, B., Simmons, R., and Singh, S. (2005). Results in sliding autonomy for multi-robot spatial assembly. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, pages 5–8.

[Henderson et al., 2008] Henderson, J., Lemon, O., and Georgila, K. (2008). Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics*, 34(4):487–511.

[Herskovits, 1986] Herskovits, A. (1986). *Language and Spatial Cognition*. Cambridge University Press.

[Hobbs, 1978] Hobbs, J. (1978). Resolving pronoun references. *Lingua*, 44(4):311–338.

[Horacek, 1997] Horacek, H. (1997). An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 206–213. Association for Computational Linguistics.

[Jarvis, 1973] Jarvis, R. (1973). On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21.

[Jones et al., 2004] Jones, C., Abdelmoty, A., Finch, D., Fu, G., and Vaid, S. (2004). The spirit spatial search engine: Architecture, ontologies and spatial indexing. *Geographic Information Science*, pages 125–139.

[Jones et al., 2003] Jones, C., Abdelmoty, A., and Fu, G. (2003). Maintaining ontologies for geographical information retrieval on the web. *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 934–951.

[Jurafsky et al., 2000] Jurafsky, D., Martin, J., Kehler, A., Vander Linden, K., and Ward, N. (2000). *Speech and language processing*. Prentice Hall New York.

[Kasami, 1965] Kasami, T. (1965). An efficient recognition and syntaxanalysis algorithm for context-free languages. Technical report, DTIC Document.

[Kate et al., 2005] Kate, R., Wong, Y., and Mooney, R. (2005). Learning to transform natural to formal languages. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1062. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Kelleher and Costello, 2009] Kelleher, J. and Costello, F. (2009). Applying computational models of spatial prepositions to visually situated dialog. *Computational Linguistics*, 35(2):271–306.

[Kelleher and van Genabith, 2004] Kelleher, J. and van Genabith, J. (2004). Visual salience and reference resolution in simulated 3-d environments. *Artificial Intelligence Review*, 21(3).

[Kelleher and van Genabith, 2006] Kelleher, J. and van Genabith, J. (2006). A computational model of the referential semantics of projective prepositions. *Syntax and Semantics of Prepositions*, 29:211–228.

[Kelleher and Costello, 2008] Kelleher, J. D. and Costello, F. J. (2008). Applying computational models of spatial prepositions to visually situated dialog. *Comput. Linguist.*, 35(2):271–306.

[Khatib, 1986] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90.

[Klein and Manning, 2003] Klein, D. and Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

[Kollar et al., 2010] Kollar, T., Tellex, S., Roy, D., and Roy, N. (2010). Toward understanding natural language directions. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 259–266. ACM.

[Kruijff et al., 2006] Kruijff, G., Zender, H., Jensfelt, P., and Christensen, H. (2006). Clarification dialogues in human-augmented mapping. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 282–289. ACM.

[Kruijff et al., 2007] Kruijff, G., Zender, H., Jensfelt, P., and Christensen, H. (2007). Situated dialogue and spatial organization: What, where... and why. *International Journal of Advanced Robotic Systems*, 4(2):125–138.

[Kuipers, 2000] Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 119(1):191–233.

[Lakoff and Johnson, 1980] Lakoff, G. and Johnson, M. (1980). Metaphors we live by.

[Lappin and Leass, 1994] Lappin, S. and Leass, H. (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

[Larsson et al., 1999] Larsson, S., Bohlin, P., Bos, J., and Traum, D. (1999). Trindikit manual. Technical report, Tech. rept. Deliverable D2.

[Larsson et al., 2000] Larsson, S., Ljunglöf, P., Cooper, R., Engdahl, E., and Ericsson, S. (2000). Godis: an accommodating dialogue system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 7–10. Association for Computational Linguistics.

[Larsson and Traum, 2001] Larsson, S. and Traum, D. (2001). Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural language engineering*, 6(3&4):323–340.

[Leech, 1983] Leech, G. (1983). *Principles of Pragmatics*, volume 30. Longman Pub Group.

[Lemon et al., 2006] Lemon, O., Georgila, K., and Henderson, J. (2006). Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the TALK TownInfo evaluation. pages 178–181.

[Levinson and Wilkins, 2006a] Levinson, S. and Wilkins, D. (2006a). *Grammars of space: Explorations in cognitive diversity*, volume 6. Cambridge Univ Pr.

[Levinson and Wilkins, 2006b] Levinson, S. C. and Wilkins, D. P. (2006b). *Grammars of Space: Explorations in Cognitive Diversity*, chapter 1, pages 4–5. Cambridge University Press.

