Abstract

Speech Acts can be considered as utterances that are produced not only to describe states of affairs, but rather perform actions that are executed in order to have some effects on the hearer.

A conversationally proficient human-computer interface shall not only have the quality to comprehend textual description, rather the ability to understand human speech acts utterances hence perceiving the direct or indirect intention of the user in terms to provide helpful assistance.

Multi-agent computing utilises the same theory of speech acts to describe communicative behaviours of distributed agents. A competent speech acts natural language interface would provide a bridge to transparently embody human agent in the co-operative computing agent paradigm.

This thesis describes present parsing techniques and theoretical speech acts analysis. The design and implementation of an English Grammar covering common speech-acts sentences. This Grammar and Parser are closely coupled with a demonstration program illustrating the fusion of human speech acts behaviour in an agent world.
Acknowledgements

I would like to thank my supervisor Jim Cunningham for his excellent motivation and guide with this project. Jeremy Pitt for his past works on natural language takes me through the first step of the project.

A special thanks goes to my friends and colleagues for their encouragement and advice on the demonstration. Finally, I would like to thank Flora Kan for her moral support throughout the length of the project.
CHAPTER 1...................................................................................................................................................4
1.1 MOTIVATION AND AIMS ......................................................................................................................4
1.2 HOW TO READ THIS REPORT ..............................................................................................................6
1.3 BACKGROUND AND BASIC INTUITIONS............................................................................................7
1.3.1 Background ........................................................................................................................................7
1.3.1.1 Syntactic Theories and Grammar .................................................................................................7
1.3.1.2 Semantic of Speech Acts ..............................................................................................................12
1.3.1.3 Current Problems and Solutions ..................................................................................................15
1.3.2 Basic Intuitions .................................................................................................................................16

CHAPTER 2 ....................................................................................................................................................18
2.1 THE SYNTACTIC FOUNDATION OF THE GRAMMAR ........................................................................19
2.1.1 QGSL ..................................................................................................................................................19
2.1.1.1 Statements ......................................................................................................................................19
2.1.1.2 Questions .......................................................................................................................................20
2.1.1.3 Commands .....................................................................................................................................20
2.1.2 HPSG ................................................................................................................................................21
2.1.2.1 Sorted Feature Value Structure ....................................................................................................21
2.1.2.2 Schemas and Principles ..............................................................................................................23
2.1.2.3 Lexicons and Lexical Rules .........................................................................................................26
2.1.3 Merging QGSL and HPSG ...............................................................................................................29
2.2 THE SEMANTIC FOUNDATION OF THE GRAMMAR .........................................................................34
2.2.1 HPSG Semantics Principle .............................................................................................................34
2.2.2 Recognition of Primary Speech Acts .............................................................................................35
2.2.3 Recognition of Indirect Speech Acts ...............................................................................................37

CHAPTER 3 ....................................................................................................................................................40
3.1 CHOICE OF IMPLEMENTATION LANGUAGE .......................................................................................41
3.2 IMPLEMENTATION ................................................................................................................................43
3.2.1 Type Hierarchy and Feature Structure ............................................................................................43
3.2.2 Lexicon ...............................................................................................................................................46
3.2.3 Schemas Implementation ................................................................................................................48
3.2.3 Semantics Extraction .......................................................................................................................49
3.2.4 Primary Speech Acts Recognition ..................................................................................................50
3.2.5 Indirect Speech Acts Recognition ...................................................................................................52

CHAPTER 4 ....................................................................................................................................................54
4.1 DEMONSTRATION SYSTEM ...................................................................................................................55
4.1.1 The Blocks World .............................................................................................................................56
4.2 EXAMPLES .............................................................................................................................................58
4.2.1 Detailed Interpretation .......................................................................................................................58
4.2.2 Other Examples ...............................................................................................................................59

CHAPTER 5 ....................................................................................................................................................63
5.1 CONCLUSION .........................................................................................................................................63
5.1.1 Achievements ....................................................................................................................................63
5.1.1. Insights ...........................................................................................................................................64
5.1.2 Future Research ..............................................................................................................................65

Appendix 1 - Bibliography / References ......................................................................................................67
Chapter 1

Introduction

1.1 Motivation and Aims

Speech Acts can be considered as utterances that are produced not only to describe states of affairs, but rather perform actions that are executed in order to have some effects on the hearer.

A conversationally proficient human-computer interface shall not only have the quality to comprehend natural language description, rather the ability to understand human speech acts utterances hence perceiving the direct or indirect intention of the user in terms to provide helpful assistance. With the advent of advance speech recognition technology, a speech act analysis engine would be essential to comprehend human speech. This will unify the communication behaviour of human and computer agent providing a natural user-interface for assisted problem solving.

Traditional truth analysis in analysing natural language description cannot assess the truth of speech acts. Austin [Austin, 1962] pioneered a theory of speech acts that accounts for the semantic analysis of such utterance. Computer scientists engaged in Agent Computing employ the theory to serve as a framework to describe the behaviour of multiple computers communicating in a distributed environment performing co-operative tasks. Agent Communication Language (ACL) such as KQML [Labrou and Finin, 1993] and FIPA [FIPA, 1997] deliver models of speech acts communication. These computational approaches resembling human speech acts
behaviour motivates this project to integrate the two serving as an onset to a human-computer interfacing agent in a multi-agent environment.

This thesis describes the development and implementation of a fragment of English Grammar focusing on the ability to interpret speech act sentences. This project has the following objectives:

- Our grammar shall cover the English fragment that resembles the following most common speech acts; Inform, Request and Query.
- We show how an adaptation of the syntactic theory of Head-driven Phrase Structure Grammar [HPSG, 1994] in association with the prescriptive English grammar of Quirk et al [QGSL, 1985] can be implemented using the Attribute Logic Engine [ALE, 1999] to recognise the designated English subset.
- We will employ FIPA ACL as the object language of the parser. The content language would be a prolog clause resembling first order logic extended with modal operators that could be interpreted against a world model by a prolog interpreter.
- We will develop an inferencing routine to recognise indirect speech acts from the surface form.
- We will incorporate the parser and the interpreter into a demonstration program that user can input an utterance that will inform, query or request a virtual robots to manipulate or answer questions about a blocks world environment.
1.2 How to read this report

This thesis draws a wide base of theoretical works from computational linguistics, to avoid including excessive details of such broad field of works, readers are advised to read Sag and Wasow [Sag and Wasow, 1999] for a formal introduction of the syntactic theory of unification grammar exploited in this thesis. Brady and Berwick [Brady and Berwick, 1983] compiles a series of essays on computational discourse modelling concerning the theory of speech acts.

This report is structured as a logical sequence:

- **Chapter 1 – Overview and Background**: The motivations and aims of the project outlining the goals and objectives. We inspect the background context in which this research is based on. A comprehensive description of the syntactic theories and semantic analysis related to speech acts. Similar related studies are examined.
- **Chapter 2 – Design**: This chapter contains details of the design of our grammar fragment and its computational relatedness. Methods of indirect speech acts analysis are examined.
- **Chapter 3 – Implementation**: Methods to construct a parser for the grammar specification is outlined. The inferencing algorithm of indirect speech acts is discussed.
- **Chapter 4 – System Operations**: This chapter outlines the capability of the demonstration program and its incorporation of the parser developed in the preceding chapter.
- **Chapter 5 – Evaluation and Conclusion**: This chapter evaluates the system we have developed. This section highlights the strength and weakness of the system, and concludes by examining the deliverables that met the initial goals.

Readers more interested in the syntactical aspects of the grammar should focus on chapter 1.3.1.1. and 2.2.. For an analysis of English grammar in relation to speech act shall concentrate on chapter 2.1.1.. Those interested in the semantic aspects of speech acts and the FIPA Agent Communication Language shall read chapter 1.3.1.2 and 2.2.
1.3 Background and Basic Intuitions

1.3.1 Background

The intended application of our system was user intention recognition in a human computer dialogue. The system would respond to the user intention by producing relevant actions. This study surveys various fields of computational linguistic covering syntactic analysis, semantic interpretation and speech acts analysis. Let us now look at each in depth.

1.3.1.1 Syntactic Theories and Grammar

The acceptance of a valid input to a natural language system is described by the underlying grammar.

Quirk et al [QGSL, 1985] provides a comprehensive account of traditional English Grammar based on empirical study. QGSL suggests the following taxonomy of clause type or mood:

- Statements
- Questions
- Commands
- Exclamations

The clause type would in turn has a surface speech act meaning of inform, query, request and expressive. However, as Searle [Searle, 1979] suggested there is no one-to-one mapping from syntactic mood to the corresponding speech acts. Searle defends that there is a basic assumption that speaker always mean what they literally say, and only by the literal meaning that any indirect speech acts can be inferred. Therefore, one could follow QGSL’s taxonomy of clause type to identify the surface speech acts, then further inferences based on world knowledge, mutual beliefs and social conventions is essential to recognise the true intentions of the speaker.
By merging the prescriptive English Grammar of QGSL, and a language independent generative grammar, we will have the machinery to formally describe the grammar of our English fragments.

Generative Grammar began in the 1950s with Noam Chomsky’s minimalist approach [Chomsky, 1957]. It provides a formally definable computational device that enumerates a set of valid strings. Many theories span out from the school of Generative Grammar, notably the Generalised Phrase Structure Grammar [Gazdar et al, 1985] and Head-Driven Phrase Structure Grammar. We will survey the two for their applicability to this project.

**Generalised Phrase Structure Grammar**

Gazdar et al formulated GPSG as a context-free and transformation-free analysis of syntax [Gazdar et al, 1985]. It describes generalisation of linguistic phenomenon and movement without using transformations.

With a phrase-structure syntax, the categorical symbol is replaced with a collection of **features**. The features are specified as pair naming the feature and its value from a given range. The set of pairs represents a conventional monadic symbol of categorical grammar.

GPSG comprises 6 major components:

- **Linguistic categories** are complex symbols of a set of feature-value pairs.
- **Immediate Dominance (ID)** and **Linear Precedence (LD)** rules. ID rules govern the admissible immediate daughter category of a mother category in a categorial grammar syntax tree. Linear Precedence determines the order of the sister nodes under a given mother category.
- **Metarules** generate new rule from existing one.
- **Three universal feature conventions** govern the feature distribution and agreement between categories.
- **Feature Specification Defaults (FSD)** defines default value assumed for features if they are not specified.
• **Feature Co-occurrence Restriction (FCR)** restricts the feature dependency in a category restricting possible categories in the grammar.

The GPSG formalism observes a version of compositionality by associating a syntax rule to semantics rule. A complex syntactical category is built recursively by the admissible syntactic rules, and the semantic of the highest initial category can also be built with the one-to-one syntax-semantic rule to build the semantic representation compositionally. This idea is attributed to Frege's Principle of Compositionality. More precisely, the principle of Compositionality can be expressed by, "The meaning of an expression is a monotonic function of meaning of its parts and the way they are put together."

Following Pitt, the components of GPSG have to be directly implemented or reconstructed to render them computationally tractable [Pitt, 1991]. It is a major restriction and a loss of generality of linguistic principle when strictly used in implementation.

**Head-driven Phrase Structure Grammar**

HPSG [HPSG, 1994] evolves directly from GPSG. It synthesises ideas from a variety of perspectives, including situational semantics, data type theory and a variety of syntactic theories of the early and mid-1980s. This hybrid approach provides a wealth of the best features from other established syntactic theories. It emphasis on heads of phrase which are linguistic component that exerts control syntactically and semantically on other constituents of the complements [HPSG, 1994].

HPSG is based on a sorted feature structure to represent linguistic object. A sorted feature structure consists of a set of feature-value pairs. The value of a feature is determined by the sort of the feature. It enables complex nested feature, which gives it a powerful representational structure. In HPSG, linguistic category is represented by feature structures and is displayed orthographically as Attribute Value Matrix (AVM) preceded by a subscript indicating sort of the enclosed structure. Example of a HPSG structure representing the lexicon "Dog" is shown below.
Linguistic entity is represented via type. A noun sort introduces its internal features and inherits features from its parent sort. The “Dog” lexicon is of type cn_lxm which is inherited from a noun type which introduces features (syn, arg_st, sem). The hierarchical approach enables inheritance of common properties important to many linguistic objects.

HPSG is strongly lexicalist. Subcategorisation features of GPSG coded in the phrase structural rule are refined to move to the lexical level. It relies on a wealth of complex lexicons, but a substantially small set of general phrase structural rule that claimed to be language independent. The phrase structural rules or schemas identified by Sag and Pollard [HPSG, 1994] governs the distribution of the projection of subject, head, modifiers, complements, markers and adjuncts of a phrase.

HPSG also factorise the phrase structural properties of GPSG into general principles and constraints governing the formulation of phrase projected by the schemas, therefore limiting the possible phrases to be formed. Those principles maybe language dependent and semantic translation is provided through the principles.
This theory is highly praised in the computational linguistic community. Implementations of HPSG are being developed at numerous institutions, e.g. the LINGO project at the birthplace of HPSG at CSLI [LINGO, 1999]. It has a strong emphasis on rich lexical information but general projection schemas. This approach enables already existed online dictionary to be integrated into current implementation for a wider coverage of the target language. It has also been adapted in many languages other than English, such as Chinese and Japanese and German. However, the LINGO project currently has no experimentation on parsing speech act phrases.

HPSG has also incorporated a version of situational semantic theory into the formalism. Although it places no restriction on the semantic interpretation, an analysis such as Montague's PTQ [Montague, 1974] can be adapted in the framework of the syntactic analysis due to its Compositionality. HPSG presents the semantic representation as a situation schemata [Fenstad et al, 1987]. It can be viewed as a notation for situation-theoretic objects, it sums up the linguistic form, and it could be treated in various kind of semantic interpretations such as Montague's intensional logic to determine its reality. Fenstad has provided an account to the semantic interpretation of situational schemata in a model-theoretic manner.

With the benefit of direct implementation and generality of language independence, we would favour the HPSG paradigm in this research. We will develop a HPSG style grammar based on the implementation of the preliminary version of HPSG on the Attribute Logic Engine by Penn and Carpenter [ALE, 1999].
1.3.1.2 Semantic of Speech Acts

A truth-conditional analysis of sentence semantic usually provides a useful account of the meaning of a statement. However, in the analysis of speech act sentences, they are not apparently used with any intention of making true or false statement:

- I apologise that I have stepped on you.
- I promise I will come
- I hereby name this ship as H.M.S. Flounder

The examples above are described by Austin [Austin, 1962] that they are not used only to describe a state of affairs, but rather actively perform things.

Issues of truth and falsity have been of central interest throughout linguistics study. Truth-conditional analysis of the meaning of sentence has its limitation when applied outside the subset of sentences making a statement [Levinson, 1983, P.251]. In Austin's theory of speech acts, he set about demolishing the traditional positivist approach in the analysis of sentences which apparently make no intention to convey a truth condition. He termed these sentences as performatives, in contrast with the constatives which makes a statement and assertion.

He claimed we cannot access the truth of such utterance, however, they could be infelicitous whereas meaning some pre- and post-conditions have to be true in terms for the performatives to be felicitous. *Illocutionary act* refers to speech act identified with reference to the communication intention of the hearer. As Quirk suggested the illocutionary act is dependent on context, it is left to the hearer to draw inference on the intention of the speaker based on the preceding utterance and common knowledge.

In natural language understanding, although we could not extract out the truth from the speech acts of the user input, we would like to show whether the input is felicitous based on a conjunction of successful conditions. Searle proposed a classification into four kinds of condition, *propositional content*, *preparatory preconditions*, *conditions on sincerity*, and the *essential post-conditions* [Searle, 1969]. Our
Application could take the natural language input and check it against the above conditions to retrieve the intention of the input, and verify such input is felicitous with the assumption of the situation.

The Agent Communication Language FIPA [FIPA, 1997] provides a representation of speech acts and their corresponding semantics through the mental attitudes of the communicating agents. This is roughly a parallel to the conditions devised by Searle. If one could parse a natural language sentence into the FIPA language, one would have a formal semantics of the speech act sentence described by the agent’s belief, goals and other modality.

The FIPA representation of speech act is described by the ACL message. The message has a set of parameters denoting the sender, receiver, propositional content and other communication and ontology description attributes. Each message is identified by the type of speech act, and for each type of speech acts, there is a precise semantic expressed in terms of belief of the communicating agents. The belief formulae can be checked against a possible world semantic, therefore giving a formal meaning of speech acts.

The goal of successful analysis of the semantics of a speech act utterance is to translate the sentence into the surface linguistic act. Searle suggests that there is a basic assumption where speaker always mean what they literally say, and only by the literal meaning that any indirect speech acts can be inferred.

The literal meaning can be extracted from the sentence by the syntactic analysis. With Frege’s Principle of Compositionality, the meaning of the sentence is composed of the meaning of the parts. HPSG provides an account of building the semantic representation through the semantic principle. Montague’s PTQ has one-to-one syntax and semantic rules which builds the semantic from the parsed syntactic tree.

One could extract the propositional content of a speech act sentence using the traditional techniques described above, and the primary illocutionary force by the language dependent convention prescribed by the grammar, such as clue word,
‘please’ which denotes an indirect request and a imperative sentence describes a surface request.

A sentence could be attributed to many illocutionary forces with each describing different intention of the speakers with a single utterance. Searle suggests a series of generalisation which could be used to extract the secondary illocutionary force (indirect speech acts) of an utterance [Searle, p45, 1979].

Allen [Allen, 1983] uses a plan-inference technique to extract speaker intention from a surface speech acts. He specified a model representing knowledge about the world, goals, actions and speech acts. Speech Acts are actions (parameterised procedures) that will change the state of the world. Given an initial world $W$, a goal state $G$, a plan is a series of actions that transform $W$ to $G$. Therefore the hearer can attempt to reconstruct the speaker’s plan based on the actions the speaker performed. This depends on the hearer’s knowledge of a rational plan and his beliefs about what goals the speaker is likely to have.

Another approach of recognising indirect speech acts relate to the politeness phenomenon [Ardissono, Boella, Lesmo, 1995]. The speech acts are represented as actions in a plan library. The action is activated on the basis of the presence of syntactic and semantic features of the parsed input utterance. The analyser uses the politeness indicator to climb up the decomposition and generalisation hierarchy of acts in the library. The possible paths represent the possible secondary speech acts.

The TRAINS project [TRAINS, 1994] is an attempt to build a planning assistant to interact with a human agent in problem solving tasks. It utilises probabilistic method to recognise the user intention by studying large corpus to extract such heuristics.
1.3.1.3 Current Problems and Solutions

All the attempts we have looked at are domain specific to the problem. There are no direct emphasis to incorporate natural language parsing of speech acts utterance into an agent language that could be used in a multi-agent environment to serve as an interface between the human agent and the computational agents. The highly praised HPSG focuses on situational semantics as the semantic representation which caters for the traditional truth-analysis, but insufficient to interpret speech acts utterances.

Our project is a hybrid approach to experiment with the integration of HPSG with Agent Communication Language and the recognition of illocutionary forces based on the various methods examined. We would show such a natural language system could be used in the wider context of distributed computing.
1.3.2 Basic Intuitions

The system analyses a sentence syntactically and extract and compose the whole semantic representation from the semantic parts of the sentence. The illocutionary force can be identified from the linguistic device, such as clue words, surface form and mood. The speech act representation can then be used as the input of the indirect speech act analyser, which suggest possible secondary illocutionary acts from the primary. Then application can use the compiled possible speech acts to perform domain action based on the force.

This intuition can be captured in the following example:

- User utters to the robot: “Can you move to room 1?”
  The sentence syntactically falls into the classification of a question because of the subject-auxiliary inversion. In turn, it dictates a surface speech act of querying the ability of the hearer of passing the salt that can be captured in the following representation:

  In FIPA ACL syntax,
  \[
  (\text{query} \\
  \quad :\text{sender} \ \text{speaker} \\
  \quad :\text{receiver} \ \text{hearer} \\
  \quad :\text{content} \ \text{can(hearer, moveto(hearer, room1))} \\
  )
  \]

  Or in a more compact form:
  query(speaker, hearer, can(hearer, moveto(hearer, room1)))

\[\text{Notes: Other ACL parameters have been omitted}\]
**speaker**, **hearer** and **room1** are terms denoting individual and object in the domain. The content value is a first order logic extended with modal operators such as **can** and **want**. The three place predicate **moveto** denotes an action of hearer moving to room1.

The speech act analysis would begin from the above input, and draw inference from a set of rules. Based on Searle’s generalisation, we could infer that the speaker has a goal of indirect requesting the hearer to move to room 1, because he is asking for the ability (can) to move. Therefore the analysis would produce a list \[[query(speaker, hearer, can(hearer, moveto(hearer, room1))), request(speaker, hearer, moveto(hearer, room1))\] that constitutes the possible illocutionary force of the utterance. The domain interpreter would interpret the speech act representation and produces a response such as answering ‘Yes, I can’ then move to room1.