[Lewis II and Stearns, 1968] Lewis II, P. M. and Stearns, R. E. (1968). Syntax-directed transduction. *J. ACM*, 15(3):465–488.

[Lindkvist, 1950] Lindkvist, K. (1950). Studies on the local sense of the prepositions in, at, on, and to, in modern English. CWK Gleerup.

[Logan and Sadler, 1996] Logan, G. and Sadler, D. (1996). *Language and space*, chapter A computational analysis of the apprehension of spatial relations, pages 493–529. MIT Press.

[Matsui et al., 1999] Matsui, T., Asoh, H., Fry, J., Motomura, Y., Asano, F., Kurita, T., Hara, I., and Otsu, N. (1999). Integrated natural spoken dialogue system of jijo-2 mobile robot for office services. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, pages 621–627. American Association for Artificial Intelligence.

[Maye et al., 2010] Maye, J., Spinello, L., Triebel, R., and Siegwart, R. (2010). Inferring the semantics of direction signs in public places. In *Proc. of The IEEE International Conference on Robotics and Automation (ICRA)*.

[McNamara, 1986] McNamara, T. (1986). Mental representations of spatial relations. *Cognitive Psychology*, 18(1):87–121.

[Minsky, 1974] Minsky, M. (1974). A framework for representing knowledge.

[Nguyen et al., 2008] Nguyen, L.-M., Shimazu, A., Phan, X. H., and Nguyen, P. T. (2008). Online structured learning for semantic parsing with synchronous and lambda-synchronous context free grammars. In *Tools with Artificial Intelligence, 2008. ICTAI '08. 20th IEEE International Conference on*, volume 2, pages 135 –142.

[Olivier and Tsujii, 2004] Olivier, P. and Tsujii, J.-I. (2004). Quantitative perceptual representation of prepositions semantics. *Artificial Intelligence Review*, 8:147–158.

[Pattabhiraman and Cercone, 1990] Pattabhiraman, T. and Cercone, N. (1990). Selection: Salience, relevance and the coupling between domain-level tasks and text planning. In *Fifth International Workshop on Natural Language Generation*, pages 79–86.

[Pechmann, 1989] Pechmann, T. (1989). Incremental speech production and referential overspecification. *Linguistics*, 27(1):89–110.

[Pereira and Warren, 1980] Pereira, F. and Warren, D. (1980). Definite clause grammars for language analysis–a survey of the formalism and a comparison with augmented transition networks. *Artificial intelligence*, 13(3):231–278.

[Poon and Domingos, 2009] Poon, H. and Domingos, P. (2009). Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 1–10. Association for Computational Linguistics.

[Regier and Carlson, 2001] Regier, T. and Carlson, L. A. (2001). Grounding spatial language in perception: An empirical and computational investigation. *J. Exp Psychol Gen*, 130(2):273–298.

[Sacks et al., 1974] Sacks, H., Schegloff, E., and Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. *Language*, 50(4):696–735.

[Schmidt et al., 1978] Schmidt, C., Sridharan, N., and Goodson, J. (1978). The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1-2):45–83.

[Searle, 1976] Searle, J. (1976). A Taxonomy in Illocutionary Acts.

[Searle, 1978] Searle, J. (1978). Literal meaning. *Erkenntnis*, 13(1):207–224.

[Skubic et al., 2001] Skubic, M., Chronis, G., Matsakis, P., and Keller, J. (2001). Generating linguistic spatial descriptions from sonar readings using the histogram of forces. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 485–490. IEEE.

[Skubic et al., 2004] Skubic, M., Perzanowski, D., Blisard, S., Schultz, A., Adams, W., Bugajska, M., and Brock, D. (2004). Spatial language for human-robot dialogs. 34(2):154–167.

[Skubic et al., 2002] Skubic, M., Perzanowski, D., Schultz, A., and Adams, W. (2002). Using spatial language in a human-robot dialog. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 4143–4148. IEEE.

[Smith and Hipp, 1994] Smith, R. and Hipp, D. (1994). *Spoken natural language dialog systems: a practical approach*. Oxford University Press, USA.

[Stalnaker, 1978] Stalnaker, R. (1978). Assertion. *Syntax and Semantics: Pragmatics*, 9.

[Steedman and of Techn, 1996] Steedman, M. and of Techn, M. M. I. (1996). *Surface structure and interpretation*, volume 30. MIT press.

[Stone, 2000] Stone, M. (2000). On identifying sets. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 116–123. Association for Computational Linguistics.

[Tellex et al., 2011] Tellex, S., Kollar, T., Dickerson, S., Walter, M., Banerjee, A., Teller, S., and Roy, N. (2011). Understanding natural language commands for robotic navigation and mobile manipulation. In *Proc. AAAI*.

[Tellex et al., 2010] Tellex, S., Kollar, T., Shaw, G., Roy, N., and Roy, D. (2010). Grounding spatial language for video search. In *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, page 31. ACM.

[Thrun, 2003] Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Auton. Robots*, 15(2):111–127.

[Tomko and Winter, 2009] Tomko, M. and Winter, S. (2009). Pragmatic construction of destination descriptions for urban environments. *Spatial Cognition and Computation*, 9(1):1–29.

[Traum and Hinkelman, 1992] Traum, D. and Hinkelman, E. (1992). Conversation acts in task-oriented spoken dialogue. *Computational intelligence*, 8(3):575–599.

[Turner et al., 2008] Turner, R., Sripada, S., Reiter, E., and Davy, I. (2008). Selecting the content of textual descriptions of geographically located events in spatio-temporal weather data. *Applications and Innovations in Intelligent Systems XV*, pages 75–88.

[Vaid et al., 2005] Vaid, S., Jones, C., Joho, H., and Sanderson, M. (2005). Spatio-textual indexing for geographical search on the web. *Advances in Spatial and Temporal Databases*, pages 923–923.

[Van Deemter, 2002] Van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52.

[Van Noord et al., 1999] Van Noord, G., Bouma, G., Koeling, R., and Nederhof, M. (1999). Robust grammatical analysis for spoken dialogue systems. *Natural language engineering*, 5(1):45–93.

[Vasudevan et al., 2007] Vasudevan, S., Gächter, S., Nguyen, V., and Siegwart, R. (2007). Cognitive maps for mobile robots—an object based approach. *Robotics and Autonomous Systems*, 55(5):359–371.

[Viethen and Dale, 2008] Viethen, J. and Dale, R. (2008). The use of spatial relations in referring expression generation. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 59–67. Association for Computational Linguistics.

[Voorhees and Dang, 2003] Voorhees, E. and Dang, H. (2003). Overview of the trec 2003 question answering track. In *Proceedings of TREC*, volume 12.

[Walker et al., 1997] Walker, M., Litman, D., Kamm, C., and Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. Association for Computational Linguistics.

[Walker and Passonneau, 2001] Walker, M. and Passonneau, R. (2001). DATE: a dialogue act tagging scheme for evaluation of spoken dialogue systems. In *Proceedings of the first international conference on Human language technology research*, page 8. Association for Computational Linguistics.

[Ward, 1989] Ward, W. (1989). Understanding spontaneous speech. In *Proceedings of the workshop on Speech and Natural Language*, pages 137–141. Association for Computational Linguistics.

[Ward and Issar, 1994] Ward, W. and Issar, S. (1994). Recent improvements in the cmu spoken language understanding system. In *Proceedings of the workshop on Human Language Technology*, pages 213–216. Association for Computational Linguistics.

[Webber, 1978] Webber, B. (1978). A formal approach to discourse anaphora. Technical report, DTIC Document.

[Wei et al., 2009] Wei, Y., Brunskill, E., Kollar, T., and Roy, N. (2009). Where to go: Interpreting natural directions using global inference. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3761–3767. IEEE.

[Weizenbaum, 1976] Weizenbaum, J. (1976). *Computer power and human reason: From judgment to calculation*. WH Freeman San Francisco.

[Wilske and Kruijff, 2006] Wilske, S. and Kruijff, G. (2006). Service robots dealing with indirect speech acts. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 4698–4703. IEEE.

[Wong and Mooney, 2006] Wong, Y. and Mooney, R. (2006). Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 439–446. Association for Computational Linguistics.

[Wong and Mooney, 2007] Wong, Y. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 960.

[Yeap and Jefferies, 2000] Yeap, W. and Jefferies, M. (2000). On early cognitive mapping. *Spatial Cognition and Computation*, 2(2):85–116.

[Young et al., 2006] Young, S., Williams, J., Schatzmann, J., Stuttle, M., and Weilhammer, K. (2006). D4. 3: Bayes Net Prototype-the Hidden Information State Dialogue Manager.

[Zettlemoyer and Collins, 2007] Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed ccg grammars for parsing to logical form. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007*. Citeseer.