The interpretation process shall have the following attribute:

- The output of the first phase is the true surface meaning of the input utterance.
- The content representation has a formal semantic. Our chosen representation is similar to the SL content language specified as a candidate content language of the FIPA Language (FIPA, Annex B, 1998).
- The speech act analyser will select the secondary illocutionary acts based on the input, and the inference process can be explained by the mental attitude of the agents.
- The domain interpreter would select the most appropriate intention of the speaker and find the most appropriate response based on those actions.
- The quality of the selection is based on the heuristic chosen.
Chapter 2

Syntactic and Semantic Foundations

In this chapter

In section 2.1.2, we would look at the semantic analysis based on our syntactic treatment of the English subset. We will look at possible methods to extract illocutionary force from surface linguistic acts. We describe and motivate the linguistic and computational theories used in the development of our system. The target English subset is formalised.

Section 2.1.1 surveys the use of Head-driven Phrase Structure Grammar [HPSG, 1993] in association with the English Grammar of Quirk et al [QGSL, 1985] as the basis to recognise our English Fragment.
2.1 The Syntactic Foundation of the Grammar

2.1.1 QGSL

The grammar has its English coverage dictated by the definition of clause type given in QGSL. We will focus only on the following types:

<table>
<thead>
<tr>
<th>Clause Type</th>
<th>Surface Speech Act</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements</td>
<td>Inform</td>
</tr>
<tr>
<td>Questions</td>
<td>Query</td>
</tr>
<tr>
<td>Commands</td>
<td>Request</td>
</tr>
</tbody>
</table>

Table 1 Relation of clause type and surface speech act

The clause type has their own sub-types differentiated by the sentential form.

2.1.1.1 Statements

QGSL prescribes a statement as a sentence in which the subject is always present and generally precedes the verb. Huddleston [Huddleston, 1984] observes all kernel clauses are declarative. Several linguistic factors can override declarative clause type in determining the illocutionary force of an utterance:

(a) Performative verbs such as promise, request, order, ask, advice, etc. overrides the declarative's surface speech act. When a performative verb is used as the main verb of a main clause, the subject of the performative verb is in the first person and the direct object is the second person, then the illocutionary act of the clause is taken from the force of the performative verb.
(b) Deontic use of the modals in a declarative can be used with the illocutionary force of a request / directive.
(c) Intonation or punctuation can alter a statement to a question. We will ignore this fact in this research.

2.1.1.2 Questions

Questions are sentences marked by one or more of the following criteria:

(a) The placing of the operator immediately in front of the subject. This type is denoted by the yes-no question.

(b) The initial positioning of an interrogative or wh-element.

2.1.1.3 Commands

Sentences which normally have no overt grammatical subject and whose main verb is a finite verb. It is implied in the meaning of a command that the omitted subject of the verb is the second person pronoun you.

The above clause types defines the set of English Grammar that our grammar shall handle in the system.
2.1.2 HPSG

The grammar we are developing is guided by the syntactic formalism of Head-driven Phrase Structure Grammar (HPSG, 1994). HPSG promotes a small set of universal principles and schemas to govern the formation of word string. Most of the complexity of the grammar is encoded in the lexicons as sorted feature structures. In our study, we will implement a modified version of the HPSG feature structure devised by Pollard and Sag to model the phenomenon of speech act sentences. The universal principles and schemas in our grammar would closely model the original HPSG formalism.

2.1.2.1 Sorted Feature Value Structure

The feature structures employed here have a close resemblance to other syntactic theories such as GPSG. The feature structure in HPSG is sorted as well as well-typed. GPSG models a syntactic category using a set of feature-value pairs, where each feature has a set of possible values. HPSG introduces a nested feature structure, where each feature corresponds to a sort and the sort is partially ordered. The sort also defines the attributes a feature of that sort could have, whereas the attributes are themselves sorted-feature structures.

Syntactic categories are modelled by feature structures of sort synsem. The hierarchy enables the sub-sorts to inherit attributes from synsem. It also allows us to define constraints on the sub-sorts so that restriction of the possible values in the feature structures can be propagated down the hierarchy.

Synsem is the most important sort in the grammar. The major attributes that we will discussed later can be readily described in the following Attribute Value Matrix (AVM).
The *sem* attribute constitutes the semantic information of the phrase that contains it. This will be examined in depth later in the discussion. The *nonloc* attribute carries information for long-distance dependency.

The *syn* attribute of the *loc* sort can be regarded as the syntactic category of the word or phrase in question. The *head* value of a word or phrase is its part of speech, analogous to the information in an X-Bar theory category and the Head features of GPSG. *Pos* are further extended by sub-sorts such as *noun, adj, prep* and *verb* to provide specific information about the parts of speech in concern.

The *subj* and *comps* attributes are termed as the valence features of a word or phrase. They are list of sort *synsem* that provides a specification of what signs to combine in order to become saturated. *Subj* defines the possible subject and specifier, while *comps* defines the complements.

To further illustrate the feature structure in HPSG, the *synsem* structure describing the verb lexeme *walk* is shown by the AVM on the right.

The lexeme *walk* can be modelled by a sort *sintran_verb_lex* which is a subsort of *lexeme*. Therefore it would inherit the structure from *synsem*. Its part of speech is a verb in the base form. The most interesting part of the structure is the valence features. *Comps* is an empty list [], therefore *walk* doesn't take any complement agreeing with the intransitive property of the verb. Since a word must take a noun phrase as its subject, *subj* denotes a singleton list of NP. NP is a
shorthand for a \textit{synsem} structure having \textit{noun} as \textit{head}, and empty valence features.

This encoding allows flexible construction the lexicon. We can also introduce constraints on the sort to limit the possible values of the features in that sort, i.e. \textit{head} must be \textit{of sort verb} and \textit{comps} = \textit<> if it is a \textit{sintran\_verb\_lex}.

\section{2.1.2.2 Schemas and Principles}

In contrast with GPSG and other generative grammar, HPSG employs a small set of schemas that formulates the rule governing the projection of grammatical sentences. The principles associated with the schemas will further describe the conditions for the rules to be applicable. The grammar we are developing consists of the following schemas which are closely related to the original HPSG counterpart with slight modification.

\textbf{The Immediate Dominance Schemas}

\textbf{Schema 1: Head-Subject Schema}

\begin{center}
\begin{tikzpicture}[level distance=1.5cm,
level 1/.style={sibling distance=3cm},
level 2/.style={sibling distance=2cm},
level 3/.style={sibling distance=1cm}]

\node {phrase}
  child {node {subj \textit<>}
    child {node {comps \textit S}}
    edge from parent node [above] {$\Rightarrow$}
  }
  child {node {phrase}
    child {node {subj \textit 1}}
    child {node {comps \textit S}}
    child {node {\textit H}}
    edge from parent node [above] {$\Rightarrow$}
  }
  edge from parent node [above] {$\Rightarrow$}

\end{tikzpicture}
\end{center}

The Head-Subject schema licenses phrase such as "John walks" assuming there is a rule called word-promotion that will transform a \textit{synsem} of \textit{word} sort to a \textit{phrase} with all values copied, i.e. A proper-noun into a noun phrase, an intransitive verb into a verb phrase.

\textit{Notational Note:} \textit{H} denotes the \textit{Head} of the schema. Boxed value denotes a variable of the corresponding sort specified by the attributes. Therefore \textit{S} is structure shared between the Mother phrase and the Head phrase. For clarity, the attributes introducing the \textit{subj} and \textit{comps} attributes are not shown.
Schema 2: Head-Complement Schema

\[
\begin{align*}
&\text{phrase} \\
&\text{comps} \left[ < > \right] \\
&\text{subj} \left[ < S > \right] \Rightarrow H \\
\text{word} \\
&\text{comps} \left[ 1 .. n \right] \\
&\text{subj} \left[ < S > \right] \\
\end{align*}
\]

Example: go home

Schema 3: Head-Specifier Schema

\[
\begin{align*}
&\text{phrase} \\
&\text{subj} \left[ < > \right] \\
&\text{comps} \left[ S \right] \Rightarrow 1 \\
\text{phrase} \\
&\text{loc|syn|head|spec} \left[ 2 \right] \\
&\text{subj} \left[ < > \right] \\
&\text{comps} \left[ < > \right] \\
\text{phrase} \\
&\text{subj} \left[ 1 \right] \\
&\text{comps} \left[ S \right] \\
\end{align*}
\]

Example: a box

Schema 4: Head-Modifier Schema

\[
\begin{align*}
&\text{phrase} \\
&\text{subj} \left[ S \right] \\
&\text{comps} \left[ C \right] \Rightarrow \\
\text{phrase} \\
&\text{loc|syn|head|mod} \left[ 1 \right] \\
\text{phrase} \\
&\text{subj} \left[ S \right] \\
&\text{comps} \left[ C \right] \\
\end{align*}
\]

The Head-Modifier schema also licenses modifier phrase that post-modify the head phrase.

Example: red box
Schema 5: Head-Filler Schema

The Head-Filler schema is used to handle sentences such as "Who did Kim claim _ left?" which introduces a "trace" at the _ position. The trace will introduce a missing element and such element is stored in the inherited|slash set. The slash would be passed higher up into the parse tree until a filler phrase can be bound to that slash. In the example, "Who" is the noun phrase bounding the trace introduced by the missing direct object of 'claim'.

The Principles

The above schemas are termed as headed schemas because the head daughter selects the possible non-head daughters in the rule. The following principles pose extra constraints on the headed rules to how they license grammatical sentences.

**Head Feature Principle**

The loc | syn | head value of the mother is token-identical to the loc | syn | head value of the head daughter.

**Semantic Principle**

The loc | sem value of the mother is token-identical to the loc | sem value of the head daughter unless it is applied in the head-modifier schema whereas it is identical to the loc | sem value of the modifier daughter.

**Non-local Features Principle**

For each non-local feature F (i.e. slash) in nonloc | inherited and nonloc | bind, the value for the nonloc | inherited | F of the mother phrase is the set difference of the
union of all values on the daughters and the value of the \texttt{nonloc} \mid \texttt{bind} \mid \texttt{F} on the head daughter.

\textit{Trace Principle}

The \texttt{synsem} value of any trace must be a member of the \texttt{comps} list of a substantive word (i.e. \textit{head is a subsort of subst}).

\textit{Valence Principle}

The union of the \texttt{subj} and the \texttt{comps} values of the mother phrase is the set difference of the union of the \texttt{subj} and the \texttt{comps} values of the head daughter and the sisters.

\subsection*{2.1.2.3 Lexicons and Lexical Rules}

The sorted features structure formalism certainly provides an elegant way to handle polymorphism and unification of feature structure. Since HPSG is highly lexicalistic, the wealth of information encoded in the lexicon would undoubtedly increase the complexity in managing large amount of lexical entries. We could reduce the redundancy of the lexicon by a hierarchy of lexical sorts and the sort is differentiated by the associated feature appropriateness constraints. The motivation being the lexicon can be classified in group differentiated by minor difference in their syntactic form. All feature constraints would propagate down the hierarchy, as we shall see, the lower the hierarchy, the more specific the lexical sort. Lexical entries are associated with the leaf sorts in the hierarchy and additional lexeme specific features such as semantic representation is defined at the lexical definition level.
The following tree illustrates a sort hierarchy for the verbs.

Firstly, we shall notice the \textit{lexeme} sort is sub-typed by \textit{infl\_lex} (Inflecting lexeme) and \textit{const\_lex} (Constant Lexeme). Lexicons defined as a sub sorts of \textit{infl\_lex} would be under the transformation of some lexical rules that will morphologically and syntactically modify the base form specified in the lexical database. Sub sorts of \textit{const\_lex} would not be eligible for any lexical rule transformation.

If we direct ourselves to the sub-tree rooted at \textit{verb\_lex}, we could classify a verb as transitive or intransitive. These would then be sub-typed as prepositionally transitive, strict transitive and ditransitive and likewise for the intransitives. The list is not exhaustive, further distinction can be made by sub-typing these sorts. The advantage of representing information in the type-hierarchy or as constraints is a subjective judgement.

We could posit each sort with their corresponding constraint.

\[
\begin{array}{c}
\text{verb\_lex} : \\
\begin{bmatrix}
\text{synhead} \\
\text{verb} \\
\text{rform} \\
\text{synsubj}
\end{bmatrix}
\end{array}
\]

\text{verb\_lex} restricts the \textit{head} value to be a verb of base form, and it has to have a subject of any \textit{synsem} value.
A transitive verb should have at least one complement as well as all the constraints inherited from verb_lex.

A prepositional transitive verb will have the second member of its complement to be a prepositional phrase. PP is a shorthand for a phrase structure having a saturated valence value and a prep as head.

We could also reduce redundancy in the lexicon by introducing lexical rules which transform stored lexicons in their base form to other morphological and syntactical similar form. All lexical rules will take a lexeme associated with an infl_lex sub-sorts as input and produce the inflected form and promote it to the const_lex sort. As we have seen from the schemas, they operate on phrase and word sort. We could have a lexical-satisfaction rule to promote a const_lex to a word by copying the loc and nonloc values across.
2.1.3 Merging QGSL and HPSG

Our objective is to formally specify the prescriptive grammar of English subset in the framework of HPSG. QGSL provides a natural fit into HPSG. QGSL views a simple sentence of English as containing a number of clause elements classified into the following categories and subcategories;

- subject \( (S) \)
- verb \( (V) \)
- object
  - direct object \( (O_d) \)
  - indirect object \( (O_I) \)
- complement
  - subject complement \( (C_s) \)
  - object complement \( (C_o) \)
- adverbial \( (A) \)

The different clause types are determined by the verb class of the main verb within the verb element which determines the complementation and the subject. In table 2, we exemplify each clause type.

**Table 2 Major Clause Type**

<table>
<thead>
<tr>
<th>Type</th>
<th>Subject</th>
<th>Verb</th>
<th>Object (s)</th>
<th>Complement</th>
<th>Adverbial</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV</td>
<td>You</td>
<td>walk</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(intransitive)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVO</td>
<td>You</td>
<td>move</td>
<td>the rectangle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(monotransitive)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVC</td>
<td>The rectangle</td>
<td>is</td>
<td>red</td>
<td>(subject complement)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(copular)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVA</td>
<td>The rectangle</td>
<td>is</td>
<td>in room 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(copular)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVOO</td>
<td>You</td>
<td>give</td>
<td>me</td>
<td>(indirect object)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(ditransitive)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the circle</td>
<td>(direct object)</td>
<td></td>
</tr>
<tr>
<td>SVOC</td>
<td>You</td>
<td>paint</td>
<td>it</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(complex-transitive)</td>
<td></td>
<td>(object complement)</td>
</tr>
<tr>
<td>SVOA</td>
<td>You</td>
<td>move</td>
<td>the box</td>
<td>to room 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(complex-transitive)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We adopt these canonical forms of clause types as the permissible structures of declarative sentence.

These clause types can be directly encoded in HPSG by specifying the \textit{subj} and \textit{comps} features of the main verb. We will treat clause elements O, C and A as the complementation features that will be specified in the \textit{comps} list of the verb. The permissible Subject (S) of the verb will be directly encoded in the \textit{subj} feature. Together, they form the valence features that have to be satisfied by the corresponding projection of the clause type. The clause is termed as saturated when all elements of the \textit{subj} and \textit{comps} lists are satisfied.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Variants} & \textbf{Example} \\
\hline
\textbf{Copular [SVC and SVA]} & \\
[A1] Adjectival Cs & \textit{The rectangle is red} \\
[A2] Nominal Cs & \textit{The rectangle is mine} \\
[A3] Adverbial complementation & \textit{The rectangle is in room1} \\
\hline
\textbf{Monotransitive [SVO]} & \\
[B1] NP as O & \textit{The robot paints the circle} \\
[B2] That clause as O & \textit{I promise that I will move to room1} \\
[B3] To infinitive as O & \textit{I want to move the rectangle} \\
[B4] PP as O & \textit{The robot moved to room1} \\
\hline
\textbf{Complex-transitive [SVOC and SVOA]} & \\
[C1] Adjectival Co & \textit{The robot paints the ball red} \\
[C2] O + adverbial & \textit{The robot moves the box to room1} \\
\hline
\textbf{Ditransitive [SVOO]} & \\
[D1] NP as O$_{i}$ and O$_{d}$ & \textit{The robot give me all the balls} \\
[D2] With prepositional O & \textit{The robot move all the balls to room1} \\
[D3] O$_{i}$ + that-clause & \textit{I inform you that the robot is in room1} \\
\hline
\end{tabular}
\caption{Permissible variants of verb valence features.}
\end{table}

The verb variance can be encoded as a verb lexical hierarchy as described in Section 2.1.2.3. Therefore each lexical entry can be associated with a lexical type constraining its valence features. The devised clause types licenses declarative sentences and performative sentence such as \textit{I inform you that the circle is in room1} given by [D3].
The imperative dictates a sentence fronted by an action verb, such as *move, pick, paint*, etc. instead of the copular or the performative verbs. The complementation is the same as declarative sentence, but the subject is an explicit missing second person NP. We could devise a language-dependent schema which licenses the imperative.

**Imperative Schema**

\[
[\text{phrase } S] \Rightarrow \begin{array}{c}
\text{phrase } S \\
\text{VP} \quad \text{loc|syn|subj:} \left[ \text{NP loc|syn|head|agr:} (\text{sg2;pl2}) \right] \\
\text{loc|syn|comps:} []
\end{array}
\]

*NP* is a *synsem* structure denoting a Noun Phrase, and *agr* is a head feature denoting the agreement (number, gender, person) of the nominal. (sg2 and pl2 are extensional types denoting second person singular and second person plural.)

The interrogatives identify two distinct syntactic forms:

<table>
<thead>
<tr>
<th>yes-no question</th>
<th>op-SVC</th>
<th>Placing the operator before the subject. If no item in the verb phrase can acts as operator, do is introduced</th>
</tr>
</thead>
<tbody>
<tr>
<td>wh-question</td>
<td>wh-VC</td>
<td>Placing a Q-element at the front of the sentence, with the Q-word in the first position of the Q-element. When the Q-element is not the subject, there is an inversion of subject and operator.</td>
</tr>
<tr>
<td></td>
<td>wh-op-SV</td>
<td></td>
</tr>
</tbody>
</table>

If we treat the operator as auxiliary verb taking different forms of main verb as its complement. E.g. *has* takes a past participle as its complement, and the complemented verb share the subject with the auxiliary verb. The *subject_aux_inversion* lexical rule operating on the *aux_verb_lex* can perform the subject-operator inversion.

\[
\text{subject_aux_inversion} \quad \Rightarrow \begin{array}{c}
\text{inv_aux_verb_lex} \\
\text{loc|syn|subj:} [] \\
\text{loc|syn|comps:concat(Subj, Comps)}
\end{array} \Rightarrow \begin{array}{c}
\text{aux_verb_lex} \\
\text{loc|syn|subj:Subj} \\
\text{loc|syn|comps:Comps}
\end{array}
\]

The rule will preserve the structure sharing between the subject of the first element of the *Comps* list (verb after the auxiliary) and the auxiliary’s subject. The structure
sharing concords the subject-verb agreement between the subject and the verb and the subject with the auxiliary.

The `inv_aux_verb_lex` will ensure the verb would have the head boolean features `inv` and `aux` set to true signifying it as a inverted auxiliary. Therefore any projected clause from this verb will result as an inverted clause identifying a question.

Wh-question demands long distance dependency to handle the missing element in the clause. A wh-question “Where do you move the rectangle to _”, the underscore denotes a missing prepositional complement. The word *Where* in the above example acts as a pronoun. The missing element is said to be filled by the filler NP required as the complement of *to*.

This phenomenon can be handled by the head-filler schema. The prepositional complement *to* introduces a missing NP carried up the tree by the non-local feature `inherited | slash`. The `slash` list feature collects all the missing element of its sub-tree, and the element is removed from the list when a proper filler is found higher up in the tree. This allows the semantic value to be handled in place while the syntactic structure is transformed.
Figure 6 Projection of "Where do you move the rectangle to?"
2.2 The Semantic Foundation of the Grammar

2.2.1 HPSG Semantics Principle

Following Montague [Montague, 1974] we adopt the compositional method of modelling natural language semantics. The semantic principle of HPSG dictates the semantic value of a mother phrase is token-identical to the value of the head daughter. The logical form of a phrase is composed from the logical forms of its sub-parts, and the sub-parts of a headed phrase are selected by the head daughter, therefore, we could encode the composition routine in the head daughter.

Example 1: "John walks"

"John walks" can be translated into first order logic as $\text{walk'(john')}$. Where $\text{john'}$ is a logical constant and $\text{walk'}$ is a one-place predicate. The translation can be interpreted in the model-theoretic analysis devised by Montague. The translation is compositionally formed by function application using lambda calculus. Let we assume: $\text{john} \Rightarrow \text{john'}$, $\text{walk} \Rightarrow \lambda P.\text{walk'}(P)$

By functional application,

$$\lambda P.\text{walk'}(P)(\text{john'}) = \text{walk'}(\text{john'})$$

Example 2: "A man walks"

"A man walks" can be translated into first order logic as $\exists x. [\text{man}(x) \land \text{walk}(x)]$ where $\text{a}$ is a logical constant and $\text{man}$ is a one-place predicate. The translation can be interpreted in the model-theoretic analysis devised by Montague. The translation is compositionally formed by function application using lambda calculus. Let we assume: $\text{a} \Rightarrow \lambda P.\lambda Q.\exists x\{P(x) \land Q(x)\}$, $\text{man} \Rightarrow \lambda R.\text{man'}(R)$

By functional application,

"a man" $\Rightarrow \lambda P.\lambda Q.\exists x\{P(x) \land Q(x)\}(\lambda R.\text{man'}(R))$

$\Rightarrow \lambda Q.\exists x\{[\lambda R.\text{man'}(R)](x) \land Q(x)\}$

$\Rightarrow \lambda Q.\exists x\{\text{man'}(x) ^ Q(x)\}$

"a man walks" $\Rightarrow \lambda Q.\exists x [\text{man'}(x) ^ Q(x)](\lambda P.\text{walk'}(P))$

$\Rightarrow \exists x[\text{man'}(x) ^ (\lambda P.\text{walk'}(P))(x)]$

$\Rightarrow \exists x[\text{man'}(x) ^ \text{walk'}(x)]$
Since "A man walks" is formed by the head-subject schema. "walks" being the head of the phrase selecting a noun phrase "a man" as subject. We could encode the semantic composition in the "walk" lexeme.

\[
\begin{array}{c}
\text{sintran\_verb\_lex} \\
\text{subj} [\text{NP}_x [\text{loc} \text{\_sem}(S)]] \\
\text{sem} [S(\lambda P.\_\text{walk}(F))]
\end{array}
\]

Notational Note: NP a shorthand for a synsem value representing a noun phrase. The sem value is the result of the functional application of the function S from the semantic of the NP with the lambda expression.

2.2.2 Recognition of Primary Speech Acts

The grammatical framework we have build so far licenses English sentences of fair complexity to be translated into first order logic by encoding the sub-categorisation, modification, specification and functional application of semantic components of sub-parts in the lexicons. The speech act parsing technique we are developing is motivated by the speech act taxonomy promoted by Searle (Searle, 1979). He suggested a classification of five basic illocutionary acts. FIPA provides a small set of performatives that each of them loosely conforms to one or more of the five illocutionary acts. Our aim is to translate speech act sentence into the FIPA agent language. This can be intuitively illustrated by the following examples:

Example 1: "I inform you that the rectangle is in room1"

FIPA Translation:

\[
\begin{array}{c}
\text{inform} \\
:\text{sender user} \\
:\text{receiver computer} \\
:\text{content is\_in (rectangle, room1)}
\end{array}
\]

This translation assumes that the content language is a prolog predicate denoting formulae in First Order Logic, and the ontology is ignored for the present purpose. The user and computer is valid agent IDs identifying the user and the agent he is
communicating respectively. The content can also be represented as a FIPA proposed reference content language such as SL.

The example illustrates the kind of speech act utterance termed by Searle as Explicit Performative Sentences. The illocutionary act is identified by the main verb of the sentence, i.e. "inform", and the propositional content by the that-clause, "the rectangle is in room1". Also, the subject of the performative verb is in its first person form because it is the speaker who is proposing the act. The addressee of the utterance is the direct object of the performative verb or implicit 'you' if the direct object is optional.

Within the HPSG framework we are developing, we could represent the verb "propose" in the lexicon as a \textit{stran\_verb\_lex} (strict transitive verb) having a second person singular NP as the first complement and the that-clause as the second complement. We devise two semantic sort called \textit{proposition} and \textit{speech\_act} as a subsort of \textit{sem}. Performative verb such as "request" and "inform" would have \textit{speech\_act} as their semantic sort, and ordinary verb would have a \textit{proposition} sort which represents a first order logic formula.

The \textbf{semantic principle} ensures that the semantic value is taken from the head of the phrase constructed at each node of the tree. Therefore the \textit{speech\_act} semantic value is constructed by the "inform" verb, then pass up the head-subject rule. The that-clause semantic a first order logic formula compositionally constructed by the "is in" prepositional verb.
2.2.3 Recognition of Indirect Speech Acts

The speaker utters a sentence and means exactly and literally what he says in the explicit performative sentence. But, notoriously, not all cases of meaning are this simple. In hints, insinuations, irony and metaphor, the speaker means what he says and a lot more. In the light of this argument, we cannot determine the one and the only meaning of an utterance, but a set of intentions or illocutionary acts the speaker is performing in saying so.

Searle proposed that such utterance can be explained by the felicity condition for the successful performance of the speech acts - preparatory conditions, propositional content conditions and sincerity conditions - and their use to perform indirect speech acts consists of indicating the satisfaction of an essential condition by means of asserting one of the other condition.

Searle generalises the following: (S = speaker, H = hearer, A = some action)
S can make an indirect request by

- Asking whether or stating that a preparatory condition concerning H’s ability to do A obtains.
- asking or stating that the propositional condition obtains
- stating the sincerity condition obtains, but not by asking whether it obtains
- Stating or asking whether there are good or overriding reasons for doing A, except where the reason is H wants or wishes, etc., to do A, in which case he can only ask whether H wants, wishes, etc., to do A.

By utilising techniques such as Searle’s generalisation, we can infer the secondary acts from the primary act denoted by the parsed semantic representation.
A set of abstract inference rule can capture Searle’s generalisation.

The inference rule has the following notation

\[
\begin{align*}
[\text{SpeechAct}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{Proposition}] & \Rightarrow \\
[\text{SpeechAct'}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{Proposition'}]
\end{align*}
\]

indicates that if S utters a Proposition with illocutionary force of SpeechAct, then H may infer that S has performed a secondary act of SpeechAct’ with content Proposition’.

Query-Hearer-Ability Rule

\[
\begin{align*}
[\text{query}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{can}(H, \text{Act})] & \Rightarrow \\
[\text{request}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{Act}]
\end{align*}
\]

if Act is an action with action agent = H and \text{can}(A, X) is a modal operator denoting agent A can perform X.

Query-Own-Ability Rule

\[
\begin{align*}
[\text{query}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{can}(S, \text{Act})] & \Rightarrow \\
[\text{inform}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{want}(S, \text{Act})]
\end{align*}
\]

if SAct is an action

Inform-Speaker-Want-Action Rule

\[
\begin{align*}
[\text{inform}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{want}(S, \text{Act})] & \Rightarrow \\
[\text{request}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{Act}]
\end{align*}
\]

if Act is an action with action agent = H and \text{want}(A, X) is a modal operator denoting agent A want to perform X.

Inform-Speaker-Want-Proposition Rule

\[
\begin{align*}
[\text{inform}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{want}(S, \text{Prop})] & \Rightarrow \\
[\text{request}, \text{sender}: S, \text{receiver}: H, \text{content}: \text{Act}]
\end{align*}
\]

if Prop is a proposition and \text{want}(A, X) is a modal operator denoting agent A want the action Act to be performed. By performing Act, proposition Prop would be satisfied.
Inform-Speaker-Want-Speech Act Rule

\[ \text{[inform, sender:S, receiver:H, content:want(S, SA)]} \Rightarrow \text{SA} \]

if SA is a speech act where the actor of the speech act is S.

The inference rule can be applied recursively starting from the surface speech act. Therefore a secondary speech acts satisfy the following constraint:

\[ \text{SA'} \text{ is a secondary act of } \text{SA} \text{ if } \]
\[ \text{SA infers SA'} \text{ or } \]
\[ \text{SA infers SA'' and SA is a secondary act of SA''}. \]

Note the above definition would allow alternatives to be found by exploring all possible inference paths. Each path denotes a possible inference to a different secondary speech acts inferred from the surface form.

The inference rule listed is not intended to be complete, more of these conversational postulates can be added to capture other abstraction given empirical study of conventional speech act behaviour. This approach of recognition of indirect speech acts is motivated by the study of conversational postulates by Gordon and Lakoff [Gordon and Lakoff, 1975]. Most commentators on indirect speech acts have remarked on the role of politeness, however, our approach would suffer from cultural specific variation of indirectness as suggested by study of indirectness of request in English and German (House and Kasper 1981) and English and Russian (Tbomas 1983).
Chapter 3

Implementation

In this chapter

We will survey the use of Attribute Logic Engine in conjunction with other grammar development tools to implement the design devised in Chapter 2. Key details of implementation are described. We will show how the Attribute Logic Engine can be integrated into prolog as a reusable parser generator for Natural Language Application. We will show how a path searching technique is used to implement the indirect speech acts inference engine.
3.1 Choice of implementation language

Recall from the last section, the HPSG framework is based on the formalism of sorted features. Sorted-features have lent itself to computational implementation, owing to the use of definite clause program in the logic programming paradigm in which the terms are sorted-feature structure. Logic programming system such as LOGIN and PROLOG-II, and linguistic programming system such as PATR-II have implemented a feature structure system.

The Attribute Logic Engine by Penn [ALE, 1998] provides an alternative implementation of a feature structure system as a sub-system of ANSI prolog. Terms in grammars and logic programs are specified in ALE using a typed version of Rounds and Kasper’s attribute-value logic with variables. The ALE system has a strong emphasis for the use of specifying Phrase Structure Grammar employing feature structure, such as GPSG, HPSG and LFG. The Phrase Structure Grammar is specified in a manner similar to definite clause grammar with procedural attachment. Lexical development is supported by lexical rules operating on surface lexicons. ALE compiles a Prolog-optimised bottom-up dynamic chart parser from the grammar specification. The parser can be incorporated into prolog programs via the use of ALE’s prolog interface.

The Attribute Logic Engine was chosen as the implementation language for the following reasons.

- **Reference implementation:** Various implementation of Head-driven Phrase Structure Grammar on ALE have been cited, readers are encouraged to read [Carpenter, 1997]. Our implementation is an adaptation of the above reference approach, with our emphasis focusing on speech act parsing and the use of an ACL semantic representation.

- **Ease of use:** The HDRUG system is a complementary graphical interface for the development of ALE grammar. It effectively simplifies the development efforts on the system.
• **Platform dependency:** ALE is designed as a sub-system of prolog and currently implemented on various widely-used prolog packages, e.g. Sicstus and Quintus. Therefore the platform dependency of ALE is only constrained by the platforms the above prolog system have been implemented on.

• **Logic Programming Paradigm:** ALE is a complete logic programming system that could interface with the ordinary prolog engine. Our design has suggested an indirect speech act inference engine that can be easily implemented by a logic programming language.

• **Natural Language Generator:** ALE has a built-in *Semantic Head-Driven Generation* generator engine using the same grammar specification of the parser engine. Therefore it provides a natural two-way parsing and generation engine for our design.

![Figure 6 HDRUG running on LINUX showing the parse tree and an AVM](image)
3.2 Implementation

Most HPSG features can be directly implemented using the constructs of ALE:

3.2.1 Type Hierarchy and Feature Structure

ALE is a language of strong typing. The inheritance hierarchy of sorts is defined by the “\texttt{type1\ sub\ [type2, type3, \ldots\ type_n]\ intro\ [f1:type_a, f2:type_b]}” construct governing the sub-sorting relations between types. Intuitively, the definition says type2, type3 and type_n are sub-types of type1, therefore they will inherit the constraints and the set of feature-value pairs \texttt{f1} and \texttt{f2} of type \texttt{type_a} and \texttt{type_b}.

However, ALE does not provide an explicit list type. The implementation of list is defined by a cyclic sort:

\begin{verbatim}
list sub [e_list, ne_list].
  e_list sub [\].
  ne_list sub [\]
  intro [hd:bot, tl:list].
\end{verbatim}

This specification describes a list can be an empty list (\texttt{e_list}) or non-empty list (\texttt{ne_list}). It implicitly declared that an empty list has no features defined for it. A non-empty list would have a head (\texttt{hd}) of type \texttt{bot}, and a tail (\texttt{tl}) of type \texttt{list}.

Recall the type-hierarchy of HPSG discussed in section 2.1.2.1, can be directly encoded as an ALE type-hierarchy. The constraints associated with the lexical hierarchy can be applied to a sort by the type constraint construct.

Our grammar introduced a set of features describing the linguistic form of the sign. The head features govern the parts of speech (\texttt{pos}) of the clause. The \texttt{pos} is either functional (\texttt{func}) or substantive (\texttt{subst}), the former denotes a marker or a determiner, while the later defines the ordinary parts of speech.
<table>
<thead>
<tr>
<th>Features</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>subst</strong></td>
<td></td>
</tr>
<tr>
<td><strong>mod</strong></td>
<td>A specification of a sign that is modified by this parts of speech. E.g. an adjective will specifies a NP as the mod value, so that the modifier-head schema will license a modified phrase to be projected</td>
</tr>
<tr>
<td><strong>noun</strong></td>
<td></td>
</tr>
<tr>
<td><strong>agr</strong></td>
<td>The number and person of the nominal. Used in noun-verb concord.</td>
</tr>
<tr>
<td><strong>case</strong></td>
<td>The case of the nominal. i.e. Nominal or Accusative.</td>
</tr>
<tr>
<td><strong>verb</strong></td>
<td></td>
</tr>
<tr>
<td><strong>vform</strong></td>
<td>The form of the verb, e.g. base, finite, infinite, past, past-participle, present participle.</td>
</tr>
<tr>
<td><strong>aux</strong></td>
<td>Boolean specifying whether the verb is an auxiliary or a verb phrase is headed by an auxiliary</td>
</tr>
<tr>
<td><strong>inv</strong></td>
<td>Boolean specifying whether the verb phrase is inverted</td>
</tr>
<tr>
<td><strong>prep</strong></td>
<td></td>
</tr>
<tr>
<td><strong>pform</strong></td>
<td>The form of preposition. e.g. to, from, by, on. A verb could select its prepositional complement by specifying the pform of the prepositional phrase.</td>
</tr>
<tr>
<td><strong>func</strong></td>
<td></td>
</tr>
<tr>
<td><strong>spec</strong></td>
<td>Specifies a specification of a sign which this sign will specify. This is used by the specifier-head schema.</td>
</tr>
<tr>
<td><strong>marker</strong></td>
<td>Marker head, such as the ‘that-clause’, that is a marker of a finite sentence. The head-marker rule uses this feature to transmit the marked feature up the projected tree. So that verb such as “inform” which takes a that-clause as a complement can specify the marking = that.</td>
</tr>
<tr>
<td><strong>det</strong></td>
<td>It is introduced to distinguish a determiner from a marker, so that head-marker schema would only operate on the marker.</td>
</tr>
</tbody>
</table>

Figure 7 Head Features
As our design suggested, all linguistic objects are specified as a sub-type of `synsem`.

<table>
<thead>
<tr>
<th>Features</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>synsem</strong></td>
<td></td>
</tr>
<tr>
<td><strong>local</strong></td>
<td>Store information about the local information of the sign</td>
</tr>
<tr>
<td><strong>syn</strong></td>
<td></td>
</tr>
<tr>
<td><strong>head</strong></td>
<td>The possible head features we have discussed which specifies the parts of speech of the sign</td>
</tr>
<tr>
<td><strong>subj</strong></td>
<td>A list of synsem object denoting the unsatisfied subject valence of this sign. Used by subject-head schema to determine the valid subject-head projection. Although English only licenses a single subject, a list is retained to allow cross-language use.</td>
</tr>
<tr>
<td><strong>comps</strong></td>
<td>A list of synsem object denoting the unsatisfied complement valence of this sign. Used by head-complement schema to license English sentence of the form SVO and SVC.</td>
</tr>
<tr>
<td><strong>marking</strong></td>
<td>Carry the marker of the sign. e.g. that, whether.</td>
</tr>
<tr>
<td><strong>sem</strong></td>
<td>The semantic value of the sign. It is implemented using the (a_) type specifier allowing un-typed prolog term to be used. It allows us to compile a complex prolog term encoding the semantic that could be passed to a prolog interpreter via the ALE-Prolog interface.</td>
</tr>
<tr>
<td><strong>non-local</strong></td>
<td>Store information collected from the sub-tree of the projection</td>
</tr>
<tr>
<td><strong>inherited</strong></td>
<td></td>
</tr>
<tr>
<td><strong>slash</strong></td>
<td>A list of local values collecting the traces from the lower sub-tree of the projection</td>
</tr>
</tbody>
</table>

Figure 8 synsem features
3.2.2 Lexicon

With the type system fully specified, we could implement the lexicon using the lexicon development constructs of ALE. A lexeme is defined by its surface morphological form. Each lexeme is associated with a type from the hierarchy, additional constraints can be added and feature value specified to instantiate a distinct lexeme.

We observed that the verb is rich in morphological and valence differences. Verbs differ in their complementation has no morphological relationship, but verbs of different form bears a morphological transformation with their valence features intact. Therefore we can conclude that our implementation can base on a division of verb type into two sub-classes: subcat_verb_lex, vform_verb_lex.

The subcat_verb_lex constraints the valence features of the verb. It could have possible sub-types of transitive_verb_lex and intransitive_verb_lex that specifies an empty complementation in the later case.

The vform_verb_lex constraints the morphological features and the form of the verb. It could have possible sub-types of past_verb_lex and past_participle_verb_lex that under-specify the valence features, but define the head feature of the parts of speech.

HPSG suggests a multiple-inheritance paradigm allowing a lexeme to be associated with more than one type. A past form of “walked” would inherit from intransitive_verb_lex and past_verb_lex. However, ALE disallows such specification.

We observe that all verbs can be syntactically associated with any vform_verb_lex, but only one from subcat_verb_lex. Therefore, we could achieve the same effect by using lexical transformation rule.
A lexical transformation rule in ALE allows lexicon unified with a type specification to be morphologically transformed, and the sign specification to be changed to another sign. Our grammar implements four lexical rules:

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sg3_pres</td>
<td>Transform lexicon from subcat_verb_lex to a third person singular finite verb</td>
</tr>
<tr>
<td>non_sg3_pres</td>
<td>Transform lexicon from subcat_verb_lex to a plural or non-third person singular finite verb</td>
</tr>
<tr>
<td>base_verb</td>
<td>Transform lexicon from subcat_verb_lex to its base form</td>
</tr>
<tr>
<td>subject_aux_inversion</td>
<td>From a aux_verb_lex to a inv_aux_verb_lex</td>
</tr>
<tr>
<td>lexicon_satisfaction</td>
<td>Promotes all lexeme to a word sign, except subcat_verb_lex because all subcat_verb_lex has to subject to a lexical rule transformation</td>
</tr>
</tbody>
</table>

The specification of the lexical hierarchy has already taken away the complexity of coding the lexicons. ALE provides a MACRO features that allow feature-structure to be defined as a parameterised-macro, so that the feature can be replaced whenever a macro description is found in the source.
3.2.3 Schemas Implementation

HPSG Schemas are highly general phrase structure rule operating on linguistic categories defined by type-feature system, the applicability of the rules are govern by a set of language independent principles. The poly-morphic nature of type system allows only six PS rules to be the major schemas defining the projection of grammatical sentences.

We encode Phrase Structure rule in ALE using definite clause grammar, see [Pereira and Warren, 1980]. Definite clause grammars are an extension of the well-known context-free grammars. A grammar rule in Prolog takes the general form:

\[ \text{mother} \Rightarrow \text{body (conditions)} \]

The head is a non-lexical category, and the body is a sequence of categories that have to be satisfied for the `mother` to be derived. Extra conditions can be specified to constraint the applicability of the rule.

HPSG Schemas can be define as usual PS rules in definite clause grammars, and the principles are specified as predicates in the conditions of the rule. Since all the principles we have discussed in 2.1.2.2. operates on all the headed schemas, all the PS rules will satisfy the `general_principles`.

\[
\text{general_principles}(\text{Mother}, \text{Head}, \text{Daughters}, \text{SemanticHead}) : - \\
\text{head_feature_principle}(\text{Head}, \text{Mother}), \\
\text{semantic_principle}(\text{SemanticHead}, \text{Mother}), \\
\text{non_local_features_principle}(\text{Head}, \text{Daughters}, \text{Mother}), \\
\text{universal_trace_principle}(\text{Head}, \text{Mother}).
\]

where

- `Mother` is the left hand side category of a PS rule,
- `Head` is the head daughter in the body of the rule,
- `Daughters` is a list of categories resembling the set difference of the body and the Head,
- `SemanticHead` is the category defining the semantic of the headed phrase. For a modifier phrase, the semantic head is the modifier. For a specifier phrase, it is the specifier. For other phrases, it is the head selecting the daughters.

The valence principle is explicitly coded in the PS rules.
3.2.3 Semantics Extraction

In section 2.2.1, we discussed the use of lambda reduction to compute the semantic representation of a sub-tree of a projection. We implement lambda reduction in ALE using unification and partial application.

Suppose, we have

\[ \text{walk} = \lambda P. \text{walk}'(P) \]
\[ \text{john} = \text{john}' \]

therefore, “john walks” = \( \lambda P. \text{walk}'(P)(\text{john}) = \text{walk}'(\text{john}') \)

In ALE, we could encode the semantic representation of walk in the \texttt{sem} feature.

“walks” \( \Rightarrow \) \texttt{sem : (a_ P^walk(P))} \\

“john” \( \Rightarrow \) \texttt{sem : (a_ john)}

Note: The (a_ Y) is an explicit type in ALE allowing any content of Y to be specified. “^” is chosen to be a prolog operator denoting the lambda operator.

However, for lexeme such as determiner “a”

“a” \( \Rightarrow \lambda P \lambda Q. \exists x(P(X), Q(X)) \)

we cannot encode it in ALE as \( P^Q^\exists(x, \text{and}(P(X), Q(X))) \) since prolog and ALE forbids a variable to be a functor.

However, the problem can be solved by using unification and partial application

Since “walks” selects the subject NP “john”, and lexeme “john” has a \texttt{sem} value:

\[ \text{sem : (a_ (john^S^S))} \]

In the sign of “walk”, the \texttt{subj} feature specifies the NP.

\[ \text{syn:subj:[ \{ @NP, \texttt{sem:(a_ P^walk(P)^Q)} \} ]} \]
\[ \text{sem:(a_ Q)} \]

Since \( P = \text{john}, \ S = \text{walk}(P), \) therefore \( Q = \text{walk}(\text{john}) \)
Therefore, the semantic of “john walks” equals the instantiated value of walk(john) because the semantic principle ensures the semantic is taken from the head (walk).

Using unification, we could encode complex sentence such as “a robot walks”

```
"a" ⇒
  sem : (a_ ( (X^S2)^exists(X, and(S1, S2))))
  syn : head : spec :(@NOUN, sem:(a_ (X^S1)))

"robot" ⇒
  sem : (a_ (Y^robot(Y)))

"walks" ⇒
  syn:subj:[ (@NP, sem:(a_ P^walk(P)^Q)) ]
  sem:(a_ Q)

"a robot" ⇒
  sem : (a_ (Y^S2^exists(Y, and(robot(Y), S2))))
    because X = Y and S1 = robot(Y)

"a robot walks" ⇒
  sem:(a_ exists(Y, and(robot(Y), walk(Y))))
    because
      Y=P, S2=walk(P),
      Q = exists(Y, and(robot(Y), S2))
```

### 3.2.4 Primary Speech Acts Recognition

Recall from the last section, the semantic information is collected by the \textit{sem} value of the sign. The \textit{sem} feature is sub-typed by \textit{proposition} and \textit{speech-act}. While a \textit{speech-act} type is only introduced by an explicit performative verb, other semantic information built so far is a \textit{proposition}. As a result, we could introduce an explicit schema \textit{speech-act-satisfaction} as a high level schema which will take a sign with semantic value of speech-act into a valid input to the speech act analysis engine. When such schema is activated, it triggers the speech act analysis engine to infer on possible secondary acts.
Speech-act-satisfaction Schema

\[
\begin{align*}
\text{[speech-act } S] & \Rightarrow \left[ \begin{array}{c}
\text{phrase} \\
\text{loc|sem|}\text{(speech-act } S)
\end{array} \right]
\end{align*}
\]

For each clause type, we have a schema to derive the associated surface speech-act semantic from the propositional content.

**declarative_rule Schema**

\[
\begin{align*}
\text{phrase } P \\
\text{loc|sem|}\text{(speech-act inform}(A, B, \text{Prop})) \\
& \Rightarrow \left[ \begin{array}{c}
\text{phrase} \\
\text{loc|sem|}\text{(proposition } \text{Prop})
\end{array} \right]
\end{align*}
\]

if \( P \) is an unmarked finite non-inverted sentence, \( A \) and \( B \) are the terms denoting the speaker and the hearer respectively derived from the context.

**imperative_rule Schema**

\[
\begin{align*}
\text{phrase } P \\
\text{loc|sem|}\text{(speech-act request}(A, B, \text{Sem})) \\
& \Rightarrow \left[ \begin{array}{c}
\text{phrase} \\
\text{loc|syn|subj:}\text{Subj} \\
\text{loc|syn|comps:}\text{} \\
\text{loc|sem|Sem}
\end{array} \right]
\end{align*}
\]

if \( P \) is a base form verb phrase with unsaturated subject \( \text{Subj} \) denoting a noun phrase in second person. It places no restriction on the type of \( \text{sem} \), because a query allows an indirect speech act to be embedded.

**interrogative_rule1 Schema**

\[
\begin{align*}
\text{phrase } P \\
\text{loc|sem|}\text{(speech-act query}(A, B, \text{Sem})) \\
& \Rightarrow \left[ \begin{array}{c}
\text{phrase} \\
\text{loc|sem|Sem}
\end{array} \right]
\end{align*}
\]

if \( P \) is a inverted unmarked sentence.

Since not all interrogative derives from inverted sentence such as a wh-question with the wh-element equals the subject. We need a second rule to handle this type of sentence.
3.2.5 Indirect Speech Acts Recognition

Our design suggested an inferencing engine which takes a surface speech-acts as the input, and infers possible indirect speech-acts from the surface. With the phenomenon of embedded speech acts, it emerges that a single level of inferencing would not be sufficient to model such behaviours. We view the inferencing engine as a possible path searching algorithm; the edge of the path denotes a one-step inference from one speech-act to another. The path searching algorithm stops when no inference rule is applicable. The definition of indirect speech acts can be implemented through the trivial prolog definition.

\[
\begin{align*}
\text{interrogative_rule2 Schema} \\
\begin{array}{l}
\text{phrase} \\
P \\
\text{loc|sem|}(\text{speech-act}) \\
\text{query}(A, B, \text{Sem})
\end{array} \\
\Rightarrow \\
\begin{array}{l}
\text{phrase} \\
P \\
\text{loc|sem|Sem}
\end{array}
\end{align*}
\]

if \( P \) is an unmarked finite non-inverted sentence. The \( \text{Sem} \) value heads a \( Q \)-quantifier. We treat all \( \text{wh} \)-word as logical quantifier just like the determiners.

\[
\text{% True when C is a secondary speech act inferred from A.}
\text{secondary_act}(A, C) :- \\
\text{infer}(A, C). \\
\text{secondary_act}(A, C) :- \\
\text{infer}(A, B), \\
\text{secondary_act}(B, C). \\
\]

For all secondary acts found, they constitute the possible intention of the speaker. As Allen terms it the plan of the user [Allen, 1983]. Instead of following the belief and wants model of Allen’s system that the system requires to reason about belief and wants modality. We express the inference rule by some high level abstraction of wants and beliefs. For example, if speaker queries the hearer about his ability to do some action, we could infer that he wants that action to be executed. The hearer by being co-operative, might infer that he/she shall perform that action for the speaker. This approach can be justified by Allen’s belief model. At a meta-level, our inference rule relates to the common-sense behaviour we have learnt from experience, and pre-
programmed into the inference engine without possibly time-consuming theorem proving of modal logic.

The disadvantage of this approach being when more speech-act types add to the system, we require a set of new inference rules which will operate on this action. If we employ the belief model, we do not need to alter the inference engine because it reasons from the underlying definition of the speech-acts.

3.2.6 Limitations

The Attribute Logic Engine although provides an excellent set of construct to complement the theory of HPSG. However, several weakness of the system have limited the use of the grammar:

- The Semantic Head Generator Engine uses the same grammar specification to generate text as well as parsing text. However it suffers from infinite recursion problem for some grammar specification. Preliminary test shows that the bounded method used in the implementation of the engine does not catch such problem as claimed. It seriously reduces the usefulness of the grammar developed, and prevents us from using the grammar to generate text from a FIPA ACL message.

- The ALE system does not provide feature value default. If a default value can be automatically specified when a feature is under-specified, it will reduce the amount of redundancy in coding the lexicon.

- Our lexical hierarchy for the verb requires a lexical rule to promote a verb defined by its valence features to a verb defined by its form. A feature-value overriding features and multiple-inheritance would be favourable to achieve the same effect but with a more natural use.
Chapter 4

User Guide and Examples

In this chapter

Section 4.1 describes a demonstration program incorporating the parser and the analyser as the input interface.
Section 4.2 illustrates the fundamental system mechanics and sample interpretations are examined in detail.
4.1 Demonstration System

The demonstration program incorporates a virtual blocks-world environment that the user can communicate with a virtual robot via natural language to query or manipulate the virtual world. It facilitates the parser system we have developed in a finite domain demonstrating the possible use of the parser in a broader application.

Figure 10 Demonstration Program

The program is composed of a TCL/TK applet interfacing with Sicstus Prolog’s TCL/TK library. It demonstrates the parser can be incorporated into other programming languages given a binding exists between them. The prolog sub-system of the program incorporates a STRIPS planner as the action planner for the virtual robot, and a first order logic interpreter evaluates formula against the world model. The planner allows quantified goals to be planned, and the interpreter allows query to be answered based on Prolog’s resolution engine.
### 4.1.1 The Blocks World

The blocks world constitutes the following referable entities, attributes and operations.

<table>
<thead>
<tr>
<th>Description</th>
<th>Natural Language Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entities</strong></td>
<td></td>
</tr>
<tr>
<td>robot</td>
<td>The only virtual robot the user can communicate with.</td>
</tr>
<tr>
<td>user</td>
<td>The entity representing the user of the system.</td>
</tr>
<tr>
<td>rectangle</td>
<td>The moveable, colourable entity in the blocks world</td>
</tr>
<tr>
<td>circle</td>
<td>The moveable, colourable entity in the blocks world</td>
</tr>
<tr>
<td>room1..4</td>
<td>The rooms in the blocks world</td>
</tr>
<tr>
<td><strong>Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>A is in B</td>
<td>Object A is in Room B</td>
</tr>
<tr>
<td>A is an object</td>
<td>A is an object</td>
</tr>
<tr>
<td>A has colour B</td>
<td>Object A is of colour B</td>
</tr>
<tr>
<td>A next to B</td>
<td>Room A is next to B</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td></td>
</tr>
<tr>
<td>move robot</td>
<td>Move the robot from a room to another</td>
</tr>
</tbody>
</table>
move object  Robot picks up an object and move that object to another room
move an object  move every object to, move the rectangle to, move the circle to, relocate,
pick-up object  Robot picks up an object  pick up A, pick A up
put down object  Robot puts down an object  put down A, put A down
colour object  Robot colours the object to another colour  colour A red, colour A green, paint A red

**Figure 11 Entities, Attributes and Operations of the Blocks World**

The system operation is illustrated by the following flowchart
Responses are based on a naïve model of pragmatic. When the user submits a request, the system will try to find a plan to execute the request, if no plan is found, the system responds with a “sorry” message. The system will be able to answer a yes-no question or wh-question given a proposition known in the ontology of the system. The user can inform the system of certain facts. The system will check the fact against the world model, and deny the fact if it is inconsistent.

4.2 Examples

4.2.1 Detailed Example

The following example illustrates the system mechanics of transforming a user input into response.

User Input: “Can I ask whether you can move the box to room1”

Parse Tree:

```
User Input: “Can I ask whether you can move the box to room1”
Parse Tree:

query(user, robot, D)
D = can(user, C)
C = query(user, robot, B)
B = can(robot, A)
A = move_robot(_, room1))
```

Parse output:
```
query(user, robot,
  can(user,
    query(user, robot,
      can(robot, move_robot(_, room1)))))
```
Given the surface speech act, we can infer the indirect speech acts.

\[ \text{query}(\text{user, robot}, \text{can}(\text{user}, \text{query}(\text{user, robot}, \text{can}(\text{robot, move_robot}(_, room1))))) \]

⇒ query-own-ability-rule ⇒

\[ \text{inform}(\text{user, robot, want}(\text{user, query}(\text{user, robot, can}(\text{robot, move_robot}(_, room1))))) \]

⇒ query-own-speech-act-rule ⇒

\[ \text{query}(\text{user, robot, can}(\text{robot, move_robot}(_, room1))) \]

⇒ query-ability-rule ⇒

\[ \text{request}(\text{user, robot, move_robot}(_, room1)) \]

Therefore, the possible indirect speech acts are

1. \[ \text{query}(\text{user, robot, can}(\text{robot, move_robot}(_, room1))) \]
2. \[ \text{request}(\text{user, robot, move_robot}(_, room1)) \]

because they represent the end-nodes of inference process that could be executed by the system.

The system respond to 1 with “yes, I can” if the robot can execute \text{move_robot}(_, room1). Then it will respond to 2 by searching for a plan that will satisfy the predicate \text{move_robot}(_, room1), and execute the plan if it exists.

### 4.2.2 Other Examples

The system is evaluated with the following input. Their corresponding speech acts representation are shown respectively:

1. **Can you move the rectangle to room1?**
   - \[ \text{query}(\text{user, robot, can}(\text{robot, do}(\text{robot, move_object(rectangle, _A, room1))))} \]
   - \[ \text{request}(\text{user, robot, do}(\text{robot, move_object(rectangle, _A, room1)})) \]

2. **Can you move every object to room1?**

3. **Move the circle to room1!**
- request(user, robot, do(robot, move_object(circle, _A, room2)))

4. **May I ask you to move the rectangle to room1?**
- query(user, robot, may(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))
- inform(user, robot, want(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

5. **I ask you to move the rectangle to room1.**
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

6. **Can I ask you whether I can request you to move the rectangle to room1?**
- query(user, robot, can(user, query(user, robot, can(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))
- inform(user, robot, want(user, query(user, robot, can(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))))
- inform(user, robot, want(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))
- query(user, robot, can(user, request(user, robot, do(robot, move_object(rectangle, _A, room1)))))
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

7. **I request you to move the rectangle to room1.**
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

8. **I request you to move the rectangle to room1.**
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

9. **I want to move the rectangle to room1.**
- inform(user, robot, want(user, do(user, move_object(rectangle, _A, room1))))
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))

10. **I want to ask you whether you can move the rectangle to room1**
- inform(user, robot, want(user, query(user, robot, can(robot, do(robot, move_object(rectangle, _A, room1))))))
- query(user, robot, can(robot, do(robot, move_object(rectangle, _A, room1))))
- request(user, robot, do(robot, move_object(rectangle, _A, room1)))
11. I want to ask you whether I can request you to move the rectangle to room1.
   • inform(user,robot,want(user,query(user,robot,can(user,request(user,robot,do(robot,move_object(rectangle,_A,room1))))))
   • inform(user,robot,want(user,request(user,robot,do(robot,move_object(rectangle,_A,room1)))))
   • query(user,robot,can(user,request(user,robot,do(robot,move_object(rectangle,_A,room1)))))
   • request(user,robot,do(robot,move_object(rectangle,_A,room1)))

12. Do you want to move the rectangle to room1
   • query(user,robot,want(robot,do(robot,move_object(rectangle,_A,room1))))
   • request(user,robot,do(robot,move_object(rectangle,_A,room1)))

13. Is every object red?
   • query(user,robot,forall(A_, imp(object(A_, red(A_))))

14. Is the rectangle red?
   • query(user,robot,red(rectangle))

15. Where is the rectangle?
   • query(user,robot,where(A_, location(rectangle, A_))

16. Which object is red?
   • query(user,robot,which(A_, object(A_), red(A_))

17. Which object is in room1?
   • query(user,robot,which(A_, object(A_), location(A_, room1))

These examples suggested the parser produces correct surface speech acts for all clause type we have set out to cover. The indirect speech acts obtained can be explained by the inference rules applied. Although for some complex sentence, the large number of possible speech acts suggested a heuristic to sort the most favourable intention is required.
Chapter 5

Conclusion and Further Research

5.1 Conclusion

In this project, we have designed and built a prototype English Speech Acts Parsing System that is capable of parsing a subset of English Grammar into the FIPA Agent Communication Language. The parser is complemented by a conceptual implementation of a secondary speech acts analyser extracting deep intentions from the surface speech act. The applicability of the parser is evaluated by a concept demonstrator interpreting natural language input to produce textual responses and actions to a blocks world model.

5.1.1. Achievements

Key achievements and contributions can be summarised as follows:

1. **Linguistics and Computing:** The Syntactic Theory of Head-Driven Phrase Structure Grammar is shown to have a direct computational implementation. The emphasis on constraints, unification and derivation suggest a close relation to constraint logic programming that synthesis with existing theorem proving and AI reasoning tools.

2. **Lexicon Development:** Our grammar builds on the type feature structure formalism. We have devised an extension to the original HPSG framework by generically classifies different parts of speech through type and constraints. It simplifies the definition of lexicon by providing a rich lexical hierarchy.
3. **English Coverage:** Our grammar partially covers the English Grammar set out by QGSL. It enables common English speech act sentence resembling the declaratives, interrogatives and imperatives to be parsed into message of the FIPA Agent Communication Language.

4. **Speech Acts Analysis:** The secondary speech acts analysis system is only a proof of concept of recognition of deep intentions. It correctly extract possible intentions based on a series of inference justified by a belief model of speech acts. However, it has a main limitation of over-generating possible intentions and lacking a method to select the most appropriate intention.

5. **Concept Demonstration:** The demonstrator has shown an example of the type of application of the parser, especially in an agent framework communicating using an Agent Communication Language.

6. **Modular Design:** The parser is capable to be incorporated into other system and programming language based on its modular design, high platform coverage and broad language binding of prolog.

### 5.1.1. Insights

One might question the relevance of speech-acts in natural language computer interface. A polite indirect request expressed in a query such as, “Do you think you can move the box to room1?” is an unnatural typed input to a computer. However, a proficient natural language parser capable of comprehending conversational utterance would serve as an onset to a major sub-system of a speech understanding engine. With the advent of speech recognition technology, a speech-to-text system can serve as a pre-processor for our parser. The integrated system can be applied to computerised telephone technical support, secretary service or conversational robots. The commercial undertaking of such system is apparent in the state-of-the-art *Wildfire* personal information system in the Orange Mobile network [WILDFIRE, 1999].
This thesis suggests a proficient parser shall not only have a broad coverage of the target language, but the ability to recognise the indirect intentions of the speaker. Inferring indirect speech acts suffer all the problems of reasoning with uncertainty. Contextual information and empirical experience would aid correct reasoning. However, communication is about interaction of the agents. Ambiguity is best to be solved through further interaction of a series of ask-and-tell. As a consequence, speech acts resolution can be resolve with conversational agents engaged in discourse with the understanding of pragmatic issues.

5.1.2. Future Research

The proof of concept implementation of the parser presented in this report has shown encouraging results of future extension. This section discusses some aspects of the system that can be developed further.

- **English Coverage:** The grammar we have developed only covers a small portion of the full English Grammar concerning speech acts. The lexical hierarchy has already encodes a fair complexity of the lexicon, but certain parts of speech needs to be extended to cover more variety of verb type, adjuncts and nominal classes. Pronouns have been ignored in this study to ease the complexity of implementing the binding theory and reference resolution. These will extend the grammar to solve the limitation of utterance between only two agents.

- **Lexicon:** The HPSG formalism has a strong emphasis of complex encoding of linguistic information at the lexical level retaining a general schemas for projection. A viable extension to extend coverage is to connect to publicly available lexicon-base such as WordNet and translate the proprietary representation to a HPSG encoding.

- **Discourse Representation Theory:** As the thesis suggested a discourse understanding would aid indirect speech acts analysis. Discourse Representation Theory [Kamp and Reyle, 1991] has given major insights into the understanding of discourse fragments and computational linguists have adopted this theory as the semantics for discourse analysis.
• **Plan based Speech Act Analysis:** Our speech act analysis is based on an empirical view of speech act use. As Allen suggested [Allen, 1983], a speech act can be realised as a plan of the speaker, by re-constructing the plan based on the beliefs of the speaker in the beliefs of the hearer, the hearer can infer the intention of the speaker by executing the speech-act. This approach has been implemented before, however with similar limitation of over-generation and issues of modal logic theorem proving.

• **Parsing and Generation:** The Attribute Logic Engine provides a generation engine that will generate text using the same grammar for parsing. However, the generator suffers from infinite recursion for some grammar specification. An implementation of a bounded generating routine would be desirable so that the grammar can be used to generate text from Agent Communication Language.
Appendix 1 - Bibliography / References


[Beun, 1962] Robbert-Jan Beun, Context and Form: Declarative or Interrogative, that is the question, Institute for Perception Research, Netherlands.


[WILDFIRE, 1999] web site: http://www.orange.co.uk

The full source code of the system and grammar developed can be found at ~wwhl97/nlp/